Visible Analyst®

Operation Manual

Version VA2013

Visible
Systems Corporation

Printed and bound in the United States of America.

This manual was prepared using Microsoft Word for Windows 2000 and updated using Microsoft Word for Windows 2010. Information used in examples is fictitious and for purposes of example only.

Visible Analyst
Version VA2013
Operation Manual

Visible Systems Corporation
63 Fountain Street
Kenwood Centre Suite 301B
Framingham, MA 01702
Technical Support: 781-778-0200
Fax: 508-628-1515
Internet: http://www.visible.com
E-mail: support@visible.com

Table of Contents

Introduction

VISIBLE ANALYST

Visible Analyst is an easy-to-use Computer Aided Software Engineering (CASE) strategic planning; analysis; design; and object, data, and process modeling tool. It is supplied as a Microsoft® Windows®-based set of integrated tool sets that can be easily installed on any current Windows operating system. Its powerful yet easy-to-master diagramming, analyzing, and modeling capabilities enhance not only the productivity of software developers but also the overall efficiency of software analysis, design, and modeling practices. In addition, the software's ease of use permits people outside the traditional software development function to become involved in the development process. Even non-technical managers and system end-users are able to use Visible Analyst to effectively communicate system requirements and goals, proposed modifications, etc. The three main components of Visible Analyst are summarized below.

Visible Analyst Diagramming Tool

The Visible Analyst diagramming tool is an easy-to-use, mouse-driven graphics package that allows you to draw virtually any type of diagram. Standard symbol and line types and text entry capabilities, combined with menu-selectable editing features, allow you to create and update diagrams that are unstructured and unencumbered with any methodology or rules and to organize them on a project basis.

Visible Rules

The Visible rules are built into Visible Analyst to give you guidance and to provide consistency in carrying out your activities. The various methodologies supported by your tool set(s) are codified in the rules. They are implemented in two different ways. Some are applied as you draw items onto project diagrams. Violations of the methodologies are not allowed and certain required actions are automatically taken. Other rules are applied on request when you execute the Analyze function from the Diagram menu. Analysis also displays some error and warning messages, suggesting further action for you to take.

Visible Repository

The Visible repository includes a database that acts as a central repository for all information pertaining to a project. The Visible repository is tightly integrated with all process modeling, data modeling, and object modeling tool sets, providing data repository support for diagrams that are created under the guidance of one of the supported methodologies. All additions and modifications to diagrams are automatically reflected in the repository with corresponding repository entries. Fields of ample size are provided for each repository entry so that detailed definitions may be created for all entries. Different object types have different repository entry fields available. Wildcard and selective search facilities make navigation within the repository easy. Reports may be generated on virtually any section or subsection of the repository.

VISIBLE ANALYST TOOL SETS

Visible Analyst consists of five tool sets as described below.

- The Visible Analyst data modeling tool set allows you to create entity relationship diagrams (ERDs), with the requisite rules for disciplining data model creation and including the repository to capture the information. Visible Analyst supports ERDs based on data modeling notations defined by IDEF1X, Bachman, and Information Engineering (IE).
- The Visible Analyst process modeling tool set allows you to create data flow diagrams (DFDs), functional decomposition diagrams (FDDs), structure charts, and entity life history (ELH) diagrams. There is a choice of four sets of methodology rules for data flow diagrams: Yourdon, Gane & Sarson, SSADM, and Métrica. There are rules for the Yourdon/Constantine method for doing structured design. The data flow diagrams and structure charts share a common repository to capture all information from the diagrams.
- The Visible Analyst object modeling tool set allows you to create class, state transition, use-case, sequence, collaboration, component, deployment, and activity diagrams for object model creation including the repository to capture the information. Visible Analyst supports the UML standard, giving you a bridge between traditional structured and relational techniques and the newest object-oriented technologies, including use-case diagrams, sequence diagrams, collaboration diagrams, and activity diagrams.
- With the introduction of the Business Process Modeling Notation (BPMN) the Visible Analyst provides a modeling notation that can be communicated to and understood by all business users, from the business analysts developing the models, to the technical analysts implementing the model processes, to the business people who manage and monitor the processes.
- The Visible Analyst Custom Diagramming capability empowers users to create their own diagram types. Using custom icons or free symbols downloaded from the Web, users define a new diagram Template, specify the symbols to be used on the template, and determine the repository characteristics of the symbols. Each symbol can be created with the same properties as a user-defined object, that is, with the capability to define composition items and establish links to one or more repository entry types.

These tool sets work from a common repository. The integrated repository is accessible from any modeling tool set, and items within it may be used by all tool sets where applicable. The analysis function does some balancing of a data model against a process model according to criteria that you configure. It is also possible to create a process view of your data model that shows all entities affected by a certain process. A process model and a data model can be developed in either order. Classes can be used on entity relationship diagrams and entities can be used on class diagrams. The reporting function generates reports that cross the boundaries between the tool sets.

Visible Analyst is available in the following editions:

- Visible Analyst Professional Edition is the most robust of all the Visible Analyst editions, containing all of the Visible Analyst features. It is suitable for large projects with multiple groups of users, with advanced database engineering capabilities, multi-user access controls, and transition to object-oriented methods. Support for the Zachman Framework cell artifacts and the ability to link these artifacts to the diagrams and/or repository entries make this the flagship edition of Visible's line of modeling tools.
- The Visible Analyst University Edition supports all of the models types supported by the Professional Edition in a concurrent multi-user access capability stored in a relational repository. Individual user and group access can be specified for project collaboration. The Assignments feature enables instructors to easily setup courses and assignments for students. Instructors create their own list of courses and assignments and then add students to those assignments. Student submissions to assignments are easily viewed and assigned a score. Forward Engineering is included for those courses related to database design and implementation.
- The single user Student Edition is a sub-set of the University and supports all of the model types in a relational repository. This edition is limited to 2 active projects at one time and users are limited to 10 diagrams per diagram type per project. Custom diagramming is not included.
- The Free Editions of the Visible Analyst support a specific limited subset of the Visible Analyst Professional Edition's modeling capabilities enabled for 90 days from the date of installation. The free editions are only available as single user versions.
    - The “Concept and Idea Modeler” edition supports the custom diagramming feature. Users can create any diagram type using custom symbols. Neither diagram analysis nor any of the other methodology diagrams are included in this edition.
    - The “Business Process and Activities Modeler” edition supports the BPMN diagramming capability. Diagram analysis is included in this edition.
    - The “Data Entity and Entity Relationships Modeler” support the creation of entity relationship diagrams, SQL DDL Generation, Key Analysis and Key Synchronization as well as diagram analysis.
    - The “UML Use Case, Class and Object Modeler” supports all of the UML diagrams supported in the Professional Edition. Diagram analysis, model linkage and repository support is included.

The Professional Edition of the Visible Analyst is available in two versions:

- A single-user version, licensed for use on a single workstation.
- A LAN version with true multi-user capabilities. The LAN version supports simultaneous access to shared projects or groups of projects. It can be licensed for as many nodes as you require.

The University Edition is only available as a concurrent multi-user version.

Zachman Framework

Today the Zachman Framework has become a standard for Enterprise Architecture used by many of the most successful organizations in the world. Evidence of the acceptance of the Framework has been apparent at the annual forums conducted by the Zachman Institute for Framework Advancement (ZIFA, www.zifa.com). At each forum, attendees hear presentations on the many different aspects and practical uses of the Framework. *Visible* fully supports both the concept and philosophy of the Zachman Framework. *Visible* helps clients gain greater control of their information systems and technology requirements through development of an enterprise-wide architecture.

Visible takes an engineering approach to developing an enterprise architecture. We use a combination of forward and reverse engineering to establish the enterprise architecture. Forward engineering tasks include business planning and data and process modeling. Reverse engineering tasks include analysis and documentation of all existing structures for the organization. The result is a model that represents an integrated view of the enterprise architecture framework, with redundancies and discrepancies resolved and documented. All conceptual and logical architecture components can all be maintained in Visible Analyst.

Visible Analyst supports the tasks and techniques involved in the creation and management of an enterprise architecture, with sufficient flexibility to integrate and support other approaches to software engineering. Visible Analyst captures business plans of multiple organization levels and maintains the hierarchy of planning components (mission, goals, strategies, measures, business rules, etc.).

Strategic Planning

Planning and requirements identification is often the initial phase in an enterprise-engineering project. During the planning phase, you develop a comprehensive strategic business plan that meets the identified mission and purpose of the organization. Visible Analyst not only allows you to create these statements, but also to link them to other objects in your repository. This allows you to track the software development process from the planning stages through analysis, design, and implementation. Linking planning statements to model objects helps you determine the significance of each object and ensures that each object is essential in supporting the organization’s business plan. Strategic planning is available in the Corporate Edition of Visible Analyst.

Visible Analyst Data Modeling Tool Set

The data modeling tool set gives you the ability to create several varieties of the entity relationship diagrams originally developed by Peter Chen. All of the capabilities of the diagramming tool are incorporated into this tool set. Rules for this type of data modeling are intelligently applied to your diagrams. You can use four different notations to display relationship cardinality: IDEF1X notation, crowsfoot notation, double arrow notation, and Bachman notation. Visible Analyst’s flexible symbol template and custom symbol implementation give you control over the other data modeling symbols you use. The analysis capabilities of the data modeling tool set help you by detecting syntactical and some normalization errors, as well as by doing key analysis and migration. An Entity Life History (ELH) diagram, a component of the SSADM and Métrica methodologies, can be exploded to from an instance of an entity on an entity relationship diagram. The ELH shows how events in a system affect data entities. The data modeler automatically draws views for you based on information from the repository that you choose to include in those views. You even have the ability to create views too large to edit in Visible Analyst; the repository maintains them, and you can print them. Because large views can become cluttered and hard to understand, Visible Analyst allows you to limit detail and still show an overall picture by creating cluster diagrams. On these, a cluster of entities is represented by a single symbol, while still showing relationships between entities in different clusters.

Visible Analyst Process Modeling Tool Set

The process modeling tool set enables the rules of popular structured analysis and design methodologies to be intelligently applied to your diagrams. Yourdon/DeMarco, Gane & Sarson, SSADM, and Métrica methodologies are supported for structured analysis diagrams (data flow diagrams). The Yourdon/Constantine structured design rules are supported for structured design diagrams (structure charts). All of the capabilities of the diagramming tool are incorporated in this tool set.

Rules are provided for creating and validating diagrams, including automatic process numbering, level-to-level data flow balancing, decomposition of data by splitting data flows, complexity ratings for structure charts, consistency and completeness checks, and much more. Functional decomposition diagrams allow you to do high-level project planning. From your FDDs, you can have Visible Analyst generate a set of data flow diagrams, from which you can proceed with a detailed analysis. Diagrams in a project are automatically organized in a tree structure. For DFDs, the tree structure is hierarchical, and this lends itself nicely to the concept and process of top-down decomposition. A process decomposition diagram can be generated for all or part of a process model. This unstructured diagram illustrates the top-down decomposition of a process and all of its descendants.

Visible Analyst Object Modeling Tool Set

The object modeling tool set gives you the ability to create class, use case, collaboration, component, sequence, deployment, state, and activity diagrams using a variety of notation styles. All of the capabilities of the diagramming tool are incorporated into this tool set. The analysis capabilities of the object modeling tool set help you by detecting syntax and user errors. Information stored in the repository about classes includes attributes, methods, and friends. The object modeler automatically draws views for you based on information from the repository that you choose to include in those views. You even have the ability to create views too large to edit in Visible Analyst; the repository maintains them, and you can print them. The object modeler is tightly integrated with the data modeler so that classes can be used on entity relationship diagrams, and entities can be used on class diagrams.

Visible Analyst Business Process Modeling (BPMN) Tool Set

The primary goal of BPMN is to provide a modeling notation is an effective communication medium across all the constituencies in computing technology-supported organizations. BPMN is specifically designed to communicate process behavior information in a manner easily understood by business end users while providing supporting technology organizations with sufficient information about process execution, flow and dependencies to understand the workings of the business processes being modeled. BPMN notation is designed therefore to support the needs of not only business end users but also the business analysts who develop models and technical analysts who implement the model processes.

BPMN models describe business process behavior and as a result use an event based paradigm. Both parallel and conditional behavior is supported in the modeling notation and also in the Visible Analyst's implementation of BPMN. A number of symbols are used to describe process flows, events and decisions and allow the viewer to easily differentiate between sections of the BPMN diagram.

Visible Analyst provides support for Business Process Modeling Notation (BPMN) diagrams based on the Business Process Modeling Initiative developed by the Object Management Group (OMG). The complete specification is available for download from the OMG website, www.omg.org.

The Visible Analyst Custom Diagramming Tool Set

The custom diagramming capability allows users to create their own diagram templates, specify symbols for each diagram type, and define the repository entries for the diagram symbols. Users can create Network Diagrams, Security Diagrams, Flowcharts, or any diagram type that meets their modeling needs.

Local Area Networks

The LAN version is compatible with Novell's Advanced NetWare™ versions 1.0 and above. Adding this capability to your Visible Analyst allows an unlimited number of users to have access to Visible Analyst. For more detailed information on the LAN version of Visible Analyst, refer to the manual chapter Local Area Networks.

There are two network versions of Visible Analyst available. The Visible Analyst Novell LAN version uses the file and record locking mechanisms within Novell NetWare. The Generic DOS LAN version supports networks other than Novell that perform standard DOS file locking.

Chapter 1

Getting Started

STARTING OUT

Getting started with Visible Analyst is easy. All you need to do is verify your PC‘s compatibility, install Visible Analyst onto your hard disk, and configure the hardware configuration menu. All of this should take less than 10 minutes.

SYSTEM REQUIREMENTS

Hardware Configuration

Before you install Visible Analyst, verify that your PC components meet the hardware configuration and random access memory (RAM) storage requirements.

To run Visible Analyst, you need a hardware configuration that supports Microsoft Windows XP, Vista, Windows 7, 8, Windows 2000, 3003, or 2008 server. This can cover a range, from the minimum processor and memory size absolutely necessary to the “fastest with the mostest.” Visible Analyst runs anywhere in this range; but as with many other Windows applications, the software runs faster and with less swapping to disk at the higher end of the range. Graphics put a substantial load on the processor, so a faster processor improves performance. Another factor that affects performance under Windows is the number of applications you have running and the amount of system resources they consume. One gigabyte of RAM, or better still two gigabytes or more will also improve performance. The Visible Analyst files consume approximately twenty eight megabytes of hard disk space. This does not include your project files.

Note

- If you are using one of the supported RDBMS engines (Centura SQLBase®, SYBASE SQL Server®, or Oracle Server®) as your Visible Analyst database manager, your hardware requirements may be different. Refer to the next section for details.

The Visible Analyst Database Manager

Project information is stored in a relational database format using one of several database engines, including Btrieve®, SQLBase, Microsoft or SYBASE SQLServer or OracleServer. Each project you create can be stored in a different database format.

Btrieve

Visible Analyst has traditionally used the Btrieve record manager as its underlying database engine. For the Windows version, this is implemented as a dynamic link library (DLL) named VAWBTR32.DLL. By default, all database processing is performed on the workstation where Visible Analyst is running.

If you have the LAN version of Visible Analyst, you have the option of using the client-server version of Btrieve, Pervasive SQL, so that most of the database processing is handled by the file server. To implement the client-server version of Pervasive SQL, do the following:

Note

- It is not necessary to use the client-server version of Btrieve to get all of the multi-user capabilities of the LAN version of Visible Analyst. However, in some circumstances it improves network performance.

- Install Pervasive SQL on the file server. This is a separate product that can be purchased directly from Pervasive Software (www.pervasive.com). You will need to purchase the same number of Pervasive SQL licenses as the number of Visible Analyst Nodes. You will have to reboot the server after installation.
- Install the client Pervasive software on each PC that will access the Visible Analyst.
- Edit the PATH of the computer to include the local Pervasive SQL 2000 BIN directory in the PATH of the PC: <local drive>\Program Files\Pervasive Software\PSQL\bin
    - Rename the following files in the Visible Analyst install folder as shown here:W32MKDE.EXE to W32MKDE_orig.EXE
    - W32NKDE.REG to W32NKDE_old.REG
    - W32MKRD.DLL to W32MKRD_orig.DLL
    - VAWBTR32.DLL to VAWBTR32_orig.DLL
- Copy the file WBTRV32.DLL from the Pervasive SQL folder into the Visible Analyst install folder and rename the file to VAWBTR32.DLL
- While logged into the client PC as an admin level user, start the Visible Analyst so that the Pervasive Registry key is created on the local PC. The users logging into the PC using non-admin level logon ID's will have to be assigned full access rights to this setting.
- To improve client performance make the following changes on the server:
    - Open the Pervasive Control Center and Documentation program and click the "Configure Microkernel Router" option.
    - Select the Performance Tuning listing and check the option to use the Cache Engine. You must click the Apply button to save the change.

  - o Select the Cache Engine listing and check the option to Allocate Resources at Startup. Again, click Apply, then OK to save these changes.

Note

🗒 You can install the client Pervasive software on more client PC’s than the number of Visible Analyst nodes purchased to connect to the Visible Analyst. The Visible Analyst tracks the number of connected clients and prevents connections that exceed the purchase number of nodes.

If you installed Visible Analyst specifying a relational database engine for your repository format and would like to create a Btrieve project, you can do this by selecting Btrieve at the Repository dialog box when creating a new project. You can also convert a relational database project to Btrieve and vice versa. To do this, select Rebuild from the Tools menu, and check the database format you want the project to be rebuilt in. If you want to create or convert a project to a relational format, you are asked to enter site-specific database information in the Project Database Location Information dialog box.

An additional procedure to convert a Btrieve project to a relational database project, or to convert a relational database project to a Btrieve project, is to create a Tools | Backup of the project, and choose Tools | Restore. On the Restore dialog, choose the appropriate database format.

SQLServer

SQLServer can be used as the Visible Analyst database engine. Versions 4.x, 6.x, 7.x, 2000, 2005 and System 10/11 are supported. The advantage of creating a project using a relational format is the ability to use third-party report writers to access information in your data, process, object, and structure models.

When using Visible Analyst SQLServer, be aware of the following:

- The Visible Analyst SQLServer driver cannot create a database; you must create the database that will be used by Visible Analyst beforehand.
- There is no automatic clearing of SQLServer log files. This must be done manually. Logs should be purged frequently, especially when rebuilding a project.
- When rebuilding or creating new projects, SQL DDL commands are issued that are not included in the SQLServer log files. A restore of the log files may not be complete.
- SQLServer requires space for each database created. The log files grow quickly. For large projects with many concurrent users, approximate the space needed by adding the size of the database and index and multiplying by 10. As an example, the LIB project shipped with the product requires 8 MB of space to run properly. You can use the “sp_space” command in ISQL to determine the space values.

- If you are running Visible Analyst under Windows NT, the 16-bit version of the drivers must be accessible.

Oracle Server

Oracle Server can be used as the Visible Analyst database engine. To use Oracle 7.x, 8.x, 9x or 10x Server as the database engine, you must have Oracle Server installed and the proper DLLs included in your path during the Visible Analyst installation. If you performed a Typical installation, these files were installed. If you performed a Custom installation and did not elect to install these files, perform a Custom installation again and choose these files to install. When you choose to create or convert a project to Oracle format, you are asked to enter site-specific database information in the Project Database Location Information dialog box (described below).

When using Oracle Server, be aware of the following:

- Visible Analyst Oracle Server driver cannot create a database, you must create the database that will be used by Visible Analyst prior to creating a project.
- Visible Analyst uses the SQL*Net driver (ociw32.dll). This driver must be in your path.
- The LIB project shipped with Visible Analyst requires 2 MB of space in the database to run if converted to an Oracle server format.

ODBC

ODBC (Open Database Connectivity Driver) can be used as the Visible Analyst database engine. ODBC supports:

- Informix. This driver should be used to access Informix databases.
- DB2 5, 6, 7, 8. This driver should be used to access DB2 5, 6, 7, 8 databases.
- Access. This driver should be used to communicate with Access databases.
- ODBC Compliant. This driver should be used to create projects if the ODBC driver supports ON DELETE CASCADE.
- ODBC Entry Compliant. This driver should be used to create projects if the ODBC driver does *not* support ON DELETE CASCADE.

To use ODBC, you must have the ODBC driver manager installed, as well as the specific driver you are using to access a database. If you add this software after the initial Visible Analyst installation, run the installation again and the proper Visible Analyst DLLs will be installed.

Connecting to an RDBMS Engine

If you choose to create or convert a project to a relational format, you are asked to enter site-specific database information in the Project Database Location Information dialog box. The dialog box prompts you for the following:

- **Connect String** defines where the DBMS is located. For local versions of the database server this field should be left blank. For databases accessed across a network, use the following syntax: for Oracle Server "protocol letter:servername" or the database alias, for SQLBase "servername", and for SQLServer "servername" as defined in the local configuration. All users must use the same connect string to look at a database.

- **Database** is where the project files are located. If you are using SQL*Net SPX for Netware or SQL*Net TCP/IP this entry should be left blank as these versions of Oracle Server do not support multiple active databases. In the case of SQL*Net TCP/IP, the sid or instance name should be specified in the connect string as follows "protocol letter:servername:instance." (If you enter any characters in the database field and then decide you should leave it blank, make sure you delete the field using the DELETE key or blanks are left in the field and your connection fails.)
- **Schema** or authorization ID is used to create and access the project tables for Visible Analyst in a specific space within the specified database. You must have access rights to the space you specify. If you leave this blank, the default schema assigned to your database username is used, and all other users must log on as that schema name.
- **Prefix** is unique to Visible Analyst and is a prefix that must be attached to any tables created within a project if your site stores multiple projects within the same schema. The maximum length is six characters and the default is VA<ROOT>, where <ROOT> is the project name.
- **Username/Password** is your user ID and password to access the database. You must have the appropriate rights. The minimum rights needed to access Visible Analyst projects are insert, update, and delete. The owner of a project also needs create rights to the database.

VISIBLE ANALYST INSTALLATION

Installation can be accomplished in approximately 10 minutes by carefully following the procedures that follow. If you have any questions or problems, please call the Visible Systems technical support staff at:

✂ **781-778-0200**

In addition, you can visit our web site www.visible.com for a wealth of information regarding product support, free "How To…" videos and consulting services.

Another resource that you have available is the Visible Community Forum, located at www.visibleforum.com. The Visible Community Forum Discussion Groups are an open forum for users of our products to exchange ideas, discuss modeling issues, and get solutions to problems. This is a great resource for data analysts, data administrators, IRM analysts, quality assurance analysts, and others dedicated to providing high quality information systems. Topics that are included in our discussions include:

- Installation
- Upgrades
- Model Management
- Forward and Reverse Engineering
- Enterprise (Scaling) Issues
- Wish List (Enhancements)

Prior to Installing Visible Analyst

If you are installing an updated version of Visible Analyst, the installation procedure copies the new version directly over the old version. You should not delete the old version; the installation program does this for you, if you want it to. Your project data files and any symbol set modifications you made are not affected by this installation procedure.

Near the beginning of the installation procedure, you are prompted for the install-to path. You may simply press ENTER to install to \VA; while we recommend this path, you may specify a different path. If you specify a different path, you need not include VA in the specification; if you do not, it will be assumed. For example, typing \CASE is equivalent to typing \CASE\VA.

The default path for a location installation of the Visible Analyst is the C:\Program Files\Visible\VA folder on 32-bit Windows 7, 8 while the default path on Windows 7,8 64-bit is C:\Program Files (x86)\Visible\VA. In both cases a separate default System Data Path of C:\ProgramData\Visible\va\ is created. The system data files are written to this path, while the projects are written to a separate Projects sub folder.

It is always good practice to maintain up-to-date backups of your files. If you have an existing version of Visible Analyst on your computer, we recommend that you back up your projects prior to installing a new version. You should do this using the Backup function of Visible Analyst, rather than a DOS-based backup, because it is easier to restore projects to Visible Analyst using the internal procedure should you ever need to.

If you are installing to a LAN:

- Visible Analyst must be installed onto a file server from the server console.
- You must be able to refer to the install-to drive with a letter designation (for example, F:).
- If you are installing to a Novell server, the procedure assumes that you are familiar with NetWare. Specifically, you should be fluent with its security system and know how to set search paths.
- It is recommended that the system supervisor perform the installation; however, as long as the above points are satisfied, any user can do it.

.

Installation Procedure

To install Visible Analyst:

1. Place the Visible Analyst installation media into the appropriate drive. The auto-run process will begin the installation process. If you downloaded the Visible Analyst installation file, double click on the file to begin the installation procedure
2. If the auto-run process does not begin, access the VA folder and double click the setup.exe file
3. Click Next at the Welcome screen
4. Accept the License Agreement.
5. In the dialog box, type your name and company information. If Visible Analyst has previously been installed, you cannot change this information.
6. Type the serial number of your copy of Visible Analyst. Make sure that the letters are entered in upper case and that you include the dashes.
7. Select the drive and folder where you want the Visible Analyst program files to be placed, or accept the default folder when installing the single user version. NOTE: Do not install to the root of the drive, but to the VA or another named folder.
8. Next, type the drive and folder where you want the Visible Analyst data files to be placed. By selecting separate folders for programs and data, you can assign different access rights. Refer to Local Area Networks for more information. *This option is not available in the University, Student, and Free edition versions.*
9. Select the type of installation: typical, custom, or minimum. We recommend typical.
10. Click Next or press ENTER. The installation program copies and expands the necessary files to your hard disk. When the installation is complete, a message to that effect is displayed on your screen. You are asked if you wish to review updates.

Completion of Single User Installation

When the installation procedure is completed, Visible Analyst resides in the selected path of your install-to drive. For information on starting Visible Analyst, refer to Basic Operating Principles in the section that follows.

Other installation Information

In order to perform the installation on the Windows 2000, Windows XP, Vista and Windows 7, 8 operating systems, the user performing the installation must be logged into the PC as an Administrator level user.

Uninstalling Visible Analyst

If you should ever need to uninstall Visible Analyst from your hard disk, perhaps to move it to another machine, you need only follow these steps:

- Open the Add/Remove Programs in Control Panel, select Visible Analyst and click the Remove button.
- Run the installation again to display the Modify, Repair or Remove dialog. Choose the Remove option, click Next, and click the Remove button.

Additional Installation Considerations for LAN

There are additional steps that must to be performed when installing the LAN version of Visible Analyst. Refer to Local Area Networks, for specific instructions.

Security in Single-User Visible Analyst

Visible Analyst is shipped with a default supervisor name of SUPERVISOR. To log on as a supervisor for the first time, use this user ID. No password is necessary. Thereafter, you may change this user ID as you wish and add any password you like. Make sure you keep a record of your passwords and user ID, as they are encrypted within Visible Analyst and can’t be looked up if you forget them. Security is off by default.

The VAW.INI File

There is a file in your \VA directory named VAW.INI. It holds the default values for numerous Visible Analyst settings. If you are using the LAN version of Visible Analyst, you also have a file named VAW#.INI, where # is user number of your node on the network. Visible Analyst copies the contents of VAW.INI into VAW#.INI when it creates it and thereafter doesn’t use the original any more. The single-user version always uses the original version of the file. The only reasons for you to edit this file are:

- To establish default paths for network users.
- To define a minimum line length.
- To add user-defined attributes to the repository.
- To allow a number to be the first character of a line or symbol.

A FEW WORDS ABOUT ERROR MESSAGES

There are many messages displayed as the result of an activity you do that doesn’t conform to the rules of a methodology, as a result of some other occurrence (missing or corrupted files, etc.), or even because of some error made by Visible Analyst. You, as the user, need to know what happened, why it happened, and how to fix it. Where you turn for advice depends on the kind of activity performed when the error occurred. Below is a list of the types of activities that cause errors and where you can turn for an explanation:

- If you are running one of the project analysis functions (Analyze, Key Analysis, Key Synchronization**,** or Model Balancing), errors supplied at the end of the process relate to the sometimes complex rules of the methodologies that underlay the operation of Visible Analyst. An explanation of these errors can be found in The Visible Rules chapter. In this chapter, error messages are organized by the diagram type of the diagrams being checked.

- Messages produced by import, export and shell code or DDS generation are generally simple, direct and self-explanatory. However, if you find a message that baffles you, please call Visible Systems technical support number for an explanation.
- Other messages, such as those that are displayed when you attempt some illegal drawing operation, are listed alphabetically, with explanations, in Appendix A.
- All error messages are described in the online help system.

BASIC OPERATING PRINCIPLES

Before you begin using any of the capabilities supplied with Visible Analyst, take about 20 minutes to become familiar with the following basic operating principles that apply to all of the Visible Analyst tool sets. By understanding these basics, you'll get off to a good start and become more effective with Visible Analyst in a shorter period of time.

Accessing Visible Analyst

You access Visible Analyst in the normal Windows fashion, by double-clicking on the Visible Analyst icon.

Using the Mouse

Although Windows does not absolutely require it, we recommend using a mouse with Visible Analyst, which is used to perform a variety of functions in Visible Analyst. The standard Windows functionality for the mouse is implemented here. The specific ways the mouse is used to perform various tasks are discussed throughout this manual as those tasks are introduced.

In general, the left mouse button is used to select objects and confirm actions, while the right mouse button displays an object menu listing actions that are valid for the current object.

Notes

- Unless stated to the contrary, instructions to click a mouse button refer to the left button. Instructions for the right button are explicitly mentioned.
- Left-handed mouse users: if you use a mouse with the buttons reversed, you should reverse references to left and right mouse buttons in this text.
- For detailed instructions how to use the keyboard entries and keyboard shortcuts to draw diagrams and edit the repository without using a mouse, contact our support department at support@visible.com or call 781-778-0200.

Finding Help Whenever It's Needed

Visible Analyst includes an extensive help system that displays information and instructions about the use of the program. The help screens are accessible by pressing the F1 key on your keyboard from anywhere in Visible Analyst or by using the Help menu. Clicking the "?"

symbol on a dialog and clicking on a field will display the Help topic associated with the field.

Context-Sensitive Help

The Visible Analyst Help system has context-sensitive help for all menus and dialog boxes. You can access it in either of the standard Windows ways.

- First, you can highlight a menu item and press F1 to display the Help topic for the menu item you specified. Once a menu has opened, you cannot select a menu item for the purpose of getting Help by simply clicking on it with the mouse. If you try to do this, the function is executed before you have the opportunity to request Help for it. You should use either the keyboard arrow keys to highlight the item and press F1 or use the mouse to drag the highlight to the desired item and press F1 before you release the mouse button.
- Second, you can press SHIFT+F1 and Visible Analyst changes into Help mode and the cursor changes as shown in Figure 1-1. Use the mouse to select any menu item. When you do, the cursor changes back to the standard cursor, indicating that Visible Analyst is no longer in Help mode, and the help topic for the menu item you selected displays.

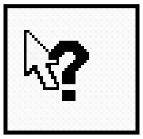


Figure 1-1 Help Cursor

Keyboard Shortcuts

Many menu functions can be accessed with keyboard shortcuts in the standard Windows manner. The shortcut key is shown on the menu next to the command. To use the Edit menu functions Cut, Copy and Paste from edit boxes, you must use the keyboard shortcuts. The following table lists all keyboard shortcuts used in Visible Analyst.

Table 1-1 Visible Analyst Keyboard Shortcuts	
CTRL+A	Analyze
CTRL+C	Copy to Clipboard
CTRL+D	Define Repository Object
CTRL+E	Connect Selected Symbols
CTRL+F	Find Diagram Object
CTRL+L	Line, Add to Diagram
CTRL+N	New Diagram

Table 1-1 Visible Analyst Keyboard Shortcuts	
CTRL+O	Open Diagram
CTRL+P	Print Diagram(s)
CTRL+Q	Query Report
CTRL+R	Report
CTRL+S	Save
CTRL+T	Text, Add to Diagram
CTRL+U	Clear (Deselect) Diagram Object or Block
CTRL+V	Paste from Clipboard
CTRL+X	Cut to Clipboard
CTRL+T	Snap Symbols to Horizontal or Vertical Row
SHIFT+F10	Display Repository Object Menu for Field Options
CTRL+Z	Undo (Erase Partially Drawn Line or Undo Moved Line)
ALT+R	Delete Project with No Project Files
DEL	Delete Object from Diagram
F1	Help, Context Sensitive
SHIFT+F1	Enter Help Mode to Get Help on Menu Item
Insert	Add a line segment to an existing line

Recognizing Illegal Commands

An extensive array of message boxes is displayed by Visible Analyst to alert you to illegal actions and procedures and to notify you of other irregularities that may occur.

Don’t Forget To Save and Back Up

As with any software program, it is good practice to save your work at regular intervals while working on a project and to back up your files each time you make significant changes. While a computer and its programs can be very helpful, it can also be extremely insensitive if a power outage or another unscheduled event should occur.

To avoid the frustration that goes along with lost work, always remember to save and back up your work. The function to Save the diagram on which you are working can be found on the File menu. The Backup function, as well as the Restore function, can be found on the Tools menu. These functions are described in more detail later.

THE VISIBLE ANALYST WORKSPACE

Whenever you access Visible Analyst, the workspace is presented as shown in Figure 1-2. The workspace is the starting point for all functions, and it has five major sections:

- The menu bar at the top gives you access to all of the menus in Visible Analyst containing all of the functions of the tool.

- The control bar below the menu bar gives you quick access to commonly used Visible Analyst functions.
- The diagram area in the center of the screen contains the diagrams you have open, whether maximized and ready to edit or minimized to an icon.
- The help bar at the bottom of the screen displays the current menu item, the name of the current object, the current zoom level, and current project.
- The object browser displays a list of the objects in the repository in a resizable window.

The control bar, the help bar, and the object browser can be toggled off (or on again) from the Options menu if you don’t want them visible.

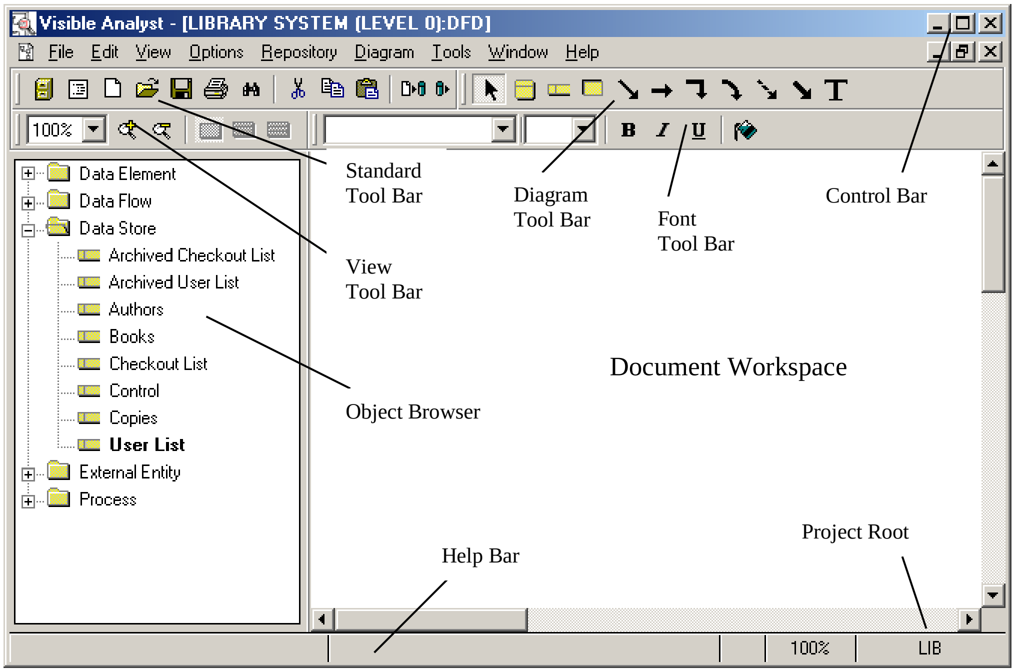


Figure 1-2 Visible Analyst Workspace

Information About Your Copy of Visible Analyst

Information about your copy of Visible Analyst can be accessed using the About Visible Analyst selection from the Help menu. This screen shows you:

- The version of Visible Analyst you are running.
- The serial number of your copy.
- The network under which you are operating (LAN version only).
- The number of network nodes (users) that can concurrently use your copy of Visible Analyst (LAN version only).
- Your user name (the name you used to log in to Visible Analyst if you are using a single user version and Security was turned on from the Options menu, or your network login name for the LAN version).
- Your user number (LAN version only).
- A Purchase button allowing you to upgrade to a different edition of the Visible Analyst or Purchase an edition if you installed a demonstration copy of the software.
- The capabilities of the tool in the configuration that you purchased. Capabilities available but not included in your configuration are shown in gray.

Visible Analyst Menu Functions

The functions available on Visible Analyst menus provide the starting point for all Visible Analyst operations. A list of these functions, with a description for each, is referenced below.

Table 1-2 Menu Functions

Menu	Command	Description
File	New Project	Select New Project to create a new project. A dialog box displays allowing you to specify information about the new project.
	Select Project	Select Select Project to open an existing project. A dialog box displays a list of the project from which to choose.
	Project History	Select Project History to display a box that provides a descriptive overview of the current project. The following descriptive information is included: • Project name or root • The name of the project manager • A short description of the project • The rules methodology that applies to the project • Repository enabled or disabled for the project • Who created the project (if security has been enabled) • DOS path where project files are stored • List of diagrams in the project • List of dates showing when each diagram was last edited
	Current Activity	Select Current Activity to display the users who are currently using Visible Analyst and provide some information about what each is doing (LAN version only).
	Modify User List	In the LAN version of Visible Analyst, select Modify User List to display a submenu that allows you to add or delete user IDs of users who are allowed access to the current project. You can also define users who may change project information or just view it.
	Zachman Framework	Select Zachman Framework to display or close the Zachman Framework interface. Only available in the Zachman Framework edition.
	Strategic	Select Strategic Planning to open the Planning Statement

Table 1-2 Menu Functions

Menu	Command	Description
	Planning	hierarchy window. Use this window to create new planning statements and parent-child relationships between statements.
	New Diagram	Select New Diagram to create a new diagram. To add new diagrams to an existing tree file of data flow diagrams, select a point in the project tree where the new diagram is entered. To add diagrams to an existing tree file of non-data flow diagrams, the New Diagram command automatically places them in alphabetical order among existing diagrams.
	Open Diagram	Select Open Diagram to open an existing diagram. A dialog box displays a list of the diagrams from which you can choose. To open a diagram in Read-Only mode, right-click on a diagram displayed in the list, then select Open Read Only from the menu displayed.
	View of Data Model	Select View of Data Model to create a view of a data model. Select the type of view you want to create: global, new, process, or cluster diagram. For a cluster diagram, you also select the method to create the diagram: global, made from the current view, or a custom cluster diagram.
	Draw VIRTUAL Chart	VIRTUAL Chart generates a structure chart from a virtual diagram in the repository created from data imported from the Hypersoft Application Browser. Refer to Hypersoft's Application Browser for more information or contact Visible systems support at support@visible.com for more information.
	Close	Select Close to close the active document. If the active document is a diagram, you are prompted to save the changes.
	Save	Select Save to save work done on the active diagram.
	Save With New Name	Select Save With New Name to save the current diagram under a new name.
	Erase Changes	Select Erase Changes to undo all editing changes that have been made to the current diagram since it was last saved.

Table 1-2 Menu Functions		
Menu	**Command**	**Description**
	Nest	Select Nest to display a submenu to select one of the following: • Explode any process symbol in a diagram in order to create a lower level diagram that represents the detailed functional aspects of the process being exploded. This only applies when the diagram type is set to data flow. When rules are disabled, symbols other than processes may be decomposed. If the symbol has already been exploded, this allows you to move from the diagram containing the parent symbol to the child diagram. The parent diagram remains open. • Move to the parent diagram of the current diagram. • Detach a child diagram from its parent (break the parent/child link). • Automatically generate a decomposition diagram.
	Spawn	Select Spawn to generate a set of leveled data flow diagrams from a low level function and its tree of processes on a functional decomposition diagram. It appears on the File menu only when an FDD is the current diagram.
	Page	Select Page to create a multiple page diagram. This is generally only useful for structure charts, although you can use it for DFDs as well. It is not available for other diagram types. For data flow diagrams, the Analyze function does not apply the rules across the page connections. For structure chart diagrams, the rules are applied across multi-page charts created with the Page function. A submenu allows you to connect or disconnect two diagrams.
	Print	Select Print to print the current diagram, to select diagrams to add to the print queue, or to print the current project tree. If you select print queue, you can (1) view its contents to see how many project diagrams are currently entered into the queue; (2) add or delete project diagrams from the queue; and (3) print all diagrams that are listed in the queue.

Table 1-2 Menu Functions

Menu	Command	Description
	Print Setup	Select Print Setup to change the printer you configured for Windows. Click the Setup button to change settings for the printer driver.
	Recent Diagrams	Select Recent Diagrams to display a list of the eight most recently opened diagrams.
	Recent Projects	Select Recent Projects to display a list of the most recently opened projects.
	Exit	Select Exit to end the Visible Analyst session and return to Windows.
Edit	Undo	Select Undo to erase a partially drawn line or restore a moved line to its original position.
	Cut	Select Cut to remove the selected object or block and place it in the Windows Clipboard.
	Copy	Select Copy to copy the selected object or block into the Windows Clipboard.
	Paste	Select Paste to copy what is in the Windows Clipboard and place it on the current diagram.
	Clear	Select Clear to deselect the current object or block on the current diagram.
	Select All	Select Select All to select all objects on the current diagram.
	Delete	Select Delete to remove the selected object or block without placing it in the Windows Clipboard. You are asked to confirm your deletion.
	Copy To	Select Copy To to store the current object or block as a Windows metafile or JPEG file.
	Find	Select Find to search for an object on the current diagram. If the object is found, it is selected and brought into focus.
	New Statement	Select New Statement to create a new planning statement. If

Table 1-2 Menu Functions

Menu	Command	Description
		an existing statement is selected, the new statement is a child of it. Otherwise, it is inserted at the top of the hierarchy.
	Promote	Select Promote to move the selected planning statement one level higher than its present position.
	Demote	Select Demote to move the selected planning statement down one level lower than its present position.
	Move Up	Select Move Up to move the selected planning statement up one position.
	Move Down	Select Move Down to move the selected planning statement down one position.
View	Zoom	Select Zoom to reduce the display size of the diagram so that a larger portion may be viewed on the screen. The zoom level is expressed in terms of zoom percentage that is selected from the menu. Diagrams can be edited and printed in zoom mode. The zoom level is also displayed on the help bar.
	Show Level	When the current document is the planning statement hierarchy, select Show Level to select how much of the tree is displayed. Level 1 shows only the top-level nodes; Show All displays all nodes. You can change individual branches by clicking the + to the left of each planning statement.
	Filter	Select Filter to choose a particular branch of the planning hierarchy to display. This is useful for projects that contain a very large number of planning statements.
	Grid	Select Grid to turn the diagram grid on and off. The grid is useful for positioning objects on a diagram. It does not appear on printed diagrams.
	Ruler	Select Ruler to display or inhibit the screen ruler throughout the vertical and horizontal extremes of the diagram. The ruler is useful for positioning objects on a diagram. It does not appear on printed diagrams.

Table 1-2 Menu Functions		
Menu	**Command**	**Description**
	Show Line Names	Select Show Line Names to display or inhibit the names of lines on a diagram.
	Show Symbol Names	Select Show Symbol Names to display or inhibit the names of symbols on a diagram.
	Show Discriminators	Select Show Discriminators to turn on and off the display of discriminators on entity relationship or class diagrams.
	Show Statement Type	Select Show Statement Type to display or hide statement types in the planning window.
	Show Priority	Select Show Priority to display or hide the priority of a statement in the planning window.
	Class	Select Class to choose the type of information displayed on a class diagram.
	Entity Display Options	Entity Display Options allows you to select display options for the entity attributes displayed on the data model diagram. • Select Entity Level to display only entity names within entity symbols on an ERD. • Select Primary Key Level to display entity names and attributes composing the primary key within entity symbols on an ERD. Refer to Entity Attributes Displayed on Data Model Diagrams in The Visible Rules for more information. • Select Attribute Level to display entity names and all attribute names within entity symbols on an ERD. • Select IDEF1X Notation to turn on IDEF1X notation for the current entity relationship diagram. If there are no diagrams open, this acts as a default. • Select Name Inside Box to display the name of an IDEF1X entity inside the entity box. It is disabled if IDEF1X is not the current notation for an entity relationship diagram. • Select Expand Associators to display a foreign key that is represented by an associator element with the names of the elements that comprise the primary key in the form *associator.(column one, column two, ...).*

Table 1-2 Menu Functions

Menu	Command	Description
	Physical Schema	Select Physical Schema to display the physical name and properties of an entity to be used during SQL generation.
	Events	Select Events to select change the information that is displayed for events on an activity diagram.
	Messages	Select Messages to select change the information that is displayed for messages on a sequence or collaboration diagram.
Options	Auto Label Symbols	Auto Label Symbols automatically labels symbols as they are drawn. This function can be toggled between On and Off to enable or disable it. When On, you are prompted to enter a label each time a symbol is drawn on your project diagram.
	Auto Label Lines	The Auto Label Lines function automatically labels lines as they are drawn. This function can be toggled between On and Off to enable or disable it. When On, you are prompted to enter a label each time a line is drawn on a diagram.
	Line Settings	The Line Settings function allows you to specify default settings for the diagram type, line type, terminator type, terminator end type, and line orientation.
	Text Settings	Text Settings allows you to specify a default font and format for each label or caption entered onto a diagram. You can specify that one particular typeface and format be used for all symbols, another for all process/data store numbers, another for all lines and data flows, another for all paragraphs, etc.
	Security	Security can be used to prevent unauthorized access to a single-user version of Visible Analyst. When enabled, a user must enter a valid user ID and optionally a password to gain access to Visible Analyst. User IDs may be added by the supervisor or a system manager via the Users function on the Tools menu. To enable security, be sure you are defined as the supervisor. Select this function in the menu and enter the supervisor user ID and, optionally, a password via your keyboard; then press ENTER.
	Colors	Colors allows you to assign colors to different screen objects,

Table 1-2 Menu Functions		
Menu	**Command**	**Description**
		to enhance the appearance and clarity of your diagrams.
	Auto Connect	Auto Connect allows lines to be drawn between symbols without worrying about attaching them exactly to the border of the symbol. With the Auto Connect selection checked, if you start a line within one symbol and conclude it within another, Visible Analyst draws the line using the currently selected line and terminator types and automatically attaches the line to the border of each symbol. Note that, for FDDs, following this procedure draws a two-segment elbow line connector as is usual for that diagram type. When Auto Connect is checked, it also affects moving lines. If you drag the endpoint of a line to within another symbol, Visible Analyst attaches the line to the border of that symbol.
	Auto Position Text	Auto Position Text allows you to turn Auto Position Text either On or Off. When on, after manually positioning a line label, moving the line results in automatic repositioning of the text based on the text position algorithm built into Visible Analyst. When off, after manually positioning a line label, moving the line results in the text item moving relative to where it was placed. The only time the latter would not occur is if a line is radically moved on the diagram or the line is moved to a location where the text does not fit. In this case, the text is repositioned using the text position algorithm.
	Include Connections	Include Connections causes lines connected to a symbol to remain connected and move when you move a symbol or to be deleted when you delete a symbol.
	Grid Settings	Grid Settings allows you to specify the grid settings for placing objects on a diagram both horizontally and vertically.
	Interaction Diagrams	Select Interaction Diagrams to select change the way objects and messages are created on sequence and collaboration diagrams.
	Classic User Interface	Classic User Interface changes the method by which component items are added to the repository. If checked, a free-form composition field is used. If not checked,

Table 1-2 Menu Functions

Menu	Command	Description
		attributes are added with a name, type, and reference value. This setting only affects some repository entries, such as entities, data flows, data stores, and data structures.
	Control Bar	Control Bar allows you to customize the control bar, the row of buttons above the diagram workspace that gives you quick access to commonly used functions.
	Help Bar	Help Bar turns the display of the help bar on and off. The help bar displays the current menu item, the name of the current object, the current zoom level, and the current project.
	Object Browser	Object Browser turns the display of the object browser on and off. The browser displays a list of all objects in the repository. When no diagrams are open, or the current window is the diagram list, all objects are displayed. When a diagram is open, only objects valid for that diagram are displayed. If an object appears on that diagram, it is displayed in bold.
	ERD Balancing Rules	*All Fundamental Elements Must be Used on a DFD:* If you choose Yes, Visible Analyst tries to balance ERDs against DFDs in the same project; and it lets you know which entity attributes (data elements) have not been used on any DFD. If you choose No, it does not perform this check. *Each Entity Must Correspond to a Data Store:* There is some correlation between the entities appearing on ERDs and data stores appearing on DFDs in a project. You can make this a one-to-one correlation by choosing Yes. Visible Analyst then enforces this correlation in its balancing analysis and in other ways.
	SQL Dialect for Schema Generation	SQL DDL (Data Definition Language) syntax can be generated for all entities in the repository for a project. Dialects include: Access 97 Paradox 7x, 8x

Table 1-2 Menu Functions

Menu	Command	Description
		Access 97, 2000 ANSI 92 CA Datacom 8x CA Open Ingress II Centura SQL Base 5x DB2/2 2x, 5x, 6x, 7x 8x DBASE IV Informix 7x Ingres 6x InterBase 4x, 5x Progress 7x, 8x, Native 7x, 8x MS SQL Server 4x, 6x, 7x, 2000, 2005 Netware 1x Oracle 7x, 8x, 9x, 10x Paradox 7x, 8x Progress 7x, 8x, Native 7x, 8x SYBASE SQL Anywhere 5x SYBASE SQL Server 4x, 10x Teradata SQL V2 2.1.0 Unify 2000 User Defined Vax RDB 6x Watcom 3x xdb 1x XML Each dialect supports a different subset of some theoretically complete SQL. If the SQL dialect you use is not yet supported by Visible Analyst, you can either pick the supported dialect that is closest, or you can choose User-Defined SQL and customize it to meet your needs.
	DDS Name Translation	This selection allows you to tailor how object names are generated during AS400 DDS generation and how they are mapped to Visible Analyst object names during DDS import. There are three mapping schemes available: • Logical to Alias. A Visible Analyst object name is mapped to the ALIAS( ) field. • Logical to Colhdr. A Visible Analyst object name is mapped to the COLHDR( ) field. • Logical to Text. A Visible Analyst object name is mapped to the TEXT( ) field. These mappings only occur if the object name cannot fit in the DDS record/element name field (column positions 19-28). If Logical to TEXT is not selected, the TEXT( ) field contains the Visible Analyst description.
	Code Generation Options	This selection allows you to make choices about how shell code generation is done from your project.

Table 1-2 Menu Functions

Menu	Command	Description
	Define User Attributes	Define User Attributes allows you to add, modify, or delete fields associated with repository entries.
	Define User Objects	Define User Objects allows you to define site-specific objects in the repository. These new objects can then be linked to standard repository objects and reference other standard objects within the composition field of the user-defined object.
	Planning Statement Types	Select Planning Statement Types to define additional site-specific statement types in the repository. Visible Analyst is shipped with a predefined set of types such as Mission, Goal, and System Requirement; but you can add new types and define how they can be used.
	Framework Settings	Select Framework Settings to specify which Model and Object Types are associated with a specific cell. Click on a cell to enable the Cell Settings.
	Symbol Templates	Select Symbol Templates to substitute a different symbol image for an existing methodology symbol. New template image symbol sets can also be created.
	Artifact Stereotypes	Select Artifact Stereotypes to add custom stereotype labels and custom stereotype images that will be displayed within the artifact symbol.
Repository	Define	Select Define to go directly to the data repository for the current project. A repository dialog box is displayed to execute repository functions. If no diagram object is selected, the dialog box is empty. The Visible Analyst repository dialog box allows you to add and change descriptive information about all objects that appear in methodology-based projects.
	Divisions	Select Division to create, modify, or delete divisions and add users to divisions. Divisions are used to restrict access to repository objects and to transfer data information between an enterprise and satellite project.

Table 1-2 Menu Functions		
Menu	**Command**	**Description**
	Key Analysis	The analyze function identifies errors in the primary and foreign key specifications you entered in the repository of your project for the entities in your ERDs.
	Key Synchronization	This function begins the analysis synchronization function to review all of the primary and foreign keys you specified in the repository of your project. It then generates foreign keys in various entries, as necessary, to properly represent relationships between entities.
	Model Balancing	This function balances a data model against a process model according to the ERD Balancing Rules you previously selected on the Options menu.
	Syntax Check	This function analyzes a class of diagrams according to the rules of a methodology.
	Generate Database Schema	This function generates a database statement file schema for your project that can be used to create the tables in your database.
	Compare Model Against Schema	This command compares the current model against an existing database schema and shows the differences between them.
	Generate DDS	This command begins the generation of a DDS description of your project.
	Generate Code	This command begins the process of shell code generation for your project.
	Reports	Select Reports to display the repository reports dialog box for the current project if the repository was enabled when the project was created. The dialog box allows you to set report formats and generate a variety of reports based upon the current project repository and other project information.
	Reports Query	Select Report Query to bring up the custom reports dialog box for the current project if the repository was enabled

Table 1-2 Menu Functions

Menu	Command	Description
		when the project was created. The dialog box allows you to generate custom reports based upon the current project repository and other project information.
Diagram	Symbols	Select Symbols to display the symbol submenu add symbols to a diagram. The cursor changes to the symbol cursor.
	Lines	Select Lines to add lines to a diagram. The cursor changes to the lines cursor.
	Text	Select Text to add or change diagram text. The cursor changes to the text cursor.
	Picture	Select Picture to insert an image file onto the diagram.
	Construct	Select Construct to create, load, or delete constructs. Constructs can be subsections of any diagram.
	Change Item	Select Change Item to change the label and/or process or data store number of a symbol, line or caption, as well as its appearance.
	Stylize	Select Stylize to add enhancements (bold lines, relief, etc.) to a symbol, or to change the size of any symbol.
	Connect	Select Connect to connect a function or process with subsidiary functions or processes, or two diagram objects.
	Snap Symbols	The Snap Symbols function causes subsequent symbols enclosed in a block to be aligned in a strictly horizontal or strictly vertical row.
	Snap Lines	Snap Lines causes lines enclosed in a block to be drawn in a strictly horizontal or strictly vertical plane. Lines drawn between symbols are connected to the center points of the symbols.
	Analyze	Select Analyze to analyze a diagram or an entire project according to the rules of a methodology.
	Split Data Flow	Select Split Data Flow to divide data flows into component

Table 1-2 Menu Functions

Menu	Command	Description
		subflows. This function is used for data flow diagrams only, and only with the rules enabled.
	Regenerate Chart	Select Regenerate Chart to redraw the open structure chart diagram, updating it with any information in the repository that might not already be on the diagram.
	Modify View	Select Modify View to modify an existing view of a data or object model by adding and deleting entities and relationships.
	Settings	Settings allows you to change the size, orientation, and scale of an existing diagram. If you initially selected the easel format, you cannot decrease the page size to standard once your diagram has been created and saved if the content of the diagram would prevent it from fitting on a standard diagram. The same principle applies when changing from a multi-page diagram to one of the smaller sizes.
Tools	Backup	Select Backup to back up a project to a floppy disk or to a hard disk.
	Restore	Select Restore to restore a project from a floppy disk or from a hard disk.
	Copy Project	Select Copy Project to copy a project.
	Delete Project	Select Delete Project to delete a project.
	Rename/Move	Select Rename/Move to rename or move a project to a different subdirectory. You can move it with or without changing its name.
	Export	Select Export to export repository information.
	Import	Select Import to import repository information.
	Rebuild	Select Rebuild to rebuild projects that have become corrupted.
	Enterprise Copy	Select Enterprise Copy to copy a division between an

Table 1-2 Menu Functions		
Menu	**Command**	**Description**
		enterprise project and a satellite project.
	Enterprise Tag Maintenance	Select Enterprise Tag Maintenance to remove the link between an Enterprise project and a Satellite project.
	Copy Diagram	Select Copy Diagram to copy a diagram or a branch (data flow diagram and all of its children/grandchildren). This command allows you to copy similar methodology diagrams and branches from one project to another. All data repository information associated with the diagram(s) is also copied into the new project.
	Delete Diagram	Select Delete Diagram to delete a diagram, or a branch (a data flow diagram and all of its children/grandchildren).
	Users	Select Users to define user security information, including User Type (System Manager, Project Manager, or User). For single user and generic DOS network version of Visible Analyst, you can define user names and passwords to secure Visible Analyst access to only those users that you have defined. Thereafter, anyone logged in as supervisor or system manager can select any user from the list box and view or change that user ID.
	Prototyper	Select Prototyper to start the Visible Prototyper. If you purchased this tool, you can work with the prototyping and simulation capabilities.
Window	Tile	Select Tile to arrange open diagram windows as tiled windows.
	Cascade	Cascade arranges open diagram windows as cascaded windows.
	Arrange Icons	If you have minimized diagrams to icons, this function arranges those icons on your application workspace.
	Open Diagrams	The name of every diagram you have open appears on this menu. You can move among them by selecting them here.

Table 1-2 Menu Functions

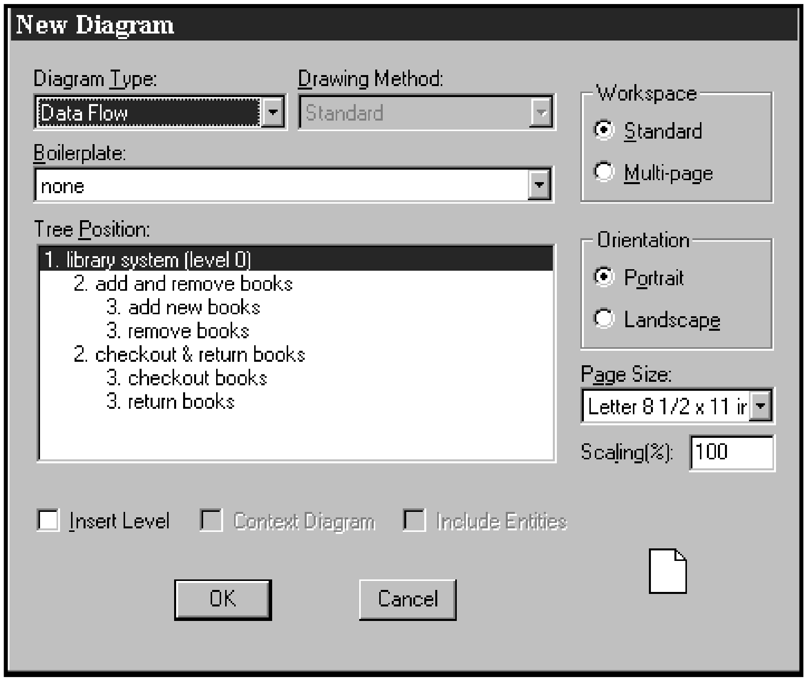
Menu	Command	Description
Help	Contents	Use Contents to display areas of help or to access the Index search function.
	Glossary	Glossary contains definitions of key terms used in Visible Analyst.
	Menu Functions	Menu Functions allows you to find information about the menus in Visible Analyst.
	Tutorial	Selecting Tutorial opens the Visible Analyst tutorial file in Adobe Acrobat Reader.
	Analysis Errors	Analysis Errors contains an explanation of analysis error messages.
	Miscellaneous Errors	Miscellaneous Errors contains an explanation of miscellaneous error messages.
	Check for Updates	Check for Updates communicates with the Visible System's servers and updates the Visible Analyst with the latest versions of the Visible Analyst files.
	Visible on the Web	Visible on the Web takes you to the Visible Systems Corporation web site.
	Visible Community Forum	Visible Community Forum takes you to the Visible Forum site on the Web.
	About Visible Analyst	About Visible Analyst contains information about your version of Visible Analyst. The Purchase button on the About dialog allows you to renew or upgrade your copy of the Visible Analyst.

UNDERSTANDING PROJECT TREES

Visible Analyst organizes diagrams on a project basis. Before working in Visible Analyst, you must select a project name where the work is to be kept. Every subsequent function performed by Visible Analyst addresses the selected project, with the exception of New Project, Select

Project, Print, Recent Projects, Restore, and Copy Diagram. The selected project remains the current project until a different project is selected.

Within each project, diagrams are automatically stored in a diagram list referred to as the project tree or tree file (see Figure 1-3). There are separate diagram lists within a project, one for the project's data flow diagrams and others for activity, business process model, class, collaboration, component, data flow, deployment, entity relationship, entity life history, functional decomposition, sequence, state transition, structure chart, use case, custom and unstructured diagrams. The diagram list for data flow diagrams is arranged in a hierarchical structure, and makes use of indentation to depict relationships between diagrams on different levels, child/parent relationships, etc. This type of diagram list organization complements the concept and process of top-down functional decomposition.



New Diagram

Diagram Type: Data Flow

Drawing Method: Standard

Boilerplate: none

Tree Position:

```
1. library system (level 0)
    2. add and remove books
         3. add new books
         3. remove books
    2. checkout & return books
         3. checkout books
         3. return books
```

Workspace: ● Standard ○ Multi-page

Orientation: ● Portrait ○ Landscape

Page Size: Letter 8 1/2 x 11 ir

Scaling(%): 100

□ Insert Level □ Context Diagram □ Include Entities

Figure 1-3 A Typical DFD Project Tree

The activity, business process model, class, collaboration, component, deployment, functional decomposition diagram, entity relationship, entity life history, sequence, state transition, structure chart, use case, custom and unstructured diagram lists do not represent diagrams with a hierarchical structure. They are not based on the concept of top-down decomposition, and the diagram list has a flat structure that lists all of the project diagrams in alphabetical order.

Each project is associated with a project root designator for the project. You can change the currently selected project selection using Select Project or Recent Projects on the File menu.

Every process-modeling project must begin with a top-level data flow diagram, from which subsequent levels of data flow diagrams can be decomposed. The remaining levels of data flow diagrams in the project are structured below the top-level diagram.

- You can store an unlimited number of projects in Visible Analyst.
- Each project is designated by a project root and has its own diagram list for data flow and other types of diagrams.
- Each data flow diagram in a project can have only one parent, but may have up to 40 children.

Note

- This information applies to both data flow diagrams and to other diagram types. All references to parent and child diagrams, and to the copying, adding, deleting, etc., of branches and child diagrams refer only to data flow diagrams.

- The Visible Analyst allows users to create nested relationships between certain methodology symbols and other methodology diagrams to create a parent child relationship. The table below lists the diagrams that can be generated for these methodology symbols.

Methodology Symbol	**Child diagram**
Activity	Activity Diagram
Business Process Model Activity	Business Process Model diagram
Class	Activity or State Transition diagram
Component	Component diagram

Data Flow	Data Flow diagram
Deployment	Deployment diagram
Entity	Entity Life History diagram
System Boundary	Use Case diagram
Use Case	Activity, Collaboration, Sequence diagram

To remove the Nest relationship between diagrams, open the nested diagram, right mouse click on the diagram background and choose “Detach” from the Object menu. Save the change to complete the detachment. The diagram can now be nested to another symbol when exploded. The Detach option is also available from the Nest option on the File menu.

The next topic “Modifying a Data flow Diagram Project Hierarchy”, below has detailed instructions how to rearrange stored diagrams and copy diagrams from one project to another.

Modifying a Data Flow Diagram Project Hierarchy

Visible Analyst gives you the capability to restructure a data flow diagram project hierarchy. You can rearrange project data flow diagrams, as well as add or delete them at any level. You may also copy a diagram or branch from one project to another, inserting the diagram at any level in the receiving structure. All repository information associated with diagrams remains attached to the diagrams when they are moved with the Copy Diagram function. However, whenever you rearrange or add a diagram or branch to the diagram hierarchy, you may have to recreate nest relationships to maintain the integrity of any parent/child relationships that are impacted by the modification. This is explained in the following pages.

Inserting New Diagrams into a DFD Project Hierarchy

The New Diagram function at the File menu allows you to create and insert new diagrams into the project hierarchy of an existing project. The dialog box asks you to pick an existing diagram in the project hierarchy.

- If you want the new diagram to become the hierarchical parent of the diagram you select, check Insert Level. When the inserted diagram is saved, it bumps all subsidiary levels of the hierarchy down one level.
- Otherwise, the new diagram becomes the hierarchical child of the diagram you choose.

Any nest relationship that existed between the diagram that had occupied the insert position and its parent are removed. However, the new diagram being inserted is nested from its new parent diagram. There is no nest relationship between the inserted diagram and the diagram that had occupied its new position (the diagram that was bumped down one level); however, you can establish a nest relationship between them using the Nest function described in the Drawing Diagrams chapter.

If you did not make a context diagram when you created the first DFD of a Yourdon-rules project, you can check Insert Level and Context Diagram and select the top-level diagram in the hierarchy. Your new diagram then becomes the project context diagram.

Figure 1-4 illustrates an example of the before and after results of inserting a new diagram into a project hierarchy. In the example, diagram 8 is inserted at the position that diagram 2 had occupied. Notice that the branch of the tree containing diagrams 2, 3, and 4 is therefore bumped down one level. The nest relationship between the old parent and child is broken, and a nest relationship between the old parent and new child must be established manually so proper data flow balancing can be done. All other nest relationships remain undisturbed.

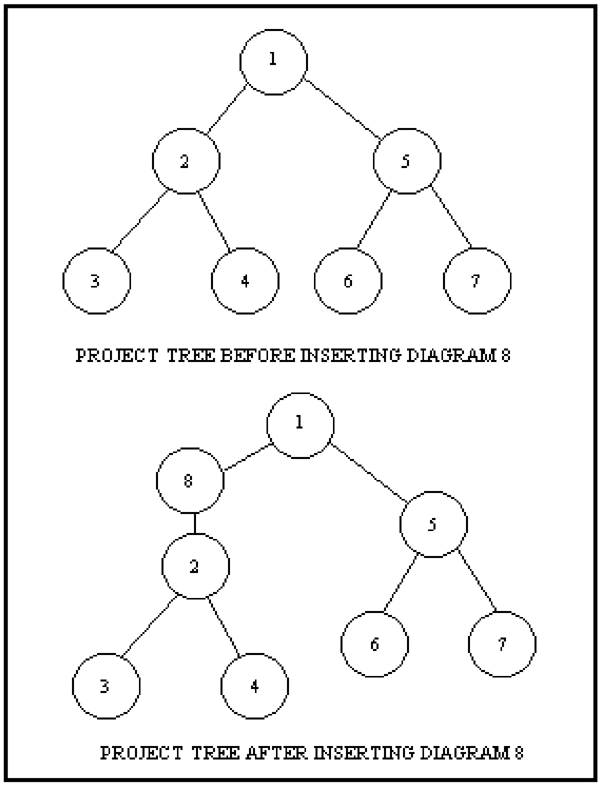

1
2
5
3
4
6
7
PROJECT TREE BEFORE INSERTING DIAGRAM 8
1
8
5
2
6
7
3
4
PROJECT TREE AFTER INSERTING DIAGRAM 8

Figure 1-4 Inserting a New Diagram into the Hierarchy

Deleting Diagrams from a Project Hierarchy

Deleting a diagram from the project hierarchy is accomplished using Delete Diagram on the Tools menu. Upon selecting Delete Diagram, you must select a diagram from the hierarchy and indicate whether to delete the diagram only, or to delete it and all of its children. If you delete the diagram only, it has the same effect as removing a diagram level from the hierarchy and bumping the children up one level. Any nest relationships that existed between the diagram being deleted and its children are removed, and there is no nest relationship to the diagram(s) that takes the place of the deleted diagram. You must establish any desired relationship using the Nest function.

You can also delete boilerplate diagrams with Delete Diagram.

Figure 1-5 is an example of the result of deleting a diagram from a project hierarchy. Notice the hierarchy structure before diagram 2 is deleted, then notice how the children of diagram 2 are bumped up one level after the diagram is deleted from the hierarchy. There are no nest relationships between diagram 8 and diagrams 3 and 4, although any nest relationship between diagrams 1 and 8 is maintained.

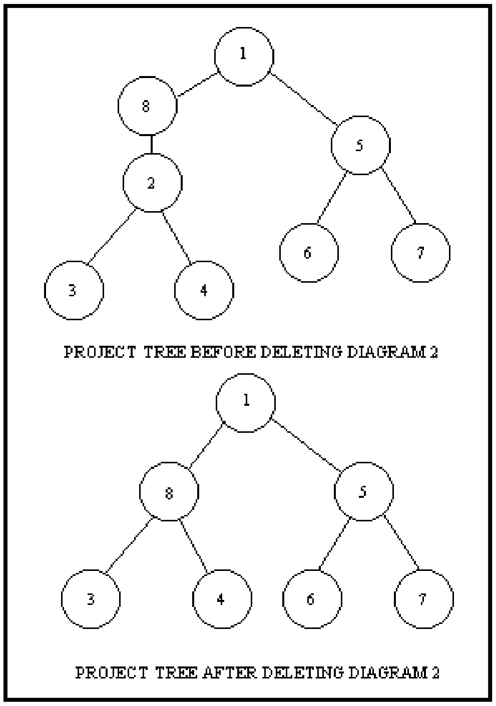

1
8
5
2
3
4
6
7
PROJECT TREE BEFORE DELETING DIAGRAM 2
1
8
5
3
4
6
7
PROJECT TREE AFTER DELETING DIAGRAM 2

Figure 1-5 Deleting a Diagram from the Hiuerarchy

Copying Existing Diagrams Between Projects

This section describes how to copy an existing diagram from one project to another (or from one level in a project to another) using the Copy Diagram function on the Tools menu. Upon selecting Copy Diagram, you must first define which project root you want to copy from. Then you must select the diagram or diagrams you want to copy. You can copy diagrams from more than one diagram type at one time. Make selections for all of the diagram types you want to copy and then click OK to copy them.

For non-DFDs, the standard Windows multiple selection techniques can be used. Click on individual diagrams in any sequence to select; click on a diagram a second time to deselect it.

Because of their hierarchical nature, DFDs work slightly differently. There is a check box in the Copy Diagram dialog box named Mark Branch. If this is checked and you click on a diagram, all of its descendant diagram branches are selected. If Mark Branch is not checked, only the diagram name you click on is selected. Note that if you deselect a diagram and there are superior diagrams in the branch selected, all of its children and grandchildren, etc., deselect. You cannot skip levels within a project branch when selecting diagrams, as this would disrupt nest relationships in potentially unintended ways. If you select diagrams to copy in different branches of the project, the top diagrams of all of the branches you select become sibling diagrams (on the same level) in the Copy Diagram destination.

You must also indicate the project to which you want to copy the diagram(s) and, for DFDs, destination parent diagram.

For DFDs, nest relationships are not automatically set up between any diagram(s) copied and the parent diagram being copied to. You must establish any desired nest relationships via the Nest function described in the Drawing Diagrams chapter. When this is done, all processes (on data flow diagrams) are renumbered according to their new locations in the project hierarchy. Figure 1-6 is an example of how a diagram labeled 4 is copied into a project hierarchy with diagram 2 the parent diagram copied to.

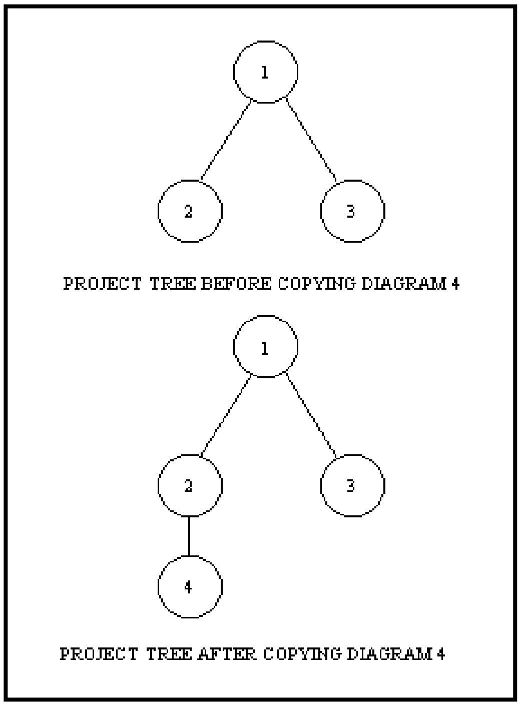

1
2
3
PROJECT TREE BEFORE COPYING DIAGRAM 4
1
2
3
4
PROJECT TREE AFTER COPYING DIAGRAM 4

Figure 1-6 Copying a Diagram into a Different Hierarchy

Notes

- If rules are enabled for the project into which diagrams are being copied, Visible Analyst performs a check to assure that no diagram labels are duplicated as a result of the copy. You are prompted to enter a new label for any diagram that has a duplicate name to another diagram that already exists in the destination project. Also note that if an object label in the diagram being copied conflicts with an object label in the project being copied to, Visible Analyst does not copy the duplicate object.

You cannot copy both boilerplate diagrams and diagrams in other diagram types in a single execution of the Copy Diagram function. If you select both boilerplate diagrams and diagrams in other diagram types, only one group is copied. If Boilerplate appears as the diagram type when you click OK, the boilerplate diagrams are copied and you are prompted for a new label for the copied diagram. Otherwise, the rest of the diagrams are copied.

If the projects involved in a copy operation have repository entries, then the repository entries associated with the diagram(s) being copied from the source project are copied into the repository of the destination project. These repository entries include the entries for all objects on the diagrams being copied, as well as any derivative entries associated with the Alias and Composition fields of these objects. (This assumes that they are copied into a project using the same methodology rules, and that there are no conflicting labels).

In summary, the following repository functions are performed when copying a diagram between projects:

- Visible Analyst copies all repository information that is necessary to recreate the entries in the destination project.
- If the destination project has a repository entry with the same label and same entry type as an entry being copied, then Visible Analyst assumes that it is the same entry. It does not overwrite any existing fields that are already defined in the destination project. It does copy only those entries that are not currently defined in the destination project.
- New repository entries are created in the destination project for all aliases that are included in the entries being copied.
- New repository entries are created in the destination project for all composition items that are included in the entries being copied.

PRINTING YOUR WORK

Printing diagrams can be accomplished from the File menu.

Print Options

When you print, you have the following options:

- You can select the target printer and access its properties. You can:
  - Select a paper source.
  - Select a number of other options depending on the target printer.
- You can print the diagram in the active window. By using the Print Range options, you can determine how much of the diagram to print. All prints the entire diagram; Selection prints the current selection; and Pages indicates only certain pages of a multi-page diagram are to be selected. Type page numbers separated by commas, or a range of pages separated by a hyphen, or click the Select Pages button to graphically choose the pages.
- You can view the Visible Analyst print queue, where you can add or delete diagrams. If no diagram is open, you can print the diagrams in the queue.
- You can print a snapshot of the tree structure for any diagram type in the current project.
- If you have chosen to print a global view of a data model or a very large decomposition diagram or cluster diagram, Visible Analyst enters a special item into the print queue that is easily identifiable. These special items segment themselves and print over several pages. A key to the segmentation page is printed to aid in reassembling the pages of the view print for display. If you interrupt the printing of the queue when it contains one or more of these special items and then redisplay the print queue, you see identifiers to the individual diagrams in the queue.
- If you are printing one or more easel (11 x 15) or multi-page diagrams, you might want to take advantage of scaling the diagram so that it fits on a single page. When Scale To Page is checked, the size of everything on the diagram is reduced so that it fits on whatever page size you selected for the diagram.
- If you choose this option for an easel diagram and your diagram is set to use portrait orientation, the diagram objects appear quite small and there is considerable white space at the bottom. In this case, you should also change your page orientation to landscape. Note that for multi-page diagrams, the scaling option causes things to become quite tiny (unless you are printing to a single page on a large plotter). Note further that the only part of a multi-page diagram that is scaled and printed is the part that you have actually used, not the entire 180 x 176 inch area that is possible to use.
- You can choose to scale a diagram a specific amount by entering a value in the Scaling Percentage field. Unless you check Override Diagram Scaling, this value is combined with the scaling for each diagram. For example, if the scaling factor for a diagram is 50% and you set the scaling percentage to 50%, the diagram is actually be printed at 25% of its actual size.
- You can print the diagram in color or in black and white. If the target printer does not support color, any colors used on the diagram are dithered.
- If you are printing from the queue, you can either remove all items from the queue when the print job has completed, or retain the list for a future print job. Retaining the queue is useful if you plan to print a set of diagrams repeatedly.
- You can use the printer defaults for page size and orientation instead of the settings stored with each diagram. This allows you to print a set of diagrams with the same page characteristics, without having to change the settings for each diagram in the queue. You

can change settings for the printer driver, such as page orientation (portrait or landscape), print to file, etc., by clicking the Setup button in the Print dialog box.

- You can choose the number of copies of each diagram to print.
- When printing multi-page diagrams, you can choose to have the page number printed in the upper left hand corner of each page.
- Neither the grid nor the ruler is included in the printed copy of a diagram, even if you have it displayed on the screen.

The procedure for printing is very straightforward. Choose Print from the File menu. At the dialog box, choose from among the options described above and click one of the available buttons.

- If you choose Modify Queue, the dialog box expands to show the current print queue. If you want to delete diagrams from the queue, select them by clicking on them and clicking Delete. If you want to add diagrams to the queue, click Add. Another dialog box displays, showing you a list of all of the diagrams for the current diagram type in the active project. As above, click on the diagrams you want to print and click the Add button. You can also choose different diagram types and different projects from which to add diagrams to the queue. When you are finished selecting diagrams, click OK to return to the print queue display. You can then print the diagrams in the queue (if there are no diagrams open), adjust the printer setup, continue the add and delete process or cancel.
- You can print a layout page that details how the separate pages of a multi-page diagram should be assembled.

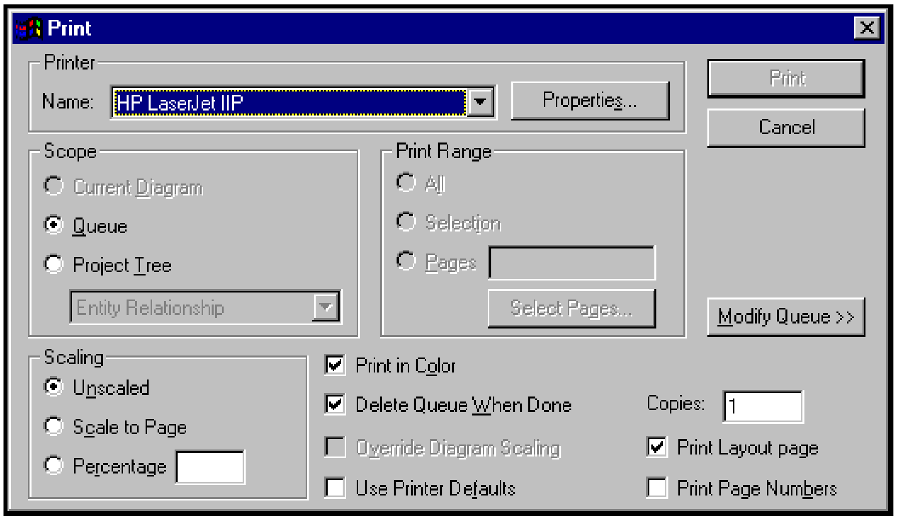
Selecting Pages for Printing

When printing a multi-page diagram, you can use the Select Pages button on the Print dialog box to display a reduced view of the current diagram. The diagram is divided into the pages that can be selected. When you select pages, you have the following options:

- To select a page, either click with the left mouse button to select a single page, or press the left mouse button and drag the mouse to select a rectangular area.
- To deselect a page, click a second time with the left mouse button. You can also change the drag mode by selecting Clear or by clicking the right mouse button. Now when you click and drag the mouse, pages are de-selected.
- To select all pages, click Select All.
- To reverse your selections (select those pages that are not selected and de-select those pages that are), click Invert.
- To deselect all pages, click Clear.
- When you have completed your selections, click OK. The Pages field on the Print dialog is filled with your choices.

Printing

Choose Print from the File menu. At the dialog box (refer to Figure 1-7), select from the available print options as described above. If there are multiple diagrams open in Visible Analyst, you can only print the currently active one.



Printer
Name: HP LaserJet IIP

Scope
○ Current Diagram
● Queue
○ Project Tree
Entity Relationship

Print Range
○ All
○ Selection
○ Pages

Scaling
● Unscaled
○ Scale to Page
○ Percentage

☑ Print in Color
☑ Delete Queue When Done
☐ Override Diagram Scaling
☐ Use Printer Defaults

Copies: 1
☑ Print Layout page
☐ Print Page Numbers

Figure 1-7 The Print Dialog Box

Note that if there are multiple diagrams open in Visible Analyst, you can only print the currently active one.

Chapter 2

Strategic Planning

Planning and requirements identification is often the initial phase in an enterprise-engineering project. During the planning phase, you develop a comprehensive strategic business plan that meets the identified mission and purpose of the organization. Visible Analyst not only allows you to create these statements, but also allows you to link them to other objects in your repository. This allows you to track the software development process from the planning stages through analysis, design, and implementation. Linking planning statements to model objects helps you determine the significance of each object and ensures that each object is essential in supporting the organization's business plan.

PLANNING PHASE

The Planning phase allows senior-level management to capture and document the business vision it has for the organization. Once this business vision is captured, management can communicate the vision effectively to the people who will implement it.

During the Planning phase, management develops a comprehensive strategic business plan through formal or informal planning sessions.

When the Planning phase is complete, participants have a dynamic plan that meets the identified mission and purpose of the organization. This plan is captured in a set of planning statements that forms the foundation for the project.

Planning session participants use the following standard strategic planning techniques to produce a business plan:

- External and internal assessment
- Goal analysis
- Strategy and objective formulation

PLANNING STATEMENTS

As the product of the Planning phase, planning statements communicate the business vision and rules that govern the organization. Written in business language, the planning statements provide the framework to ensure that the data model developed in subsequent phases meets the information requirements of the business.

Each planning statement is assigned a statement type. Many predefined statement types come with Visible Analyst:

- Vision
- Assumption
- Mission
- Strength
- Weakness
- Opportunity
- Threat
- Goal
- Strategy
- Issue for Resolution
- Critical Success Factor
- Objective
- Policy
- Tactic
- Task
- Business Event
- System Event
- System Requirement
- System Design Objective

Your organization may use different terms for these types. You may add different statement types using the Planning Statement Types function on the Options menu (as described later in this chapter).

Statements support each other in a hierarchical relationship according to their types (objectives support the mission, policies support objectives, etc.). You form this hierarchical structure in the Planning Outline window.

Object Links

Planning statements are integrated into the analysis and design phase by links to the model objects (entities, attributes, processes, classes, etc.) that are created to support them.

Linking planning statements to model objects helps you determine the significance of each object you add to the model. This ensures that each model object is essential in supporting the organization's business plan.

For example, an organization's business plan includes the following strategy: "When an order is received, it is immediately processed and invoiced." This planning statement would be linked to the entities ORDER and INVOICE, since those entities support this strategy. These entities can also be linked to other planning statements, and this strategy can also be linked to other model objects.

Statement Priority

You can assign a priority level to each statement. The priority level conveys the importance of the statement to the business plan.

Statement Description

You can display the planning statement description. When this option is selected, a window opens below the statement hierarchy window that displays the description of the selected planning statement. If you want to change the description, simply click on the window (or press the Tab key) and begin typing. If you do not have rights to modify the object, or if another user is editing it, you will not be able to change the description. Once the focus is set back to the statement hierarchy, changes that you made are saved.

To change the size of the description window, move the cursor over the hierarchy/description border. When the cursor changes to a vertical splitter ╪, press the left mouse button and drag the mouse until you reach the desired window size.

Note

Do not use these special characters when naming a planning statement because they are reserved repository characters. ! @ # $ % ^ & * ; :

DEFINING PLANNING STATEMENT TYPES

You can define your own statement types, customizing Visible Analyst to your specific strategic planning needs. This feature not only allows you to create new types, but define how these types are related to other planning statements and linked to other objects in your repository.

To create a planning statement type:

Select Planning Statement Types: 1 Select Planning Statement Types from the Options menu. The Planning Statement Types dialog box appears.

Select New or Existing Project: 2 Click either New Project Defaults or Current Project.

- New Project Defaults. Choose this option to modify the list of statement types added to all new projects.
- Current Project. Choose this option to modify the list of statement types defined for the current project. If this option is selected, types can either be added or removed, but the definition of existing types cannot be changed. If a statement type is being used, it cannot be removed.

Set Type Name: 3 This is the name of the new statement type being created. Whenever you define a planning statement, this name appears in the list of available statement types.

Set Composite Type: 4 If you choose this option by clicking your left mouse button on the check box labeled Composite Type, you allow any newly defined planning statements of this type to contain other planning statements. This allows a hierarchical relationship to be created between statements according to their types (Objectives support the Mission, Policies support Objectives, etc.). For every item created as a child of another statement, a location reference for this entry is created in the Locations field of the statement's repository entry. See the Composition section of the Repository chapter for information on a related topic. If Composite Type is not checked, planning statements of this type are at the lowest level in the hierarchy.

Set the Link Option: 5 This option allows you to establish a link or series of links between planning statements of this type and other repository objects and user-defined objects. To choose this option, click the left mouse button on the check box.

Set the Link Cardinality: 6 This list box describes the cardinality between any linked objects. The default setting is 1:1. This means that only One planning statement object can be linked to one repository object. There are four cardinalities allowed. The entry to the left of the colon always refers to the planning statement. The entry to the right always refers to the linked repository object.

Set the Link To: 7 This field describes the repository types that can be Linked to the planning statement. The default is All. You can click the drop-down arrow for a list of options.

Save the Object: 8 Click Add to save the planning statement type.

Notes

- If you modify the New Project Defaults list, you must create a new project in order to implement newly defined planning statement types. They ARE NOT valid with the currently selected project unless you select Add to Current Project or you choose the Current Project option in Step 2.
- If using a network version of Visible Analyst, every planning statement type is accessible with any subsequently created project. If you create a Requirement type, anyone with access to Visible Analyst also has that Requirement type in any project they may create.

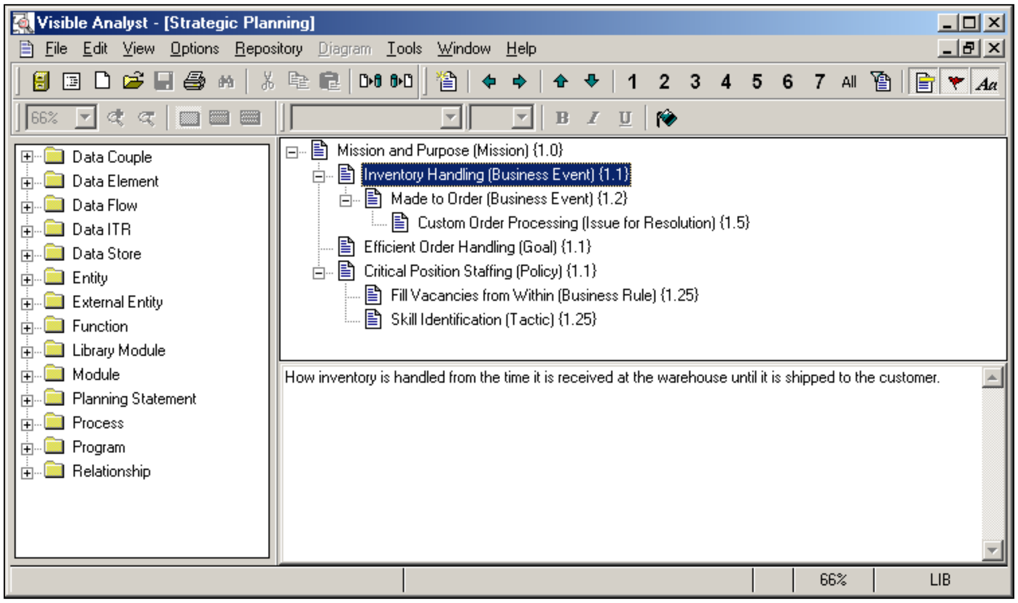
PLANNING WINDOW

Planning statements are captured and refined in Visible Analyst through the Strategic Planning window. This window allows you to add statements to and delete statements from your repository, as well as to edit and to organize them. In addition, the planning windows let you link planning statements to other modeling objects such as entities, attributes and classes created during later phases of the software development process.

To open the Strategic Planning Outline window:

Open the Window: 1 Choose Strategic Planning from the File menu, or

Click the Strategic Planning icon ▤ on the control bar. Planning is only available in the Zachman Framework and Corporate Editions of Visible Analyst. If this option is grayed out, it means your version of Visible Analyst does not support planning statements.



- Mission and Purpose (Mission) {1.0}
  - Inventory Handling (Business Event) {1.1}
    - Made to Order (Business Event) {1.2}
      - Custom Order Processing (Issue for Resolution) {1.5}
  - Efficient Order Handling (Goal) {1.1}
  - Critical Position Staffing (Policy) {1.1}
    - Fill Vacancies from Within (Business Rule) {1.25}
    - Skill Identification (Tactic) {1.25}

How inventory is handled from the time it is received at the warehouse until it is shipped to the customer.

Figure 2-1 Strategic Planning Outline Window

The Planning Outline window provides a hierarchical view of planning statements defined in the project repository.

Use the Planning Outline window to:

- Add, edit, and delete statements.
- Move and copy statements to other positions within the outline.
- Link statements to other repository objects.
- Assign priorities to statements.

Each planning statement defined in the repository is shown in a tree-link fashion in the planning window. For each statement, the following information can be displayed.

Statement Icon – A symbol indicating if any levels are collapsed beneath the statement.

- A plus symbol (+) means that one or more levels are collapsed beneath the statement.
- A minus symbol (-) means that all levels beneath the statement have already been expanded.
- No symbol means that there are no levels below the statement.

Statement Title – The title of the statement.
Statement Type – The type of statement. Statement type is shown in parentheses ( ) after the title.
Statement Priority – The priority level assigned to the statement. The Statement priority is shown in curly brackets { } after the title and type.
Statement Description – A description of the selected planning statement shown in a window below the statement hierarchy window.

The Planning Outline window displays planning statement titles in a tree arrangement similar to the directory tree in Windows Explorer. Like Explorer, you can expand and collapse branches of the outline to vary the level of detail displayed in the window.

The tree is hierarchical. Statements may have child (hierarchically subordinate) statements that in turn may have their own child statements if the statement type allows it. You may expand all hierarchy branches or collapse one or more of them to show only parent statement branches.

You can change the appearance of the statement window by using commands on the View menu, by right clicking on the window to display the Property menu, or by clicking on the appropriate buttons on the control bar.

Using the Planning Window

The Planning window provides a view of the titles of statements visible in a single branch or the entire repository. Because only the titles are displayed, you can easily rearrange the statements to reflect their hierarchical relationships to one another. This window does not provide detailed information about each statement. (That is the function of the Define dialog box. See the Repository chapter of this manual.)

Use either the Planning window or the Define dialog box to add planning statements. Only the planning window allows you to organize statements hierarchically. If you use the Define dialog box, the new statement is added at the end of the hierarchy on the first level.

Building a Planning Outline

You begin building a planning outline either by adding statements to the Planning Outline window or by rearranging existing statements into the appropriate hierarchical order.

Planning statements are usually arranged from the top down (following strategic management principles). This means that strategic statements, such as the mission statement, are placed in the highest level of the outline. Operational statements, such as policies, are placed in the lower levels.

New statements appear as a child of the currently selected statement; otherwise, they appear at the top of the outline. You can use the mouse or the control bar buttons to select, move or copy statements to other locations in the outline.

Control Bar Buttons

The control bar for the Strategic Planning Outline window is shown below.

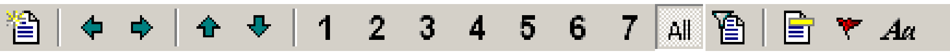


Figure 2-2 Strategic Planning Outline Window Control Bar

Insert a new planning statement.

⇦ Promote the selected planning statement one level higher than its present position.

⇨ Demote the selected planning statement down one level lower than its present position.

⇧ Move the selected planning statement up one position

⇩ Move the selected planning statement down one position.

1 2 3 4 5 6 7 All

Show Level – Select how much of the tree is displayed. Level 1 shows only the top-level nodes, while All displays all nodes. You can change the display of individual branches by clicking on the + to the left of the planning statement.

Click on Filter to choose a particular branch of the planning hierarchy to display.

This is useful for projects that contain a very large number of planning statements.

Display statement types

Display statement priority.

Aa Display the statement description.

In addition to the control bar, options for changing the statement hierarchy are available on the Edit and View menus, as well as by right clicking on the planning window to display the Properties menu.

Adding a New Statement

To add a new planning statement to the hierarchy:

Select the Parent: 1 Click on the planning statement to which you would like to add a child statement. If no statement is selected, the new statement will be added to the top of the hierarchy.

Open the New Statement Dialog Box: 2 Click the New Statement icon on the control bar, or right-click the parent and select New from the Properties menu to display the Add Planning Statement dialog box.

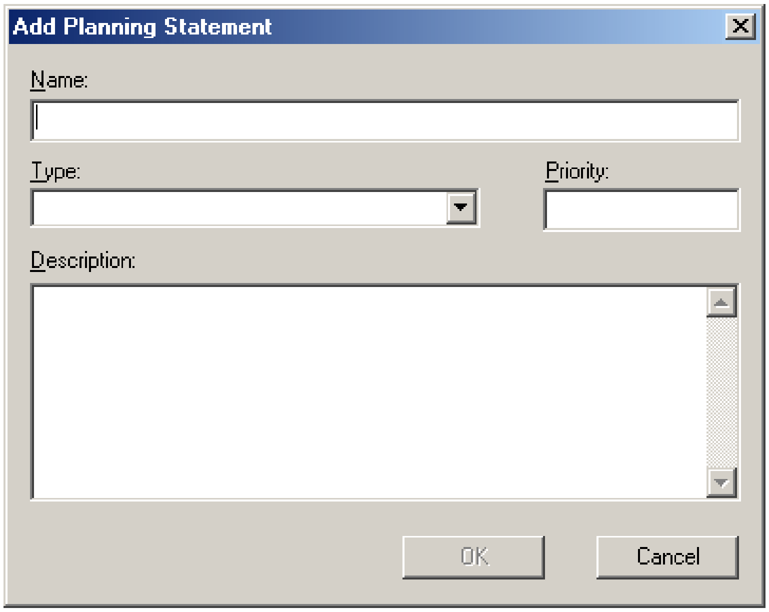


Figure 2-3 Add Planning Statement Dialog Box

Define the Statement: 3 Enter a name for the statement. Statement names must be unique. NOTE: Do not use these special characters in the name of the statement because they are reserved repository characters: ! @ # $ % ^ & * ; :

4 Select the statement type from the list.

5 Set the priority. This is an option field that can contain any value that you like. Generally, a numeric value is used to set the relative priority against other statements.

6 Describe the statement.

Save the Statement: 7 Click OK. The statement is added to the hierarchy.

Moving Statements

There are several ways that the hierarchy can be modified. The Move Up and Move Down options on the control bar, the Edit menu, and the Properties menu move the selected statement up or down one position. In addition, you can use the mouse by holding down the left button and dragging the statement to its new position. While you are moving the mouse, an insertion marker appears showing you where the statement will be placed.

Adding Additional Detail to a Planning Statement

Like all objects that are stored in the repository, planning statements can have more information associated with them, such as a short description, notes, and especially links to other objects. To modify a statement's complete definition, click on the desired statement and select Define from the Repository menu, right-click on the item and select Define from the Properties menu, or double-click on the item. For detailed information on the Define dialog box, see The Visible Repository chapter of this manual.

Chapter 3

Drawing Diagrams

THE DIAGRAMMING PROCESS

In its most simplified terms, Visible Analyst diagramming is a three-step process consisting of adding symbols to a diagram, adding lines to a diagram, and entering text. The sequence in which you perform the process is entirely up to you. For example, you may choose to position all of the diagram symbols before positioning any lines. Then you could position text to label each symbol and line; or you could enable the Auto Label function and be prompted to label each symbol or line as it is positioned. You can also perform editing functions on any diagram, allowing you to stylize, reposition, or re-label any object as you develop the diagram; or you can return to the diagram during a subsequent work session to perform edits.

Detailed procedures that can help you perform the diagramming and editing processes are provided in this chapter. These procedures provide all of the information you need to draw symbols and lines and label them. Editing procedures are also described that allow you to stylize, copy, move, or delete any symbol, line, or text entry.

Note that all of the diagramming procedures are supplemented by helpful information when using rules and repository functions. Other information explaining the use of nesting functions, creating and loading constructs, etc., is also provided to help you understand the complete Visible Analyst diagramming process. Finally, although some explanations discuss only data flow diagramming, they are valid for all diagram types unless otherwise noted.

Before You Begin

Although the Visible Analyst diagramming procedure is easy to learn, you should understand the basic operating principles, project diagram hierarchies, and the menu selections presented in Getting Started before proceeding. If you understand the basic Visible Analyst operating principles and the menus, you can interactively follow along and perform all of the procedures presented in this chapter.

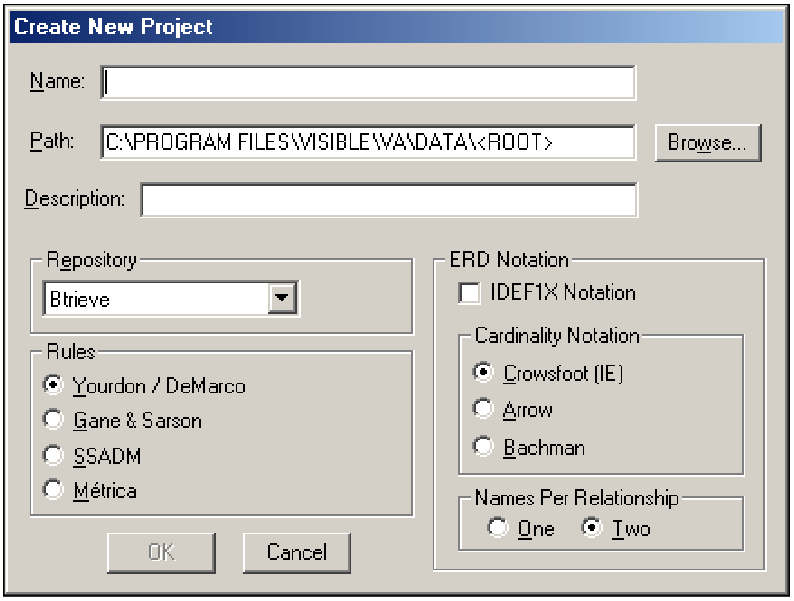
Before using Visible Analyst, you should also be aware of a few basic operation aspects and important decisions you should make prior to creating a new project and drawing the first diagram. These operation aspects are listed below and described in the following paragraphs.

- Rules and repository considerations.
- Changing diagram types (data flow, functional decomposition, structure chart, entity relationship, entity life history, business process model, class, state transition, use case, activity, sequence, collaboration, unstructured or boilerplate).
- Enabling and disabling boilerplates.

- Using boilerplate keywords.
- Adding image files to a diagram or boilerplate
- Selecting a page size.
- Selecting a project.

CREATING A NEW PROJECT

Identify the Project: 1 At the File menu, first select New Project. The Create New Project dialog box appears.



Name:

Path: C:\PROGRAM FILES\VISIBLE\VA\DATA\<ROOT>

Description:

Repository: Btrieve

Rules
● Yourdon / DeMarco
○ Gane & Sarson
○ SSADM
○ Métrica

ERD Notation
□ IDEF1X Notation

Cardinality Notation
● Crowsfoot (IE)
○ Arrow
○ Bachman

Names Per Relationship
○ One ● Two

Figure 3-1 Create New Project Dialog Box

2 Type a project name up to 200 characters long (if you are creating a project on a drive that does not support long file names, or using a database other than Btrieve you are limited to 4 characters). Then type a short description. Identify the path where the project is to reside. The default location is in a subdirectory beneath Visible Analyst with the same name as the project root.

Rules and Repository Considerations

Rules Considerations

Rules are applied to diagrams on a project-wide basis; that is, all diagrams in a particular project must have the same rules. You must therefore be sure that the desired Rules are selected prior to drawing the first diagram for any new project. You *cannot* go back and change your selection for a project once the project has been created. You can use Copy Diagram to copy the diagrams to a project using a different rule set. The appropriate methodology symbols will be changed from the old rule set to the current projects rule set. See Copy Diagram later in this manual for additional information.

You can select Yourdon/DeMarco, Gane & Sarson, SSADM, or Métrica methodology for data flow diagrams. Yourdon/Constantine methodology is used by default for all structure charts if the structure chart diagramming mode is selected. Refer to The Visible Rules section of this manual for detailed information about applying rules to a project.

To make your rules selection for any new project:

Select a Methodology While Creating a New Project:

3 Select Yourdon/DeMarco, Gane & Sarson, SSADM, or Métrica as a methodology for your project.

4 Continue by defining the repository function for your new project, as described below.

Data Repository Considerations

Rules are a prerequisite for the Visible Analyst repository; for example, Yourdon/DeMarco, Gane & Sarson, SSADM, or Métrica methodology must be selected. Like rules, the repository function is also applied to diagrams on a project-wide basis, where all diagrams in a particular project share the same repository. Therefore, you must be sure that the repository rules you want to use are enabled prior to drawing the first diagram for a new project. If a project is created using one set of rules, you *cannot* go back later and change the repository rules for that project. You can use Copy Diagram to copy the diagrams to a project using a different rule set. The appropriate methodology symbols will be changed from the old rule set to the current projects rule set. See Copy Diagram later in this manual for additional information.

Specific instructions for using the capabilities of the repository are provided in The Visible Repository manual section.

To enable the data repository selection for any new project:

Select the Repository Prior to Creating a New Project:

5 Select the database engine to be used for the repository.

6 Continue by defining the ERD notation for the project, as described below.

ERD Notation

Relationship Cardinality Notation

A relationship's cardinality shows how many instances of one entity type relate to how many instances of the entity type at the other end of the relationship. To show a relationship cardinality of "Many," a line terminator must be used. Visible Analyst gives you a choice of four:

- Crowsfoot Notation - a three-way fork.
- Arrow Notation - a double arrowhead.
- Bachman Notation - a single arrowhead.
- IDEF1X

IDEF1X Notation

Click this check box to select IDEF1X as the default notation for your entity relationship diagrams. Features include entity notation (independent entities, dependent entities), category notation, cardinality notation, and group attributes. If you select IDEF1X as the relationship cardinality when creating the project, you would then select Crowsfoot, Arrow or Bachman as an alternate cardinality notation.

Choose a Cardinality Notation:

7 Select one of the three choices presented.

8 Continue by defining the number of names for relationships, as described below.

Number of Names for Relationships

Relationships are inherently bi-directional. However, you may not want to save relationship names in both directions. You may save two names for each relationship or just one.

Choose the Number of Relationship Names to Use:

9 Select one of the two choices presented.

10 Click OK to create the project.

Selecting an Existing Project

Once projects have been created and saved, you can easily load different projects into Visible Analyst by choosing Select Project from the File menu and selecting the desired project from the list. When an existing project is selected and loaded into Visible Analyst for access, all of the original project parameters stored for the project are automatically restored.

The Uses of Different Diagram Types

The different diagram types are discussed in detail in the Drawing Diagrams section of this manual. The paragraphs that follow are summaries to make the diagramming processes clearer.

Functional decomposition diagrams (FDDs) give you the ability to do high-level planning of business functions diagrammatically while concurrently populating the repository. You can enter business functions onto diagrams and break them down into successively finer gradations. At some point, one that is entirely up to you, you can decompose business functions (hereinafter called simply functions) into processes semantically equal to those that appear on DFDs. The processes can themselves be decomposed into smaller parts (still lower-level processes) on FDDs.

Data flow diagrams (DFDs) represent the functional tasks of your project analysis. For example, a data flow diagram might represent a particular functional area of a department within your company. That data flow diagram might also be an integral part of a project that consists of multiple data flow diagrams that represent multiple levels of department functions and tasks.

Note

- A *functional* decomposition diagram is very different from a *process* decomposition diagram. The former is a full diagramming methodology for doing business planning. The latter is simply an unstructured diagram laying out the hierarchy of processes that are descendants of an indicated process.

Structure charts (SCs) are a graphical representation of the top-down design of a project, showing the program modules that carry out the system functions defined in a data flow diagram. They also show the hierarchical relationship between these modules and how they invoke one another. The Entity Life History (ELH) diagram is a component of both the SSADM and Métrica methodologies and is similar to a structure chart. The ELH shows how events in a system affect data entities.

Entity relationship diagrams (ERDs) graphically describe the data and the relationships among data items in your project. You can draw entities (or, more properly, entity types) and the relationships between them, including relationship names in both directions. Later, the data elements (also called attributes) composing these entities can be added to the repository and, with the Key Synchronization function, key information can be generated for relationships. Later, if you choose, you can balance your data model against your process model to further enrich the information stored in your project.

The Visible Analyst supports the concept of an SQL view for use on an ERD, which can be thought of as a derived or virtual table. A view is similar to an entity in that it has a composition but the items that appear in the composition of a view must belong to other entities or be expressed based on data elements used by another entity. See the topic SQL View Support (IntelliViews™) for additional information.

Business Process Model Notation (BPM) models describe business process behavior and as a result use an event based paradigm. Both parallel and conditional behavior is supported in the modeling notation and also in the Visible Analyst's implementation of BPMN. A number of symbols are used to describe process flows, events and decisions and allow the viewer to easily differentiate between sections of the BPMN diagram.

Class diagrams (CLDs) graphically describe the object model that contains classes (an object is an instance of a class) and the relationships between classes in your project. You can draw classes that contain information about attributes and member functions and relationships to indicate association, inheritance, and aggregation.

State transition diagrams (STDs) are graphical representations of the dynamic model that contain states and events that cause a change in state. State transition diagrams can be linked to classes through use of the Nest function. An activity diagram is a special form of state diagram where states represent performance of activities or subactivities, and transitions are triggered by completion of activities or subactivities.

Use case diagrams show the relationship between a user and a computer system. A use case diagram captures some user-visible function.

Sequence diagrams are one type of iteration diagram that describe how objects collaborate in some behavior. Collaboration diagrams are another type of iteration diagram that show interaction organized around the objects in the interaction and their links to each other.

Unstructured diagrams are not linked to the repository. They are simply free standing diagrams. They can be put to any use you want. Cluster diagrams and views, generated from ERDs, and process decomposition diagrams, generated from DFDs, are unstructured diagrams.

Users can create Symbol Templates via the Options menu for use on unstructured diagrams. These Templates use images as the diagram symbols allowing users to create new unstructured diagrams, such as a Network diagram.

Note

- You can define defaults for the line types (straight line, dashed line, arc, etc.) and terminator types (solid arrowhead, open arrowhead, no terminator, etc.) that can be drawn on unstructured diagrams. They can be made the same as those used for any other diagram type. You can make this setting by selecting Line Settings from the Options menu. From the top box, you can choose the line and terminator values that are accessible when the diagram type is set to unstructured. However, the line values set for a particular diagram are *those in effect at the time the diagram was created*, so you should set this *before* you begin the new diagram creation process.

Boilerplate diagrams do not represent project designs, but instead represent standard text and graphics that are available for insertion into other diagrams. For example, a boilerplate diagram might be a corporate logo or standard heading that is to be included in your diagrams. You may create many boilerplate diagrams, and each is saved in a special Visible Analyst file that is specifically set aside for all boilerplates. You can then select any one boilerplate for inclusion in subsequently created diagrams if you also enable a boilerplate when you create a diagram, as described below. It is important to understand that rules and data repository capabilities do not apply to boilerplate diagrams.

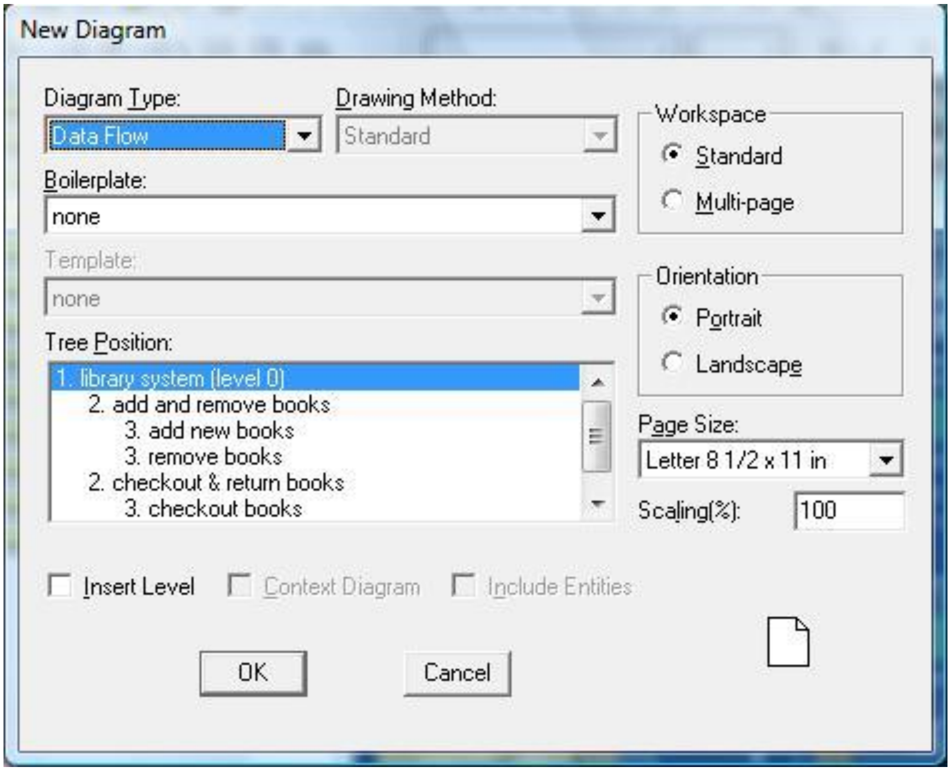
Unstructured Diagrams

When you create a project, you can have unstructured diagrams coexisting with methodology diagrams in the project. The methodology-based diagrams are linked to the repository.

Creating a New Diagram

Visible Analyst allows you to work with many different diagram types, so the first step in creating any new diagram is to:

Open the Dialog Box: 1 Select New Diagram from the File menu. The New Diagram dialog box appears.



New Diagram

Diagram Type: Data Flow

Drawing Method: Standard

Boilerplate: none

Template: none

Tree Position:

1. library system (level 0)
    2. add and remove books
        3. add new books
        3. remove books
    2. checkout & return books
        3. checkout books

Workspace: ● Standard ○ Multi-page

Orientation: ● Portrait ○ Landscape

Page Size: Letter 8 1/2 x 11 in

Scaling(%): 100

□ Insert Level □ Context Diagram □ Include Entities

Figure 3-2 New Diagram Dialog Box

Set the Diagram Type: 2 Set the desired diagram type.

Note

Once you begin drawing your new diagram, you *cannot* go back and change that diagram to another type.

Selecting a Drawing Method

If you are creating an entity relationship, class, or unstructured diagram, choose the method by which to draw the diagram. Standard is the normal manual process where you control how objects are added to the diagram. The view method automatically creates a diagram based on the type of view selected. Refer to The View Functions later in this chapter for more information.

Select the Drawing Method: 3 Set the desired drawing method.

4 If you are creating a class diagram using one of the view methods, and you want entities to be included as classes, select Include Entities. Once an entity is added to a class diagram, it is converted to a class in the repository, with an entity subtype. It can be used freely on both entity relationship diagrams and class diagrams.

Selecting a Page Size

When choosing the page settings for a diagram, either when creating a new diagram (New Diagram from the File menu), or changing an existing one (Settings from the Diagram menu), the following options are available.

Workspace

- Standard defines a single page drawing area. The dimensions of the page are determined by the page size, scaling factor, and orientation. Prior to version 6.1, standard defined a 9 x 11 inch drawing area.
- Multi-page defines a 180 x 176 inch drawing area. On a multi-page diagram workspace the places where the diagram breaks over pages when printed are indicated. These marks are determined by the page size, scaling factor, and orientation, and do not print on your diagram. While it is permissible to use this multi-page size for any diagram type, it may not always be the wisest thing to do. Remember that one of the principles of analysis is to break things up into small sections that take up no more than one standard page to make the concepts more manageable and easier to understand.

Orientation

Portrait indicates the page is taller than it is wide; Landscape indicates the page is wider than it is tall.

Page Size

Choose one of the available paper sizes for your printer. The size you choose affects the size of the diagram workspace. If you have chosen the standard workspace, a diagram is printed on a single piece of paper. If you chose multi-page, the number of printed pages depends on the page size selected.

Scaling (%)

Scaling determines the size of objects on a diagram when the diagram is printed. If you choose a scaling factor that is less than 100 percent, you have a larger workspace for each page when editing the diagram; however when the diagram is printed, the objects are smaller. If the scaling factor is greater than 100 percent, the workspace is smaller; and, when the diagram is printed, the objects are larger. Scaling affects the page breaks on a multi-page diagram. You can choose a scaling percentage between 10 and 400.

Note

- In some cases, the combination of workspace, page size, and scaling factor selections can result in an invalid page size. This can occur if the page size is very large and the scaling factor is very small, or the objects on a diagram do not fit within the boundaries of the new page size. If this happens, the scaling factor is reset to a value that results in a valid setting.

You may select a page size according to your diagram requirements. At a later time you can change the diagram size from the Diagram menu. However, be aware that if you initially select the multi-page format, you *cannot* go back and decrease the page size to standard once your diagram has been created and saved if the content of the diagram would prevent it from fitting on a standard diagram.

Set the Diagram Page Size:

5 Select a Workspace, Page Size, Orientation, and Scaling (%) that corresponds to the size you want.

6 Continue by defining the Boilerplate you want to appear on the diagram, if any. If you are creating a boilerplate diagram, you cannot include another boilerplate diagram in it. Boilerplates are described in detail later in this chapter.

Hierarchy Position of the New Diagram (DFDs Only)

If this is the first diagram of a project, it is at the top of the hierarchy. If you have chosen the Yourdon/DeMarco methodology, you can also check the box identifying this as a context diagram. Note that when you select a position in the hierarchy, you are selecting the *parent* of the new diagram. If you check the box labeled “Insert Level,” the selected existing diagram becomes a *child* of the new diagram. If you choose the top level diagram of a Yourdon/DeMarco project and you have not yet identified a diagram as a context diagram, you can insert a level above the existing top-level diagram and make it a context diagram by checking that box.

Note

- All of the warnings given in the discussion about inserting diagrams into the hierarchy apply here. Actual parent/child relationships are created only with the Nest function. Diagram insertions simply position diagrams so that they *can be* nested.

Select the Hierarchy Position: 7 Select the position in the project tree hierarchy where you want the new diagram to be (data flow diagrams only).

Create the New Diagram: 8 Click the OK button or press ENTER. An empty diagram window appears, waiting for your input.

Opening an Existing Diagram for Editing

Normally when you access Visible Analyst, the current project is the one that was last selected. In some instances, however, such as when you delete a project and then exit the program, Visible Analyst has no current project the next time it starts. In this case and whenever you want to work in a different project, you must choose a project before you can open the diagram you want to edit.

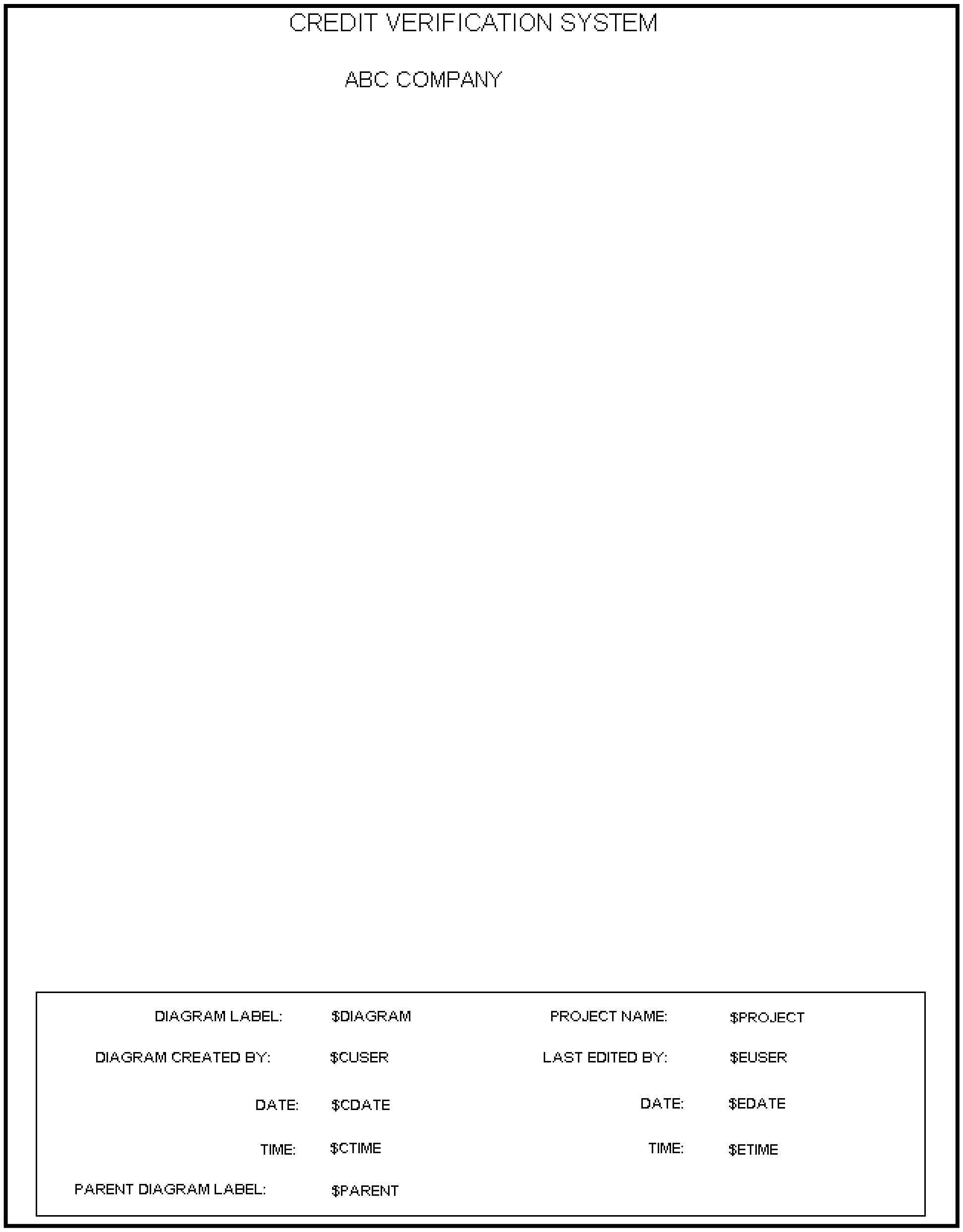
Once a project is selected, choose Open Diagram from the File menu, select the diagram type and pick the diagram you want to edit from the Diagram box. The diagram you choose opens in a window for you to edit.

USING BOILERPLATES

Enabling and Disabling Boilerplate

When creating a new diagram, the Boilerplate box allows you to select any existing boilerplate drawing for inclusion in the new diagram that you are about to draw. There are three important things to remember about this:

- If the diagram type is set to Boilerplate, Visible Analyst does not allow you to enable a boilerplate for the new diagram because you cannot include an existing boilerplate in a new boilerplate diagram.
- If a boilerplate is enabled and the diagram type is set to something other than Boilerplate, Visible Analyst automatically draws the selected boilerplate into the new diagram window when a new diagram is created. The added boilerplate is always displayed as part of the diagram whenever it is accessed. Once a boilerplate is enabled, a new diagram is created, and the diagram is saved, the boilerplate cannot be removed as one unit from the existing diagram; however, the individual parts making up the boilerplate can be removed.
- If disabled, no boilerplate is included in your new diagram. As described below, a boilerplate can be added to an existing diagram using a Construct.
- Picture files, such as a company logo or other images, can be included on any boilerplate diagram. The image will be included as part of the new diagram when this boilerplate is enabled.



CREDIT VERIFICATION SYSTEM

ABC COMPANY

DIAGRAM LABEL:	$DIAGRAM	PROJECT NAME:	$PROJECT
DIAGRAM CREATED BY:	$CUSER	LAST EDITED BY:	$EUSER
DATE:	$CDATE	DATE:	$EDATE
TIME:	$CTIME	TIME:	$ETIME
PARENT DIAGRAM LABEL:	$PARENT		

Figure 3-3 Blank Boilerplate Diagram With Key Words

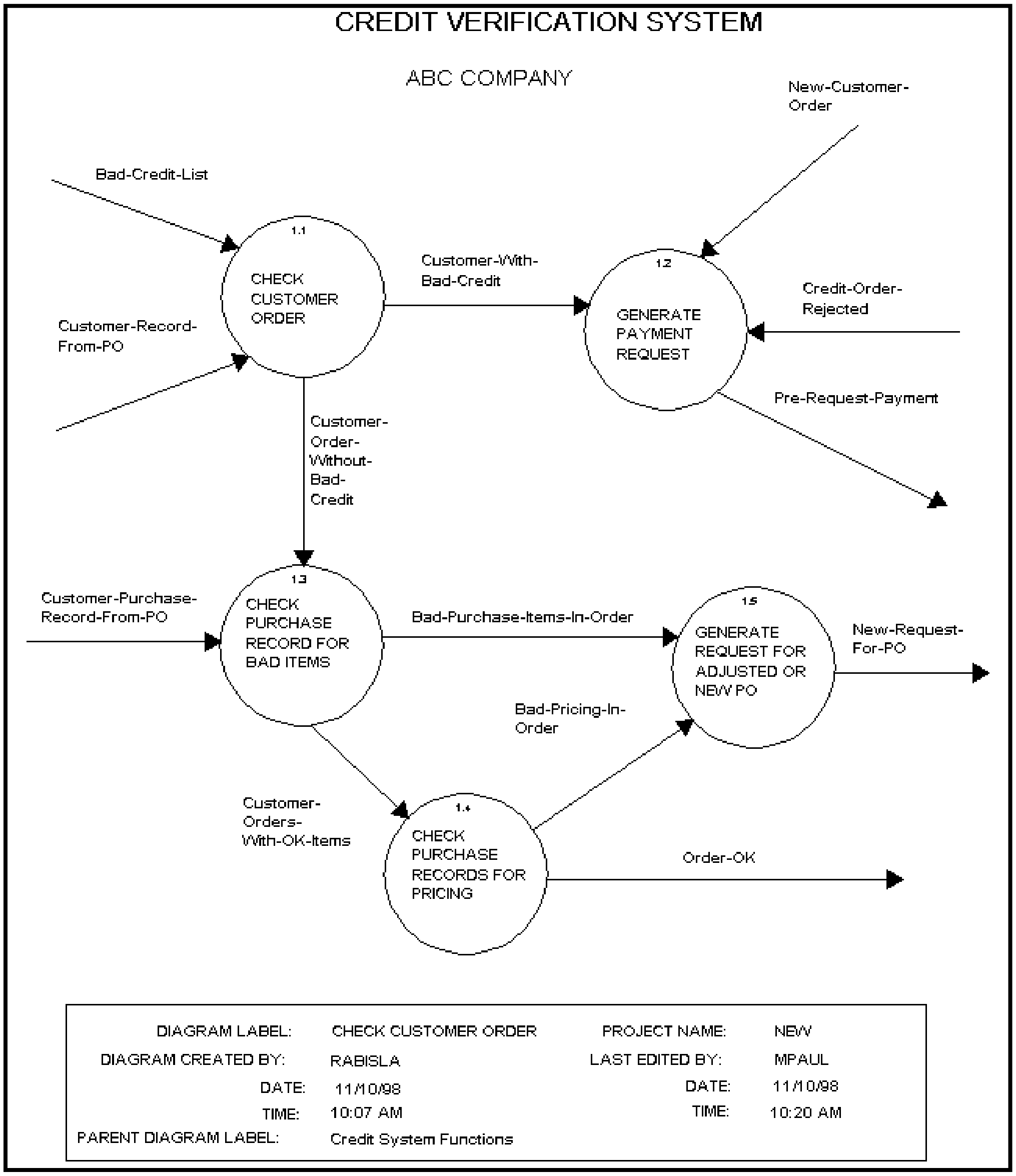

CREDIT VERIFICATION SYSTEM
ABC COMPANY
New-Customer-Order
Bad-Credit-List
1.1
CHECK CUSTOMER ORDER
Customer-With-Bad-Credit
1.2
GENERATE PAYMENT REQUEST
Credit-Order-Rejected
Customer-Record-From-PO
Pre-Request-Payment
Customer-Order-Without-Bad-Credit
1.3
CHECK PURCHASE RECORD FOR BAD ITEMS
Customer-Purchase-Record-From-PO
Bad-Purchase-Items-In-Order
1.5
GENERATE REQUEST FOR ADJUSTED OR NEW PO
New-Request-For-PO
Bad-Pricing-In-Order
Customer-Orders-With-OK-Items
1.4
CHECK PURCHASE RECORDS FOR PRICING
Order-OK

DIAGRAM LABEL:	CHECK CUSTOMER ORDER	PROJECT NAME:	NEW
DIAGRAM CREATED BY:	RABISLA	LAST EDITED BY:	MPAUL
DATE:	11/10/98	DATE:	11/10/98
TIME:	10:07 AM	TIME:	10:20 AM
PARENT DIAGRAM LABEL:	Credit System Functions		

Figure 3-4 Data Flow Diagram With Keywords in Use

Using Boilerplate Diagram Keywords

Keywords are available for use in boilerplate diagrams. If any of the keywords are present in a boilerplate that is loaded into a diagram, they are automatically replaced by information such as when the diagram was created, when it was last edited, etc. You may place the keywords anywhere within a boilerplate and they are automatically translated when the boilerplate is loaded into any diagram. They are also updated each time the diagram is accessed by Visible Analyst. All keywords must be placed on the diagrams as *separate* text entries in order to function properly. The following keywords may be used:

- $CDATE - Identifies the date that the diagram was created.
- $CTIME - Identifies the time that the diagram was created.
- $CUSER - Identifies the user who created the diagram.
- $DIAGRAM[1] - Identifies the diagram label.
- $EDATE - Identifies the date that the diagram was last edited.
- $ETIME - Identifies the time that the diagram was last edited.
- $EUSER - Identifies the user who last edited the diagram.
- $PARENT - Identifies the parent diagram label (data flow diagrams only).
- $PROJECT - Identifies the diagram project root.

More Information about Keywords

Boilerplate keywords may not be embedded within text; instead, they must be *separate* text entries. For example, the text entry “Last edit: $EDATE” does not produce the desired result. Instead, “Last edit:” and “$EDATE” must be separate text entries.

- The font that you use for the keyword is the font that is used for the text that replaces it.
- Keywords are case insensitive.
- A boilerplate can be added to existing diagrams by using Constructs. To do this:
  1. Create a new diagram with the desired boilerplate enabled.
  2. From the new diagram, save a construct that contains the entire boilerplate and *nothing else*.
  3. Load the construct into existing diagrams. The keywords for create date, time, and user take on the values of the diagram from which the construct is created, not the diagram to which they are added. All other keywords are updated appropriately.

[1] These keywords are continuously updated as necessary: $DIAGRAM, $EDATE, $ETIME, $EUSER, $PARENT, and $PROJECT.

A LOOK AT THE DOCUMENT WORKSPACE

Whenever you create a new diagram or edit an existing diagram, the diagram or application workspace is displayed as shown in Figure 3-5. The document workspace is the starting point for creating, editing, and reviewing all of your diagrams. The center of the document workspace provides the area into which you draw each diagram. When the first diagram is opened during a work session, it is opened maximized and fills the entire application workspace. You can also expand the application workspace to fill your entire screen. This makes a maximized diagram even bigger. You can also, however, shrink it to a window or minimize it to an icon, if you wish.

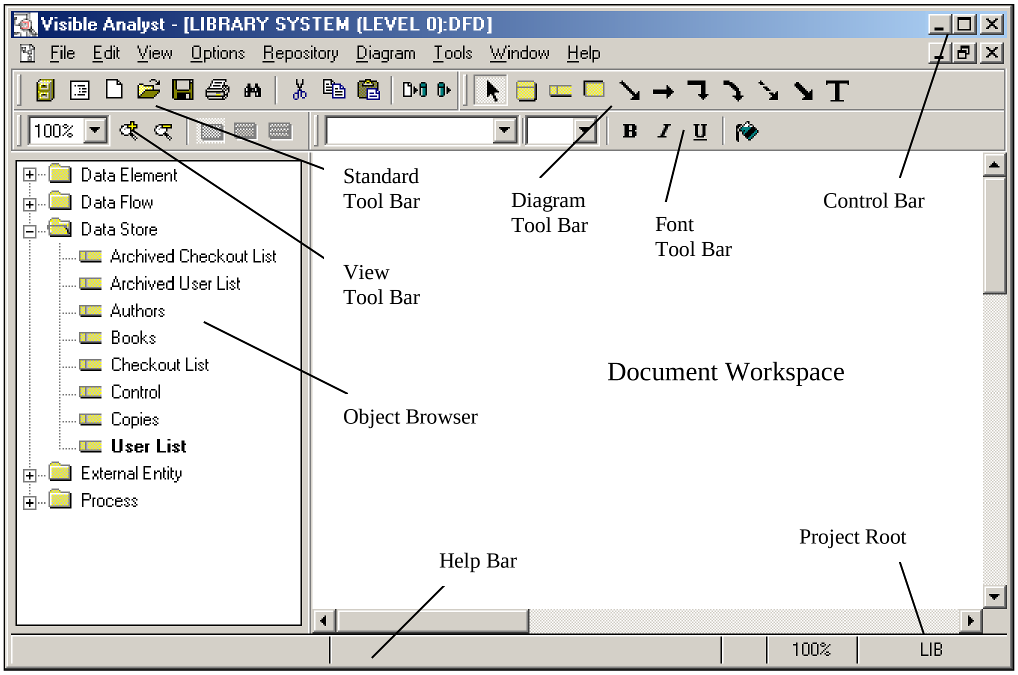


Figure 3-5 The Document Workspace

The Document Window

Document windows are sizable in the standard Windows manner. If you have a maximized document open and you open a second document of the same or a different diagram type, both are displayed as windows; neither is maximized. You can later choose to maximize either document. You can control how your individual windows appear and how your windows are arranged from the View and Window menus, as described below. Note that Visible Analyst appends the diagram type to the name of that diagram in the title bar of the document window.

The Help Bar

At the bottom of the workspace is a line that gives a brief description of the menu function currently highlighted; and displays the current zoom level, the current project, the current object, and whether the diagram was opened as Read Only (RO). The help bar can be toggled on and off from the Options menu.

The Object Browser

At the left side of the workspace is the object browser. The object browser displays a list of all the objects in the repository. It can be toggled on or off from the Options menu. When there are no documents open, or the current window is the diagram list, all objects are displayed. When a diagram is open, only those objects that are valid for that diagram type are displayed. If an object appears on the diagram, it is displayed in bold.

The Control Bar

At the top of the workspace is the control bar, explained later in this chapter. Control bar options are selected from Customize Control Bar dialog box displayed when you select Control Bar from the Options menu.

The View Menu

Zoom

Selecting Zoom from the View menu allows you to reduce the display size of a diagram, so that a larger portion may be viewed within the workspace. Select the zoom level (expressed in percentage of object sizes compared with those in the 100% level) you wish to use. All diagram editing functions can be used in the zoom mode, although the on-screen resolution of text entries in some font styles may deteriorate at the smaller display-size zoom levels. These and other View menu functions are illustrated in Figure 3-6.

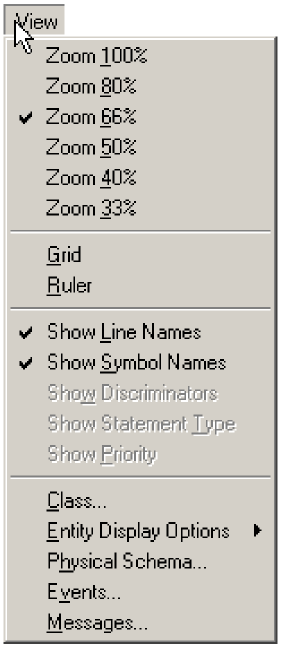


Figure 3-6 The View Menu

Grid and Ruler

A diagram can be displayed with ruler lines along the top and left sides of the window to aid you in the regular placement of items on the diagram. A grid can also be placed on the diagram. The two features can be used simultaneously. Neither prints on a diagram.

Symbols and Lines Without Labels

In order to be referenced in the repository, every methodology symbol or line (except invocation lines and data and control connections on structure charts) must carry a text label. However, there are times when a diagram structure is clearer with these labels absent. You can toggle the display of symbol and line labels on and off from the View menu. If you choose to print a diagram with labels turned off, the labels do not appear on the printed diagram.

Note

- Turning line or symbol labels off is *not* the same as not labeling them. A line or symbol that has never been labeled *does not exist* as far as the repository is concerned, except in the case of supertype/inheritance relationships.

Entity Attributes Displayed on Data Model Diagrams

When viewing a diagram in a data model, the level of detail can be set to show only the entity names, the entity name and attributes that are part of the primary key, or all attributes. When changing the detail level, entity symbols are automatically resized and the relationships are reconnected properly.

To use this feature:

Open the View Menu: 1 Open the View menu.

Select the Detail Level: 2 Select Entity Display Options, then select the level of detail you to see on the diagram and other entity options.

- Select Entity Level to show only the entity names.
- Select Primary Key Level to show the entity name as well as its primary key attributes.
- Select Attribute Level to show all attributes, key and non-key, of the entity.
- Select IDEF1X Notation to use IDEF1X modeling notation.
- Select Name Inside Box to display the entity name inside the entity symbol.
- Select Expand Associators to display the elements from the parent entity that make up the foreign key when displaying foreign keys on an entity relationship diagram.
- Selecting Colors on the Options menu allows you to assign specific colors for the Primary, Foreign and Alternate keys. These colors are displayed on the diagrams and when the diagrams are printed in color.

Physical Schema Displayed on Data Model Diagrams

When viewing a diagram in a data model, you can either show the logical model, or the physical model that is created when SQL Schema Generation is performed. The attributes displayed for each table are controlled by the current entity view level. When changing the detail level, entity symbols are automatically resized and the relationships are reconnected properly.

To use this feature:

Open the Dialog Box: 1 Open the View menu and select Physical Schema.

Select the Display 2 Select the display options to be used.

Options:

- Check Display Physical Schema if you want physical names to be displayed instead of logical names. Physical names are translated logical names that the target SQL RDBMS can understand. The translation process may change characters or truncate the names depending the limits of the current SQL dialect. When you select Display Physical Schema, the other options are available.
- Check Display Data Type if you want an attribute's data type to be shown. If the data type has not been determined or is not valid for the current SQL dialect, it is marked as undefined.
- Check Display Null Option if you want an attribute's null specification to be shown. If the attribute is part of a primary, alternate, or mandatory foreign key, it cannot be null.
- Check Expand Domains if you want attributes that reference a domain to display the data type of the domain. Otherwise, the domain name is displayed instead of the data type. If the current SQL dialect does not support domains (or user-defined data types), domains are always expanded.
- Check Use Alias Name if you want an attribute's alias name to be used in place of its actual name. You may want to do this if you have long logical names and your target RDBMS supports only short names. If more than one alias is defined, the first item in the list is used.

If the SQL dialect is changed, the physical attributes are reloaded based on the capabilities of the new dialect.

Class Attributes Displayed on Object Model Diagrams

When viewing a diagram in an object model, the level of detail can be set to show only the class names, the class name and attributes, or methods. When changing the detail level, class symbols are automatically resized and the relationships are reconnected properly.

To use this feature:

Open the Dialog Box: 1 Open the View menu and select Class**.**

Select the Notation: 2 Select the notation to be used for the class diagram.

Select the Detail Level: 3 Select the items that you want to see on the diagram.

- If you want attributes displayed, check the type of attributes to include. *Public* attributes have global visibility. *Private* attributes can only be used by member functions. *Protected* attributes are accessible to member functions of derived classes. In addition to the attribute name, you can choose to include its type, initial value, and visibility.
- If you want methods displayed, check the type of methods to include. *Public* methods have global visibility. *Private* methods can only be used by other members of the class. *Protected* methods are accessible to member functions of derived classes. In addition to the method name, you can choose to include its argument list, whether it is a virtual (abstract) function, and its visibility.
- If you want relationships displayed, check the relationship options to include.
- If you selected a Stereotype when creating the class, it will be displayed when you display attributes or methods.

Events

Select Events to display setting options for events.

- Select Display Guard Condition to display guard conditions on the diagram.
- Select Display Action Expression to display action expressions on the diagram.

Messages

Select Messages to display setting options for messages.

- Select Display Message Arguments to display message arguments on the diagram.
- Select Display Argument Types to display argument types on the diagram.
- Select Display Guard Condition to display guard conditions on the diagram.

Note

- Please refer to the Strategic Planning chapter for a description of the View menu when Strategic Planning is the open document.

Window Menu

Diagram Window Arranging

When you open more than one diagram, you can have them arranged either in cascade (each slightly below and to the right of the previous) or tiled (arranged and sized so that they are all visible and fill the workspace). (Refer to Figure 3-7.) You select the arrangement you want from the Window menu. You can also size and arrange them manually.

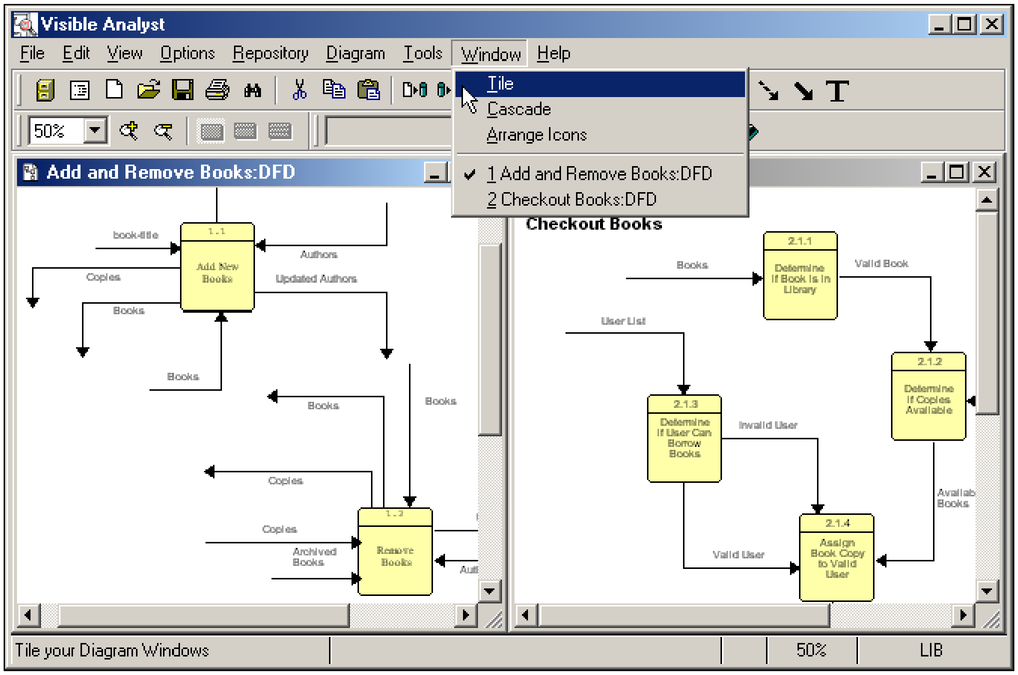


Figure 3-7 Window Functions Illustrated

For each diagram window, you can display it maximized or minimized to an icon. The Window menu lets you arrange your diagrams neatly in your application workspace as well.

Move Between Diagrams

To go from one open diagram to another, you can move the cursor to within another diagram window and click. You can also select the name of the diagram you want to move to from the Window menu. This latter method works even if a diagram is minimized to an icon. The currently active diagram name is marked with a check.

DRAWING DIAGRAMS

The process of drawing a diagram consists of entering symbols, lines, images and text and arranging them on the diagram. These processes are explained in this section.

Diagram Menu

By selecting appropriate functions from the Diagram menu (Figure 3-8), you can draw objects (such as symbols and lines) into a new diagram, choose editing functions (such as Stylize, Change Item), add a picture onto the diagram, analyze your diagrams, etc. Descriptions and procedures for most of the functions are provided in the following pages, and a summary of descriptions is included at the end of the chapter.

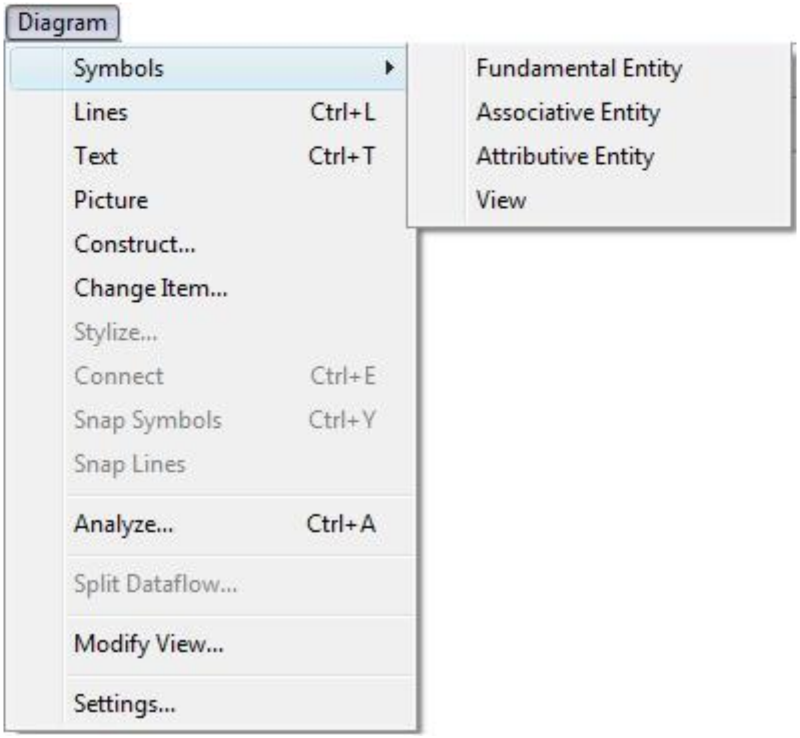


Figure 3-8 Diagram Menu and Symbols Submenu

Scrolling a Diagram

There are three ways to scroll a diagram; that is, to move the displayed area of the diagram within its window. You can use standard Windows scroll bars. You can also hold down the

CTRL key while pressing the right mouse button and bump the edge of the diagram in the direction that you want to scroll. In addition, you can use the arrow, PAGE UP, and PAGE DOWN keys or the mouse wheel to scroll a diagram window.

Drawing Modes

When you are entering or changing objects on a diagram, you can be in one of five different modes. They are editing mode, symbol entry mode, line entry mode, couple entry mode and text entry mode. Each mode has a different cursor to alert you to which mode you are in. There are several methods to change drawing mode, which is reflected in a change in the cursor. Those methods are explained in the sections that follow. The easiest way is to click on the object you want to add on the control bar.

Selecting Diagram Objects

Diagram editing in Visible Analyst is centered on the concept of the *current* or *selected* object. To make some change to or get information about an object on a diagram, click on it with the mouse. If the object is a line, click on one of its end points or handles. The object changes color. There are then a number of operations that you can perform on the object. The details of these operations are discussed in different areas of this manual. Clicking the left mouse button elsewhere on the diagram or choosing Clear from the Edit menu deselects the current object.

When a symbol or line is selected, the text label(s) it has is highlighted too. If you click on a selected object’s label, just the label is selected. If you double-click on an object, its repository entry appears for you to inspect or to make changes. If you click on an object with the *right* mouse button, or if you click the *right* mouse button on an object when it is already selected, the Object menu appears. On this menu are listed all of the operations you can perform on the object. You can select the task you want to perform from the Object menu. The selections available on the Object menu vary somewhat because different objects have different operations that can be performed upon them. An Object menu for a symbol is shown in Figure 3-9.

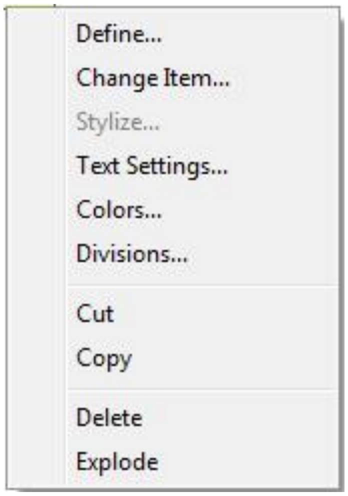


Figure 3-9 Object Menu for a Symbol

There is another way to select individual objects. If you press the TAB key while an object is selected, each of its labels is selected in turn. If you keep pressing TAB, other objects on the diagram are selected in their turn.

Selecting Blocks

You can also select a block. To do this, you must draw a rectangle that encloses all of the objects you want in the block. Point the cursor at one corner of the area you want to enclose and hold down the left mouse button. A dashed rectangle draws as you move the cursor. When the bounding rectangle encloses all of the objects you want in the block, release the button. Everything *completely* within the block is selected.

There is another, more detailed way to select items into a block. If you press and hold the SHIFT key while you click on diagram objects, they become parts of the selected block. If you change your mind, SHIFT-click an item selected into the block to deselect it. If two symbols are selected into a block, any line connecting them is automatically selected into the block. (If you don’t want the connecting line in the block, SHIFT-click on either end of it to deselect it.) Note that you can use the SHIFT-click method to remove objects within a block created by the method described in the previous paragraph.

Note

- It is important to be aware of what mode Visible Analyst is in. An object becomes the selected current object either when it is first added to a diagram or when you click on it when Visible Analyst is in *editing mode* (when the editing arrow cursor is displayed). When the item has been selected in either of these manners, clicking the right mouse button brings up the Object menu. If you click on an object when Visible Analyst displays the symbol, line, couple or caption text cursor, a symbol, line, couple or free-form caption is entered (or attempted) at that position on the diagram.

Moving Objects on a Diagram

You can move a selected object, including a block, to a new position on the diagram by pointing to it, holding down the left button and dragging it to where you want it to be. When you release the mouse button, the object redraws in its new position. If the object is a symbol and Include Connections was enabled from the Options menu, when you drag the symbol to its new diagram position, all lines connected to the symbol remain connected and stretch or "rubber band" to accommodate the move. Note that, in the case of multi-segment lines, only the last segment of the line rubber bands with the object being moved. If the resulting position of the line is not pleasing to you, you can adjust it as described in the next section. If you move a symbol and its attached lines in such a way that the line would have to retrace its steps (draw over itself) in order to adjust itself to its new position, Visible Analyst redraws the line more suitably, possibly eliminating one or more segments. (Refer to Figure 3-10.)

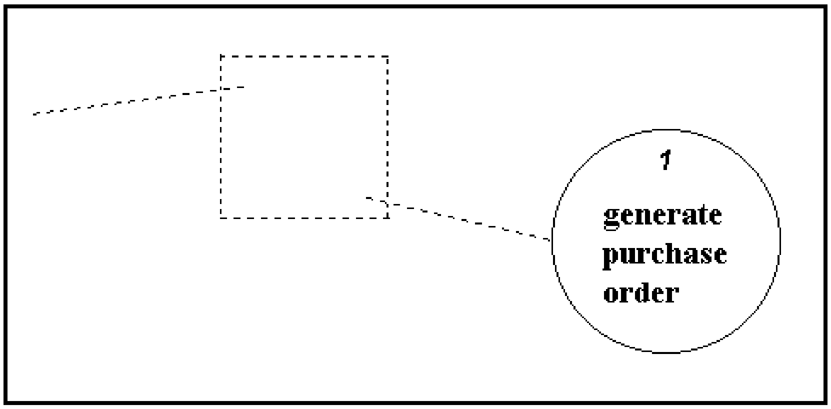


Figure 3-10 Moving a Symbol with Attached Data Flows

Moving a block is similar to moving an individual symbol. If Include Connections is enabled, lines attached both to objects *within* the block and to objects *outside* the block rubber band and remain connected.

If you move a line to connect it to a different symbol, you should move its endpoint (handle) to the border of the other symbol. If Auto Connect from the Options menu is selected, you can drag the endpoint of the line to anywhere within the other symbol and Visible Analyst attaches the line to the border of that symbol.

Notes

- Please remember the following:
  - Any text associated with symbols or lines moves automatically if the symbol or line is moved. Also, once a label has been created for these objects, it may be moved independently of the object without affecting the label/object relationship.
  - Items not *completely* enclosed by a block are not moved.
  - If you press ESC *before* you release the mouse button while dragging an object, the move is canceled and the object snaps back to its *original* position. Use the Undo operation to cancel a moved line *after* the mouse button is released, or select Erase Changes from the File menu.
  - Moving a symbol by itself without enabling Include Connections may detach it from attached lines. Detaching a relationship from an entity deletes the text on the relationship, since a relationship has no meaning without two entities; the relationship line on the diagram is not deleted. This means that the relationship entry in the repository is deleted if there is no descriptive information and if it doesn't exist on another view. Setting Include Connections does not cause this detachment. However, if you move a relationship so it connects a *different* pair of entities, the name(s) remains as long as there is no previously existing relationship with the same name(s) between the second pair of entities.
  - While moving a line, you can add or remove segments, or change its orientation by using shortcut keys. Refer to Adding or Changing Lines later in this chapter for details.

Line Handles

Whenever you add a line to a diagram or whenever you select a line, you see small colored boxes at each end of the line and at the end of every segment of the line. These are handles (refer to Figure 3-11). You can select a line by clicking on one of its end points or handles, even if the handle is not visible on the screen. You can point your mouse at any one of these handles, hold down the left mouse button and drag the handle to a new position on the diagram. When you release the button, the segment(s) involved redraws at the position you selected.

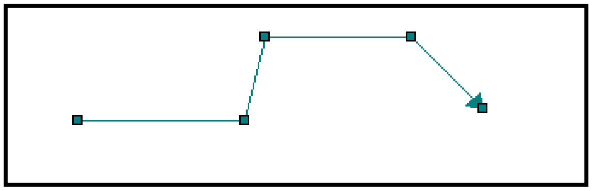


Figure 3-11 Line with its Handles

Drawing Symbols

The following sections describe the symbols used for diagramming and the manner in which they are placed into position on your diagrams.

The Symbols

The symbols that you use to create all of your diagrams are stored in Visible Analyst. Thirteen sets of symbols are provided: one each for DFDs, FDDs, ERDs, structure charts, class diagrams, state transition diagrams, business process model, ELH diagrams, use case, sequence, collaboration, activity, and unstructured diagrams. You can create new diagram template symbol sets for use on unstructured diagrams using picture files as the symbols. Refer to Using Picture Files later in this chapter for details. By selecting various symbols, then adding lines and text as described in the sections that follow, you have the versatility to build many types of diagrams that reflect your specific project analysis and design needs.

Depending on your configuration, Visible Analyst is shipped with one or more of the following toolsets:

- The three standard symbols each for the Yourdon/DeMarco data flow diagramming methodologies, as well as the symbols for functional decomposition diagrams and the standard Yourdon/Constantine symbols for structure charts.
- The set of symbols for ERDs.
- The set of symbols for CLDs and STDs.
- The set of symbols for ELHs.
- The set of symbols for activity diagrams.
- The set of symbols for collaboration diagrams.
- The set of symbols for sequence diagrams.
- The set of symbols for use case diagrams.
- The set of symbols for BPMN diagrams.

In addition, some commonly used diagramming symbols are also included with the DFD and unstructured symbol sets, as shown in Figure 3-13. The methodology symbols for each diagram type are accessed from the Diagram menu or from the control bar. The symbols available from each of these sources vary as the diagram type of the current diagram changes.

Symbol Restrictions when Applying Rules

You should be aware that rules are applied only to certain symbols for each diagram type, as described in the following paragraphs.

Functional Decomposition Methodology Symbols

There are three symbols used under the rules for functional decomposition diagrams, as shown in Figure 3-12.

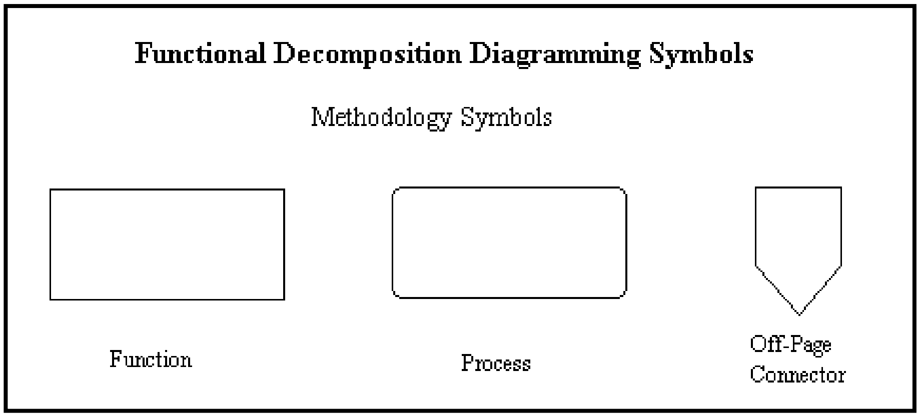


Figure 3-12 Functional Decomposition Diagramming Symbols

Functions

A business function is denoted by a rectangle. It is the first symbol for this diagram on the control bar.

Processes

A process is denoted by a rectangle with rounded corners. The conceptual dividing point between functions and processes is arbitrary and entirely up to you.

Page Connectors

A functional decomposition is viewed by Visible Analyst as one unified diagram. You can, however, spread your FDD over multiple pages. The process for linking pages with Page is very similar to that for DFDs. Unlike DFDs, however, the rules Analyze function recognizes page connectors and can transparently apply the methodology across them. The page connector symbol is the third symbol on the control bar and is identical in appearance to the off-page connector for structure charts.

DFD Symbols

Symbols Recognized by Yourdon Methodology

When Yourdon methodology is enabled for a project, rules are applied only to the symbols available from the Diagram menu or the diagram tools tool bar, as shown in Figure 3-13. Figure 3-13 shows the standard Visible Analyst symbols for data flow diagramming.

The Yourdon rules function also assigns a process number to any process symbol when it is drawn. For more information on the numbering scheme that is applied by the Yourdon methodology, refer to The Visible Rules.

Symbols Recognized by Gane & Sarson Methodology

When Gane & Sarson methodology is enabled for a project, rules are applied only to the symbols available from the Diagram menu or from the diagram tools tool bar, as shown in Figure 3-13.

The Gane & Sarson rules function also assigns a number to a process or data store when drawn. For more information on the numbering scheme that is applied by Gane & Sarson methodology, refer to The Visible Rules.

Symbols Recognized by Métrica

When the Métrica methodology is enabled for a project, rules are applied only to the symbols available from the Diagram menu or from the diagram tools tool bar, as shown in Figure 3-13.

The Métrica rules function also assigns a process number or data store number to any process or data store symbol when it is drawn. If a data store or external entity is used more than once on a diagram, or a process is exploded, it is drawn slightly differently.

Symbols Recognized by the SSADM Methodology

When the SSADM methodology is enabled for a project, rules are applied only to the symbols available from the Diagram menu or from the diagram tools tool bar, as shown in Figure 3-13.

The SSADM rules function also assigns a process number or data store number to any process or data store symbol when it is drawn. If a data store or external entity is used more than once on a diagram, or a process is exploded, it is drawn slightly differently.

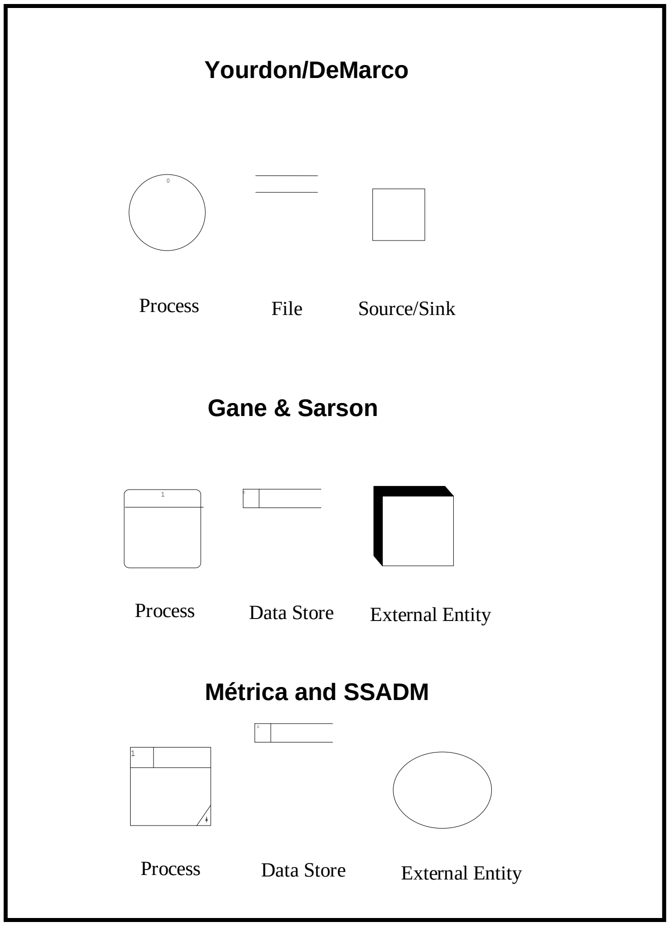


Figure 3-13 Data Flow Diagramming Symbols

Symbol Set for Structure Charts

There is a separate symbol set that contains the standard symbols for the Yourdon/Constantine method of structure chart diagramming, shown in Figure 3-14. This symbol set is automatically loaded whenever the diagram type is set for structure charts, or whenever a structure chart diagram is accessed. All of the symbols shown in Figure 3-14 are recognized by the Yourdon/Constantine rules as specific method symbols with associated identities and meanings.

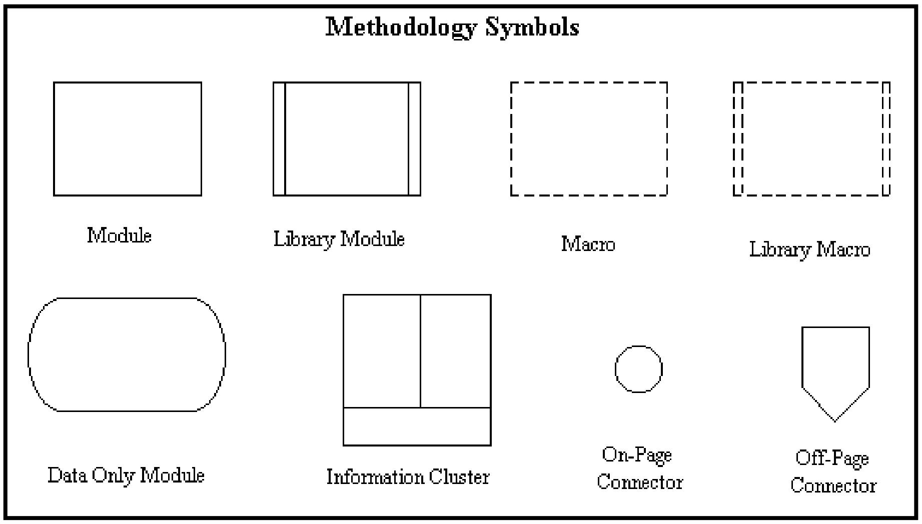


Figure 3-14 Structure Chart Diagramming Symbols

Drawing Information Clusters

The information cluster symbol in the structure chart symbol set is a compound symbol. Each can have a variable number of modules within it, as well as a shared data-only module. Place one on a diagram like any other symbol (see below); the decision as to how many modules it will contain is made in the labeling process.

ELH Symbols

There is a separate symbol set that contains the standard symbols for entity life history diagrams. The symbol set is automatically loaded whenever the diagram type is entity life history, or whenever an ELH is accessed. The symbols listed on the Symbols submenu or on the diagram tools tool bar are recognized by the SSADM and Métrica rules as specific methodology symbols with associated identities and meanings.

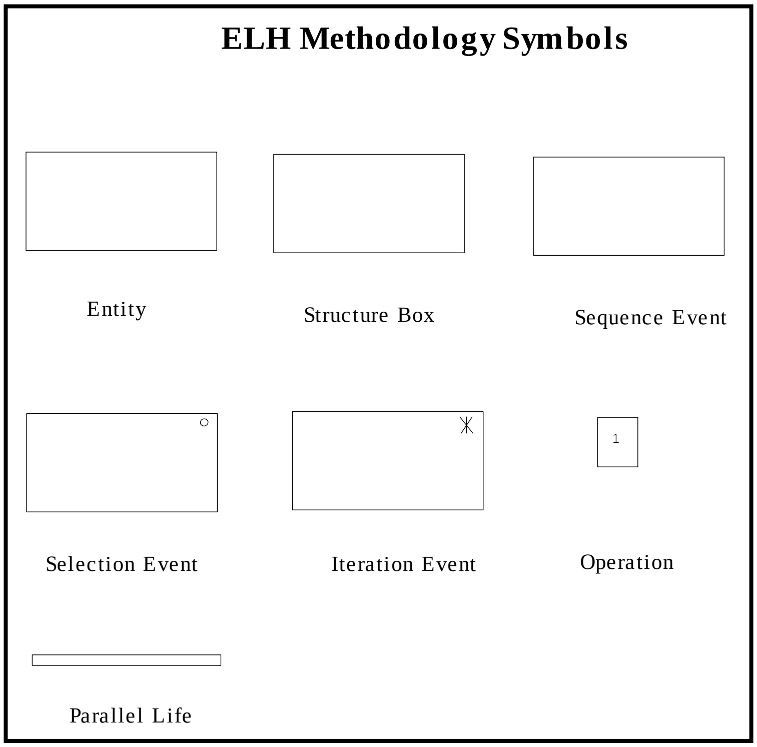


Figure 3-15 Entity Life History Diagramming Symbols

ERD Symbols

There is a separate symbol set that contains the standard symbols for data modeling (shown in Figure 3-16). This symbol set is automatically loaded whenever the diagram type is entity relationship, or whenever an ERD is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the data modeling rules as specific methodology symbols with associated identities and meanings.

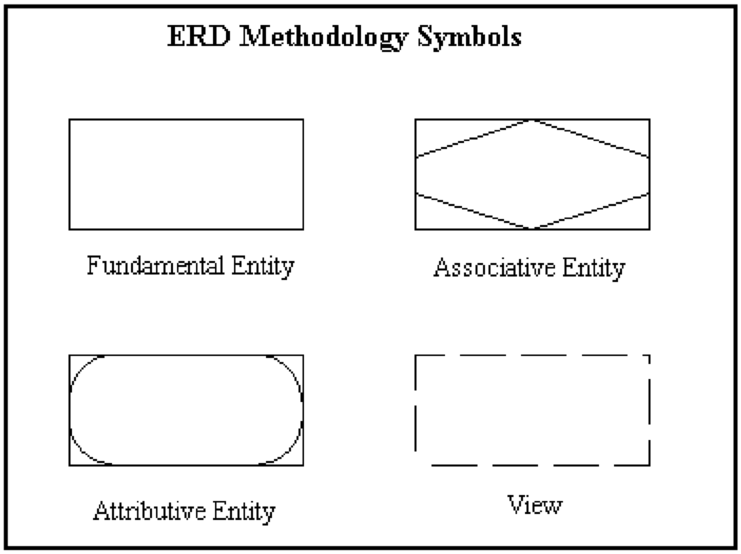


Figure 3-16 Entity Relationship Diagramming Symbols

Class Diagram Symbol

There is a separate symbol set that contains the standard symbols for object modeling (shown in Figure 3-17). This symbol set is automatically loaded whenever the diagram type is class or whenever a class diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the object modeling rules as specific methodology symbols with associated identities and meanings.

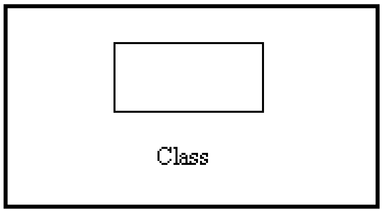


Figure 3-17 Class Diagram Symbol

State-Transition Diagram Symbol

There is a separate symbol set that contains the standard symbols for creating state transition diagrams (shown in Figure 3-18). This symbol set is automatically loaded whenever the diagram type is state-transition or whenever an state-transition diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the object modeling rules as a specific methodology symbols with associated identities and meanings.

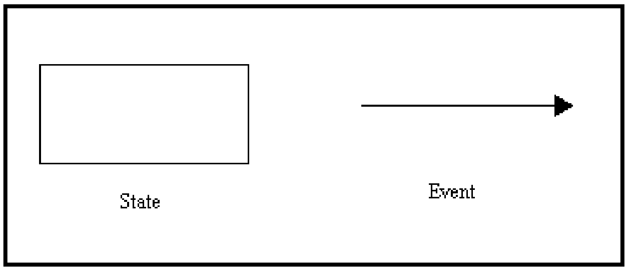


Figure 3-18 State Transition Diagram Symbols

Symbol Set for Activity Diagrams

There is a separate symbol set that contains the standard symbols for creating activity diagrams (shown in Figure 3-19). This symbol set is automatically loaded whenever the diagram type is activity or whenever an activity diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the object modeling rules as specific methodology symbols with associated identifies and meanings.

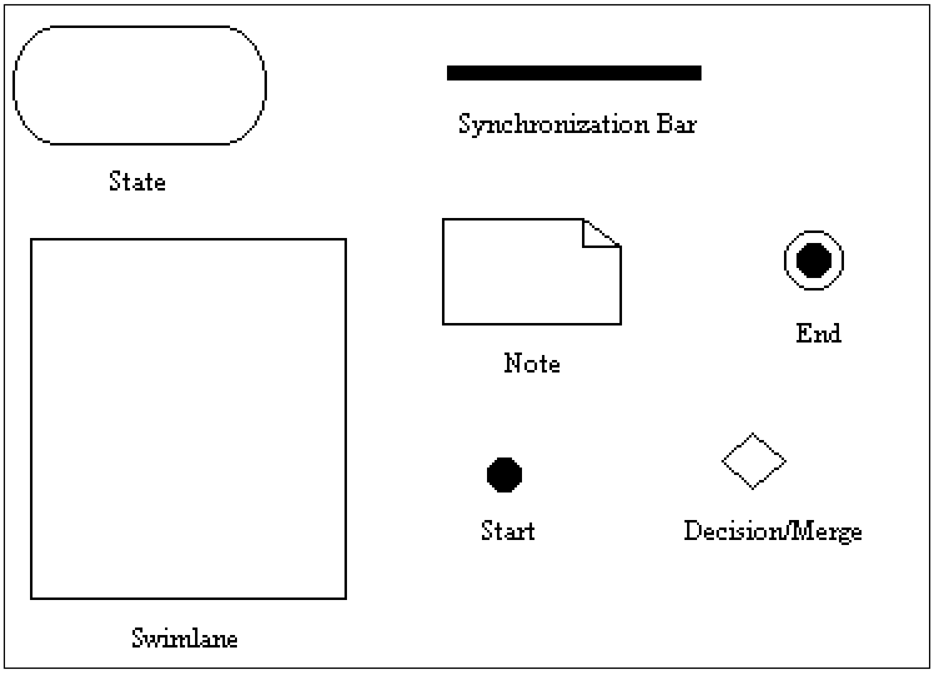


Figure 3-19 Activity Diagram Symbols

Symbol Set for Use-Case Diagrams

There is a separate symbol set that contains the standard symbols for creating use-case diagrams (shown in Figure 3-20). This symbol set is automatically loaded whenever the diagram type is use case or whenever a use-case diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the object modeling rules as specific methodology symbols with associated identities and meanings.

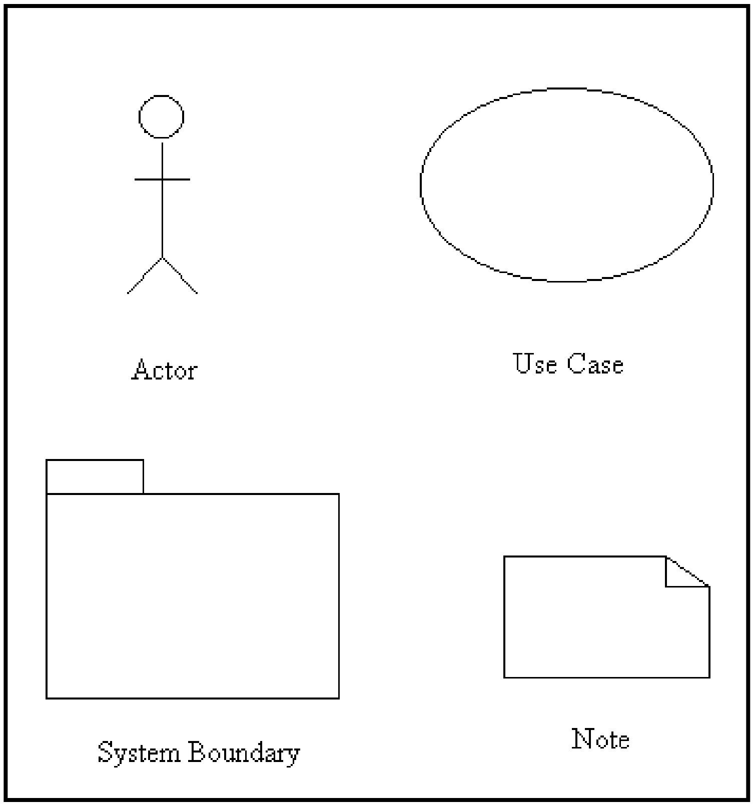


Figure 3-20 Use-Case Diagram Symbols

Symbol Set for Sequence Diagrams

There is a separate symbol set that contains the standard symbols for creating sequence diagrams (shown in Figure 3-21). This symbol set is automatically loaded whenever the diagram type is sequence or whenever a sequence diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the object modeling rules as specific methodology symbols with associated identifies and meanings.

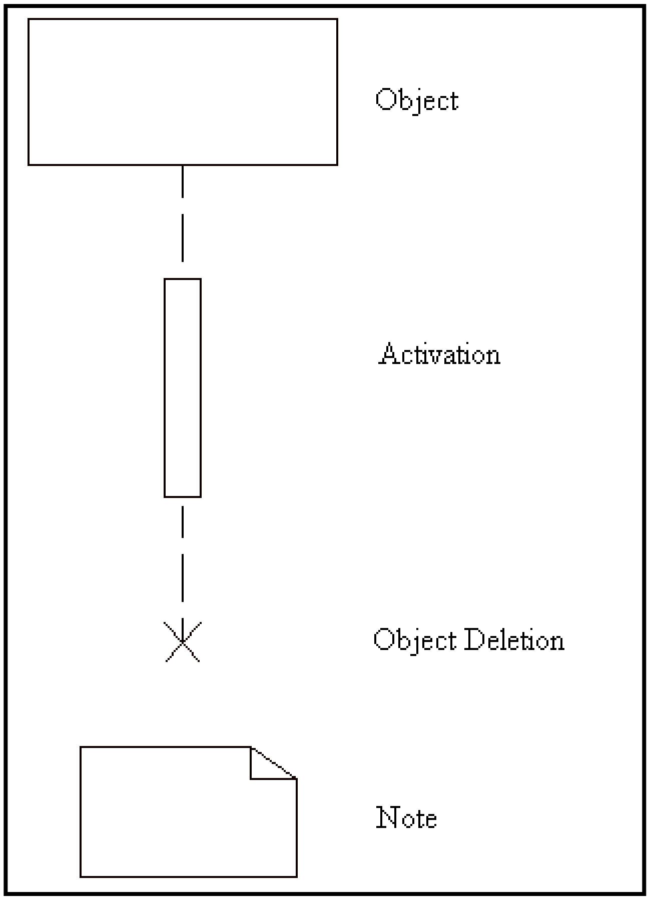


Figure 3-21 Sequence Diagram Symbols

Symbol Set for Collaboration Diagrams

There is a separate symbol set that contains the standard symbols for creating collaboration diagrams (shown in Figure 3-22). This symbol set is automatically loaded whenever the diagram type is collaboration or whenever a collaboration diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the object modeling rules as specific methodology symbols with associated identifies and meanings.

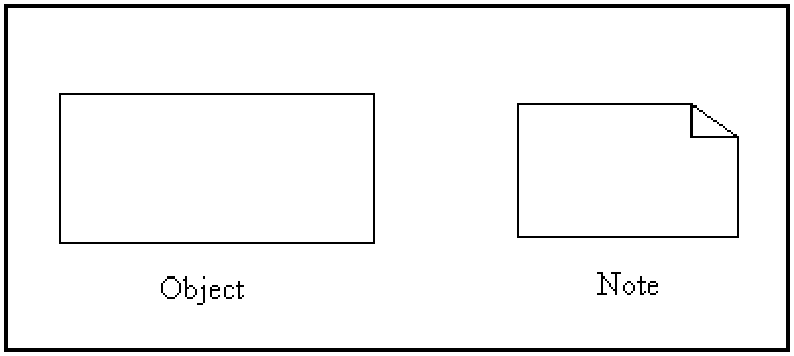


Figure 3-22 Collaboration Diagram Symbols

Symbol set for Business Process Modeling Notation (BPMN) Diagrams

There is a separate symbol set that contains the standard symbols for creating BPMN diagrams (shown in Figure 3-23). This symbol set is automatically loaded whenever the diagram type is Business Process Model or whenever a Business Process Model diagram is accessed. The symbols listed on the Symbols submenu or on the control bar are recognized by the BPMN rules as specific methodology symbols with associated identifies and meanings.

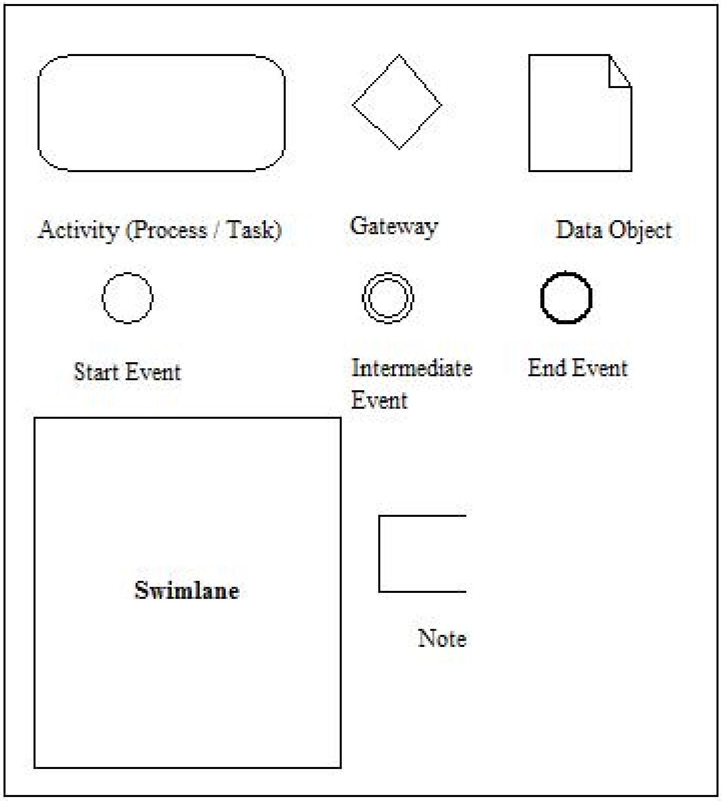


Figure 3-23 BPMN Diagram Symbols

Symbol Set for Unstructured Diagrams

There are a number of useful symbols that can be used on unstructured diagrams; and since there are no reserved or methodology symbols, you may eliminate or change any of them. They are similar to the non-methodology symbols in the DFD symbol set.

You can also create symbol templates using image files as the symbol set for use on unstructured diagrams. Refer to Using Picture Files later in this chapter for details.

Auto Labeling Symbols

Within the Options menu of Visible Analyst, there is a selection for Auto Label Symbols. The auto label function can be enabled to instruct Visible Analyst to prompt you for a label for each symbol you draw onto a diagram.

With the Auto Label Symbols function enabled, you still have the option to disregard each prompt if you just want to draw each symbol and return to it later for labeling. The auto label function can be enabled or disabled at any time before, during, or after work on a diagram. (There is also an Auto Label Lines selection that prompts you when drawing lines, described later in this chapter.)

Note

All methodology diagram symbols and line labels must begin with a letter. If you want an object to begin with a numeric value, edit the VAW.INI file and add this line to the file: Allow Numeric Prefix=Yes and then restart the Visible Analyst to enable the change. Do not use these special characters when naming a diagram object because they are reserved repository characters. ! @ # $ % ^ & * ;

Adding Symbols to a Diagram

There are three ways to add symbols to a diagram:

- Whenever a diagram is active, the Diagram menu (Figure 3-8) is enabled and you can select Symbols to add to your diagram. Visible Analyst displays a cascading submenu of the names of the methodology symbols for the diagram type of the currently active diagram.
- You can click on any of the buttons in the control bar that represent the available symbols. The symbol last entered is highlighted and ready to be used again, but you are free to choose another symbol.
- Whenever a diagram is active and the object browser is displayed, you can click on an object in the object browser list and drag it onto the diagram.

When you select symbol entry by one of the first two methods, the mouse cursor changes to the symbol cursor (see Figure 3-24). Move the symbol cursor to where you want the center of the symbol to appear and click; that symbol is drawn on the diagram. If you enabled Auto Label Symbols, a Label Object dialog box appears into which you can type the name of the symbol. If it is a process or a Gane & Sarson data store, its number also appears, and you can change it if you want. If you click the Search button on this dialog box, the names of all of the symbols of this type already in the repository are listed. You can find the one you want and select it to be the name of the symbol you just added.

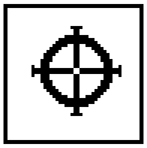


Figure 3-24 The Symbol Cursor

Note

- To return to the standard mouse cursor, you can either click the ⇖ button in the control bar, click the selected button to deselect it, press the ESC key or toggle the Symbols selection line on the Diagram menu.

When you select symbol entry by clicking and dragging from the object browser, the cursor changes to the object symbol and name. Move the object to where you want it on the chart and release the mouse button.

When you finish adding the symbol to the diagram, it is highlighted in a different color, indicating that it is the selected item. There are numerous things you can do with selected items. This subject is explained in the section entitled Selecting Diagram Objects, above.

Drawing Lines

Lines can be added to your diagrams to indicate project and process input/outputs, depict data flow between processes, to represent couples and connections between modules on structure charts, relationships between entities, etc. A wide variety of line terminators and line types are available for your selection. The line entry function is accessed by selecting Lines on the Diagram menu or you can click on a line button in the control bar.

Lines

You can draw lines thick or thin; solid, dashed, or channeled; straight, elbow or curved; with or without terminators; and select filled versus unfilled arrow terminators. You can draw your lines in *point-to-point* mode, where you use the mouse to position both the start and end points of the line, having the line drawn between the two points. Or you can draw straight lines in *snap* mode, where you point and select a starting point then an end point, having the straight line *snap* into position on the horizontal plane. Or you can draw lines containing only right angles in *elbow* mode where you point and select a starting point then an end point, having only horizontal or vertical straight lines. Except for the right angles, elbow lines work like arcs. Additionally, lines can have multiple segments that allow you to have start and end points distant from each other. The line terminators, such as the relationship cardinality terminators available for ERDs, are configurable at any time from the Line Settings selection from the Options menu

A number of BPMN sequence lines have additional line markers that indicate the flow type depending on the behavior or type of the connected symbol. See Figure 3-29.

ERD Line Terminators

Figure 3-25 explains the proper use of relationship line terminators to show the cardinality of relationships between entities.

Class Diagram Line Terminators

Figure 3-26 explains the proper use of relationship line terminators to show the cardinality of relationships between classes.

IDEF1X Diagram Line Terminators

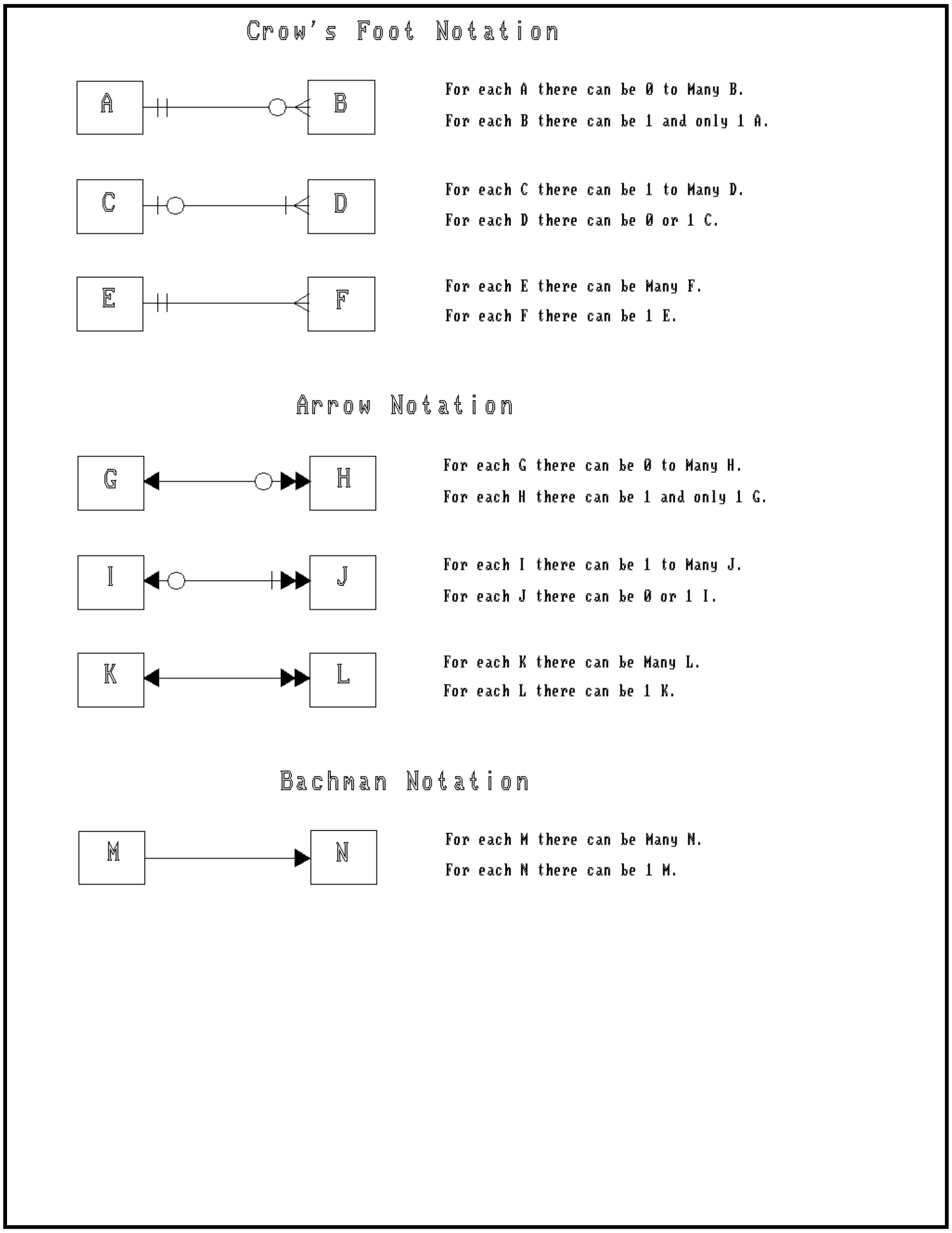
Figure 3-27 explains the proper use of relationship line terminators to show the cardinality of relationships between entities.

UML Diagram Line Terminators

Figure 3-28 explains the proper use of relationship line terminators to show the cardinality of relationships between entities.

BPMN Line Terminators

Figure 3-29 displays the different uses and types of BPMN sequence flow lines. See BPMN diagramming in Chapter 4 for an overview of sequence flow line usage.



Crow's Foot Notation

For each A there can be 0 to Many B.
For each B there can be 1 and only 1 A.

For each C there can be 1 to Many D.
For each D there can be 0 or 1 C.

For each E there can be Many F.
For each F there can be 1 E.

Arrow Notation

For each G there can be 0 to Many H.
For each H there can be 1 and only 1 G.

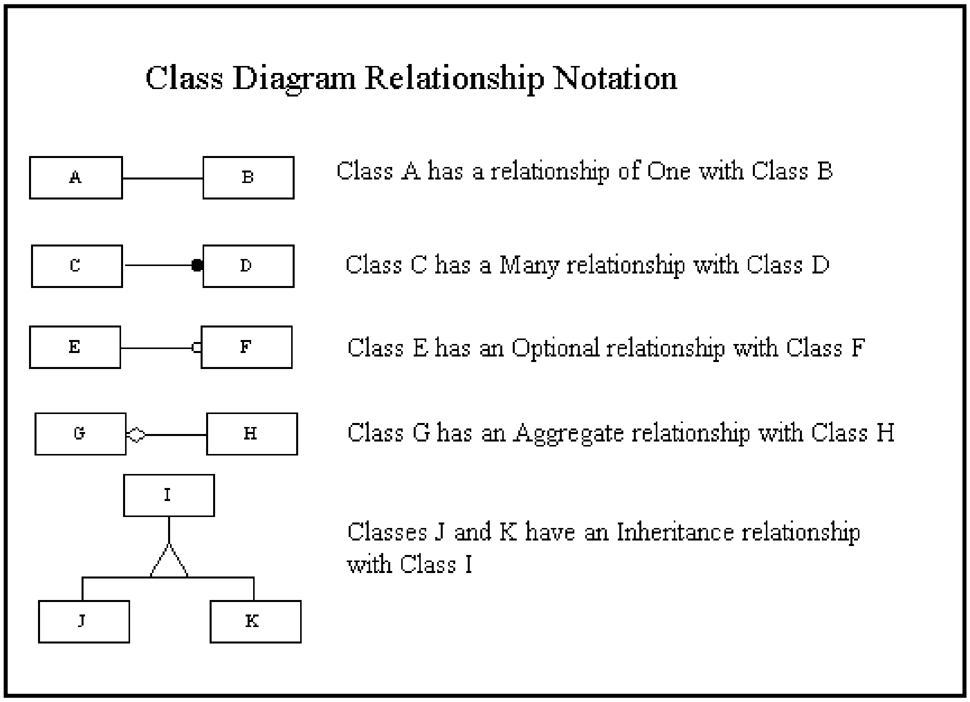
For each I there can be 1 to Many J.
For each J there can be 0 or 1 I.

For each K there can be Many L.
For each L there can be 1 K.

Bachman Notation

For each M there can be Many N.
For each N there can be 1 M.

Figure 3-25 ERD Relationship Line Terminators



Class Diagram Relationship Notation

Class A has a relationship of One with Class B

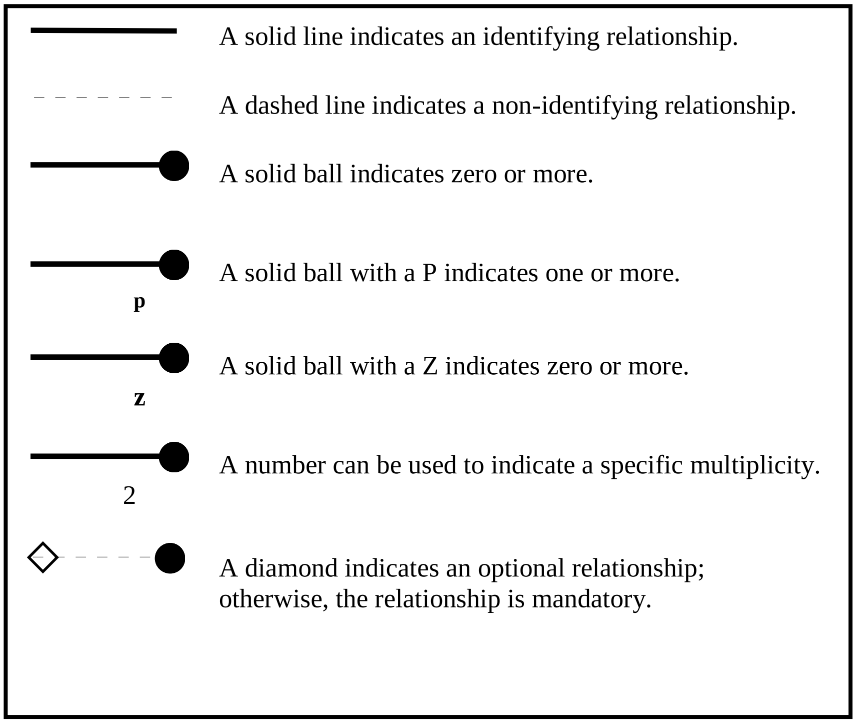
Class C has a Many relationship with Class D

Class E has an Optional relationship with Class F

Class G has an Aggregate relationship with Class H

Classes J and K have an Inheritance relationship with Class I

Figure 3-26 Class Relationship Notation



A solid line indicates an identifying relationship.

A dashed line indicates a non-identifying relationship.

A solid ball indicates zero or more.

A solid ball with a P indicates one or more.

A solid ball with a Z indicates zero or more.

A number can be used to indicate a specific multiplicity.

A diamond indicates an optional relationship;
otherwise, the relationship is mandatory.

Figure 3-27 IDEF1X Relationship Line Terminators

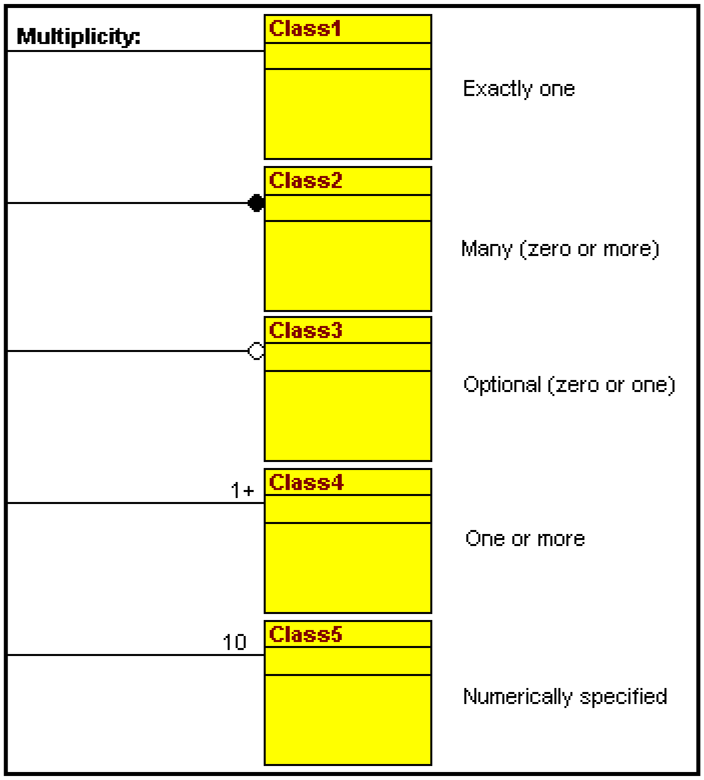


Figure 3-28 UML Relationship Notation

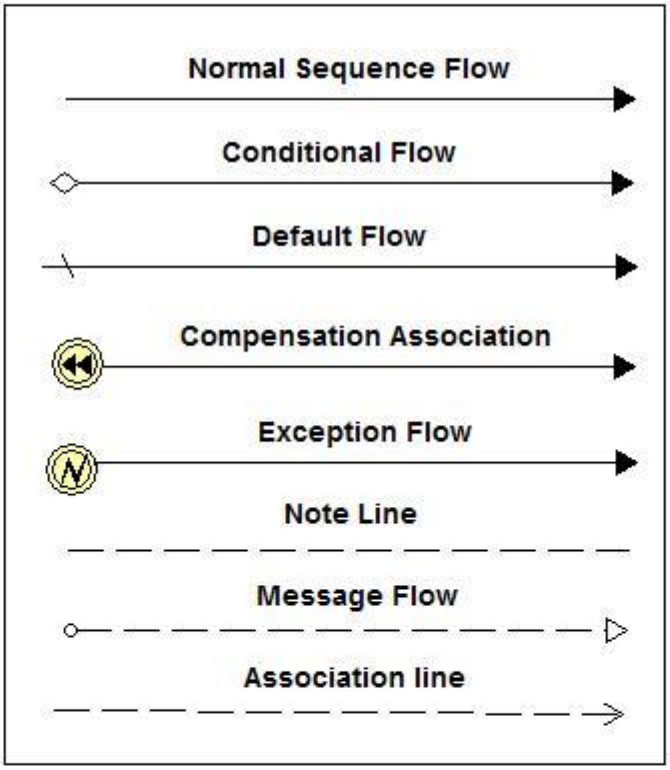


Figure 3-29 BPMN Sequence Flow Line Types

Auto Labeling Lines

Within the Options menu of Visible Analyst, there is a selection for Auto Label Lines. The auto label function can be enabled to instruct Visible Analyst to prompt you for a label for each line you draw onto a diagram. See the note on p.97 regarding the use of special characters and numbers when labeling lines or diagram objects.

With the Auto Label Lines function enabled, you still have the option to disregard each prompt if you just want to draw a line and return to it later for labeling. The auto label function can be enabled or disabled at any time before, during, or after work on a diagram. (There is also an Auto Label Symbols selection that prompts you when drawing symbols, as described elsewhere in this chapter.)

Adding or Changing Lines

There are three ways to add lines to a diagram:

- Whenever a diagram is displayed, the Diagram menu is enabled and you can select Lines to add a line to the diagram.
- You can click on any of the buttons on the control bar that represent the available line styles. The line style last entered is highlighted and ready to be entered, but you are free to choose another style.
- You can have Visible Analyst connect the symbols, using Connect from the Diagram menu, with the current line style and terminator(s). This same procedure works for connecting pairs of symbols on other types of diagrams.

When you select line entry by any of these methods, the cursor changes to the line cursor (see Figure 3-30). Move the line cursor to where you want the beginning of the line to be. Click and drag the cursor to where you want the end of the line and double-click. The line is drawn on the diagram. If you are adding a line to connect two symbols, you should place its start point and endpoint on the border of the respective symbols. If Auto Connect from the Options menu is selected, you can start and end the line anywhere within the two symbols and Visible Analyst attaches the line to the borders of those symbols.

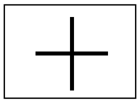


Figure 3-30 The Line Cursor

If you want a multi-segment line, simply click at the end of each segment and double-click at the end of the line. If the line style you are using is an arc or an elbow line, you might find that the loop of the arc or the angle of the elbow for a given segment is on the opposite side from where you want it. This is easy to correct. While you have the left mouse button depressed and before you click for the end of the segment or double-click for the end of the line, simply click the right mouse button; and the arc or elbow segment inverts. To invert an elbow or arc in an existing line, point to the line handle at the end of the segment, click and drag a little bit. While the left mouse button is still depressed, click the right button and the segment inverts.

If you enabled Auto Label Lines, a Label Object dialog box appears into which you can type the name of the symbol. If you click the Search button on this dialog box, the names of all of the lines for this diagram type already in the repository are listed. You can find the one you want and select it to be the name of the line you just added. If you click the Change Type button, you have the capability to change the line type, terminator type and line orientation of line you just drew. This brings up the same dialog box explained in the Default Line Selections section below. When you finish adding the line to the diagram, it is highlighted in a different color, indicating that it is the selected item. There are numerous things you can do with selected items. This subject is explained in the section entitled Selecting Diagram Objects, above.

Notes

- To return to the standard mouse cursor, you can either click on the ↖ button in the control bar, click the selected button to deselect it, press the ESC key or toggle the Lines selection line on the Diagram menu.
- When you draw a line and double-click to signal that you want the line to end at that point, you may sometimes move the mouse little. Because of the way Windows communicates with applications, it is possible that Visible Analyst will interpret your action as wanting to add another segment to the line you are drawing before terminating the line. To help prevent this, you can specify a minimum line length. If you find that this happens to you, increase the minimum line length setting. Thereafter, any motion of the mouse while double-clicking that is less than the minimum line length does not cause a new line segment to be drawn.

  To change this setting, you must edit the VAW.INI file in the directory containing the Visible Analyst executable files, usually \VA, with an ASCII text editor. (If you have a network version of Visible Analyst, the file name has your network user number appended to it. For example, if your user number is 11, the file is named VAW0011.INI. You can find out your user number by looking at the Help/About dialog box.) There is a line in this file that, by default, reads “Minimum Line=10”. The units are internal Visible Analyst length units. Changing this value to a higher number, say 20, should reduce the frequency of this situation occurring.

Changing the Number or Orientation of Line Segments

You can add, remove, or straighten segments on existing diagram lines. To remove a segment from an existing line:

Set Selection Mode: 1 Be sure you have established the standard mouse cursor for selecting diagram objects. You can either click the ↖ button in the control bar, click on the selected button to deselect it, press the ESC key or toggle the checked selection line on Symbols, Lines or Text on the Diagram menu.

Select the Line: 2 Select the line from which you want to remove a segment or to which you want to add one. The line handles of the line are displayed, one each at the beginning and end of each segment.

Select the Segment: 3 To delete, point the cursor at the handle at the beginning of the segment you wish to delete.

To insert, point the cursor at the handle at the beginning of the segment before which you wish to insert a new segment.

To straighten, point the cursor at the handle at the beginning of the segment that you wish to straighten.

4 Press the left mouse button and start to drag the handle, as if you want to move its position. Drag it just enough so that the line changes to the dashed line used to show a line being moved. Don’t release the mouse button yet.

Delete, Insert, or Straighten Segment:

5 To delete, press the DELETE key and that segment vanishes, with a new segment being drawn from the cursor position to the end of the following segment

To insert, press the INSERT key and a new segment is inserted at that point, but you won’t be able to see it because it has zero length. If you drag the mouse cursor, the new segment shows itself.

To straighten, press the S key and the line segment is snapped either horizontal or vertical depending on the plane the endpoint is closest to.

Position the Segment:

6 Release the mouse button when the new segment is positioned as you wish.

Adding a Couple to a Structure Chart Diagram

Adding a couple to a diagram is done a little differently. There are two ways to add couples:

- First, set the default line type to couple. Do this with the Line Settings function from Options menu. Then select Lines from the Diagram menu. When you move the mouse cursor over the drawing area of the diagram, it changes into the couple cursor shown in Figure 3-31.
- You can click on any of the buttons on the control bar that represent the available couple styles. The couple style last entered is highlighted and ready to be entered, but you are free to choose another style.
- Then select an invocation line. Now click on the end of the invocation from which you want the couple passed and the couple is drawn.

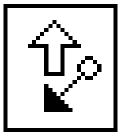


Figure 3-31 The Couple Cursor

If you have Auto Label Lines turned on in the Options menu a Label Object dialog box opens and allows you to enter the text for the couple label. That text is added in the font you chose in the Text Settings function from the Options menu. If you click the Search button in the dialog box, the repository search feature is activated and you can choose from the names of couples of this type currently in the repository. At that time, you are also able to change the type of the couple.

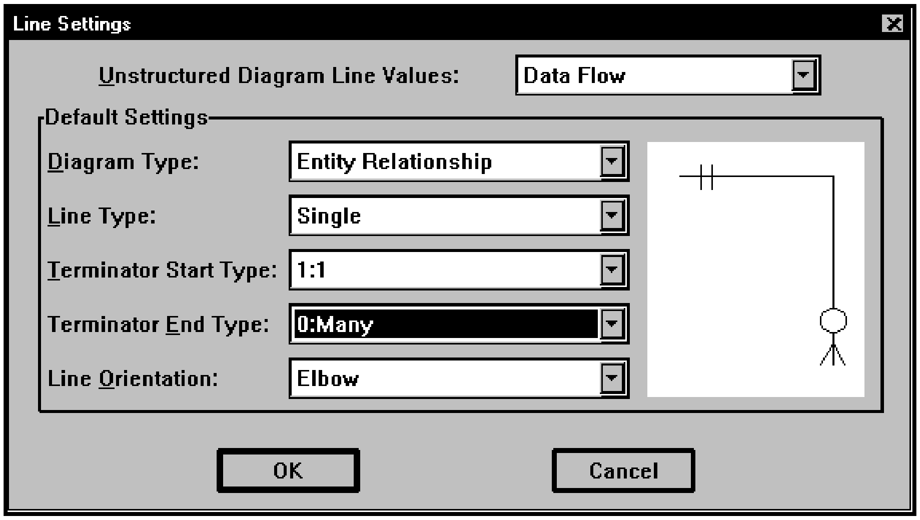
After you finish adding a couple, it remains highlighted as the selected object and the couple handles are displayed.

Note

- The cursor remains a couple cursor until you leave the couple entry mode by either clicking on the ⬉ button in the control bar, clicking the selected button to deselect it, pressing the ESC key, or toggling the Diagram menu selection off. If you don’t turn off the couple cursor in this manner, Visible Analyst starts a couple at every invocation line you select.

Default Line Selections

If you use the same type of line and terminator often in your diagrams, you can save time by using those selections as defaults every time you draw a line. You can define defaults for line type (straight line, dashed line, arc, etc.), terminator type (solid arrowhead, open arrowhead, no terminator, etc.) and line orientation (point-to-point, elbow) for each diagram type by selecting Line Settings from the Options menu. The selections available for line and terminator types vary with the diagram type. The dialog box displays a copy of a line drawn with all of the selections you choose, and the defaults are saved until you change them, as shown in Figure 3-32. Note that not all combinations of these line style settings are valid. An error message displays when you pick an invalid combination. You do not have to use the default selections for all lines on a diagram. You can draw it in the default style and change it, as described above.



Line Settings

Unstructured Diagram Line Values: Data Flow

Default Settings

Diagram Type: Entity Relationship

Line Type: Single

Terminator Start Type: 1:1

Terminator End Type: 0:Many

Line Orientation: Elbow

Figure 3-32 Line Settings Dialog Box

This dialog box is essentially the same as the one displayed when you want to change the style of an existing line. Here there is an additional selection box, the Unstructured Diagram Default Values box. You can choose the line and terminator values that are accessible when the diagram type of the active diagram is Unstructured. They can be made the same as the lines available for any other diagram type. However, the unstructured values set for a particular diagram are *those in effect at the time the diagram was created*. Note that when you display this dialog box from an unstructured diagram, the Diagram Type box does not read Unstructured, but rather the default type you set in the Unstructured Diagram Default Values box.

Bi-directional Couples and Data Flows

Bi-directional couples and data flows can be added to your structure charts as any other couple is. Set the default line type to a data flow or couple with two terminators, or change the line to one of these types after it is drawn.

Drawing Relationships

Relationships are drawn as any other line, but labeling them is a little different. This is explained in the section Labeling a Relationship, below.

FDD Connection Lines

The lines between functions, between processes, and from functions to processes are called “connectors.” They allow you to specify the hierarchical relationships between the functional elements of your model. The higher-order symbol can be referred to as a “parent” and each of the lower-order symbols connected can be called a “child” of the parent.

The connectors between functions are purely diagrammatic in that they don’t appear in the project repository. You can change them at will with no ramifications on other parts of your model. The connectors between processes represent the same type of parent/child relationships as exist on data flow diagrams. Processes with the same parent would appear on the same DFD. Connectors between a function and one or more processes are significant at the time you Spawn that function; the child processes of a function are placed on the top diagram of the process model branch spawned by the FDD, and so forth.

You can change any of these connectors at any time. However, if you change the hierarchical relationships between functions and processes, and DFDs including these processes exist, the two diagram types are then out of sync. You should correct this with the Spawn Verify function.

Drawing Connectors

The lines available for FDDs are the same as for DFDs. You can use any line style and any terminator you want to make the connections necessary to show the function/process hierarchy in your model. However, since “elbow lines” are commonly used in this type of methodology, there is a special line drawing technique for making branched elbow line connectors on FDDs. This procedure is as follows:

Block Method

To connect a function or process with subsidiary functions or processes, place the parent and child symbols in a block. (Selecting symbols with the SHIFT-click method is probably the most efficient way.) Then select Connect from the Diagram menu. Branched, elbow connectors are drawn from the parent to the child symbols.

Individual Method

- For a function being broken down into several lower-level functions or processes, draw a two-segment elbow line in the usual manner.
- For the second (and subsequent) connector, indicate the start point by pointing at or very close to the beginning of the previous connector.
- Indicate the start of the second segment by pointing to a point either along or on an extension of the second section of the previous line.
- Point to the edge of the second symbol to indicate the endpoint of the connector.

Visible Analyst superimposes the lines for as much of the length as possible, making nice, neat, branched, hierarchical connectors. The same technique can be extended for connectors with a greater number of segments. This drawing technique works for elbow lines on diagrams of all types, not just on FDDs.

Unlike data flows, FDD connector lines are not designed to be labeled. If it makes your diagram clearer to you, you can label them using the Text function from the Diagram menu. The labels are for your own information only and are ignored by the Visible rules and repository.

Entering Text

Visible Analyst allows (and encourages) you to label the objects drawn on your diagrams. An object without a label does not exist, as far as the repository for the project is concerned. For example, you can label symbols and data flows, but not invocation lines. In addition, you can enter notes or captions in paragraph form. Finally, you can edit existing text. To allow easier use of Visible Analyst with non-English languages, the entire ANSI character set can be used for text, except for characters with a value of 221-222, which are used for special purposes by Visible Analyst.

Entering Text on a Diagram

You can select Text from the Diagram menu to access the text entry function. Adding or changing labels on screen objects is done somewhat differently from caption text and is discussed later.

There are two ways to add free form caption text to a diagram:

- Whenever a diagram is displayed, the Diagram menu is enabled and you can select Text to add text to the diagram.
- You can click on the large T (text) button on the control bar.

When you select text entry by either of these methods, the cursor changes to the text cursor (see Figure 3-33). Move the text cursor to where you want the upper left of the caption to appear and click; a dialog box displays into which you can type your text. When you click OK, the text is drawn on the diagram. When you finish adding caption text to the diagram, it is highlighted in a different color, indicating that it is the selected item. There are numerous things you can do with selected items. This subject is explained in the section entitled Selecting Diagram Objects, above.

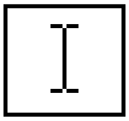


Figure 3-33 The Text Cursor

Note

- To return to the standard mouse cursor, you can either click on the ↖ button in the control bar, click on the selected button to deselect it, press the ESC key or toggle the Text selection line on the Diagram menu.

Labeling Diagram Objects

The procedure for labeling an object (symbol or line) on a diagram is simple:

- Select the object with the mouse. If you just placed the object on the diagram, it is already selected as the current object.
- Select Change Item from the Diagram menu to get the dialog box used for entering labels.
- Fill in the label field(s) in the dialog box.
- Click OK when you are done, and the object is labeled and entered into the repository.

See the note on p.97 regarding the use of special characters and numbers when labeling diagram objects.

When labeling an object on a sequence or collaboration diagram, in addition to the object name, a class must be specified. The object name is optional; however, the class is mandatory. Choose the class from the drop-down list; or you can enter a new class name if you have configured the tool to operate in this manner.

When labeling a class object you can also select a Stereotype for the class. The Stereotype will be displayed when the attributes or methods for a class are displayed.

There are variations on this procedure. You can click the right mouse button on the object, thus selecting it and displaying the Object menu. Then select Change Item and continue as above. You can click the right mouse button on an object already highlighted as the current object; this also displays the Object menu. The procedure varies slightly for some specialized objects, as described immediately below. Note that for processes and Gane & Sarson data stores, process and data store numbers become part of the labeling process. These are assigned by Visible Analyst, but you can change process numbers when labeling the symbol or later. As described under symbol and line entry, you can click the Search button in the dialog box and select the names of existing objects from the repository.

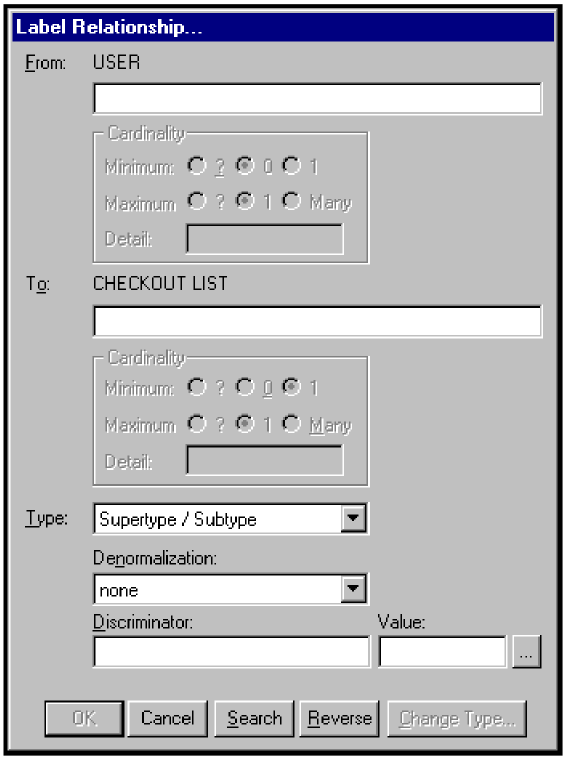
When a image is used as a methodology symbol, you are prompted to label the symbol. Non-methodology images can be labeled as described in the above paragraph. Refer to Using Picture Files later in this chapter for additional information.

Auto Label

The Options menu includes selections for Auto Label Symbols and Auto Label Lines. These automatic labeling functions can be enabled to instruct Visible Analyst to prompt you for a label as each object is drawn on a diagram. These functions are described in their own sections earlier in this manual.

Labeling a Relationship

Relationships must be independently labeled in each direction, unless you have specified that only one label is used for a relationship when you created the project. If Auto Label Lines is enabled, after you draw the line you see a dialog box in which you can enter the names of the relationships and their cardinality, the relationship type, and either denormalization information for ERDs or role/qualifier names for class diagrams. If Auto Label Lines was not enabled when you added the relationship to the diagram, or you chose not to label it at that time, you can label it as described above by selecting Change Item from either the Diagram or Object menu and completing the dialog box. With this dialog box you can also change the cardinality of an existing relationship. As with other symbols, you can click the Search button in the dialog box and pick the names of existing relationships from the repository. This can be done for each relationship to be labeled by putting the cursor in each relationship name box in turn and clicking the Search button.



Label Relationship...

From: USER

Cardinality
Minimum: ○ ? ● 0 ○ 1
Maximum: ○ ? ● 1 ○ Many
Detail:

To: CHECKOUT LIST

Cardinality
Minimum: ○ ? ○ 0 ● 1
Maximum: ○ ? ● 1 ○ Many
Detail:

Type: Supertype / Subtype

Denormalization: none

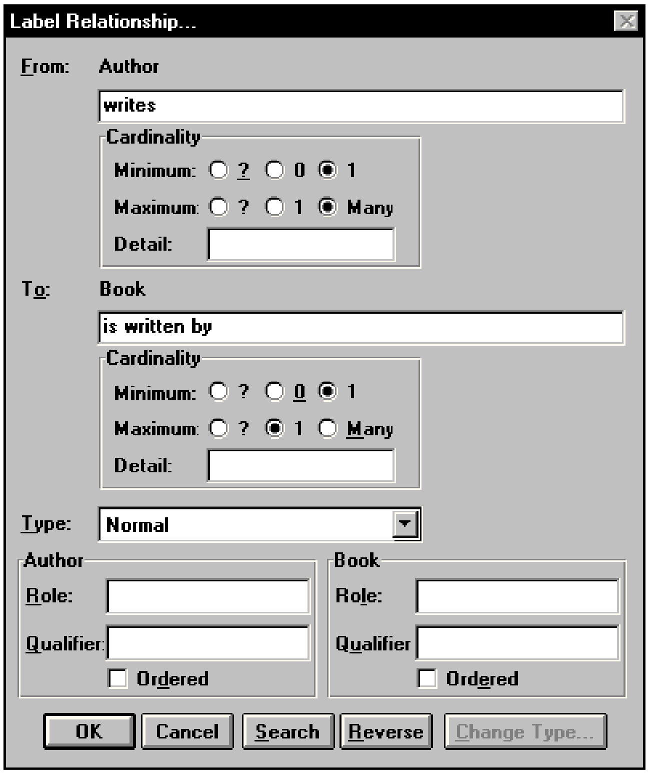
Discriminator:
Value:

Figure 3-34 Label ERD Relationship Dialog Box

The information that is maintained for a relationship is:

- **From.** The name of the parent entity or class.
- **Cardinality.** How many instances of one entity or class relates to another. The Detail field can be used to store a specific quantity, for example 1, 3 or 5+.
- **To.** The name of the child entity or class.
- **Type.** The type of relationship. An *identifying* relationship indicates the child entity cannot exist without the parent; the child is drawn as an attributive or associative entity depending on the number of identifying relationships attached. A *supertype/subtype* relationship indicates the creation of a specialized entity (subtype) that is based on a generalized entity (supertype) that shares common attributes. Only the attributes unique to the specialized entity need to be listed in the subtype object. An *aggregation* relationship indicates the parent class contains the child. An *inheritance* relationship indicates the definition of the To class is based on the From class. The base class contains the basic definition, while the derived class implements only those features that need to be different. A *normal* relationship indicates there are no special characteristics between the objects involved.
- **Incomplete.** If the relationship type is set to supertype/subtype, and all subtypes within a group appear on the current diagram, you can force the IDEF1X category symbol to be drawn as incomplete by clicking on the Incomplete option.
- **Denormalization.** The denormalization option to be used during SQL generation. None indicates the tables should not be altered during SQL generation. *Collapse Child* indicates the child columns should be added to the parent table. (If a supertype/subtype relationship is used, a discriminator should be specified.) *Duplicate Parent* indicates the columns in the parent table should be added to the child table, and the parent table should not be generated, while *Duplicate Parent and Retain* indicates the parent should be generated.
- **Discriminator.** The name of the discriminator used for supertype/subtype relationships. A discriminator is used to distinguish between different subtypes when the denormalization option for a relationship is set to collapse child. For example, if you had a supertype named Employee, and subtypes named Salaried Employee and Hourly Employee, and *Collapse Child* was the denormalization option, the discriminator Employee Type would be used to differentiate between the two types of employees
- **Role.** The role names used for the parent and child classes. For normal relationships, a role should exist as an attribute of the class at the other end of the association with the Reference type set to Address. For aggregation relationships, it should be set to Value. The Visibility in both cases should be Protected, and the type set to the opposite class.
- **Qualifier.** The qualifier names used for the parent and child classes. If a qualifier is used, an attribute of type Void is created in the class at the other end of the association with the Reference type set to Address and the Visibility set to Protected. You can change the type of the qualifier by modifying the attributes field in the repository.

- **Ordered.** Ordering indicates the objects on the many side of a relationship have an explicit order. The term set is commonly used to describe an unordered association, while a list indicates an ordered association.
- **Stereotype.** For relationships between use case symbols, this field determines the type of relationship. <<extend>> indicates the target use case adds functionality to the source use case. <<include>> indicates the target includes the functionality of the source, while <<generalize>> indicates a generalization relationship. The <<generalize>> stereotype is not displayed on the diagram because the notation (open arrowhead as opposed to stick arrowhead) indicates the type of relationship.



Label Relationship...

From: Author

writes

Cardinality

Minimum: ○ ? ○ 0 ● 1

Maximum: ○ ? ○ 1 ● Many

Detail:

To: Book

is written by

Cardinality

Minimum: ○ ? ○ 0 ● 1

Maximum: ○ ? ● 1 ○ Many

Detail:

Type: Normal

Author

Role:

Qualifier:

□ Ordered

Book

Role:

Qualifier

□ Ordered

Figure 3-35 Label Class Relationship Dialog Box

Note

- Each relationship between two entities or classes is a unique object in the repository. The fact that two relationships between two different pairs of objects

> may have the same name does not change the fact that the individual relationships are distinct. When using the Search function to name relationships, you are only selecting a *name*, not a *relationship repository object*. If, when you are finished, a pair of objects on this view is connected by relationships with the same names as relationships joining the same objects on another view in the data or object model, then both sets of relationships are considered two different instances (locations) of the same repository object.

When a relationship Type is selected as Identifying, the connected entity type is changed to reflect this relationship. One identifying relationship changes the connected entity to an Attributive entity, while two identifying relationships changes the connected entity to an Associative entity.

See the note on p.97 regarding the use of special characters and numbers when labeling lines.

Labeling an Information Cluster

As a compound symbol, an information cluster is labeled somewhat differently from other symbols. After you draw it onto a diagram, you must decide how many modules are to be contained in the cluster. If Auto Label Symbols is enabled, after you draw the symbol you see a dialog box in which you can enter the name of the information cluster, the name of the data-only module and the names of the contained modules. See the note on p.97 regarding the use of special characters and numbers when labeling lines or diagram objects. After you enter this text, the information cluster is redrawn with the number of modules for which you provided labels. If Auto Label Symbols was not enabled when you added the information cluster to the diagram, or you chose not to label it at that time, you can label it as described above by selecting Change Item from either the Diagram or Object menu and completing the dialog box. As with other symbols, you can click the Search button in the dialog box and select names of existing modules from the repository. This can be done for each module or data-only module, or even the information cluster itself, to be labeled by putting the cursor in each relationship name box in turn and clicking the Search button.

With this dialog box you can also change the number of modules in the information cluster by changing the number of module labels, or reorder the modules by reordering the labels by cutting and pasting using the Windows clipboard.

Labeling a Message

Messages must be labeled in order for an entry to be created in the repository. If Auto Label Lines is enabled, after you draw the message line, you will see a dialog box in which you can enter the name of the message, the message type, and other useful information.

If Auto Label Lines was not enabled when you added the message to the diagram, or you chose not to label it at that time, you can label it by selecting Change Item from either the

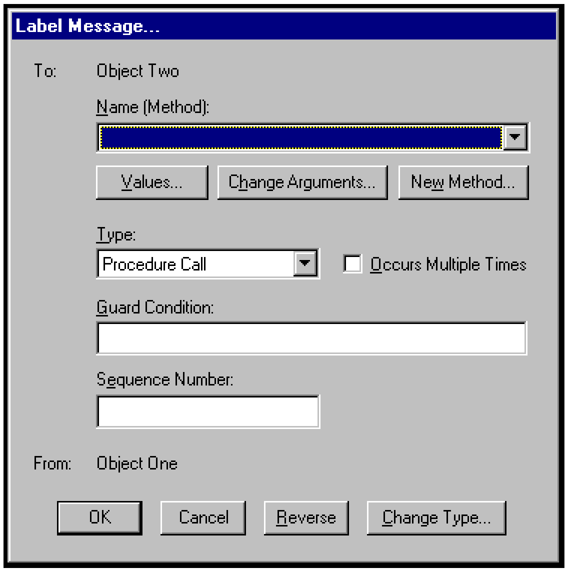
Diagram or Object menu and completing the dialog box. See the note on p.97 regarding the use of special characters and numbers when labeling lines or diagram objects.

The following information is maintained for a message:

- **To.** This is the name of the target object.
- **Name (Method).** Name is the name of the message along with its arguments. The name is based on the name of a method in the target object's class, or in a base class. Only methods from the derivation tree of the target object's class can be used. All available methods are displayed in the drop-down list. If you want to create a new method, click the New Method button. If the method has arguments, you can specify values for the arguments by clicking the Values button. By default, the name and type for the method are displayed. If you want to change the argument list of the method, click the Change Arguments button.

> **Note** The degree to which you can change method or message attributes depends on your rights to the target object's class and the interaction diagram settings.

- **Type.** This is the type of message, either Asynchronous Stimulus, Flat Flow of Control, or Procedure Call.
- **Occurs Multiple Times.** This indicates that the message will be called more than once. If this option is selected, an asterisk will appear next to the message name on the diagram.
- **Guard Condition.** Specify the guard condition that controls the firing of the message. This is a free-form text field.
- **Sequence Number.** This indicates the order of messages. This can either be a single numeric value, such as 1, 2, or 3, or a decimal such as 1.2 or 1.1.4. This option is only available on collaboration diagrams, since sequence by its very nature indicates message ordering.
- **From.** This is the name of the source object.



To: Object Two

Type: Procedure Call

□ Occurs Multiple Times

From: Object One

Figure 3-36 Label Message Dialog Box

Labeling an Event

Events must be labeled in order for an entry to be created in the repository. If Auto Label Lines is enabled, after you draw the event line, you will see a dialog box in which you can enter the name of the event and other useful information.

If Auto Label Lines was not enabled when you added the event to the diagram, or you chose not to label it at that time, you can label it as described above by selecting Change Item from either the Diagram or Object menu and completing the dialog box. See the note on p.97 regarding the use of special characters and numbers when labeling lines or diagram objects.

The following information is maintained for an event:

- **Event Name.** The event name is the name of the event.
- **Guard Condition.** Specify the guard condition that controls the firing of the event. This is a free-form text field.
- **Action Expression.** Specify the action expression that executes if the event is fired. This is a free-form text field.
- **Multiple Trigger.** Multiple trigger indicates that the event will be fired more than once. If this option is selected, an asterisk will appear next to the event name on the diagram.

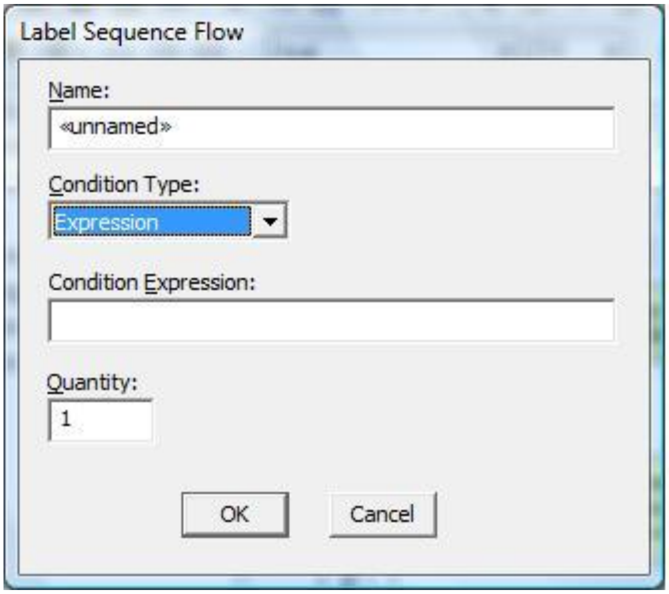
Labeling a Sequence Flow

Sequence flows used on a BPMN diagram should be labeled to fully document the business processes. If Auto Label Lines is enabled, after you draw the sequence flow, you will see a dialog box in which you can enter the name of the flow and other useful information. If you chose not to label it at that time the default label <<unnamed>> is automatically added for the flow when the diagram is saved. This insures that the sequence flow is defined in the repository. The default <<unnamed>> label is not displayed on the diagram.

If Auto Label Lines was not enabled when you added the sequence flow to the diagram, or you chose not to label it at that time, you can change the default label <<unnamed>> by selecting Change Item from either the Diagram or Object menu and completing the dialog box. See the note on p.97 regarding the use of special characters and numbers when labeling lines or diagram objects.

The following information is maintained for a sequence flow:

- **Sequence Flow Name.** The sequence flow name is the name of the sequence flow. The sequence flow will be saved with the default name of <<unnamed>> if choose not to label the sequence flow when it is drawn.
- **Condition Type.** Specifies the conditions that are evaluated at runtime to determine whether or not the flow will be used. The values are
  - **None**. No expression is defined
  - **Default**. The flow that will be used only if all other outgoing conditional flow is not true at runtime.
  - **Expression**. The condition evaluated at runtime to determine if the flow will be used.
- **Condition Expression.** Specify the expression evaluated at runtime to determine if the flow will be used.
- **Quantity.** Defines the number of Tokens that must arrive before the activity can begin. The default value is 1.



Label Sequence Flow

Name: «unnamed»

Condition Type: Expression

Condition Expression:

Quantity: 1

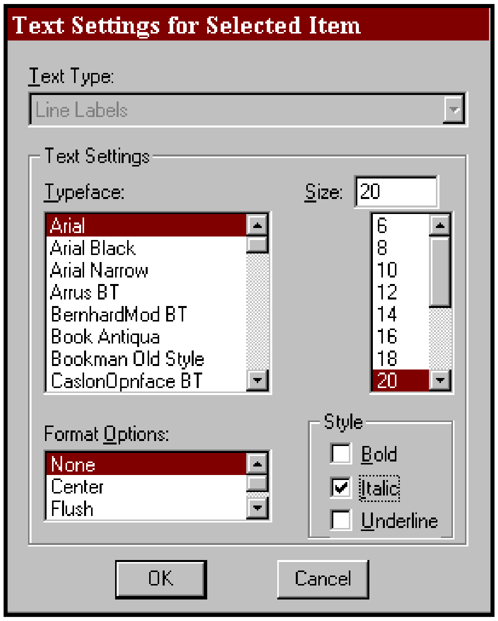
Figure 3-37   Label Sequence Flow Dialog Box

Editing Existing Text and Labels

The procedure for editing existing text is identical to that for entering it. The only difference is that you must decide whether the change is to be Individual, Local, or Global. If you choose Individual, the edit is applied to only the selected text. If the text is a label for a symbol that exists in more than one location and you change this particular instance of it, a new repository entry is created and it becomes a new repository item. If the new name is that of an existing object of that kind, it becomes a new location for that object. If you choose Local, the edit is applied to all occurrences of the selected text or label on the current diagram. If you choose Global the edit is applied to all occurrences of the selected text within the entire project.

Text Default Settings

When entering text, you can use any fonts that you have available under Windows. Selecting Text Settings from the Options menu allows you to make settings for different kinds of text used in Visible Analyst: symbol, process numbers, report headings, etc. For each of these text types, you can set the typeface and point size, as well as other characteristics such as bold, italic, underline, centered, etc. See Figure 3-35.



Text Settings for Selected Item

Text Type: Line Labels

Text Settings

Typeface: Arial

- Arial
- Arial Black
- Arial Narrow
- Arrus BT
- BernhardMod BT
- Book Antiqua
- Bookman Old Style
- CaslonOpnface BT

Size: 20

- 6
- 8
- 10
- 12
- 14
- 16
- 18
- 20

Format Options: None

- None
- Center
- Flush

Style

☐ Bold

☑ Italic

☐ Underline

Figure 3-38 Text Settings Dialog Box

Using Picture Files

Users can add one or more image files onto any diagram in the Visible Analyst. When the diagram is opened for editing, the Add Picture icon is displayed at the end of the tool bar. Clicking this icon opens the Insert Picture dialog as shown in Figure 3-29.

The image support can be implemented in the following ways.

- Insert an individual image, such as a company logo, onto a diagram.
- Add one or more images to a Boilerplate file and include the image with any new diagrams.
- Substitute an image file for a methodology symbol on a diagram. Using the Symbol Template option on the Options menu, the user is able to map an image file to replace the Visible Analyst methodology symbol on the diagrams. When the user adds the methodology symbol to the diagram, the image is substituted for the Visible Analyst symbol. All repository entries and capabilities are maintained for the symbol.
- Create a template of image files that can be used on an Unstructured diagram, to model a Network for example.

Notes

- Only System Manager level users in the multi-user version of Visible Analyst can create or modify the Template files. Project Manager and User level users can add an image to a new or existing diagram only if they are assigned Create or Modify Diagram rights in the project.

- Symbols that can display attributes on the diagram such as the Class or Entity symbols, may have an image file substituted for the default symbol. Changing the display level to display the attributes of the item will overlay the attribute list on the symbol.

The following image file types are supported:

- Bitmaps (*.bmp)
- GIF File (*.gif)
- JPEG (*.jpg)
- Icons (*.ico)
- Enhanced Metafiles (*.emf)
- Windows Metafile (*.wmf)

Adding images to a diagram

Images are added to a diagram by selecting the Diagram | Picture menu item or by clicking the Picture icon in the Diagram Tools toolbar.

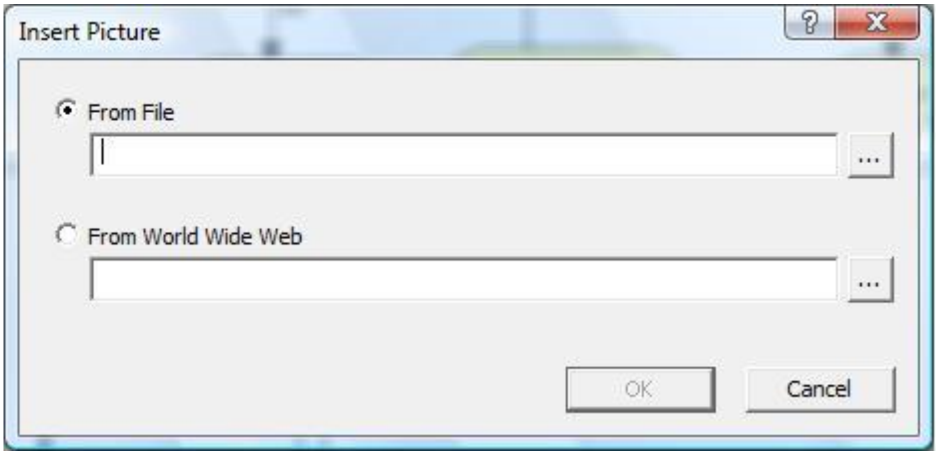


Figure 3-39 Insert Picture Dialog Box

Click the **…** button to browse to an existing image file on the local PC or on a network drive, or to select a file from the World Wide Web. A number of free image files are available on the Microsoft website and by other vendors. Once the image file is selected, click OK to display the Symbol Cursor (Figure 3-24) on the diagram. Clicking the cursor adds the image to the diagram. The image can be manipulated on the diagram in the same way as any other diagram symbol, and can be used on a diagram more than once. If Auto Label Symbols is enabled and you are adding a substitute methodology symbol, you are prompted to label the image. A non-methodology image can be labeled by selecting Change Item from either the Diagram or Object menu and completing the dialog.

Substituting an image file for a methodology symbol

Selection Mode: 1 Select the Symbols Template option on the Options menu to display the Symbol Templates dialog.

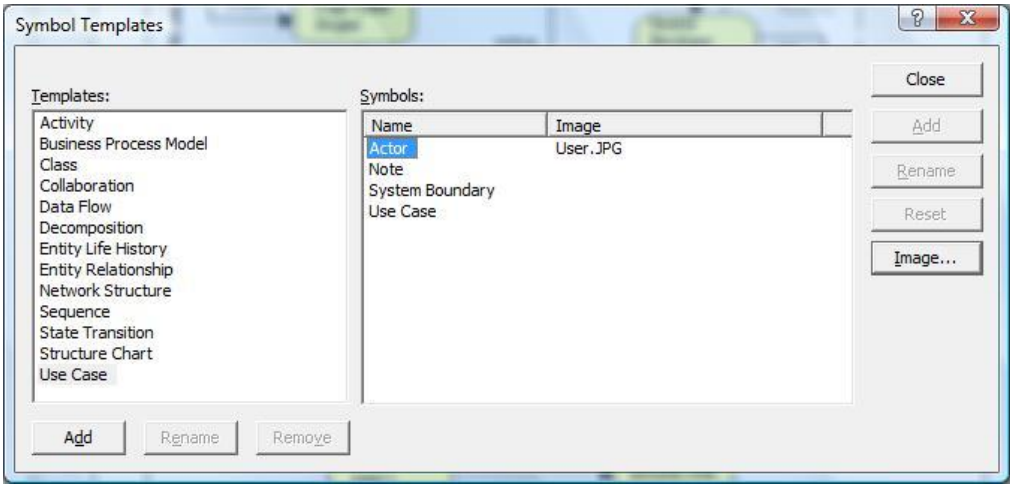
Select the Diagram Template: 2 Select the diagram template from the list of supported diagrams to display the diagram symbol types

Select Visible Analyst symbol: 3 Click on a methodology symbol name.

Click the Image button: 4 Click the Image button to display the Select Image for Symbol dialog, similar to the Insert Picture dialog box.

Select Image file: 5 Select the image file to be used as a substitute for the Visible Analyst methodology symbol and click Open. Click OK on the Select Image for Symbol dialog

The selected image file will be displayed in the Symbol Templates dialog as show in Figure 3-40



Templates:

- Activity
- Business Process Model
- Class
- Collaboration
- Data Flow
- Decomposition
- Entity Life History
- Entity Relationship
- Network Structure
- Sequence
- State Transition
- Structure Chart
- Use Case

Symbols:

Name	Image
Actor	User.JPG
Note	
System Boundary	
Use Case	

Figure 3-40 Symbol Template Dialog Box

When the Actor stick figure symbol is selected on the Use Case diagram toolbar, the picture referenced by the User.jpg file will be added to the diagram If Auto Label Symbols is turned on, the user is prompted to label the object. The new symbol has the same repository entry and capabilities as the original methodology symbol.

Image files that have been substituted for a methodology symbol can be changed to another symbol using the Change Item feature. Changing the symbol back to the original entry type will not use the image file, but the default methodology symbol supplied with the Visible Analyst.

Unstructured template files.

Unstructured template files can be defined using image files for the symbol set and used to create diagrams not represented by the methodology diagrams. The Template file is enabled for Unstructured Diagrams and can be used to diagram flowcharts, the network structure, organizational charts, etc.

Defining an unstructured diagram template symbol set is similar to substituting an image file for a methodology symbol.

Selection Mode: 1 Select the Symbols Template option on the Options menu to display the Symbol Templates dialog.

Add a new template: 2 Click the Add button under the Template section of the dialog but to add and label the template file.

Add and label the new symbol: 3 Click the Add button on the right side of the dialog to add and label a new symbol.

Select the image file: 4 Click the Image button on the right side of the dialog to select the image file associated with the symbol name.

After defining the new template file, the template can be selected on the New Diagram dialog when you select Unstructured as the Diagram Type. See Figure 3-2. When the diagram is opened for editing, the image files you selected for the symbol set are displayed in the symbol toolbar. Figure 3-41 shows 3 sample images.

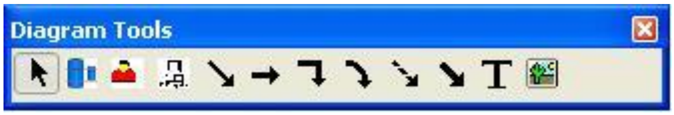


Figure 3-41 Symbol Template toolbar

The last icon in the toolbar is the picture icon, available for all diagrams. Clicking this icon displays the Insert Picture dialog as shown in Figure 3-39.

EDITING A DIAGRAM — OTHER DIAGRAMMING FUNCTIONS

Editing functions deal more with the appearance of diagrams than their underlying meaning. Other Visible Analyst functions that deal more with the semantics of diagrams or with tying diagrams to a project repository are described in the chapters on The Visible Rules and The Visible Repository, respectively. Editing functions can be performed while creating a diagram as described in the previous sections of this chapter. Editing can also be performed after the diagram is saved by initiating editing functions from File, Edit, Options or Diagram menus and by making use of other Visible Analyst functions. The latter part of this chapter is devoted to the description of some of the major editing functions. This section explains some of the smaller functions not described elsewhere. Note that some of these functions appear not only on the menus mentioned, but also in the Object menu for a selected object. The functions explained here are:

Options menu
Control Bar
Object Browser

Diagram menu
Change Item
Stylize
Snap Symbols

File menu
Project History
Erase Changes

Edit menu
Copy
Paste
Clear
Delete

Control Bar

Control Bar, on the Options menu, displays a row of buttons above the diagram workspace that gives you quick access to some Visible Analyst functions. For example, you can execute the following menu functions with control bar buttons:

- Select project to open.
- Create a new diagram.
- Open an existing diagram.
- Save the current diagram.
- Print diagram(s).
- Find an object on the current diagram.
- Generate database schema
- Import RDBMS database definitions.

Other buttons on the control bar aid you in drawing diagrams. For example, to add a symbol to a diagram, you would have to select Symbols from the Diagram menu and then pick the symbol you want to add from the submenu that displays. With the control bar, shown in Figure 3-42, you simply click the button representing the symbol you want to add, if it is not already highlighted. Adding lines and couples works in the same way.

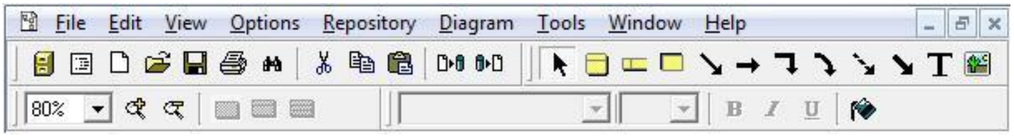


Figure 3-42 Control Bar Buttons for Gane/Sarson DFDs

The Control Bar menu option allows you to customize the control bar. When you select Control Bar, the Control Bar dialog box is displayed, allowing you to select the control bar options you want displayed. The control bar can contain up to four tool bars:

- The standard tool bar contains basic buttons (Select Project, New Diagram, Open Diagram, etc.).
- The diagram tools tool bar contains the symbol, line, text and image buttons appropriate for the current diagram.
- The view tool bar contains controls to change the zoom level and entity/class view level.
- The font tool bar contains controls to allow changing the current font characteristics.

Object Browser

With Object Browser, on the Options menu, you can choose to have the object browser, shown in Figure 3-43, display on your screen.

The object browser displays a list of all the objects in the repository. If there are no diagrams open, or the current window is the diagram list, the object browser displays all the objects in the repository. If a diagram is open, only the objects valid for that diagram type are shown. If an object appears on the diagram, it is shown in bold.

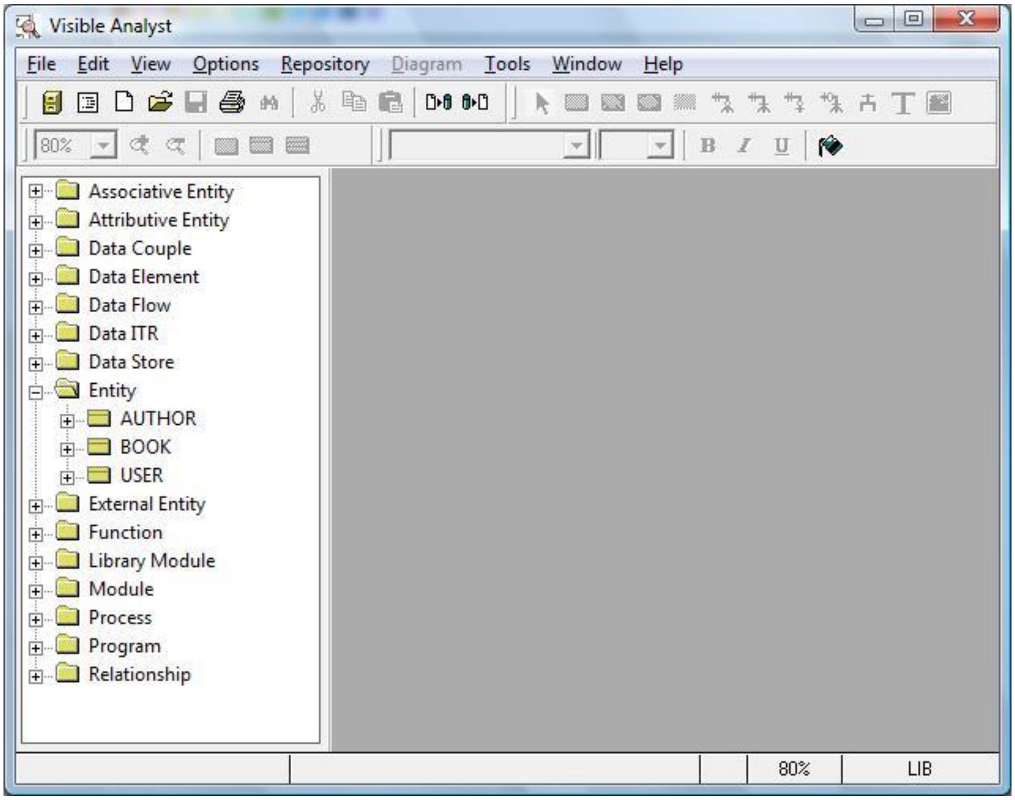


Figure 3-43 Object Browser Displayed on the Workspace

In the object browser you can:

- Double-click on a folder to expand or collapse it.
- Double-click on an object to display the Define dialog box.
- Right-click on an object to display a context-sensitive menu of options.
- Click and drag an object onto the diagram.

Right-click within the Object Browser window (but not on an entry) to display a context-sensitive menu of options for the browser: Hide turns off the browser window, Refresh reloads the repository objects, and Properties displays a dialog box that allows you to change how the object browser appears. Properties include:

- Group Objects by Type. For each object type in the repository, a folder is displayed with the object name. To see the objects, open the folder by clicking the + button or double-clicking on the item. If this option is not checked, all entries in the repository are listed alphabetically without folders.
- Differentiate Entity Types. Entities/Independent Entities, Associative Entities, and Attributive/Dependent Entities have their own folders. If not checked, all entity types are listed in one folder.
- Auto-Find Object When Selected. If this option is checked, the object is scrolled into view on the current diagram.
- Attributes. The attributes (composition) for entities, classes, data flows, data stores, and any user-defined object types created with Composite Type option are displayed.
- Primary Key. The primary key field(s) for an entity is displayed. This option is checked by default when attributes are displayed.
- Foreign Keys. The foreign key field(s) for an entity is displayed. This option is checked by default when attributes are displayed.
- Methods. The methods for a class are displayed.
- Base Classes/Entities. The classes or entities that are related to the current object via a dependent relationship are displayed.
- Physical Characteristics for Elemental Types. The type, length, and null characteristics for an elemental object (data elements or classes with an elemental subtype) are displayed.

To resize the object browser, click on the right browser margin and drag the margin to the size you want.

The Edit Menu Functions

The Undo, Cut, Copy, Paste, Clear and Delete functions from Edit menu are the standard Windows editing functions and work as usual in the Visible Analyst diagram editing context.

Undo

Undo deletes a partially completed line from a diagram. Once you double-click to complete the line, Undo no longer works. You must then use Delete.

Cut, Copy and Paste

These three functions all interact with the Windows Clipboard. The formats used to move data to and from the Clipboard are text (used when *only* text is copied to the Clipboard), Windows bitmap, Windows metafile, and a proprietary Visible Analyst format. Once data is on the Clipboard, it can be used by any software application that recognizes that particular format.

Cut, Copy

Cut and Copy are identical in function except that when Cut is executed, whatever is placed on the Clipboard is also removed from the diagram. Any object highlighted as the current object or any block selected on a diagram can be cut/copied to the Clipboard. If a symbol is selected and Include Connections is enabled, both the symbol and any connected lines (including labels for all of these) go to the Clipboard. Selecting a line with Include Connections enabled does *not* put attached symbols on the Clipboard. The formats in which items are moved to the Clipboard are:

- Anything sent to the Clipboard (symbols, lines, either of those with attached labels, caption text in a block with other diagram objects), resides there as a Windows bitmap, a Windows metafile, and in the Visible Analyst proprietary format.

As is usual with Windows applications, text from edit boxes can be cut and copied to the Clipboard by highlighting it with the cursor and pressing the shortcut key for Cut or Copy**.**

Paste

Paste moves data from the Clipboard into Visible Analyst. The only items that can currently be pasted onto a diagram are Clipboard data in either text format or Visible Analyst proprietary format. When you select Paste from the Edit menu or press the shortcut keys, the Clipboard data is displayed on your diagram as a selected block at the cursor position. You can then move the block anywhere on the diagram and fix its new position by clicking the left mouse button anywhere on the diagram outside the block.

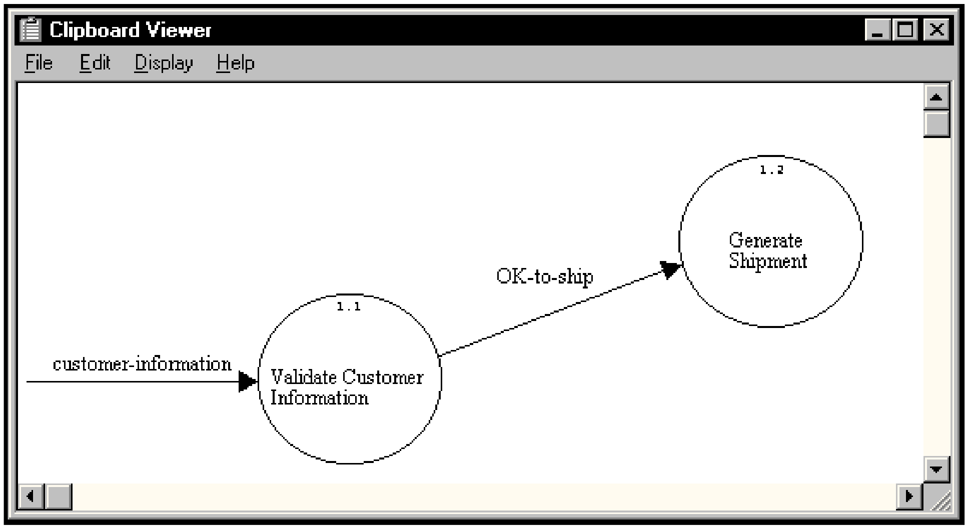


Figure 3-44 Windows Clipboard with Visible Analyst Data

When you paste Clipboard information other than simple text onto a diagram, Visible Analyst recognizes what it is and adds it to the diagram correctly: symbols are added as symbols, lines as lines, etc. If you try to paste onto a diagram an object (such as a process) that must be unique in a project, Visible Analyst does not paste it with its label if it is found to violate the uniqueness rule. It is pasted without its label.

As is usual with Windows applications, text from the Clipboard can be pasted into edit boxes by pressing the shortcut key for Paste.

Clear

Clear deselects the current object or block on the current diagram. It performs the same function as clicking on a blank area of the diagram with the mouse. As is usual with Windows applications, you can also use the shortcut key that is shown on that menu item.

Delete

Delete is identical to Cut, except that nothing is moved to the Clipboard. Since it is not reversible as Cut is, the Delete function asks you for confirmation before deleting from a diagram.

Note

- If Include Connections is enabled, deleting or cutting a symbol also removes its attached lines from the diagram. For example, if the symbol is an entity, Visible Analyst deletes any attached relationship. This means that the relationship entry in the repository is deleted if there is no descriptive information and if it doesn't exist on another view. Deleting a line with Include Connections enabled does *not* delete attached symbols.

The Change Item Function

As is mentioned elsewhere, Change Item on the Diagram menu allows you to change existing caption text, symbol and line labels, relationship cardinality, information cluster module number and order, and, for lines, the type of the line itself, as well as its terminators and orientation (point-to-point, snap, elbow).

You can also click the Search button to search the repository for the item you wish to enter into the label field. Items of the same repository object type are displayed alphabetically.

Generally, when you label an object on a diagram, Visible Analyst decides, based on diagram space available, whether the label fits all on one line or it breaks over multiple lines. However, you can force a label to spread over more than one line. Wherever you want a line break in an object label, either press CTRL+ENTER or enter a ➢➢ character by typing ALT+0187 (hold down the ALT key and type 0187 on the numeric keypad). When the label is drawn on the diagram, the label breaks where you specified.

The Change Type Button

Also in this dialog box is the Change Type button. With it you can change a selected symbol already placed on a diagram to another symbol in the symbol set in use for the diagram type currently selected. The symbols to which you can change the selected one are listed in the Change Type dialog box. Line connections, symbol label and symbol stylizing remain as on the original symbol. (If the new symbol is very differently shaped from the old, lines may appear disconnected from the new symbol and certain stylizations may change.) You can also change a selected line into one of another style or change its terminators or orientation. There are some restrictions on what symbols and lines can be changed. If a change is not possible, the individual selections or perhaps the button itself are not available.

Notes

- Change Type functions for most methodology symbols, but not for those on data flow diagrams, because DFD symbols used don’t have reasonable alternative symbols to which to change them. Change Type also works for non-methodology symbols on structure chart, boilerplate and unstructured diagrams.
- Image files that have been substituted for a methodology symbol can be changed to another symbol. Changing the symbol back to the original entry type will not use the image file, but the default methodology symbol supplied with the Visible Analyst.

The Stylize Function

The Diagram menu Stylize function allows you to enhance the appearance of symbols, or change the size of any symbol in a diagram. After selecting a symbol and choosing Stylize from the Diagram or Object menu, a dialog box displays containing a symbol of the type selected. You can choose from Horizontal and Vertical, Size, Boldness, Relief Amount, and Overlay. Size makes the symbol smaller or larger in the horizontal or vertical plane, or both. If you select Bold, you can darken the symbol. Relief adds a shadow effect to the symbol. Overlay creates a stacked symbol. The use of the Stylize function changes the appearance of symbols and can be useful for special effects and presentation graphic purposes; it does not affect the identity or recognition of methodology symbols.

For Size, Boldness and Relief Amount, use the scroll bars to select the degree of change you want to make in the symbol. At any time, you can click the Apply button to see the effect of the changes you have made. For Overlays, you must set the Count (the number of overlays) and use the scroll bars of the box to specify the relative position of the overlay to the original symbol. This relative offset is used to position each successive overlay. Again, click Apply to see the total effect of all of the stylizing you have done.

At any time you can click Reset to undo the stylizing you have done to the symbol. Once you click OK, the stylizing is applied to the symbol on the drawing. If the symbol originally selected was already stylized, Stylize modifies it or, by clicking Reset, returns it to its original state.

If you want to make a symbol size change permanent, click the Set Default Size button. Whenever you add a new symbol of this type to a diagram, the size is set to the new size.

Notes

- If you change the size of the symbol and/or add enough overlays, the symbol becomes too large to fit in the stylize box. In such cases, the symbol as displayed is scaled to fit the box. However, when you click OK, the stylizing is applied to the selected symbol on the diagram *full scale*; the scaling done in the dialog box has no affect.
- Entity and class symbols can only be stylized when no attributes or methods are displayed.
- Image symbols can be stylized as described above, but the Boldness, Relief Amount, and Overlay options are not enabled for image files.

Changing the Size of a Symbol

Whenever you add a symbol to a diagram or whenever you select a symbol, you see small colored boxes, or handles, in the four corners of the symbol. You can use the handles to change the size of the symbol by pointing the cursor at any one of these handles, holding down the left mouse button and dragging the handle to a new position. When you release the button, the symbol is redrawn.

Notes

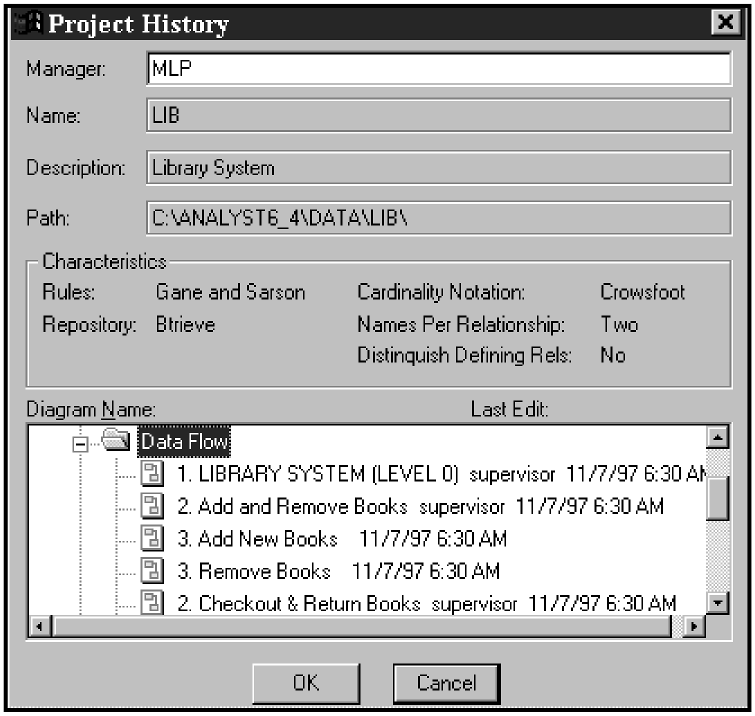
- Please remember the following:
    - Entities and classes do not have sizing handles if attributes are displayed. To change the symbol size, you must turn off attribute display mode.
    - If you press the ESC key *before* you release the mouse button while dragging a handle, the sizing operation is canceled and the object returns to its *original* size.

Snap Symbols

Snap Symbols, on the Options menu eases the alignment of symbols along a grid. If you select into a block the symbols you want to line up in one horizontal or vertical row and select Snap Symbols, the symbols in the block snap into exact horizontal and/or vertical alignment, whichever applies

The Project History Function

Project History, from the File menu, allows you to display a summary description of the project as a whole. In addition to all of the settings you made when you created it, you can view the diagram trees for each diagram type and see when each diagram was last edited. If you have the LAN version of Visible Analyst, you can see who created the project, as listed in the Project Manager field, change a project owner, and determine which diagrams have been edited and by whom.



Manager: MLP

Name: LIB

Description: Library System

Path: C:\ANALYST6_4\DATA\LIB\

Characteristics

Rules:	Gane and Sarson	Cardinality Notation:	Crowsfoot
Repository:	Btrieve	Names Per Relationship:	Two
		Distinquish Defining Rels:	No

Diagram Name: Last Edit:

- Data Flow
  - 1. LIBRARY SYSTEM (LEVEL 0) supervisor 11/7/97 6:30 A
  - 2. Add and Remove Books supervisor 11/7/97 6:30 AM
  - 3. Add New Books 11/7/97 6:30 AM
  - 3. Remove Books 11/7/97 6:30 AM
  - 2. Checkout & Return Books supervisor 11/7/97 6:30 AM

Figure 3-45 Project History Box

The Erase Changes Function

Erase Changes, from the File menu, instructs Visible Analyst to ignore all editing changes that have been made to the diagram since the last time it was saved. The diagram redraws as it was last saved and any changes you made are permanently lost.

USING CONSTRUCTS

Visible Analyst provides the capability to store entire diagrams or sections of diagrams as constructs. They may be thought of as “library” files for saving whole or partial diagrams that you may wish to reuse throughout a project or in different projects. Constructs are stored as system files and may be recalled for use in new diagrams as well as in other portions of an existing diagram. Constructs may be composed of anything that appears on a diagram (symbols, lines, text, images) and provide reusable pieces or shells of diagrams that can be used at will. There is no limit to the number of constructs that may be stored in the construct library, and you may load a construct into any diagram.

Working with constructs is very similar to using Copy and Paste, from the Edit menu, on selected blocks on a diagram. The difference is that you can save them permanently, whereas Copy and Paste deal only with the Windows Clipboard.

Creating a Construct

To create a construct:

Open a Diagram: 1 Access the diagram from which you wish to create the construct.

Enclose the Desired Objects in a Block: 2 Select a block enclosing the items you want to include in the construct.

3 Select Construct at Diagram menu. The Create a Construct dialog box displays.

Name and Describe the Construct: 4 Enter an eight-character file name for the construct and and a description of it to help you remember what the construct is.

Save the Construct: 5 Click OK to save the construct. Your construct is saved in a file and can be recalled by selecting Construct, as described below.

Loading a Construct

To load a construct onto a diagram:

Open a Diagram: 1 Access the diagram to which you wish to add the construct. Note that if you have only one diagram open and it is minimized to an icon, you can load a construct to it, but cannot complete the process until the diagram has been at least enlarged to a window.

2 Select Construct at the Diagram menu. The Load a Construct dialog box displays. The names of the constructs currently stored in your construct library are displayed, along with a description for each.

Select a Construct: 3 Select one of the constructs from the display and click the Load button. It appears on your diagram as a block enclosed in a rectangle, with the items within it shown as selected.

Load and Position the Construct: 4 Position the construct on the diagram as you would a a block, by clicking the left mouse button and dragging the block to where you want it. Clicking outside of the block finalizes the position of the construct on your diagram.

Note

📋 If an object label in the construct conflicts with an object label already in the project, then that label is not copied to the diagram as part of the construct. Also, if a construct is created from one diagram type and loaded into a different diagram type, the construct diagram appears but the objects are not recognized as methodology objects and are transparent to the Analyze function. Further, nothing within a construct is linked to the repository until it has been added to a diagram and *saved.* To move an entire diagram between projects along with all associated repository entries, see the section describing the Copy Diagram function.

Deleting a Construct

To delete a construct:

Open a Diagram: 1 Access a diagram. The Diagram menu is not activated until a diagram is open.

2 Select Construct at the Diagram menu. The Load a

		Construct dialog box displays. The names of the constructs currently stored in your construct library are displayed, along with a description for each.
Select a Construct:	3	Select one of the constructs from the list.
Delete the Construct:	4	Click the Delete button. You are asked for a confirmation before the construct is deleted from your library.

NESTED DECOMPOSITION

The Nest function, on the File menu, allows you to “explode,” or decompose a symbol, usually a process, and create a lower level diagram that represents the detailed functional aspects of the symbol being exploded. The newly created lower level diagram becomes “nested” to the higher level symbol and the diagram containing it; that is, the lower-level diagram becomes a child of the parent diagram in the project tree. (Note that only one child diagram can be nested from any process.) Nest also allows you to create a *process* decomposition. Nest functions only for data flow diagrams and only on processes; it does not apply to the other diagrams. Nest can be used to decompose any FDD function or DFD process symbol. For *functional* decomposition diagrams, it is replaced on the File menu with Spawn. The Spawn function is described later in this chapter.

The Object menu “Explode” function can also be used to connect a class to a state transition diagram. The “exploded” diagram shows the dynamic behavior of the class. Other symbols, such as Use Case, Activity, Entity, etc. can also be exploded to link the item to a child diagram.

Nesting Considerations

New data flow and business process model diagrams below the top level are normally created using the Nest function. The following considerations also apply to nested diagrams:

- Only the process or activity symbols may be exploded into a lower-level child diagrams. When other symbols are exploded, the selected symbol is linked to the new diagram type as displayed on the Nest dialog
- Process numbers are automatically added to data flow processes when using rules. Refer to The Visible Rules for additional information on the process numbering scheme.
- All data flows attached to the process being exploded are duplicated and displayed on the lower level nested diagram when using rules. That is, all net input and net output data flows are automatically “dragged down” to the lower level for level-to-level balancing purposes.
- No sequence flows are included when a BPMN activity is exploded.

The Nest Procedure

The Nest function has four subsidiary functions in its submenu: Explode, Parent, Detach and Decompose.

Each of these is explained below.

Explode

What Explode does depends upon whether the currently selected FDD or DFD symbol has an existing nest relationship to a child diagram. If it does not, you can create a nested decomposition to a child diagram. If the current symbol does have an existing nest relationship to a child diagram, you immediately open that diagram, or activate it if it is already open.

The following symbols can be used to generate child diagrams, or link a symbol with a child diagram. Click on the symbol with the right button to activate the Object menu.

- Functional Decomposition Diagram function – Spawns to a Data Flow Diagram hierarchy, explained later in this chapter.
- Data flow Process – explodes to create a child Data Flow diagram.
- Entity – links to an Entity Life History diagram
- Class – links to a State Transition or Activity diagram
- Activity – links to an Activity diagram
- Use Case – links to an Activity, Collaboration or Sequence diagram.
- System Boundary – links to a Use Case diagram
- Activity (BPMN) – explodes to a child Business Process Model Notation diagram

The procedure for exploding a symbol with no existing child and creating a lower-level nested child diagram is as follows:

Select a Symbol: 1 Click on a symbol to select it. You can also click with the right button to activate the Object menu.

Select Nest/Explode: 2 Select Nest at the File menu, then Explode from the submenu, or pick Explode from the Object menu. The Nest dialog box appears.

Select New or Existing Diagram: 3 You can click either of the buttons in the box:

- Create New. A new diagram is created and becomes the active diagram. When exploding a data flow process and rules are enabled, all net input and output data flows are dragged down to the new diagram.

When exploding a Use Case or a Class symbol, choose the new diagram type from the Diagram Type list.

- Select Existing. This allows you to nest to an already existing diagram. For DFDs, the dialog box expands, showing all diagrams in the branch of the project tree file directly below the current diagram and not already nested. You must select one of these existing diagrams to be the nested child. The selected child diagram is activated, all net input and output data flows are dragged down to the diagram and the nest relationship created. When exploding a class or other diagram symbol, a list of all applicable diagrams is displayed.

You must Save the child diagram to permanently establish the nest relationship.

Note

When a data flow process has a large number of attached input and output data flows, and the process is exploded, only a subset of the attached data flows may be dragged down to the child diagram. To add the other attached flows to the child diagram, move the dragged down flows to the middle of the diagram, and explode the parent process again. The remaining flows will be dragged down to the new child diagram.

Parent

If the currently selected diagram has an existing nest relationship to a parent process, selecting Parent opens the parent diagram, or activates it if it is already open.

Detach

If the active diagram has an existing nest relationship to a parent diagram, selecting Detach breaks or un-nests the relationship. For Data flow diagrams, the two diagrams retain their relative positions in the project tree, but the parent child relationship no longer exist. This allows the former parent process to be nested to a different child diagram and vice versa using the Explode feature.

Using Detach with other linked diagrams removes the link from the parent symbol to the linked diagram.

Decompose

Decompose generates a process decomposition diagram, as described below.

Generating Process Decomposition Diagrams

Process decomposition diagrams can be created automatically from a set of data flow diagrams. By selecting a process and choosing Decompose from the Nest submenu or from the process Object menu, a process decomposition diagram is generated that shows all subordinate processes at their appropriate levels. If you select a process without children, the Decompose menu selection is not available. The procedure for generating a process decomposition diagram for a project branch, which may cover the whole project if the highest-level process is used, is automatic once the above action is taken. Visible Analyst draws an editable *unstructured* process decomposition diagram and that new diagram is the active one.

For a large project or branch, it is possible that the process decomposition diagram is too large to edit. If this happens, you are notified and given the opportunity to send the diagram to the print queue. Visible Analyst proposes a name to identify the diagram in the print queue and you are able to change it if you wish. To print it, select Print from the File menu.

Note

- A *process* decomposition diagram is very different from a *functional* decomposition diagram. The former is simply an unstructured diagram laying out the hierarchy of processes that are descendants of a selected process. The latter is a full diagramming methodology for doing business planning.

Using Spawn with FDDs

For FDDs, the purpose of the Spawn function, from the File menu is to create a process model extension of your broad functional decomposition and allow you to carry out a detailed analysis. With Spawn, you indicate a lower-level function, one that you have decomposed on the diagram into processes, and Spawn it. Visible Analyst creates DFDs containing these processes, to which you can add all of the other elements normally contained on DFDs. While it is not impossible to continue to make changes to a function branch on an FDD after spawning DFDs from it, it would be better to do the Spawn operation after the FDD is well thought out and complete. The concept of the Spawn operation is illustrated in Figure 3-46.

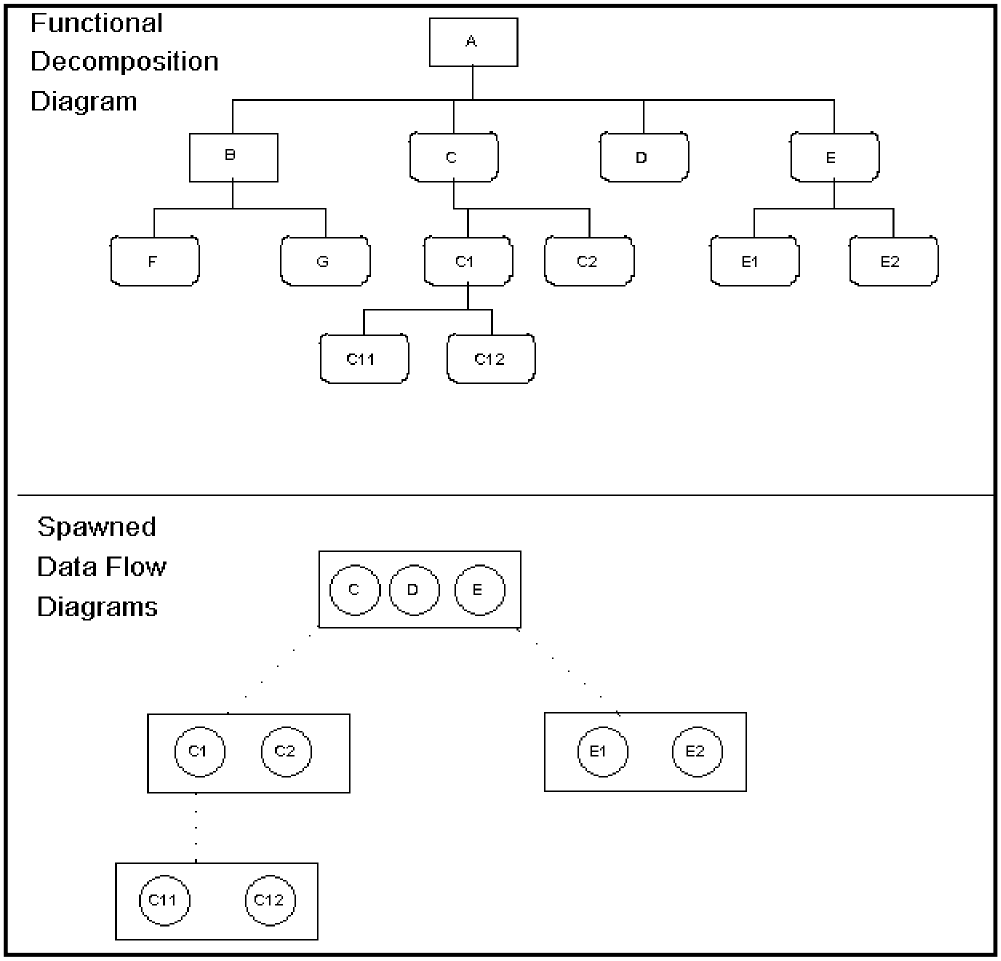


Figure 3-46 DFDs Created by an FDD by Using SPAWN

The first time you Spawn a function on an FDD, you are given the option to create a top-level diagram beneath which you can hang all branches of your DFD tree. This new top level diagram would symbolize all of the functions on your FDD. If you are using Yourdon rules, you also have the option to make this top-level diagram a context diagram. Unless you have processes hierarchically immediately beneath your highest-level function, you should accept this top-level diagram option. Otherwise, you have to manually insert a diagram as a parent for your spawned DFDs. If data flow diagrams already exist at the time of the first Spawn, Visible Analyst attempts to incorporate them into the Spawn operation.

The Spawn Procedure

The spawn procedure has four subsidiary functions: New DFD Set, Load DFD, Verify and Unlink. Each of these is explained below.

New DFD Set

What New DFD Set does depends upon whether the currently selected function has been decomposed on the diagram to a set of processes and whether DFDs have already been spawned from the function. If it has not been decomposed, Spawn advises you to decompose before you can proceed. If you created a decomposition, you can execute this function to create the set of DFDs composing this function. If the current function does have an existing set of DFDs, you can execute the other Spawn subfunctions described below.

The procedure for spawning a function with a decomposition and no existing set of DFDs is:

Select a Function: 1 Click on a function to select it. You can also click with the right button to activate the Object menu.

Select the Spawn Functions: 2 Select Spawn at the File menu and New DFD Set from from the submenu, or pick New DFD Set from the Object menu. The Spawn dialog box appears.

Position the Diagrams in the Tree: 3 From the tree display in the box, choose the diagram you want to be the parent to the new set of diagrams. When you click the Update DFDs button, the new diagrams become part of your project. Parent/child nest relationships are established among the diagrams of the new DFD set, as defined in the FDD. However, the top diagram of the newly added branch is not nested to the diagram you selected as the parent in the dialog box. You have to Nest before this relationship is established.

Load DFD

If the currently selected function does have an existing set of DFDs and you want to edit existing spawned DFDs (to add data flows, for instance), open the Spawn menu and choose Load DFD. The top diagram of the spawned branch of DFDs is opened. You can open lower level diagrams by selecting a process and exploding it.

Verify
If the current function has an existing set of DFDs, you can update and resynchronize the two sets of diagrams (the FDD and DFDs). You may want to do this if you have made changes to your FDD and want to update the spawned DFDs to reflect these changes. Bring up the Spawn menu and choose Verify. Visible Analyst compares the diagrams to find inconsistencies. If changes are needed, a dialog box describes them and asks you if you want to proceed. If you choose to continue, Visible Analyst makes the necessary changes to the DFDs and saves them. It then activates the top DFD of the branch for you to edit.

Unlink
If the current function has an existing set of DFDs, you can break the link between them with the Spawn function Unlink. You may want to do this if you want to make serious hierarchical relationship changes to your FDD or to change or delete the function from which the DFDs were spawned. These operations are not allowed on functions linked to DFDs. You can respawn later, if you choose. Note that this operation does not alter your DFDs in any way.

THE DIAGRAM VIEW FUNCTIONS

A “view” is an entity relationship or class diagram containing a selection of entity types or classes and the relationships between them. Since certain data objects may interact in different ways at different times, you have the option to look at these distinct interactions in different combinations on different views. A view is usually a subset of a larger data/object model and, in any given view, you are not obligated to use all of the relationships that exist in your model for any pair of entities/classes; you may select all of them or any subset of them, and you may create new relationships on the current view. Any new relationship you add between a pair of objects in a view is added to the repository of the appropriate model and becomes available to be added to any other view.

Implicit in any data or object model is a global view; that is, a view containing all of the information in the repository. In many models, the size of a global view may be too large to edit as a diagram (and probably too large to be visually useful, as well). That does not preclude, however, your ability to look at the whole thing should you desire. Visible Analyst draws your global view for you and enters it into the print queue.

There are three ways you can create a data or object model view. One is to open a new diagram and manually draw on it the entities/classes and relationships that you want for that view. This is how you must begin the first view in a project because it is the only way to add relationships. (Entities and classes can be directly entered into the repository.) The second way is to let Visible Analyst help you by using the View of Data Model group of functions: Global, New and Process, and also Modify View, that appears on the Diagram menu and as an option the New Diagram dialog box (Drawing Method). The third way is to select the entity, class or relationship in the Object Browser and drag the selected item onto a new diagram. If the selected item has a relationship to an existing item on the diagram, the relationship is automatically added to the diagram. Dragging a relationship onto the diagram will automatically include the attached entity or class. After your data model is well along, you can use the functions for creating cluster views. These are described later in this chapter. After the new view has been created, you may edit it as you wish.

Using the View of Data Model functions plus Modify View allow you to specify existing entities, and relationships from the repository and assure yourself that the new view is consistent with what is already in the repository. The functions then *automatically* draw your new view after you have completed specifying the contents for the view (see below).

When using the View of Data Model feature, remember the following:

- A view is like any other diagram in that it must obey the same rules. View diagram names must be unique within the data or object model for the current project.
- An entity or class may appear on any number of views at one time, but only once per view.
- The view functions work only for entity relationship and class diagrams.
- If an entity is used on a class diagram, its type is converted to class with a subtype of the actual entity type. If a class is used on an entity relationship diagram, the subtype is set accordingly.

Creating a New View

To create a new view, select View of Data Model from the File menu and then New from the submenu, or choose New Diagram from the File menu and set the Diagram Type to Class or Entity Relationship, the Drawing Method to New View and click OK. The repository is scanned and a window opens displaying all existing entities or classes, and relationships. As you move through the list and select items to go in your view by clicking on their names, they are highlighted. When you select a relationship, that relationship, the reverse relationship and both objects to which they are attached are also selected. If you change your mind about an item, you may click on it to deselect it and the highlight disappears.

Selecting the “Filter…” button on the Select Items for View dialog allows you to enter the beginning text of the entity or class names into the “Select Entities Beginning With:” field. After clicking OK, all entities or classes beginning with the filter text are selected. The “Include Relationships” checkbox includes any relationships connected to the selected entities matching the filter.

Note

- A database view is different from a model view. To create a database view, see SQL View Overview.

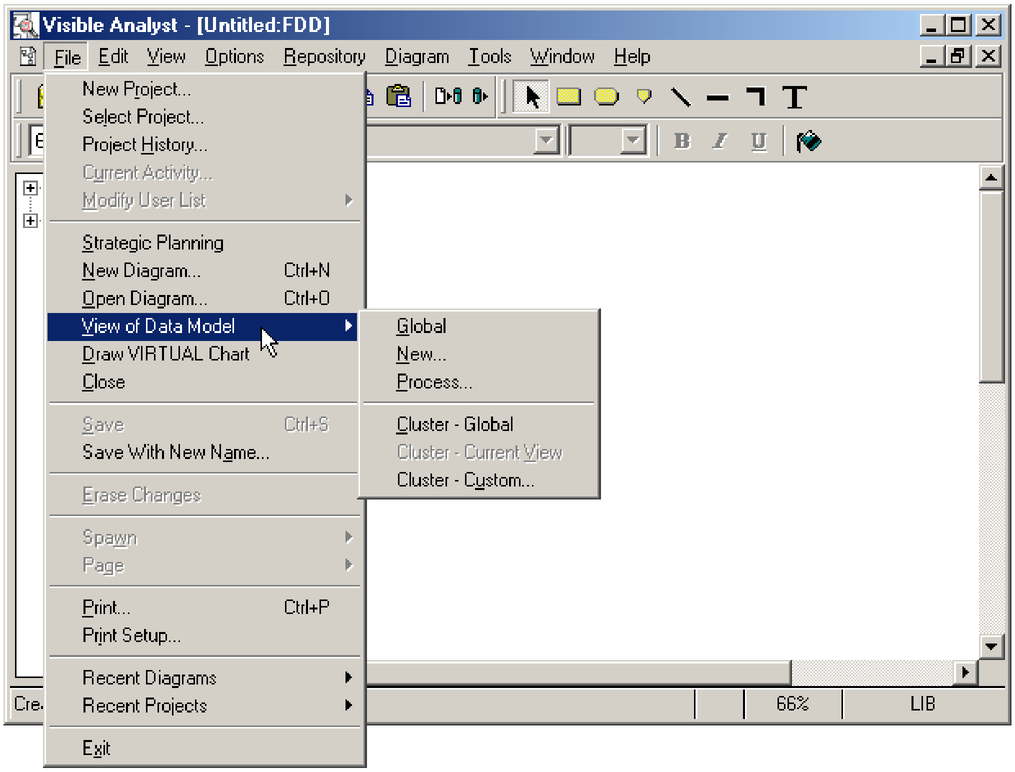


Figure 3-47 The View of Data Model Menu

After selecting the items to appear on the new view, click OK to confirm your choices and have Visible Analyst automatically draw the diagram or click Cancel to start over.

Note

- You can control the scale of a new view drawn by Visible Analyst. The size of the symbols is based on the size of the font you choose to use for symbols. The smaller the font, the smaller the entity symbols. (Symbols only get so large, so huge text does not result in a huge symbol.) The length of the relationship lines and the spacing of the symbols is based upon the size of the font used for relationship lines. The larger the font, the longer the relationship lines and the more widely spaced the entity symbols. To make these adjustments, use the Text Settings selection from the Options menu to set the defaults for these fonts before beginning the creation of the new view.

Creating a Process View

A process view is an entity relationship diagram that represents a subset of your data model and is based upon a process existing on a data flow diagram in the same project. Data elements that enter or leave the selected process in data flows and that are also contained in the composition of entities cause those entities to appear in the process view, along with relationships existing between pairs of entities. (Undesired relationships may either be eliminated manually or by editing the view and modifying it, as described below.) When you select Process, Visible Analyst prompts you to select the name of the process on which to base the view from a list in a dialog box.

After selecting the process, you have an opportunity to confirm your choice. When you do, the components of the new view are automatically selected from the repository and the diagram drawn and named for the process used. Visible Analyst then transfers control to you to edit and/or save the diagram.

Creating or Printing a Global View

When you select Global from the View of Data Model menu, or choose Global View Drawing Method when creating a new diagram, the repository is scanned. The global view (the view of the entire data or object model that exists in the repository diagram) is prepared. Global views can be too large to edit on the screen. However, Visible Analyst is capable of automatically drawing a global view and printing it. After the view is prepared, Visible Analyst lets you know if it is too large to edit and gives you the option to print it. If you decide to do this, it is entered into the print queue and can be printed from the File menu.

See the Note on p. 144 how to use the Text Settings to create a smaller diagram based on a smaller relationship font size.

Modifying the Current View

Entities, classes and relationships may be added to or deleted from a view diagram by selecting Modify View from the Diagram menu. The procedure is identical to creating a new view, except that all entities, classes, and relationships already in the current view are displayed as selected. (Selecting Modify View on a blank, newly created diagram is equivalent to choosing New from the View of Data Model submenu. Since the view is empty, nothing is displayed as already selected.) The diagram is redrawn from scratch using the new list of entities, classes, and relationships.

LINKING MULTIPLE PAGE DIAGRAMS

Visible Analyst provides links between multiple page diagrams with the File menu Page function. You can create a diagram and use Page to link it to additional diagrams. You can then move between the linked pages on your workspace by selecting Page and then Connect. You can also break the link between two pages by selecting Disconnect.

When using Page, remember the following:

- You must use a symbol as the page link between one diagram and another.
- A symbol can be linked to only one other page. For additional page links, use additional symbols.
- You can only page link a diagram to an existing diagram that is stored at the same level in the project tree, or to a new diagram.
- You cannot page link the top-level data flow diagram of any project.
- The page linking situation is somewhat different for data flow or unstructured diagrams and structure charts; see the sections below for explanations. Page doesn't apply to the other diagram types.

Note

- Page links diagrams on the same level in the hierarchy, whereas Nest links diagrams on different levels of the hierarchy.

Page Link Considerations When Using Rules

It is strongly recommended that Page *not* be used on data flow diagrams in a project that has rules enabled. This is because rules treat a multiple page diagram as separate diagrams. Rules capabilities including analysis cannot be applied across linked data flow diagram pages. They can, however, be applied across multiple page structure chart diagrams and decomposition diagrams.

Selecting Page Link Symbols for Data Flow Diagrams

Any symbol in a data flow diagram can be used as the page link symbol, unless it is already used for a nesting link. The symbols shipped with Visible Analyst include the recommended page link symbol for data flow diagrams, shown in Figure 3-48, which is the standard flow charting page symbol. However, you can select any other symbol that is already drawn into the diagram and link it to one other page or add your symbol at the time you wish to make the link. Although almost any symbol may be used, it is recommended that you select a particular symbol from the unstructured diagram symbol set and dedicate it to always represent your page links. The dedicated symbol can be any standard or custom symbol that allows you to easily recognize the link to another page. That symbol can be used repeatedly with incrementing page numbers in any diagram to represent links to additional pages. Although the illustration shows each page link symbol with a "number" label, you may use any label you desire.

Note

With the introduction of multiple pages for the diagram drawing area, most users no longer use or need the Page capability for Data Flow Diagrams.

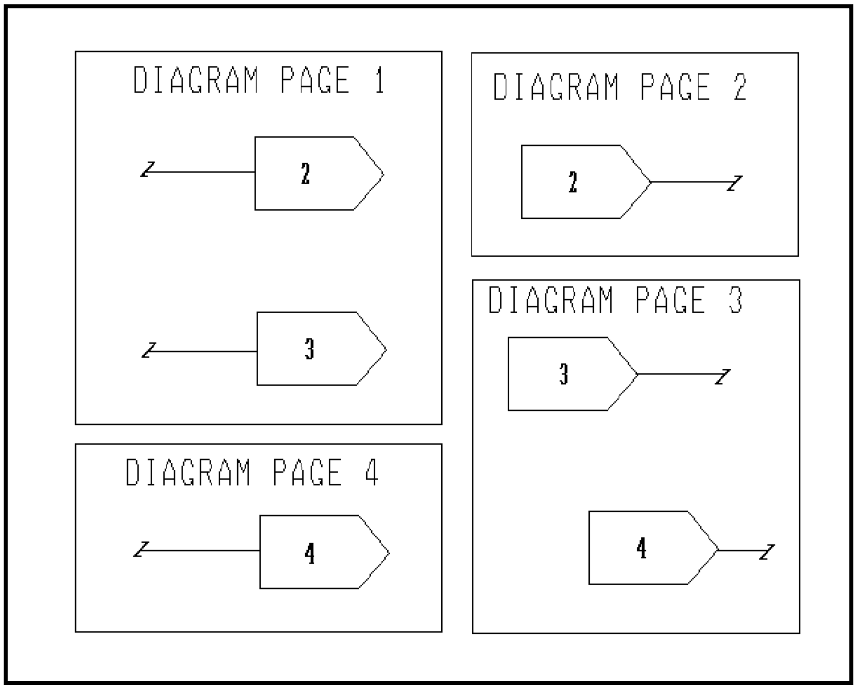


Figure 3-48 Recommended Page Link Symbols for Data Flow

The Page Link Procedure for Data Flow Diagrams

The procedure for linking multiple pages within a diagram is as follows. (Note that this example applies to a data flow diagram. The procedure for structure chart diagrams is the same except that the off-page connector symbol must be used.)

Draw a Page Link Symbol: 1 Draw and label a page link symbol on the currently active diagram. (You can also use an existing symbol already in the diagram as the page link.)

Select the Page Link Symbol: 2 Click the page link symbol to make it the selected symbol, if it is not already selected.

Start the Page Function: 3 Select Page from the File menu and then Connect. If the symbol you selected is already linked to another page, that diagram is activated. If not, the Page dialog box displays.

Choose the Page to Connect: 4 Click either the Select Existing Diagram or the the Create New Diagram button.

- Select Existing presents you with a list of diagrams eligible to be page-connected to the current one (all diagrams in the project tree that are at the same level as the diagram from which you want to page). Select one.

Note

For a structure chart, if there is an existing diagram with a page connector symbol with the same label as the one which you are currently trying to link, it is moved to the top of the list and marked with an asterisk, so you won't have to look for it in the diagram list.

- Create New Diagram creates a new, blank diagram.

Visible Analyst draws a page connector symbol and label on the linked diagram that is identical to the one you used on the connected diagram.

Save the Page Link: 5 Select Save from the File menu to make the page link permanent. The diagrams at both ends of the link are saved.

Page Linking for Structure Chart Diagrams

The page linking function works differently for structure chart diagrams because structure chart diagrams do not have hierarchical relationships that data flow diagrams have, and because it is frequently necessary to use more than one page to represent a large or complex program structure. The basic differences are:

- There are two symbols within the structure chart symbol set that are dedicated page connector symbols. These are the "on-page connector" and "off-page connector," as illustrated in Figure 3-49. Only these symbols may be used.
- The page connector symbols are fully recognized by the rules, and the analysis function for structure chart diagrams reaches across multiple pages that are connected by off-page connectors.
- For structure charts, paging is one directional. Since paging can be many-to-one, once you move across a page link, Visible Analyst doesn't always know where you came from and thus cannot bring you back across the link. However, since both diagrams are then opened, you can easily activate the original diagram.

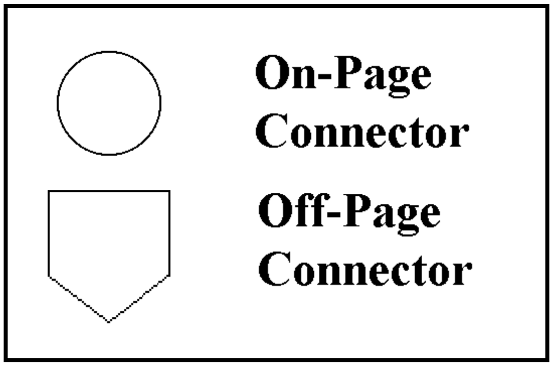


Figure 3-49 Structure Chart Page Connectors

For a detailed description of the page linking procedure, see the previous section. As with DFDs, when you are linking structure chart diagrams, the page connector symbol on the new diagram is given the same name as the originating page connector symbol, in order for the invocation line to be recognized as unbroken.

Note

- There is also a special connect process for structure chart diagrams via an off-page connector by using the Object menu. Draw the input connector on a structure chart, name it and save the diagram. Then click on the connector symbol with the right mouse button to display the Object menu for the connector. Select Connect. You have the option of selecting an existing diagram or creating a new one. After you decide and choose, the output page connector is drawn on the other diagram and the connection established when you Save. After the connected diagrams have been saved, the Disconnect option on the Object menu becomes enabled, allowing you to sever the connection. Note that after the diagrams have been connected, selecting Connect from the Object menu moves you between the diagrams rather than making a new connection.

On-Page Connector

The on-page connector allows invocation lines between structure chart symbols to be broken and then continued starting in another location on the same diagram. This can be necessary at times because of diagram space considerations. The invocation line to be broken is first terminated with an on-page connector that is labeled, and then the invocation line is started again at another on-page connector bearing the same label elsewhere in the diagram. For analysis purposes, the invocation line is considered unbroken. This is illustrated in Figure 3-50. The on-page connector cannot be used to link pages together.

With the introduction of multiple pages for the diagram drawing area, most users no longer use or need the on-page connection capability for Structure Chart Diagrams.

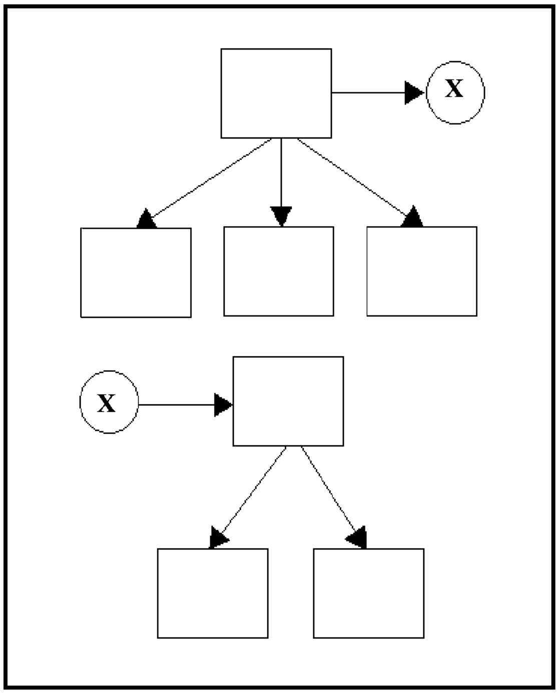


Figure 3-50 On-Page Connectors

Off-Page Connector

The off-page connector symbol serves two purposes:

- To create multiple page structure charts.
- To ensure that an invocation line that crosses a page boundary to a new diagram is recognized for analysis purposes as an unbroken invocation line.

When a multiple page structure chart is created by using the off-page connector, the analysis function can be applied across the various diagrams. The use of an off-page connector is illustrated in Figure 3-51. The same symbol is usually used to connect pages in FDDs and DFDs.

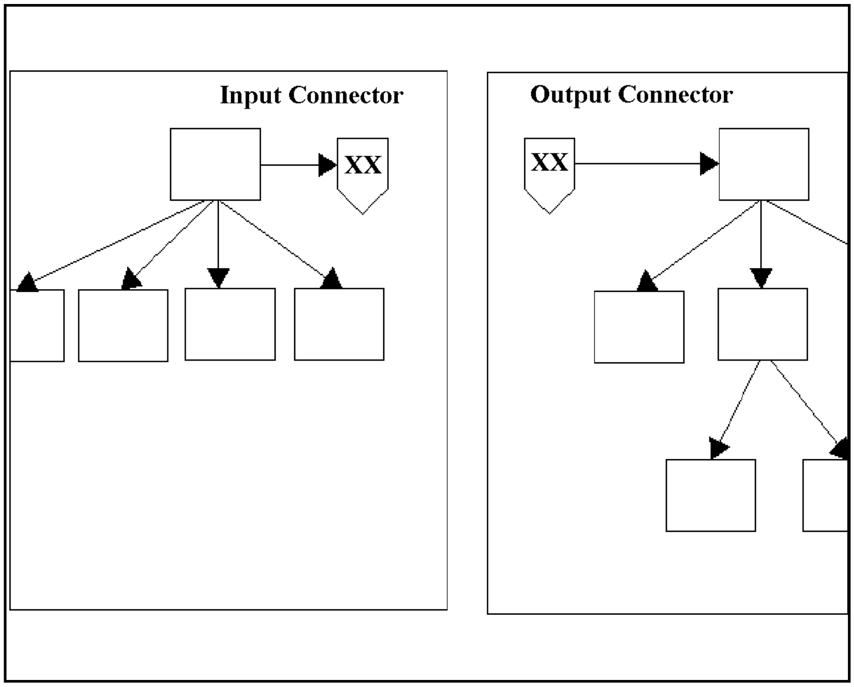


Figure 3-51 Off-Page Connector

Page Linking for FDDs

You can link pages of a functional decomposition diagram in a manner identical to data flow diagrams. The only difference is that with FDDs, the page link is transparent to the rules.

CLUSTER DIAGRAMS

Visible Analyst provides a means to make complex entity relationship diagrams easier to grasp with clusters. A cluster is a group of entities that can be displayed as one symbol on a diagram. Its purpose is to reduce the amount of displayed detail. In a cluster diagram, clusters are shown joined by pseudo-relationships wherever at least one entity in one cluster is joined by a relationship to at least one entity in the other cluster. When using the Cluster feature, remember the following:

- Cluster names should be unique within the data model for the current project.
- An entity may be a member of just one cluster at a time.

- Any cluster diagrams you create are unstructured diagrams. Creating a cluster diagram does not change your existing data model in any way.
- Cluster works only for entity relationship diagrams.

Clusters are created from within the repository. The process is described in The Visible Repository. Although clusters are stored in the repository, they have no real inherent meaning in a data model; thus, any cluster diagrams you create are not a living part of your model. Visible Analyst creates unstructured diagrams that you may edit or print as any other diagram; however, they are no longer linked to the repository and any future changes you make are not reflected in them.

First you must choose the scope of the cluster diagram; that is, whether it displays just the entities and clusters appearing on the current view, all entities and clusters in the repository or a custom-designed diagram displaying any clusters you choose from the whole repository. View of Data Model on the File menu has Cluster sub-functions that are used to generate Current View, Global, and Custom cluster diagrams. They function in the same way as to the other View of Data Model functions described previously.

Selecting any function causes a repository scan. Selecting Current View or Global immediately creates the diagram. Selecting Custom displays a dialog box containing a list of existing clusters and gives you the opportunity to select the ones you want on your cluster diagram. Visible Analyst then creates the diagram. It is possible that a global cluster diagram for a very large data model is too large to edit. If this happens, Visible Analyst lets you know and gives you the opportunity to print it. Visible Analyst sends your cluster diagram directly to the print queue, and you can print it from the File menu.

SAVING YOUR WORK

All of your work during an editing session on a particular diagram can be saved or discarded. When you edit an existing diagram, the changes do *not* become a permanent part of the diagram until it is saved. Therefore, if you want to save your work, be sure to open the File menu and select Save as the last step before quitting work on any diagram, new or old. (Note that Visible Analyst always displays messages to remind you to save your work before allowing you to close a diagram or exit Visible Analyst.) Conversely, if you want to revert to the pre-edited version rather than save the edited version, simply answer No to the prompt when you exit the diagram.

Several of the Visible Analyst functions perform an “auto save” before executing if you have made any changes in a diagram. For example, if you make changes to a diagram then select Define to access the repository, your changes are automatically saved before the function is executed. The functions that perform an auto save are Analyze, Define, Nest, Page, and View of Data Model.

Saving a New Diagram

The only difference between saving a new diagram and an existing one is that you have to provide a name for the new diagram in the dialog box that appears. The only restrictions on diagram labels are that they cannot exceed 128 characters and that they must be unique within the diagram type of the project. After that, Visible Analyst saves the diagram in the hierarchy position that you specified when the diagram was created, if it is a DFD.

EXAMPLE DIAGRAMS

The following pages contain a representative selection of the types of diagrams that can be produced with Visible Analyst, even though formal methodologies are not yet available for many of them. They were all created with Visible Analyst, then imported into the word processing package used to produce this manual.

PLANNING
Enterprise Modeling
Business Modeling Diagrams
Pert Charts

ANALYSIS
Customized Data Flow Diagrams
High-Level System Diagrams
Process Flow Diagrams

DESIGN
State Transition Diagrams

MISCELLANEOUS
Organization Charts
Requirements Matrices
Forms
HIPO Diagrams

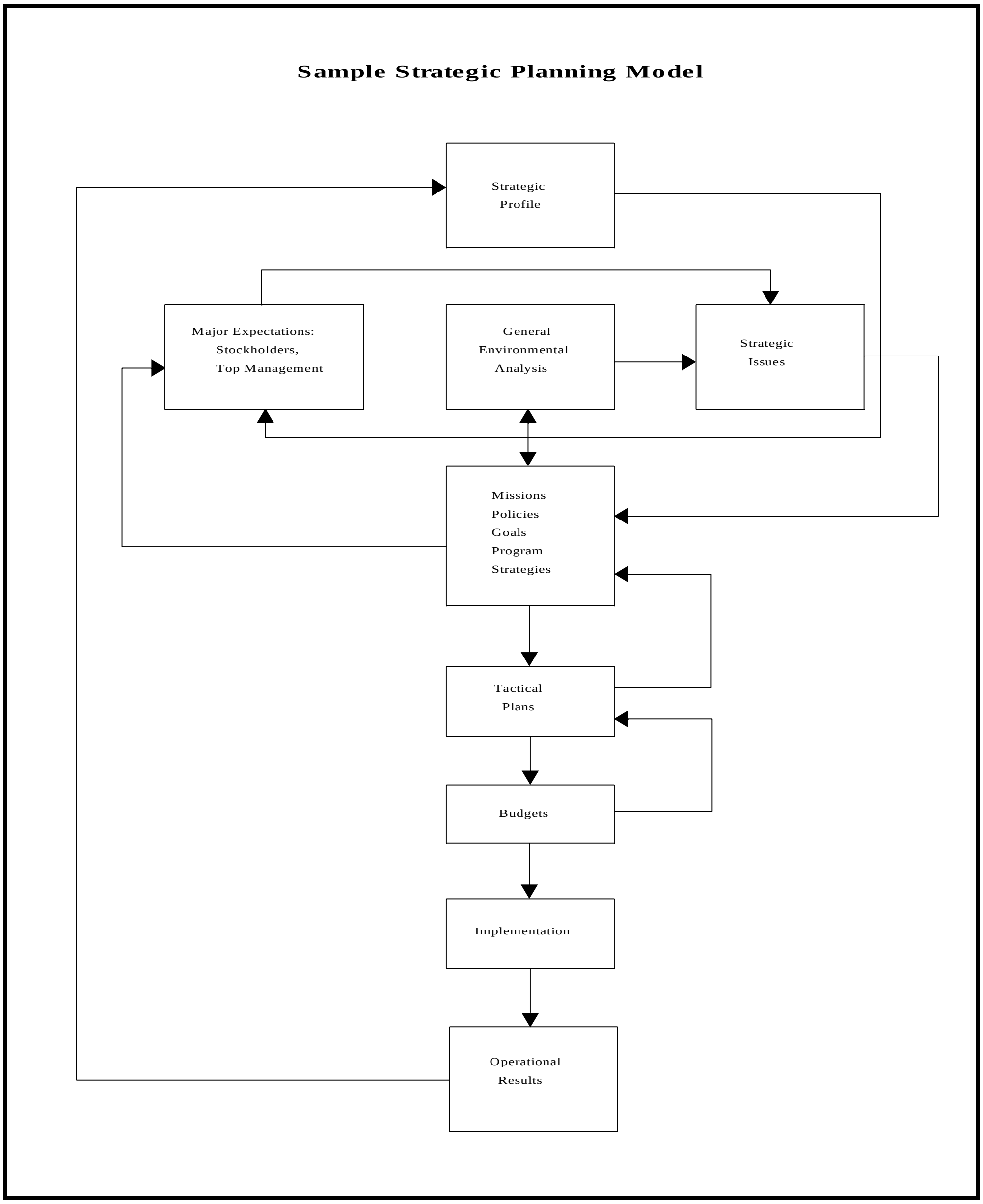

Sample Strategic Planning Model
Strategic Profile
Major Expectations: Stockholders, Top Management
General Environmental Analysis
Strategic Issues
Missions Policies Goals Program Strategies
Tactical Plans
Budgets
Implementation
Operational Results

Figure 3-52 Sample Strategic Planning Model

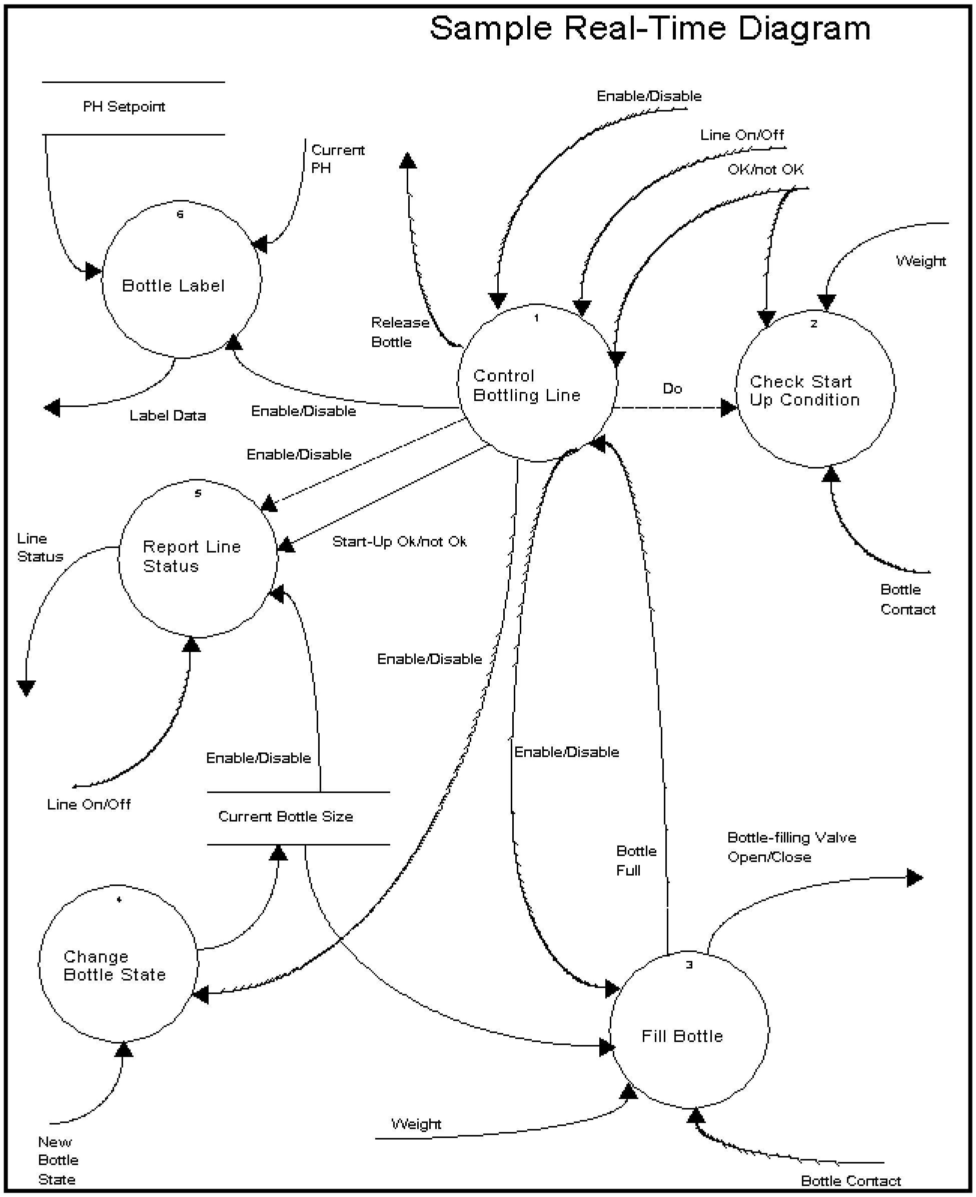


Figure 3-53 Sample Real-Time Diagram

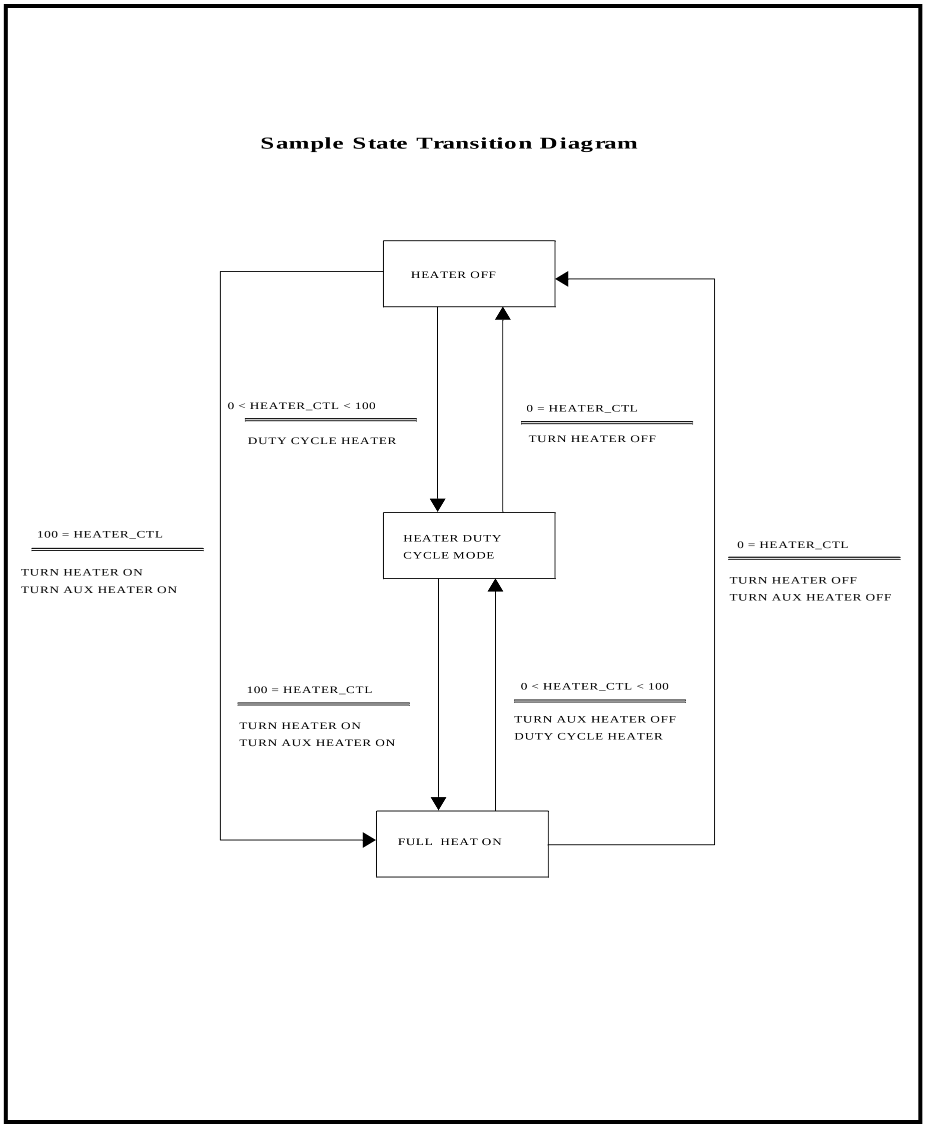


Figure 3-54 Sample State Transition Diagram

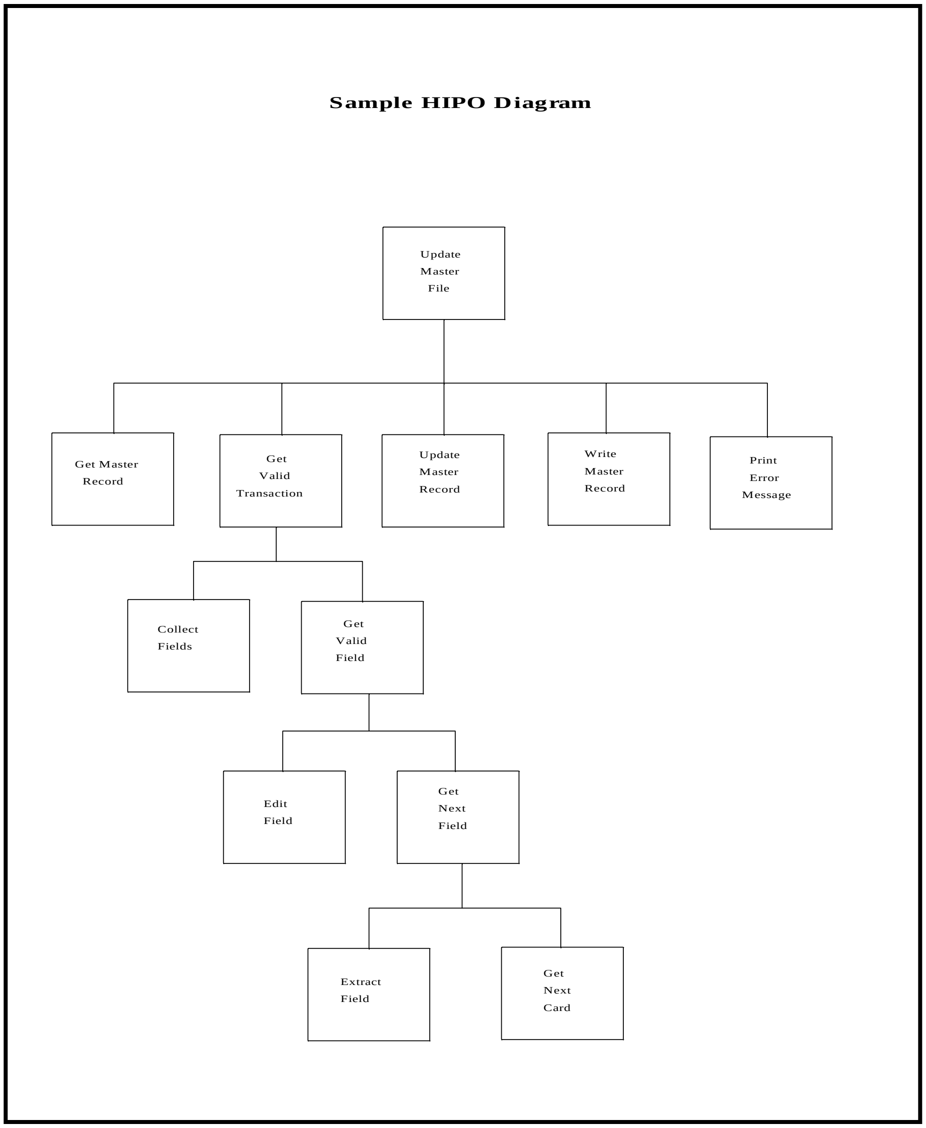


Figure 3-55 Sample HIPO Diagram

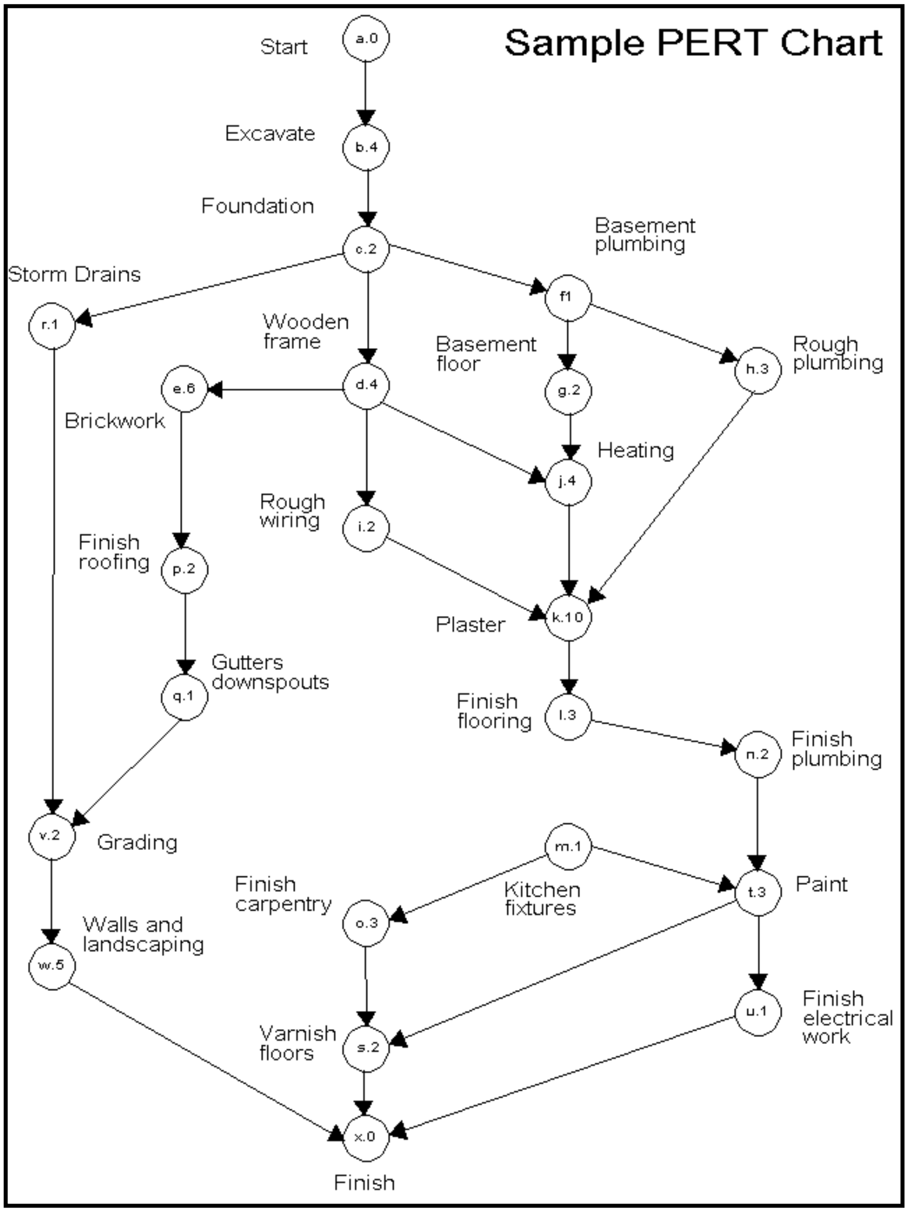
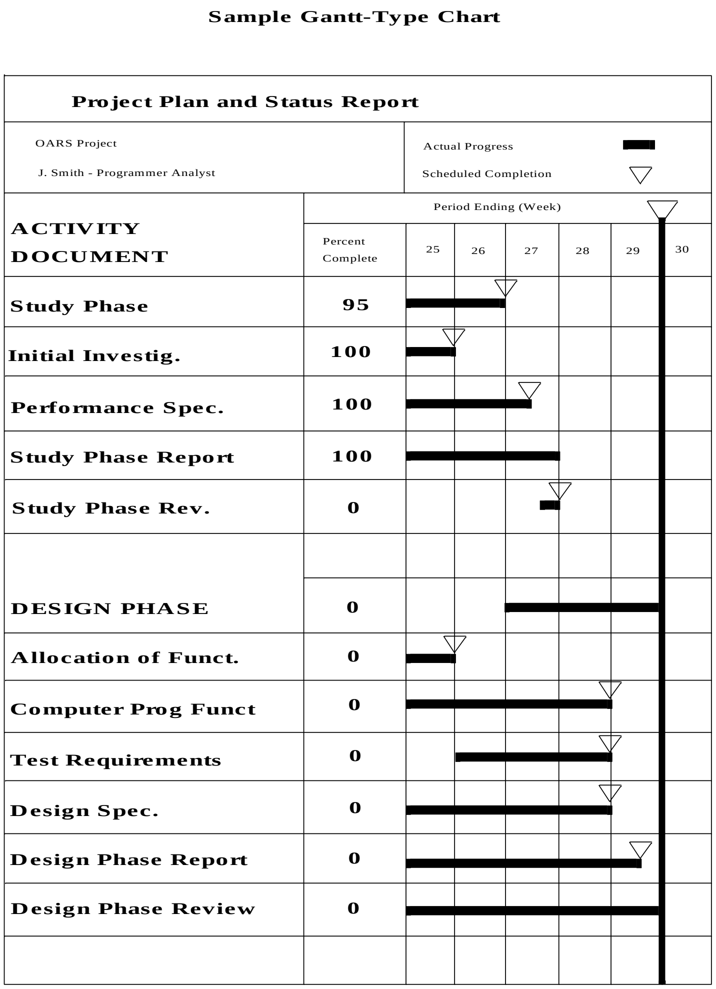

Sample PERT Chart
Start
a.0
Excavate
b.4
Foundation
c.2
Basement plumbing
f1
Storm Drains
r.1
Wooden frame
d.4
Basement floor
g.2
Rough plumbing
h.3
e.6
Brickwork
Heating
j.4
Rough wiring
i.2
Finish roofing
p.2
Plaster
k.10
Gutters downspouts
q.1
Finish flooring
l.3
Finish plumbing
n.2
v.2
Grading
m.1
Finish carpentry
o.3
Kitchen fixtures
t.3
Paint
Walls and landscaping
w.5
Finish electrical work
u.1
Varnish floors
s.2
x.0
Finish

Figure 3-56 Sample PERT Chart



Sample Gantt-Type Chart

Project Plan and Status Report

OARS Project

J. Smith - Programmer Analyst

Actual Progress ▬

Scheduled Completion ▽

Period Ending (Week)

ACTIVITY DOCUMENT	Percent Complete	25	26	27	28	29	30
Study Phase	95						
Initial Investig.	100						
Performance Spec.	100						
Study Phase Report	100						
Study Phase Rev.	0						
DESIGN PHASE	0						
Allocation of Funct.	0						
Computer Prog Funct	0						
Test Requirements	0						
Design Spec.	0						
Design Phase Report	0						
Design Phase Review	0						

Figure 3-57 Sample Gantt-Type Chart

Chapter 4

The Visible Rules

VISIBLE RULES OVERVIEW

The Visible rules module is a set of features and functions that, collectively, allow you to develop projects under the discipline of a structured methodology. Rules support commonly used methodologies, two for structured systems analysis and one for structured systems design, plus data modeling. For structured analysis, the supported methodologies are Yourdon/DeMarco, Gane & Sarson, SSADM, and Métrica. For structured design, the supported methodology is Yourdon/Constantine. For object modeling, both Object Modeling Technique (OMT) and the Unified Modeling Language (UML) can be used. The Business Process Modeling Notation (BPMN) is based on the Business Process Modeling Initiative developed by the Object Management Group (omg.org). The complete BPMN specification can be downloaded at www.omg.org.

Visible rules go further in life cycle coverage than simple support for structured analysis and design techniques. To be most effective, structured design should be based on specifications derived using structured analysis. This capability to integrate analysis and design provides you with the knowledge that your designs reflect the reality of your specifications.

DATA FLOW DIAGRAMS

Structured Analysis Methods Overview

The primary purpose of structured analysis methodologies is to provide techniques that help specify systems. Since most systems today are complex, structured analysis encourages specification of the broader aspects of the system and then decomposition of these larger views into finer and more readily understandable pieces.

The Yourdon/DeMarco and Gane & Sarson analysis methodologies have many similarities. Differences between the two are primarily syntax; however, there are superficial semantic differences that are explained in the following paragraphs. [1] Some of the major concepts of structured analysis include:

[1] For more detailed information on these analysis methodologies, you can refer to the following books:

DeMarco, Tom. *Structured Analysis and System Specification*. Englewood Cliffs: Prentice-Hall, 1978.

- Nested Decomposition
- Data Flow Balancing
- Data Flow Splitting
- Logic, Balance, and Completeness Verification
- Automated Repository Maintenance

Visible Rules functions that support the above are Nest, Split Data Flow, Analyze, and Define. In this chapter, we deal primarily with the first three. The Visible Repository provides a full description of Define. The Drawing Diagrams chapter provides a full description of the Nest function. In this chapter, we deal mainly with the consequences of a Nest operation rather than the operation itself.

To make use of a specific set of rules, Yourdon/DeMarco, Gane & Sarson, SSADM, or Métrica must be the currently enabled method. You can select a data flow methodology to use for a project when you create it, as described in Drawing Diagrams. Once selected, the methodology is applied on a project-wide basis. In other words, rules cannot be selected to apply only to an individual diagram.

As you draw each diagram using rules, the following features are employed:

- Context diagrams. The Yourdon/DeMarco methodology allows you the option of working with context diagrams. Context diagrams are simply top-level diagrams with only one process that represents the entire system to show its relation to its environment.
- Process Numbering. Visible Analyst automatically numbers each process placed on a diagram. The numbers allow you to keep track of all project processes and their nested parent/child relationships. SSADM and Métrica methodologies assign a process number to any process when it is drawn.
- Data Store Numbering. The Gane & Sarson, SSADM, and Métrica methodologies automatically number each data store drawn onto a diagram.
- Data Flow Splitting. Visible rules provide the capability to split data flows into subflows. This is accomplished using the Split Data Flow command on the Diagram menu. A split flow report can also be generated using the Reports function on the Repository menu to list split data flows and their subflows by diagram, by branch (a data flow diagram and all of its children/grandchildren), or by project.
- Analyze Project Diagrams. Visible rules allow you to analyze a project to check for errors within the structural framework of the applied methodology using the Diagram menu Analyze function. Errors detected for a diagram or an entire project can be displayed on the screen or printed in report form.
- Automatic Data Flow Balancing. Visible Analyst automatically drags down (balances) all input and output data flows from parent processes to child diagrams.

Gane, C., and Sarson, T. *Structured Systems Analysis: Tools and Techniques*. Englewood Cliffs: Prentice-Hall, 1979.

Process Modeling Graphics

Methodology Symbols

Methodology symbols for a given diagram type are listed on the Symbols submenu of the Diagram menu. In addition, they are represented by buttons on the control bar. See the topic "**Using Picture Files**" in Chapter 3 for an explanation how to substitute an image file for a methodology symbol.

Process Numbering

All process symbols are numbered. On each DFD, processes are numbered in the order that they are drawn, not in the order that they are labeled. The single process that is allowed in a Yourdon context diagram is always assigned the number 0. All other processes in Yourdon or Gane & Sarson projects are numbered by level and in sequence beginning with the number 1. For example, the first process drawn in any project diagram (other than a context diagram) is assigned number 1. If a second process is added to the diagram, it receives the number 2, the next process number 3, etc.

Process numbers also represent levels of decomposition on a project-wide basis. For example, if process 1 is decomposed into three subprocesses as shown in Figure 4-1, the subprocesses are sequentially assigned numbers 1.1 through 1.3. And if process 1.2 is further decomposed into two other processes, they are numbered 1.2.1 through 1.2.2, and so on.

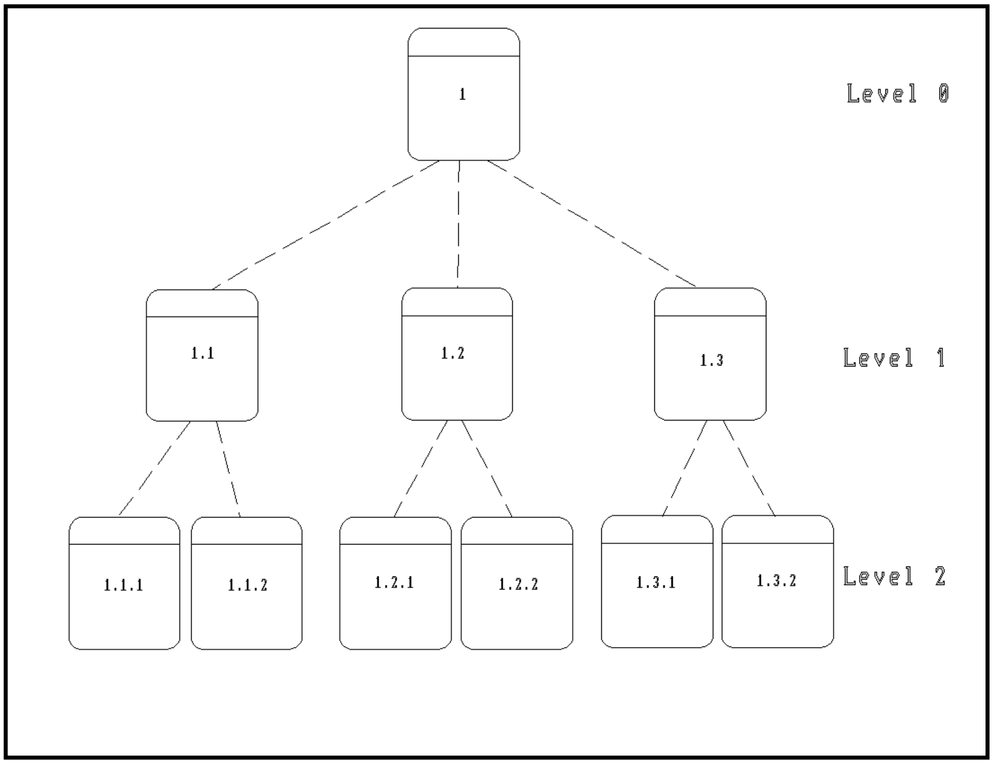


Figure 4-1 Example of Process Numbering Scheme

Although all process numbers are assigned sequentially and according to decomposition level, you can go back and modify the assigned numbers using the Change Item function if you follow a few simple guidelines.

- You cannot duplicate any process number in the same project; if a process number has already been assigned, you cannot use it when changing the number assigned to another process.
- You cannot change the decomposition level being represented; if a process is numbered 1.1.3 you can change it to 1.1.4, but you cannot change it to 1.4.
- You only have to change one number. Visible Analyst carries process number changes throughout the project on a global basis, and any processes on lower levels are automatically renumbered.
- You can reuse a process number from any process symbol that has been deleted from a project.

To change the decomposition levels of the processes on a data flow diagram, use the Detach command to detach the child diagram from the parent process. Save the diagram. Select a different parent process, choose the Explode command, and select the existing child diagram. The Detach and Explode commands are explained in chapter 3.

Data Store Numbering

When working under rules for Gane & Sarson, SSADM, or Métrica methodologies, data store symbols are numbered. Data store numbering differs from the process numbering as follows:

- Each data store number contains the prefix "D."
- Each uniquely labeled data store is assigned a unique number.
- You *cannot* change data store number assignments. When a data store is drawn, it is sequentially numbered; and it maintains the assigned number wherever it appears throughout the project.

If you delete a data store, the deleted data store number is reassigned to the next data store that is added to the project. For example, if a project has data stores numbered D1 through D10, and all data stores numbered D5 are deleted, then D5 is reassigned to the next unique data store added to the project. However, be aware that an interim Save must be performed in order for the reassignment to occur; for example, after you delete D5 you must save the diagram, then add the new data store. If the interim save is not performed, the new data store is numbered D11.

Splitting Data Flows

The Split Data Flow function is enabled on the Diagram menu when the diagram type is data flow and a data flow is selected as the current object. It also appears on the Object menu of a selected data flow. It offers the capability to divide net input or net output data flows into subflows, creating a more detailed representation on lower level diagrams. This capability greatly aids the analysis process by showing more complex data flows at high levels of the structured specification, and smaller or even atomic data flows at the lower levels of the structured specification, thus allowing better understanding of the whole system and of its parts. Only data flows that have labels and have been dragged down from a parent diagram as the result of a Nest operation can be subdivided. The relationship between subflows and their parent flows is automatically maintained within the project repository, so that data flow balancing can be assured whenever the project is analyzed.

Whenever a data flow is split into subflows, the original data flow is erased from the displayed diagram and replaced by the selected subflows. The original flow does not change on the parent diagram, however. The subflows can be repositioned on the diagram by moving them, as described in Drawing Diagrams. A split flows listing can be generated to obtain a listing of all parent flows and their subflows throughout a diagram, a branch or an entire project.

Split Data Flow Procedure

To split a data flow into subflows:

Select a Data Flow or an Existing Subflow:

1 Select a data flow or a subflow of a previously split flow to which you want to add more subflows by clicking on one end of it.

Select Split Data Flow:

2 Choose Split Data Flow from the Diagram menu or from the data flow's Object menu.

Choose How to Split the Data Flow:

3a If the selected data flow has not been previously split, Visible Analyst displays the Split dialog box, giving you three edit boxes to choose from:

- Enter Subflows allows you to manually enter labels for new subflows from your keyboard. See step 4.
- Select Components allows you to specify items that already exist in the repository Composition field of the selected data flow. There must be existing components for this box to be enabled. See step 5.
- Select Flows from Diagram allows you to select data flows that already exist on the diagram. See step 6.

You can choose any combination of the three operations or you can Cancel.

3b If the data flow you select is the product of a prior split operation, Visible Analyst asks you if you want to add more subflows to the parent flow. If you click OK, the above dialog box displays.

If Entering Subflows from the Keyboard:

4 If you chose Enter Subflows in step 3a, you can begin entering labels for as many subflows as you want to use to replace the parent data flow being split. These new flows are added to the repository and to the Composition field of the parent, and appear on the diagram.

If Selecting Components:

5 If you chose Select Components in step 3a, the components that are already defined in the repository Composition field for the selected flow are displayed in the list box. You can select one or more of them.

Note

If the selected items in the Composition field are data elements, these are converted to data flows when selected from the Component field. These flows are now available for use on a DFD. If you want to maintain these data elements as elements and not data flows, enter the new flows in the Enter Subflow section of the Split Flow dialog box.

If Selecting from the Diagram:

6 If you choose Select from Diagram in step 3a, the list box displays the names of all net input or output flows (depending upon the parent flow) on the diagram that are not already subflows of flows appearing on the parent diagram. Click on the ones that you want to designate as subflows of this parent flow. You can select one or more of them. If none are available, the box is disabled.

7 When you finish, click OK. Visible Analyst erases the data flow that is being split (if you are not adding subflows to a flow that has already been split and erased from the diagram) and each subflow is drawn at the edge of the diagram. You can then move the subflows into the positions where you want them on the diagram.

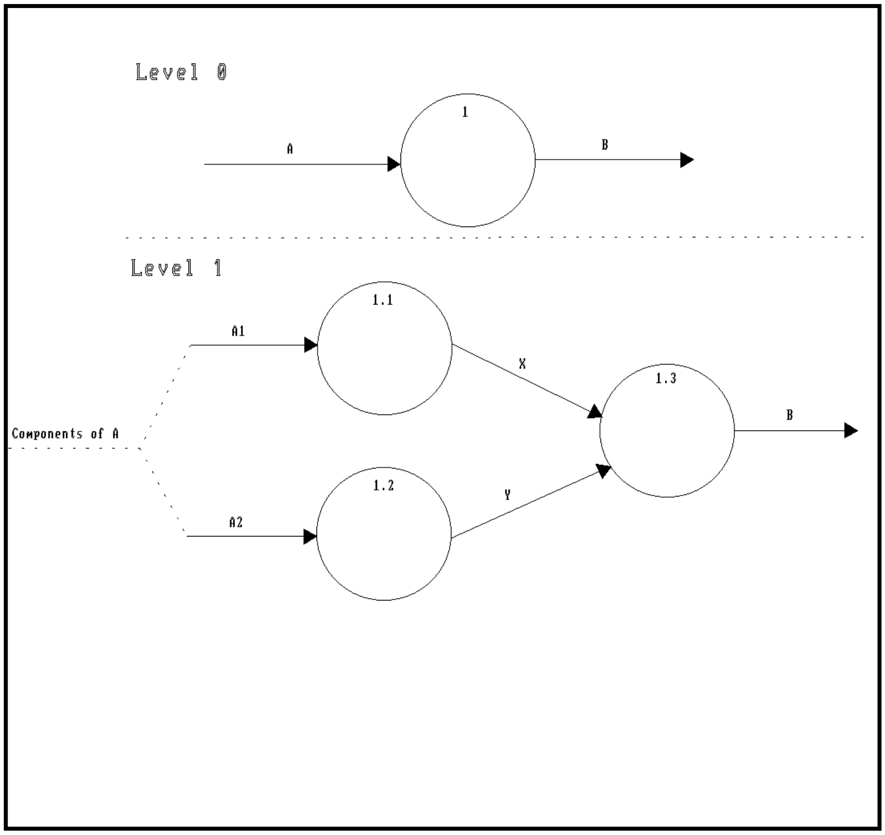


Figure 4-2 Splitting a Data Flow

Analyzing Data Flow Diagrams

As a project undergoes a number of nested decompositions, data flow splits, and various moves or other edits, there exists a significant possibility that various data flows will be incorrectly used, or that objects will be forgotten, etc. This is a natural consequence of the iterative nature of structured analysis. The Analyze function is designed to inform you of completeness and logic errors that exist at any given moment.

The Analyze function checks for a variety of balance conditions by comparing and validating data flows for a single diagram or a complete project at all levels, including the use of subflows. When Analyze is initiated, the following checks are performed:

- Checks all objects (symbols and data flows) for labels and processes for numbers.
- Checks for dangling objects (any unattached symbols or data flows).

- Checks that all processes have at least one input and one output.
- Checks data flow balance. The rule for data flow balancing is: if a data flow is used as a net input or net output (it doesn't matter which) at any level, all occurrences of that data flow on lower levels must be of the same direction (either input or output).

The Analysis Error Message Box

The Analysis error message box, including similar boxes from SQL schema generation, shell code generation, etc., is a special kind of window. It allows you to keep it on the screen, size it, and move it around while doing other things within Visible Analyst. This means that you can keep it on the screen while you correct the errors displayed. You can size it so that it displays only one or two errors at a time while you open and edit diagrams, work in the repository, etc. Working this way is an alternative to printing the list of errors and working from the printed sheet.

Data Flow Balancing Examples

The following examples illustrate data flow balancing situations:

If net input data flow "A," attached to process #1 on diagram 0, appears on diagram 1, attached as a net input data flow to process #1.1, no error condition exists. See Figure 4-3.

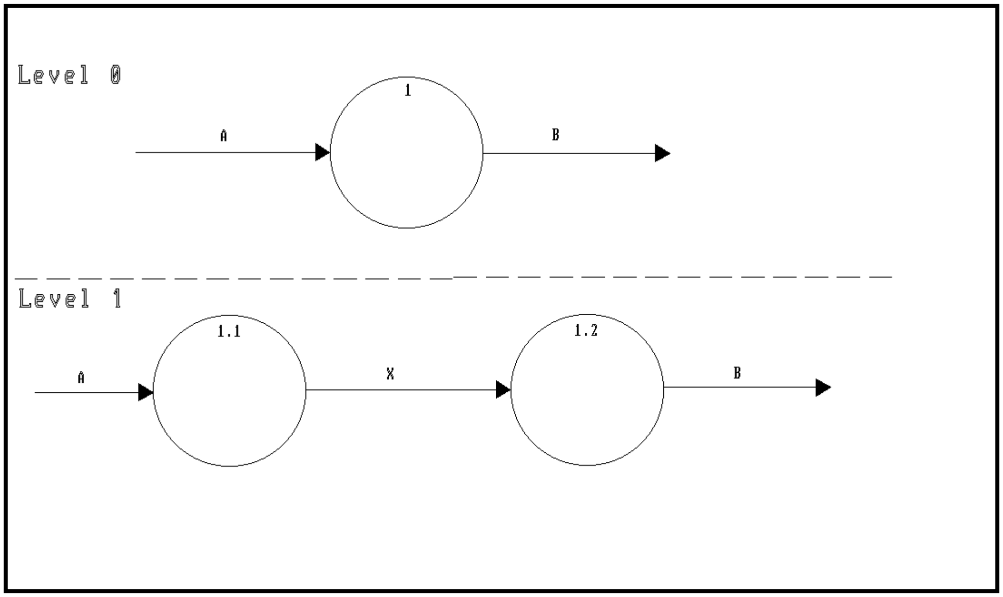


Figure 4-3 Data Flow Balancing

If net input data flow “A,” attached to process #1, on diagram 0, has subflows “A1,” “A2,” “A(n),” etc., all used as net inputs on diagram 1, attached to processes #1.1, 1.2, 1.n, etc., no error condition exists. See Figure 4-4.

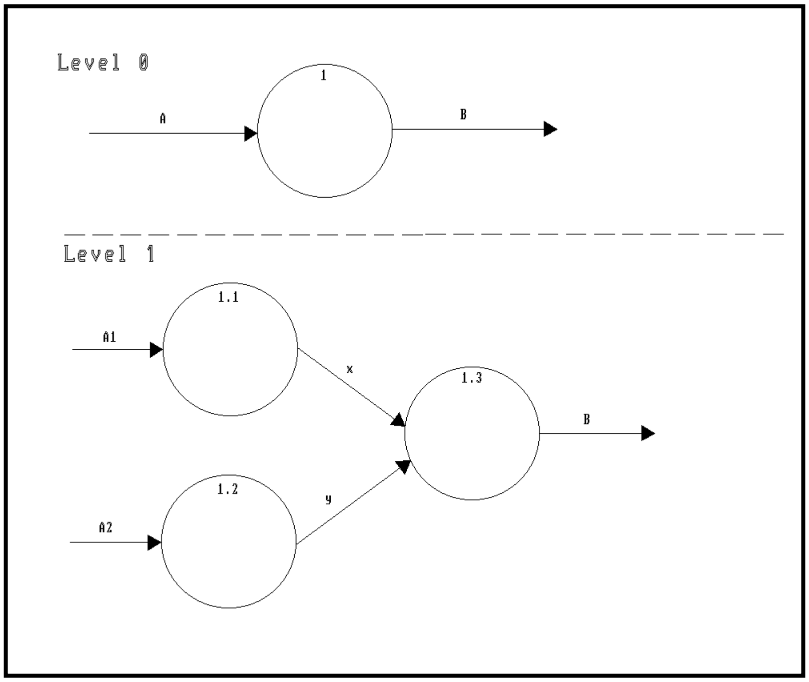


Figure 4-4 Data Flow with Subflows

If, however, a subflow like “A2” is used on diagram 1 as a net output, an error message is generated because data flow “A,” on diagram 0 is a net input, and the system becomes unbalanced. See Figure 4-5.

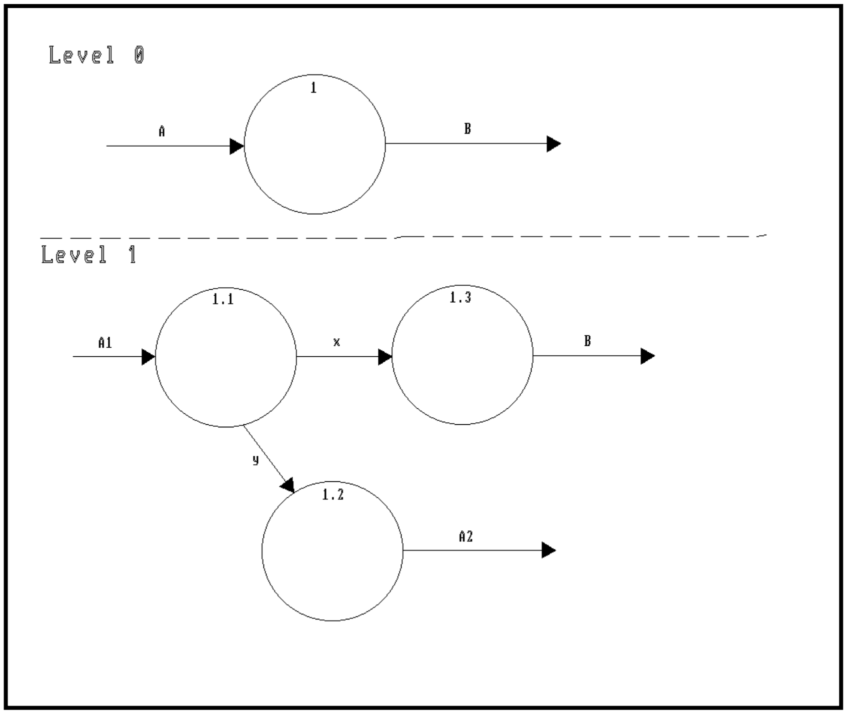


Figure 4-5 Unbalanced Data Flow

If net input data flow "A," attached to process number 1 on diagram 0 is used on diagram 1 as a net output data flow, two error messages are generated, one indicating that it isn't used on diagram 0 as a net output data flow and one indicating it isn't used on diagram 1 as a net input data flow. The reason for this is that flow "A" may very well be used as both input and output, but that the systems analysis hadn't yet indicated both needs. See Figure 4-6.

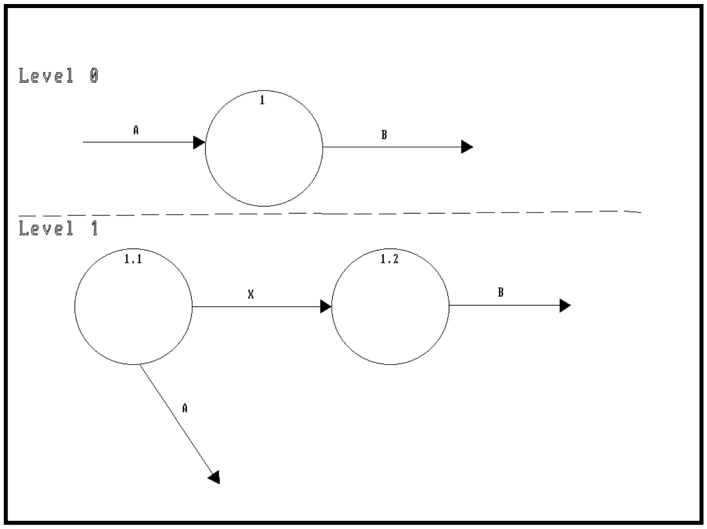


Figure 4-6 Data Flow Errors

If data flow “C” is used on diagram 1 and is not attached to any process on diagram 0, an error message is generated because it is not represented anywhere on the higher level diagram. Note that this might not be valid in the case where flow “C” was a subflow of some other flow on diagram 0. See Figure 4-7.

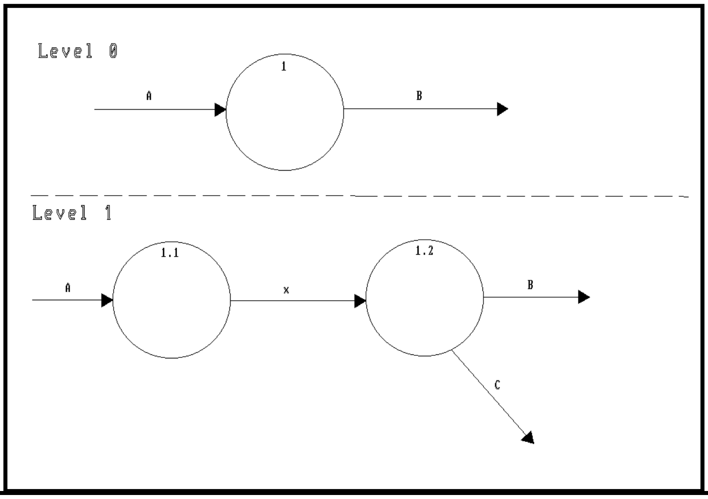


Figure 4-7 Data Flow Errors

Some types of data flows are ignored by the Analyze function for various practical reasons. These exceptions are:

1. Data flows whose labels begin with the word Error or Reject are ignored for the purposes of balancing. These labels have special meaning under the methodologies. They indicate trivial error conditions that are not necessary for the purpose of system analysis, but that are entirely necessary for understanding how a target system works. See Figure 4-8.

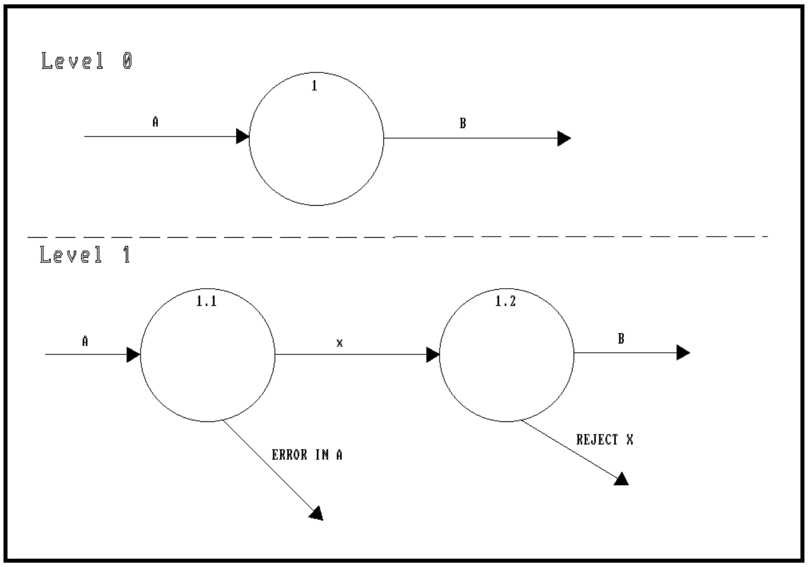


Figure 4-8 Data Flows with Error Labels

2. Local process-to-file I/O data flows are ignored for upwards balancing purposes. If a process has one or more inputs and one or more outputs to the same file (or data store in Gane & Sarson), all the data flows going to that file are considered local file I/O data flows and are not analyzed for upwards balance. However, if the process is exploded using Nest, these flows balance down to the lower level diagram and are then evaluated for balance between the two diagrams. (Figure 4-9)

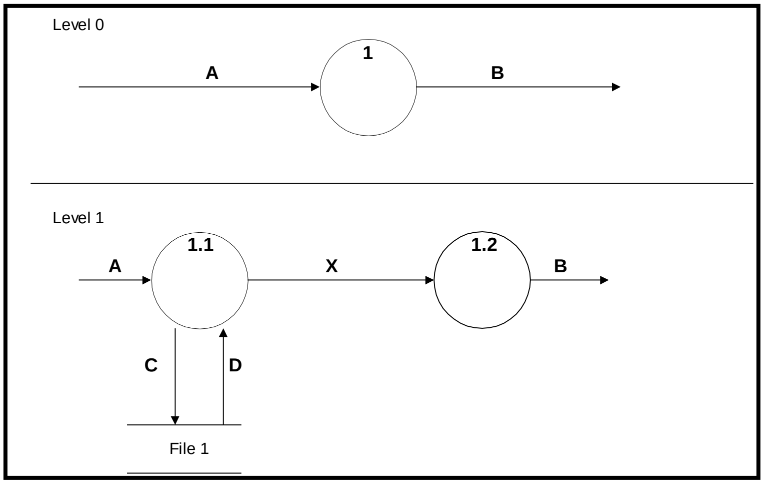


Figure 4-9

3. If a process has an output data flow to a file and a second process receives an input data flow from the same file, both the input and output flows are considered file I/O data flows, and are not analyzed for upwards balance. Here again, if one of these processes is exploded, automatic balancing is enforced and balance is checked in the new parent/child relationship. (Figure 4-10)

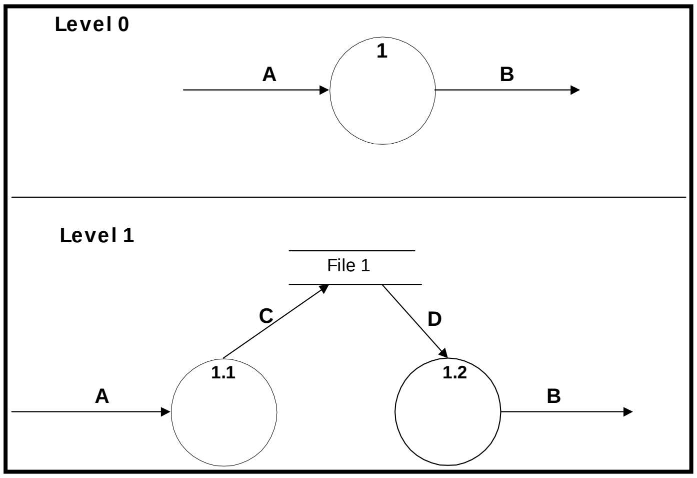


Figure 4-10

4. If a data flow is attached to two processes, it is not a net input or net output data flow, and is not analyzed for upwards balance. If one of the processes is exploded, automatic balancing is enforced, and the new parent/child relationship is validated for balance. See Figure 4-11.

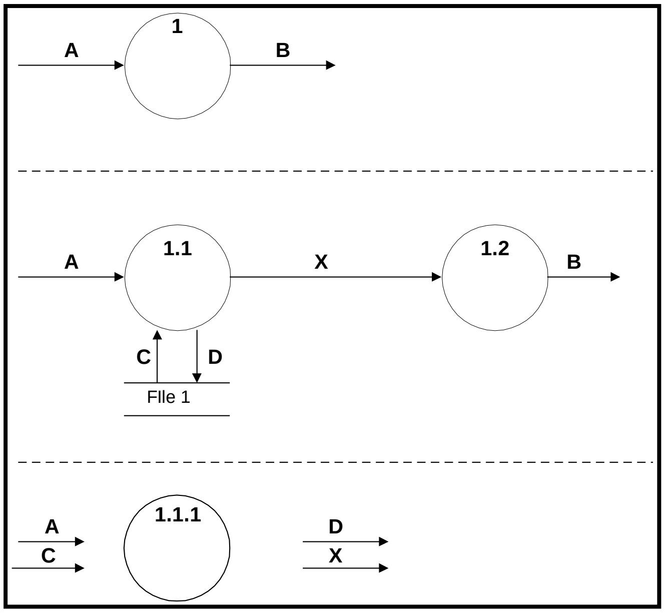


Figure 4-11

Structured Analysis Error Messages

The Visible rules module can generate a multitude of error and warning messages whenever the Analyze function is invoked, presuming there are errors, of course.

The following error and warning messages are generated. Included are brief descriptions of the meaning of the errors and warnings, as well as procedures to resolve them.

ERROR	There are ‘x’ unnamed Process(es).
ERROR	There are ‘x’ unnamed File(s).

ERROR	There are ‘x’ unnamed Data Store(s).
ERROR	There are ‘x’ unnamed Source/Sink(s).
ERROR	There are ‘x’ unnamed External Entity(s).
ERROR	There are ‘x’ unnamed Data Flow(s).

The above group of error messages applies to data flow diagram-based symbols and data flows. Unlabeled symbols and data flows generate error messages because they indicate the incompleteness of a specification. (Note that if an unnamed data flow is attached to file or data store, it is not necessarily an error.) See Figure 4-12.

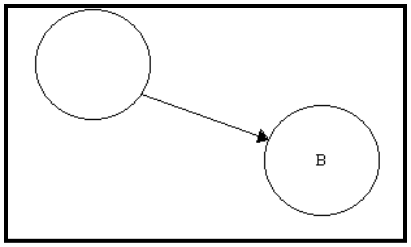


Figure 4-12

To resolve the error, use the Change Item function to label the unnamed symbols and data flows.

WARNING	Data flow attached to file labeled ‘x’ is unnamed.

This warning message indicates that the unlabeled data flow is incompletely described. Though not technically an error, unnamed data flows are flagged to indicate incompleteness. If the process this flow is attached to is nested, this flow assumes the name of the file (or data store) it is attached to when it appears on the subsequent child diagram.

To resolve the warning, use the Change Item function to label the data flow, or the Nest function to explode the process it is attached to, or leave the flow unlabeled. In this case, the data flow is interpreted to mean that it carries the entire content of the file.

ERROR	Process labeled ‘x’ is a dangling process.
ERROR	File labeled ‘x’ is dangling.
ERROR	Data Store labeled ‘x’ is dangling.
ERROR	Source/Sink labeled ‘x’ is dangling.
ERROR	External Entity labeled ‘x’ is dangling.

ERROR Data flow labeled ‘x’ is dangling.

The above error messages indicate incompleteness of analysis. They also indicate, however, that the dangling item’s relationship to the rest of the system is undefined at the point where the item is dangling. To resolve the error, either move the item(s) to attach dangling data flows to symbols, or vice versa, or add any missing flows or symbols. See Figure 4-13.

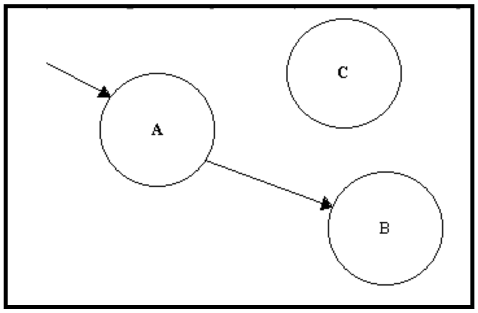


Figure 4-13

ERROR There are ‘x’ unnumbered process(es).

An unnumbered process is also unlabeled. It is illegal for the same reasons as unlabeled processes.

To resolve the error, use the Change Item function to label the unnumbered process. This causes the process to automatically be assigned a process number.

ERROR Process labeled ‘x’ is an input only process.
ERROR Process labeled ‘x’ is an output only process.

The above two error messages indicate incomplete analysis. Processes, by definition, must “process” data. A process without both input and output is therefore illogical and incomplete.

To resolve the error, create or find the necessary input or output data flow and connect it to the process. See Figure 4-14.

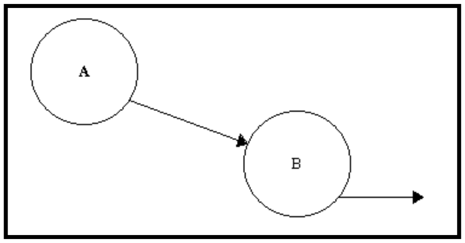


Figure 4-14

ERROR Data flow labeled ‘x’ is not attached to a process.

This error message indicates that the flagged data flow is connecting two non-process symbols (data stores or external entities).

To resolve the error, move the data flow to connect it to a process.

ERROR Diagram named ‘x’ has no parent diagram.

In structured analysis, all diagrams except the top level diagram must be based upon the decomposition of a process. If a diagram is not a top-level diagram and it has no parent, then some high level function is being left out of the specification.

To resolve the error, find or create a process on a diagram at a higher level than the flagged diagram and explode that process using the Nest function. Nest asks you whether to Create New Diagram or Select Existing Diagram. Choose Select Existing Diagram. Visible Analyst presents the portion of the project tree file containing the available diagrams, and you can select the unconnected diagram.

ERROR Input data flow ‘x’ on parent is not shown.
ERROR Output data flow ‘x’ on parent is not shown.

These two messages indicate that a data flow has been added to a higher level diagram but is not reflected in the lower level diagram.

To resolve the error, use the Parent subfunction Nest to open the parent diagram. Then find the process to which the unbalanced data flow is attached. Use the Nest function and Explode that process. This drags the unbalanced flow down to the child diagram where it needs to be used.

ERROR Net input data flow ‘x’ is not shown attached to parent process

ERROR Net output data flow ‘x’ is not shown attached to parent process.

These two error messages indicate that the flagged data flow needs to be attached to the process that was exploded to create the diagram on which the error exists. An alternative resolution is to split the data flow into a data flow previously dragged down to the current diagram. The two resolutions are shown below:

- To resolve the error, first note the process numbers on the current diagram. Then select Nest from the File menu or Object and select Parent subfunction (or just switch to the parent diagram, if it is already open). Visible Analyst loads the diagram where the parent process is located. Locate the process with the appropriate process number (if child diagram processes are 1.1.1, 1.1.2, 1.1.3, etc., locate process 1.1 on the parent). Then find or create a data flow with the same name as the flagged data flow and attach it to this process.
- To resolve the error, locate a data flow on the current diagram that is either dragged down from the parent or an already split subflow. Click on it to select it as the current object. Select the Split Data Flow function and make it a subflow of a data flow that is shown attached to the parent process.

ERROR ‘x’ should be shown as a Net input data flow.
ERROR ‘x’ should be shown as a Net output data flow.

These two errors indicate that a subflow has been incorrectly used in relation to its parent. A subflow on a child diagram of a net input (or output) flow on a parent diagram cannot be used as a net output (or input) data flow or as an internal process-to-process the data flow. Data flows “A2” in Figure 4-5 and “A” on the bottom of Figure 4- 6 would generate such an error.

To resolve the error, locate the subflow and move it to properly attach it.

ERROR Net input data flow labeled ‘x’ is attached to a different object on parent.
ERROR Net output data flow labeled ‘x’ is attached to a different object on parent.

These two error messages indicate that a net input or output data flow is attached to an external entity or global file on the current diagram and is not attached to an external entity or file with the same name on the parent diagram.

To resolve the error, change either the parent or child diagram so that the data flow attachments match.

A Complicated Error

ERROR Net data flow labeled ‘x’ is attached to a local file.

This error message requires some background explanation. Before explaining the reasons for the error message, the definition of a net data flow requires review.

Net Data Flows

Input and output data flows refer to data entering or leaving, respectively, a process to which the flow is connected. The criteria for when a flow is termed a net flow are:

- On a given diagram, a data flow is a net input or output flow if one end of the data flow is not connected to any object or if one end of the data flow is connected to an external entity, like “A” and “C” in Figure 4-15.
- On a given diagram, a data flow is a net input or output flow if one end of the data flow is connected to a non-local data store, like “F” in Figure 4-15. (This somewhat circular part of the definition is explained below.)
- A data flow is a net input or output flow if it enters or leaves the parent process of the current diagram. When a process is decomposed with Nest, the net input and output data flows defined for the parent process are also the net input and output flows for its immediate child diagram. In other words, the net flows associated with the parent process must be accounted for on its immediate child diagram; otherwise, the Analyze operation indicates a balancing error.
- A data flow is a net input or output flow if the data flow is a subflow of a net input or output flow. This means that the property of “net input” or “net output” is inheritable from a parent to a child diagram using the Nest and Split Data Flow operations.

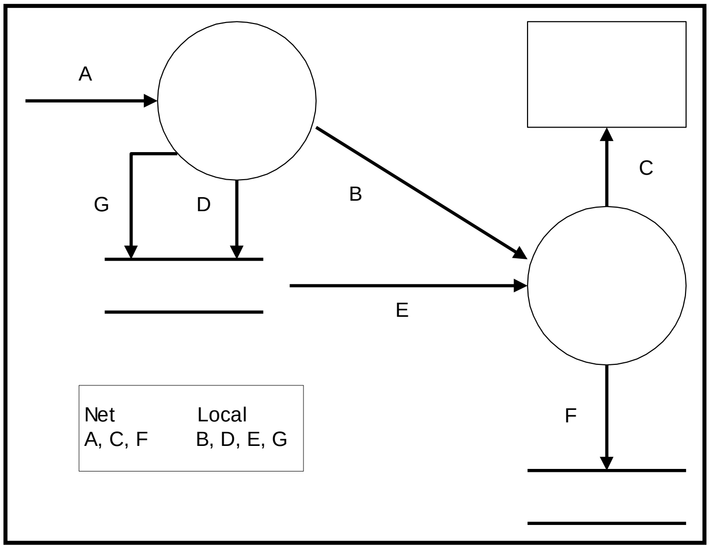


Figure 4-15 Net and Local Data Flows

Any flow not characterized as a net flow on the child diagram is considered internal to the parent process. Hence, it is local to the child diagram and is termed a local data flow.

Data Stores and Physical Files

Although the concepts of data stores and physical files are similar, the terms are not synonymous. Access to a data store must abide by the balancing principles of data flow diagramming. You should adhere to the following guidelines when you draw a data store:

- A data store is a container that temporarily holds data while it is flowing between different processes. It functions as an interface between those separate processes. Beware of cyclic connections drawn between a single process and a data store. A cyclic connection exists if data moves in both directions between a process and a data store. *This situation strongly suggests that the data flowing between the process and data store is really internal to the process.* It would be preferable not to diagram the data store at this level. Instead, decompose the process with the Nest function. On the child diagram, show the processes that have been factored out from the parent process and then draw the data store as it relates to these processes.
- When a process accesses a data store, only the net flow of data should be diagrammed. For instance, if a process withdraws data from a data store, meaningfully generates and/or transforms the data, and then returns some or all of it, only the *net* flow of data should be diagrammed at that level.
- All references to a data store (i.e., the data flows) should appear on the diagram on which the data store is initially drawn. This is called the “defining instance.” Subflows derived from these data flows maintain this data store connection even if the data store does not appear on the diagrams on which these subflows appear.
- All subsequent child diagrams should consistently access the data store. For instance, if the defining instance of a data store is referenced by process “1” with only net output data flows, then only net output data flows may be associated to this data store on the child diagram of process “1” without generating an analysis error.

The Error Message Explanation

When a local data flow is connected to a data store as an input to that data store, the data store inherits the local attribute of the data flow and *that instance* of the data store is viewed by Analyze as a local data store. This represents the “defining instance” of that *local* data store because it contains data derived within the parent process from the net inputs connected to the parent process. If a net input flow is connected directly to this local data store, then a mixture of net and derived data can occur. (Net and local *output* data flows work the same way.) Allowing this mixture of derived and net data would negate the reconciliation effort intrinsic to the methodology of data flow diagramming. Thus, Visible Analyst generates this analysis error when this situation occurs. For example, if data flow “G” in Figure 4-15 were shown on the parent diagram as a net output data flow, it would generate this error message because it is used “locally” on this diagram.

After generating this message, Analyze thereafter looks at data flow x as a local data flow, no longer having the property “net input” or “net output.” Analyze then tries to balance the net flows appearing on the parent and child diagrams and is forced to issue the additional error message:

ERROR Input/Output data flow ‘x’ on parent is not shown.

To resolve both errors at once, do one of the following:

- Make the data store to which x is connected non-local by allowing only flows entering it (if x is a net output flow) or only flows leaving it (if x is a net input flow).
- Add an additional instance of the data store to the child diagram, one to be used as a local data store and the other to be connected to data flow x.
- Disconnect data flow x from the data store, leaving one end connected to a process and the other connected to nothing. (Remember that the data store connected to x or its parent flow on the parent diagram remains implicitly connected on all descendent diagrams.) Then the data store is only a local one on this diagram.

The source of this analysis error message might indicate that your analysis is affected by thinking that is too “physical.” It is sometimes difficult to remember, especially for data stores, that data acquisition or transformation at a lower level might have to be propagated either up or down a diagram branch.

To put into practice the guidelines we have presented, we suggest that you identify the net input and output flows for a process and create the initial version of each child diagram without incorporating *any* data stores. (An equivalent data flow diagram can always be drawn without them!) Afterwards, when all the data flows on the diagram have been documented, insert the data stores, adhering to the previously discussed data store guidelines. Then your project is both methodologically correct and an accurate description of your system.

FUNCTIONAL DECOMPOSITION DIAGRAMS

Business Planning Analysis Methods

Functional decomposition diagrams (FDDs) give you the ability to do high-level planning of business functions diagrammatically while concurrently populating the repository. [2] You can enter business functions that you define onto diagrams and break them down into successively finer gradations. At some point, one that is entirely up to you, you can break down business

[2] Some of the theory behind functional decomposition diagrams can be found in:
Martin, J., and McClure, C. *Structured Techniques for Computing.* Englewood Cliffs: Prentice-Hall, 1985.

functions (hereafter called simply functions) into processes. These processes are semantically equal to the processes that appear on data flow diagrams. The processes can themselves be broken down into smaller parts (still lower-level processes) on FDDs.

Functional decomposition diagrams are viewed as the highest level of business planning. It is probably the place you want to begin when you wish to display the broadest picture of the functions of an enterprise. There is no rule that you *must* begin here, but other things are easier if you do. For designing individual projects, it might be just as effective to start with a process model or a data model (or both at once) if you feel that the project does not have the breadth to warrant planning at the FDD level.

After you diagram your highest-level business functions and decompose them somewhat, you get to the level where you feel functionality below this should be subject to the detailed type of analysis possible with DFDs. This is the point at which you want to show finer gradations of functionality as processes. You can lay out all of the processes of this branch of your FDD and their hierarchical relationships. After that, you can instruct Visible Analyst to produce a set of DFDs for this branch of processes. Then you can flesh out the DFDs and add data flows, files, external entities, etc.

Although it is possible to go back and forth between FDDs and DFDs generated from them, it is likely that the editing you do on the different types of diagrams will introduce problems, such as making incompatible changes in the hierarchical relationship between processes on the two types of diagrams. Therefore, it is recommended that you complete your business analysis on FDDs as far as possible, at least for a given branch of the diagram, and then generate DFDs and flesh them out. If you later find that you need to add processes, it is better to return to the FDD and add them there. You can then have Visible Analyst place the new processes onto the proper DFDs. Proceeding in this manner keeps the two sets of diagrams in sync and avoids compatibility problems.

Note

- A *functional* decomposition diagram is very different from a *process* decomposition diagram. The former is a full diagramming methodology for doing business planning. The latter is simply an unstructured diagram laying out the hierarchy of processes that are descendants of an indicated process.

Rules Functions

Visible rules functions that support the above are Spawn, Change Item, Analyze, Page and Define. FDDs also have full repository support, overlapping somewhat with DFDs. As explained in Getting Started, the functionality of Spawn is somewhat similar to but not identical to that used with the Nest function for data flow diagrams.

Since there is some overlap between DFDs and FDDs, it is significant which process modeling methodology (Yourdon/DeMarco or Gane & Sarson) you choose when you create a project. You should consider this carefully before you save the first diagram of a project. Also, it is helpful to be familiar with the terms and techniques of the process modeling methodology before continuing with this description of functional decomposition diagrams. This description is given in the early pages of this chapter.

Naming Rules for FDDs

There are a couple of points to be aware of in addition to the standard Visible Analyst naming rules of no more than 128-character names and labels cannot use certain delimiter characters, such as ! @ # $ % ^ & * ;. A function cannot have the same name as a process in the same project.

- Otherwise, a function can have any other valid Visible Analyst name, regardless of whether any other object is using that label within the project.
- Any process can occur on one FDD and on one DFD.

The Rules in Action

As you create functional decomposition diagrams, the following features of Visible rules are employed.

- Repository entries are created. When you show hierarchical relationships between functions and/or processes, they are known to Visible Analyst, even though you cannot see them if you look at the repository entries.
- You can Spawn a function that you have diagrammatically broken down into one or more processes. The first time you use Spawn on a function, a branch (subtree) of DFDs including these processes is created. Subsequent uses of Spawn allow you to move from your FDD to the DFD at the top of the branch, to resynchronize the DFDs with your FDD, or to break the spawn link between them.

  If DFDs exist at the time of your spawn request, Visible Analyst attempts to integrate these into the proper relationship with your FDD. Be aware, however, that this might be impossible due to extreme contradictions between how you specified relative process hierarchies on each of your diagram sets. In this case, Visible Analyst lets you know this and suggests that you run Analyze to find out just what the problems are.

- After a Spawn operation is complete, the processes that appear on the created DFDs are numbered in the usual manner for DFDs. However, these numbers do not appear on your FDD.
- Change Item, from the Diagram menu, allows you to convert a function into a process and vice versa, in case you change your mind about something after you have entered it on the FDD. You cannot change a process into a function if it already exists on a DFD, and you cannot change a function into a process if it has been spawned.
- You can Analyze your FDDs, either individually or as a group, to find inconsistencies and violations of methodology rules. These rules and the analysis process are explained below.

Analyzing FDDs

Overview

Visible rules check four classes of errors:

- Syntax errors. These are errors that make your FDD impossible for Visible Analyst to understand.
- Connection errors. These indicate violations of the rule that a function is of a higher order than a process and make it impossible to generate spawned DFDs.
- Balancing errors against DFDs. These are things wrong with a FDD that make it incompatible with existing or planned DFDs.
- Page connection errors. These are errors involving improperly specified page connections between parts of an FDD, incorrect or missing connector labels, etc.

There are two types of analysis:

- Current diagram. This checks for the first three of the above errors for the currently selected FDD.
- Entire project. This examines all of the FDDs in the current project (Analyze moves across page connections) and checks for all of the above errors.

It is recommended that you run Analyze before executing Spawn on your FDD for the first time to catch errors and allow you to correct them early, when they are easiest to fix.

Functional Decomposition Error Messages

Syntax Errors

ERROR There are ‘x’ unnamed function(s).
ERROR There are ‘x’ unnamed process(es).

These errors are the result of not labeling functions or processes on your FDDs.

To resolve the errors, find and label them using the Change Item function.

ERROR Function labeled ‘x’ is a dangling function.
ERROR Process labeled ‘x’ is a dangling process.

The above error messages are the result of not connecting the indicated items with connector lines.

To resolve the error, move the lines and/or the symbols to appropriately connect them.

ERROR Function labeled ‘x’ has more than one input connection.
ERROR Process labeled ‘x’ has more than one input connection.

The above error messages are the result of connecting the indicated item to more than one superior item, like function “D” in Figure 4-16.

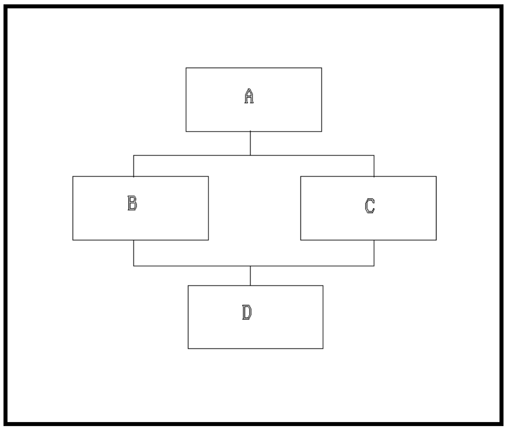


Figure 4-16

To resolve the error, move or delete lines to connect the function or process to one and only one superior symbol.

Balancing Errors Against DFDs

ERROR Process labeled ‘x’ has a different parent on a data flow diagram.

The above error message indicates incompatible function/process hierarchies on your FDD and DFDs. This is probably due either to independently making changes to these diagrams or to spawning your FDD to an already existing branch of data flow diagrams. This situation can occur either on a single FDD or across a page connection.

To resolve the error, first decide whether your FDD or your DFDs have priority. If you use functional decomposition for your overall business planning, it should probably be given priority. In this case, edit the DFD to make it matches the FDD and then rerun Spawn Verify on the parent function to synchronize the diagrams and to make sure that the error message goes away. If your DFDs have priority, you can either correct the FDD or you can do nothing and live with this error message and the fact that the FDD and DFDs are out of sync.

WARNING Process labeled ‘x’ is not used on a data flow diagram.

The above warning message indicates that either the FDD containing the process has not been spawned or that the process has been deleted from a DFD.

To resolve the warning, Spawn the FDD (if it has never been spawned), or add the process to a DFD manually or with the Spawn Verify function.

Connection Errors

ERROR Function labeled ‘x’ is placed illegally in the hierarchy structure – its parent is a process.

The above error message is the result of violating the hierarchy rule for functions and processes, like function “D” in Figure 4-17. This situation can occur either on a single FDD or across a page connection.

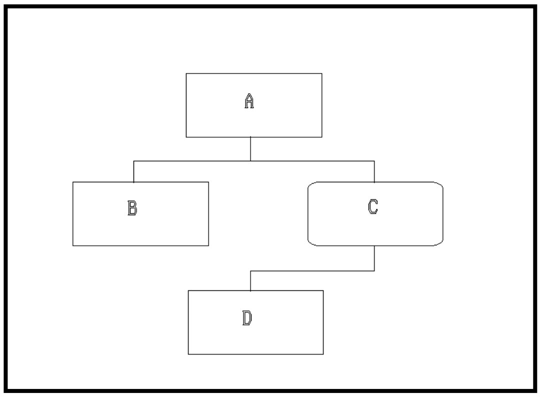


Figure 4-17

To resolve the error, use the Change Item function to make the function into a process or the superior process into a function.

ERROR There are ‘x’ lines only connected at one end to a methodology symbol.

The above error message is the result of improperly connecting two symbols, as between functions “A” and “C” in Figure 4-18. The elbow connector to function “C” was not drawn starting at function “A”, but at the elbow of the connector joining functions “A” and “B.” It is, in actuality, if not in appearance, dangling. Note that a visibly dangling connector line would also generate this error message.

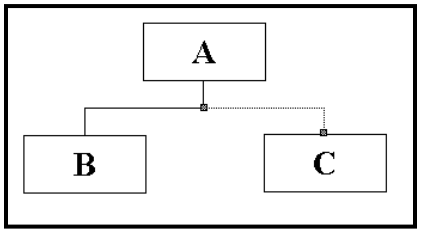


Figure 4-18

To resolve the error, redraw the line so that it truly joins functions “A” and “C.”

Page Connection Error Messages

ERROR Page connector labeled ‘x’ is dangling.

The above error messages are the result of not connecting the indicated page connector with connector lines.

To resolve the error, draw or move connector lines to connect the function or process to an appropriate other symbol.

ERROR Duplicate appearance of page connector ‘x’.

This error message is issued when two or more instances of either an input or an output page connector are assigned an identical label.

To resolve the error, either delete or change the label of the duplicate instance of the page connector.

ERROR There are ‘x’ unnamed page connectors.

These errors are the result of not labeling page connectors on your FDDs.

To resolve the errors, find and label them using the Change Item function. The page connector symbols on each end of the link should have the same name.

ERROR Page connector labeled ‘x’ is placed on wrong diagram.

This error message occurs when the input page connector symbol is located on a different diagram from the page that is actually connected.

To resolve the error, delete the misplaced page connector symbol and recreate it on the page that is actually connected. (You might want to use the Construct function to do this.)

ERROR Page connector labeled ‘x’ has more than one connection.

The above error message means that either the page connector has more than one hierarchical parent, as in Figure 4-19, or that it has been used to substitute for a connector junction (it has more than one hierarchical child) as in Figure 4-20.

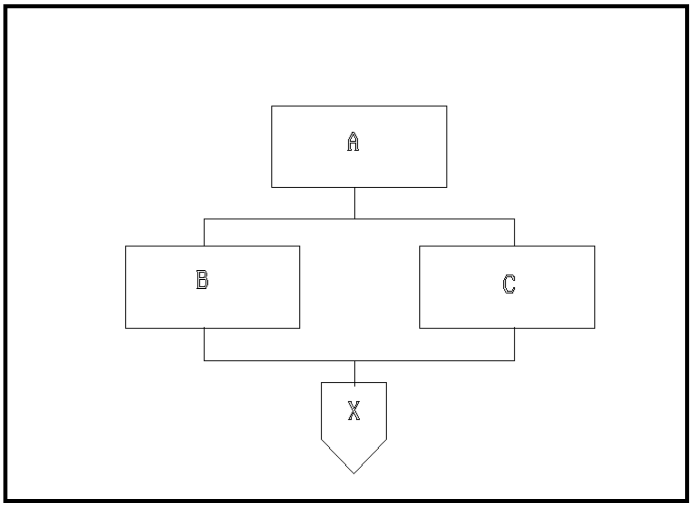


Figure 4-19

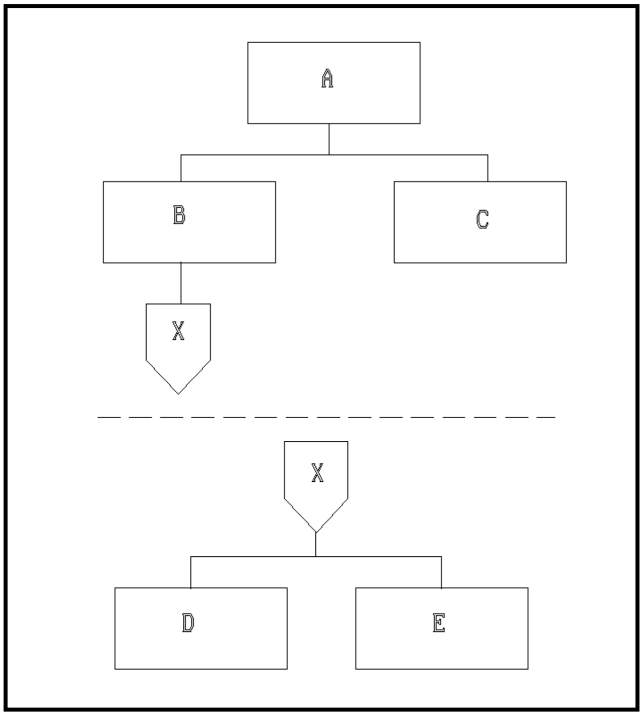


Figure 4-20

To resolve the error in the first situation, break one of the connections from a function or process to the page connector. In the second circumstance, move the parent of the page connector to the paged diagram, as in Figure 4-21.

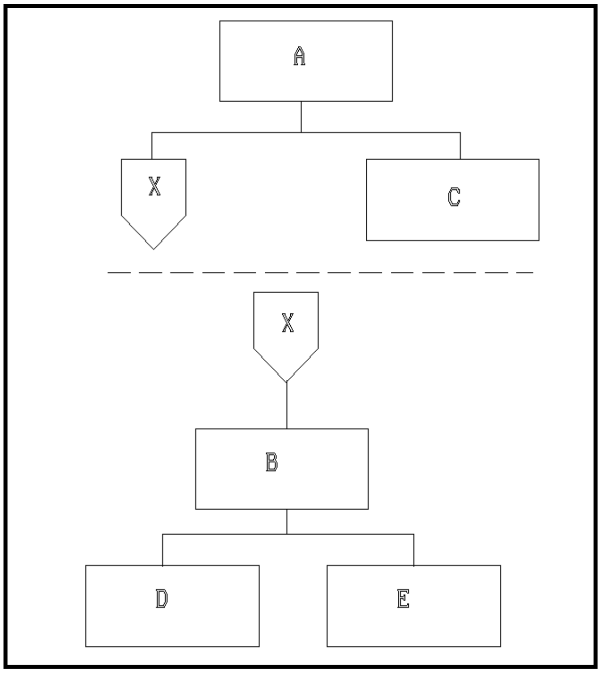


Figure 4-21

ERROR Valid matching page connector cannot be found for ‘x’.

The above error message indicates that either the two page connectors for a page connection are not labeled identically or that one or both are not labeled at all.

To resolve the error, change the name on one page connector to match the one at the other end of the connection.

ERROR Unexpected appearance of page connector labeled ‘x’.

The above error message indicates that Visible Analyst could not match up pairs of page connectors. Page connectors must be paired: one input and one output. The many-to-one page connections acceptable for structure charts are not valid on FDDs. Analyze requires unique and identical labels on each end of a page connection. If two page connectors on a diagram are not labeled identically, Visible Analyst is unable to resolve the connections, as in Figure 4-22, and issues this message. The Page operation automatically copies a labeled page connector to the newly connected page to insure the existence of both connector symbols and avoid this error.

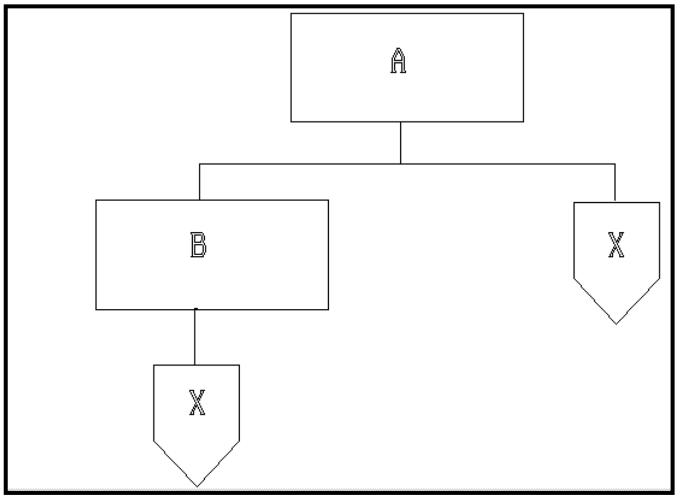


Figure 4-22

To resolve the error, either eliminate one occurrence of the named page connector or rename one of them to something else.

STRUCTURE CHARTS

Structured Design Overview

Structured design is a discipline that is complementary to structured analysis and, in fact, builds upon it. The purpose of structured design is to provide a technique for translating specifications generated using structured analysis into computer programs. As such, it is almost a graphical programming technique. Structured design is practiced in a more subjective manner than is structured analysis, meaning that there are fewer hard and fast rules for how a given project analysis should translate into a structured design. After all, even badly written programs can work.

Visible rules implementation of the Yourdon/Constantine[3] structured design methodology is intended to maintain as much design freedom as possible for the user, while providing design

[3] For more detailed information on the Yourdon/Constantine structured design technique you can refer to the following books:

evaluation against known poor design practices. The error and warning messages generated are intended to be used as guidelines rather than rules.

Visible Analyst support for the Yourdon/Constantine methodology includes four major areas. These are symbol support, line and connector support, analysis support, and repository support.

Structure Chart Graphics

Structured design is primarily a program description or specification technique. As such, the symbols, lines, and connectors used by the technique represent programming constructs. The symbols used[4] are as follows:

Structure Chart Symbols

Module
Macro
Data Only Module
Library Module
Library Macro
Information Cluster
On-Page Connector
Off-Page Connector

As with DFD symbols, they can be accessed from the Diagram menu or the control bar. All of the symbols supported under Yourdon/Constantine methodology are program modules of some type, except for the on-page and off-page connector.

Data only modules and information clusters are specialized types of modules. Modules, macros, and library modules are types of normal program modules. However, there are some differences between them.

- A module represents a contiguous sequence of program statements, in other words, a program, subroutine, function, etc., depending upon the target development language. Its interfaces to other modules (of all types) are not hidden from the designer.
- A library module is "a module that always executes in the same way on each separate activation, as if it were a fresh copy of the module."[5] It is differentiated from a module in usage by the fact that its interfaces to other modules (if any) are hidden from the designer.
- A macro is "a module whose body is effectively copied in-line during translation (e.g., compilation or assembly) as a result of being invoked by name; that is, the bounded

Yourdon, E.N., and Constantine, L.L. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Englewood Cliffs: Prentice-Hall, 1986.

Page-Jones, M. *The Practical Guide to Structured Systems Design.* Englewood Cliffs: Prentice-Hall, 1988.

[4] For a more complete and formal definition of these symbols, see Yourdon/Constantine, p.456.

[5] *Ibid*., p. 459.

contents replace the reference to the aggregate identifier." [6] The only practical difference between a macro and a module is that macros tend to make a system operate faster, but at the expense of more memory space required to operate that system.

- A library macro is a macro whose interfaces to other modules (if any) are hidden from the designer.

Note

- Each of these module varieties can have a subtype classification. It can be the standard one, as described above, or it can be a check condition, stored procedure or a trigger. These types of modules are used to store SQL procedure code to be generated with SQL schemas.

A data only module is a module that consists solely of data – data elements or structures of data elements. It is always a lowest-level module in that it may only be accessed by other module types and never calls other modules itself (though it can contain lexically included data modules).

- An information cluster is an aggregate of two or more modules and only one data-only module. Rules support information clusters with up to seven internal modules. The information cluster is intended to aggregate all the modules that would share a single data only module, for the purpose of diagram clarity and understanding. It operates as a module for all other purposes. The Drawing Diagrams chapter has a full description of how to add an information cluster to a diagram.
- A program is a structure chart item that appears only in the repository of your project. It represents a computer program and is composed of a number of modules. Its use is discussed under Shell Code Generation in The Visible Repository chapter.

Structure Chart Line Types

Structure charts use four line types:

- Invocation lines
- Data and control connections
- Data, control, and generic couples
- Data, control and generic interface table rows (ITRs)

Invocation lines, along with data and control connections, are line types that connect modules on a diagram. Invocation lines differ from data and control connections in a number of significant ways. Couples and interface table rows are a third line type and are distinguished from invocations and connections by the fact that they do not connect modules, but are shown being passed to and from modules by invocations and connections. A discussion of the differences and uses of the different line types follows.

[6] *Ibid*., p. 455.

Invocation Lines

The meaning of this relationship is a passing of control: module A calls module B. Implicit in the meaning of an invocation is the return of control to the calling module when the called module is done processing. An invocation is shown by drawing a line from the border of one module to the border of any other module except data only-modules. (See Figure 4-23.) Invocation lines can pass combinations of couples and ITRs to other modules.

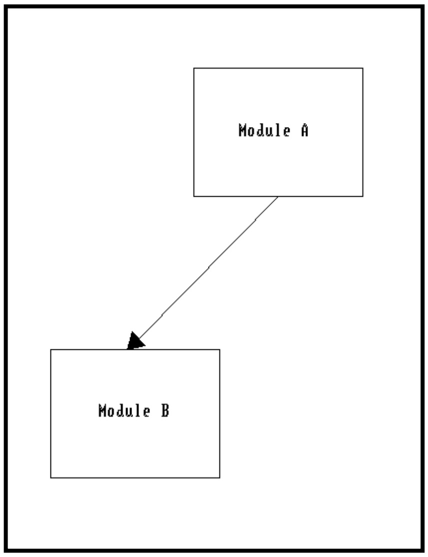


Figure 4-23

Using Invocation Lines

To use an invocation line, add it to your diagram as described in the Drawing Diagrams chapter. Any line style can be used to draw an invocation line except for the two line styles with control and data end points (solid or open circle on one endpoint) and the loop line type. Invocation lines can have a variety of terminators associated with their endpoints. These are arrowhead, conditional, and lexical inclusion. Use of a terminator is optional. Terminators are discussed later.

Data and Control Connections

Data and control connections always show data or control being passed. Implicit in their use is that control or data is passed uni-directionally. Control does *not* automatically return to the calling module as it does for invocations. They are differentiated from invocations by a number of criteria:

- They never use conditional, lexical, or loop connectors.

- Pathological connections are allowed (but not encouraged).
- Data connections can pass data couples (or DITRs) between modules; control connections can pass control couples (or CITRs) between modules.

Using Data and Control Connections
A data connection is drawn using the line selection with an open circle on one endpoint of the line. Again, arrowhead terminators are optional on the other end. Data connections are used to show that data *only* is being passed, as from module C to module A in Figure 4-24.

- A control connection is drawn using the line selection with a filled circle on one endpoint of the line. Arrowheads are optional but recommended. Control connections are used to show that only control is being passed, as from module A to module B in Figure 4-24.

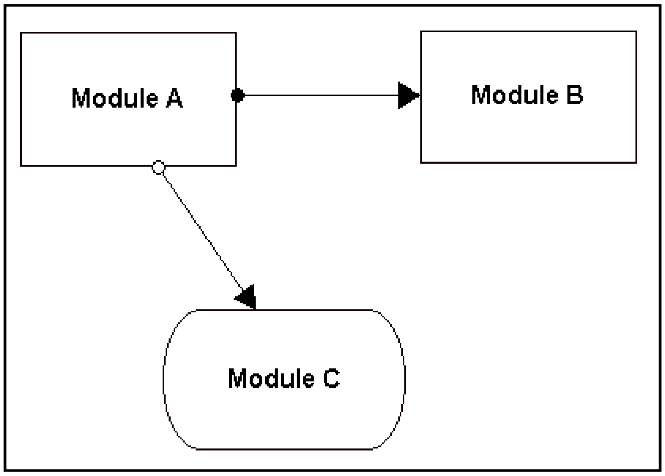


Figure 4-24

Terminators for Invocation Lines
As described above, invocation lines can use a variety of terminator types. Terminators add information about how control is passed from module to module. This is generally necessary for proper documentation when describing complex programs. Visible rules support special terminator types and one special line type used to denote the programming constructs under which invocations are made.

The terminator types and the special lines supported for invocation lines are:

- Arrowhead
- Conditional
- Loop (a line type)
- Lexical Inclusion

Conditional is the diamond shaped terminator. A conditional connector may be attached to only the “invoking” or starting point of an invocation line. It shows that conditions apply to the invocation. It is used primarily to indicate a decision. For instance, in Figure 4-25 module A calls module B under one set of conditions and module C under a second set of conditions.

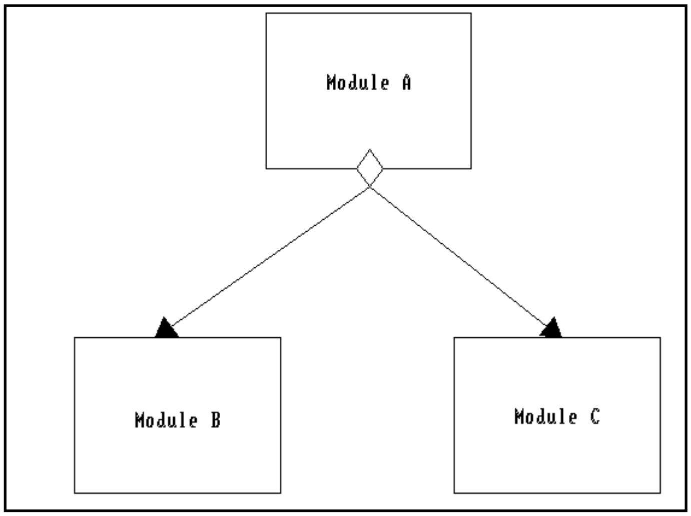


Figure 4-25

The loop line is the open ellipse line type in the structure chart Line Settings dialog box. (See Figure 4-26.) Even though it is a line, it cannot be labeled, it cannot be defined in the repository, and its primary purpose is to show that an invocation line (of any type) is part of an iterative and/or an ordered set of invocations. In use, a loop can be drawn around the start point of one or more invocation lines. Loops can be nested.

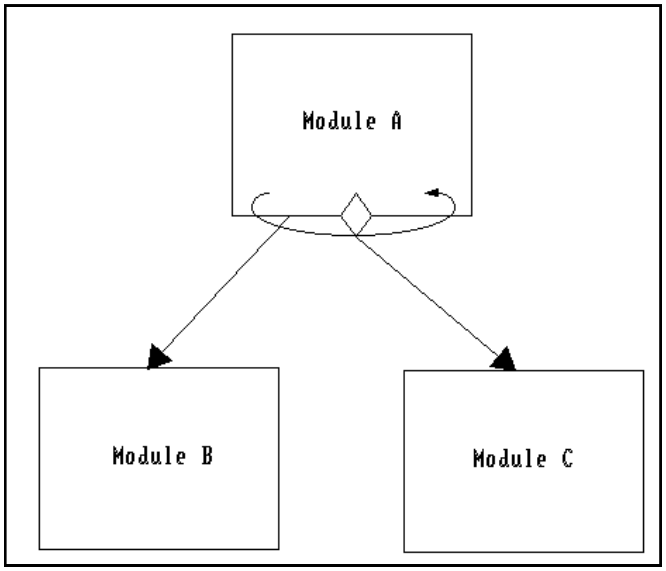


Figure 4-26

The lexical inclusion is the “split line” or “hat” shaped terminator for structure chart lines. (See Figure 4-27.) Its purpose is to show that the invoked module is logically separate but physically internal to the calling module. Its definition is “the property of one object ... being wholly contained within the lexical boundaries of another.” [7]A lexical inclusion terminator may only be attached at the end point of an invocation line.

[7] *Ibid*., p. 455.

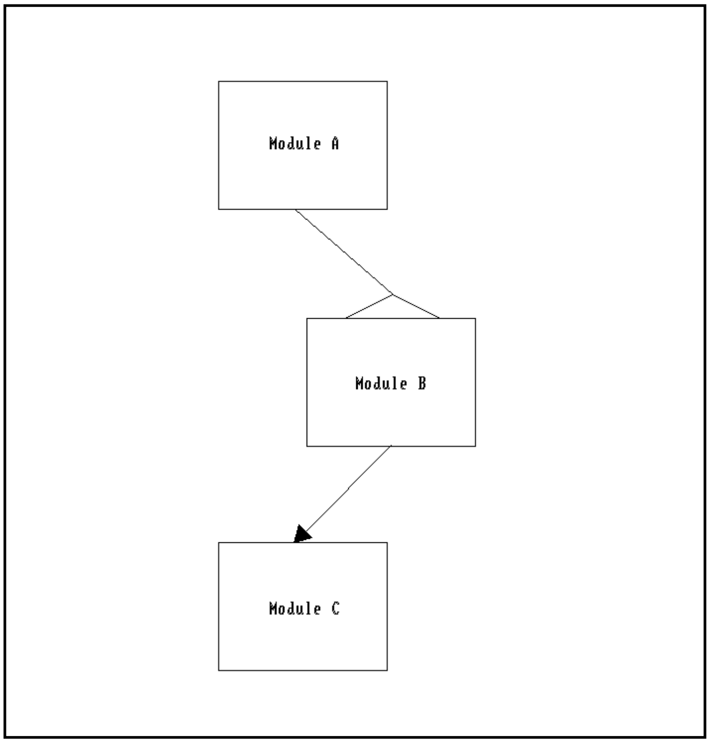


Figure 4-27

Data Couples

A data couple is a line type that is associated with invocation or data connection lines. Data couples are the data parameters that get passed between program modules. They are distinguished by an open circle at one endpoint and an arrowhead at the other. (See lines “W” and “X” in Figure 4-28.) Data couples can be bi-directional; that is, two arrowheads pointing away from each other, and one open circle in the middle of the line. Bi-directional data couples are used for indicating in-out data coupling. Refer to Drawing Diagrams for a full description of how to draw couples. A data couple can be changed into a data interface table row by modifying the couple type field in the repository, as explained in The Visible Repository chapter.

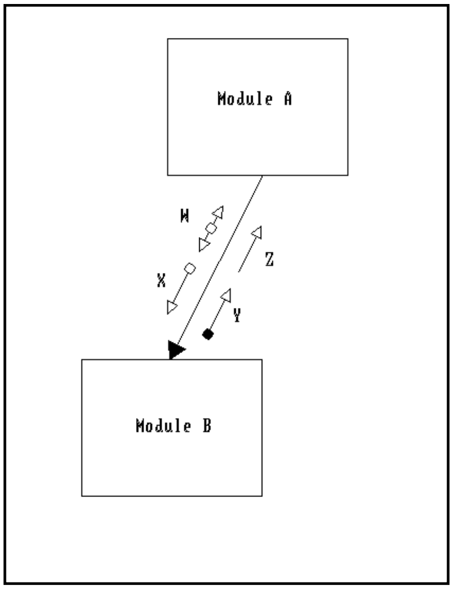


Figure 4-28

Control Couples
A control couple is a line type that is associated with invocation and control connection lines. Control couples are the control parameters that are passed between modules upon invocation or return. Control couples are distinguished by a filled circle at one endpoint and an arrowhead at the other. (See line “Y” in Figure 4-28.) Control couples can be bi-directional just like data couples and are used for indicating an in-out couple. A control couple can be changed into a control interface table row by modifying the couple type field in the repository, as explained in The Visible Repository chapter.

Generic Couples
A generic couple is a line type that is associated only with invocation lines. Generic couples show that both data parameters and control parameters are passed between modules upon invocation. Generic couples are distinguished by *not* having a circle on one endpoint of the line and by an arrowhead at the other. (See line “Z” in Figure 4-28.) Generic couples can be bi-directional (like data and control couples). A generic couple can be made into a generic interface table row by modifying the couple type field in the repository, as explained in The Visible Repository chapter.

Interface Table Rows (ITRs)

An interface table row represents a group of one type of couples and is drawn on the diagram to be a couple of that type. It is used to simplify the diagramming of complex module interfaces that include many different types of couples and make the diagram appear less cluttered. The couples that make up an ITR are listed in the composition field of the ITR in its repository entry (see The Visible Repository).

Linking Structure Chart Pages Together

Structure chart diagrams, in contrast to data flow diagrams, are not hierarchically organized. In Visible Analyst, structure charts are linked together using both on-page and off-page connectors. The Page function is used to link the structure chart modules together over diagram boundaries, without the loss of connection normally associated with multi-diagram charts. The Nest and Split Data Flow functions that connected your data flow diagrams, are disabled for the purposes of drawing structure charts. Visible Analyst recognizes page connectors when performing an analysis validation; that is, if a structure chart is composed of ten connected pages, Analyze reviews all ten pages as though they were one chart.

Off-page connectors can be either input connectors or output connectors. An input connector carries an invocation from a module to another diagram. An output connector joins an invocation from another diagram to a module on the current diagram. There are some general rules when using off-page connectors.

- Each input connector may refer to only one output connector.
- Each set of connectors must be uniquely identified. For an input connector labeled XYZ, there must be an output connector labeled XYZ.
- Each set of connectors must be connected with the same type of invocation line or data or control connection.
- Input connectors can have multiple occurrences that all refer to the same output connector.
- Input connectors can have many invocation lines or data or control connections (all of the same type) attached to them.
- Output connectors may have only one invocation line or data or control connection going from it to a module.

Analyzing Structure Charts

Visible rules analyze structure chart diagrams based upon heuristics. Fan-in, fan-out, design complexity and completeness are checked. Analysis operates at two levels, module-to-module and global. At the module-to-module level, the Analyze function reviews the connections between modules for correctness of connection, complexity of interface, and completeness of design. Module-to-module connections are in one of three conditions for any connection:

- A valid connection, for which no error message is generated.
- A conditionally invalid connection, for which a WARNING message is generated.

- An invalid connection, for which an ERROR message is generated.

All modules are analyzed for complexity of interface. The technique used is called fan-in/fan-out analysis[8]. Fan-in and fan-out measure, respectively, the number of calls to a module from other modules and the number of calls a module makes to other modules. Analyze gives warnings for low fan-in and high fan-out. Additionally, Analyze implements Card's Intrinsic Complexity Algorithm[9] to generate an index of a diagram's complexity. (See Figure 4-34 for more information.)

All modules and interfaces are analyzed for completeness of design. Modules and couples are checked for labeling and connection. If a module or couple is unlabeled, an error message is generated. If a module is unconnected, an error message is generated. If an invocation line does not connect to another module, an error message is generated. This is explained in detail below.

Structured Design Error Messages

ERROR	There are 'x' unnamed module(s).
ERROR	There are 'x' unnamed page connector(s).
ERROR	There are 'x' unnamed couple(s).
ERROR	There are 'x' unnamed information cluster(s).

These error messages are the result of not labeling symbols or couples on your structure charts.

To resolve the errors, find and label them using the Change Item function.

ERROR	Module labeled 'x' is a dangling module.
ERROR	Page connector labeled 'x' is dangling.
ERROR	There are 'x' invocation line(s) without source and/or destination.

The above error messages are the result of not connecting the indicated items with invocation lines.

To resolve the error(s), draw invocation or connector lines or move modules or lines so that they will be connected.

ERROR	Page connector labeled 'x' has both input and output connections.

[8] Refer to Yourdon/ Constantine for a full description of acceptable fan-in/fan-out parameters.
[9] Card, David N., and Glass, Robert L. *Measuring Software Complexity*. Englewood Cliffs: Prentice-Hall, 1990.

The above error is the result of using the same page connector for two-way page connections. See Figure 4-29. Rules expect page connectors to be one-way only. (This is for validation purposes.)

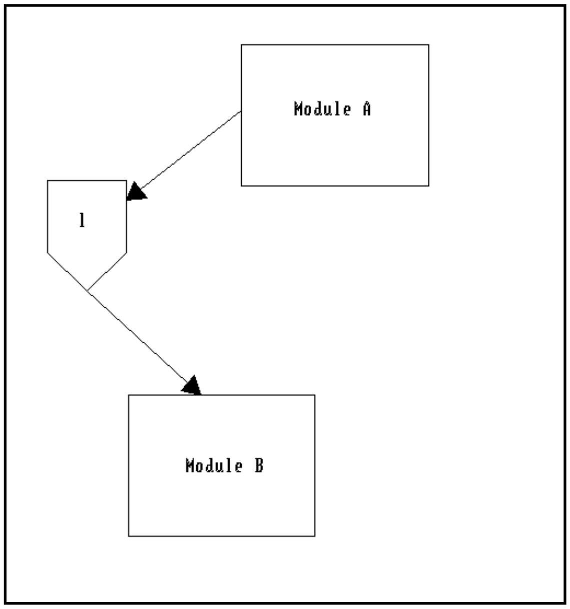


Figure 4-29

To resolve the error, create a second page connector using the Page function and use it to replace one of the directions included in the original page connector.

ERROR There are ‘x’ incorrectly placed looping structure(s).

This error message is the result of not placing a loop connector properly on the diagram. To resolve the error, reposition any loops so that they are placed with at least one endpoint attached to a module.

ERROR Illegal connection between module ‘x’ and module ‘y’.
ERROR Illegal connection(s) using off-page connector ‘x’.

The above error is the result of connecting two modules in an incorrect manner. In general, this error is the result of using a library module, library macro, or data only module to call another module, possibly using a page connector (there are some exceptions). See Figure 4-30.

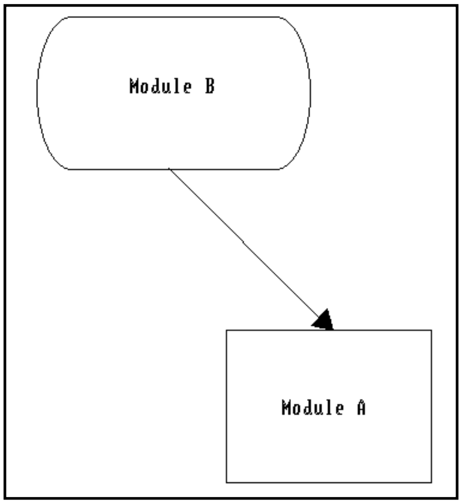


Figure 4-30

To resolve the error, review the module(s) with illegal connections, and change the connections to correct or permitted types.

WARNING Connection type between module ‘x’ and module ‘y’ cannot be accurately determined.

The above warning message indicates that rules were unable to determine a connection type, for instance, a pathological data connection that refers to an entire module.

To resolve the warning, review the diagram and check for proper use of the line type for the particular situation.

ERROR ‘x’ has more than one lexical inclusion.

The above error message indicates that the flagged module is represented as lexically included in more than one other module.

To resolve the error, either change the terminator or move the extra lexical inclusion line(s) so that only one lexical inclusion references the flagged module.

WARNING Permitted but not recommended connection between module ‘x’ and module ‘y’.
WARNING Permitted but not recommended connection(s) using off-page connector ‘x’.

The above warning messages are generated when modules are connected in ways that have been proven to have drawbacks in terms of design.

To resolve the warning(s), change the connections to more acceptable types.

ERROR Page connector labeled ‘x’ has more than one output connection

The above error is the result of using the same page connector for calling multiple modules on another page. As discussed earlier, an output connector may only connect to one module. See Figure 4-31.

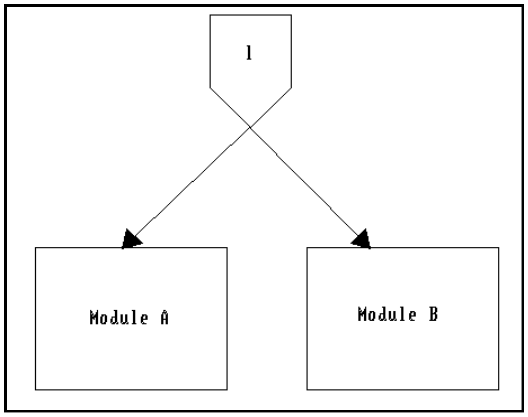


Figure 4-31

To resolve the error, reverse the positions of the calling module and the page connector; that is, replace the calling module with a page connector on the calling page. This has the effect of moving the page connector up one level in the program hierarchy. Next, place the module (the one just replaced with the page connector) on the called diagram. Use the page connector on the called page to connect to the module, then use the module to call the other modules.

ERROR Page connector labeled ‘x’ appears as output more than once.

The above error indicates that the flagged page connector is an output connector in more than one place in your structure chart. As discussed earlier, each set of uniquely named connectors may have only one output connector.

To resolve the error, remove or rename the flagged output connectors until only one output connector of that name remains in the structure chart set.

ERROR Page connector labeled ‘x’ has mismatched connection types.

The above error is the result of calling to a page connector with one type of connector, and calling from the connector, on the called page, with another type of connector. See Figure 4-32.

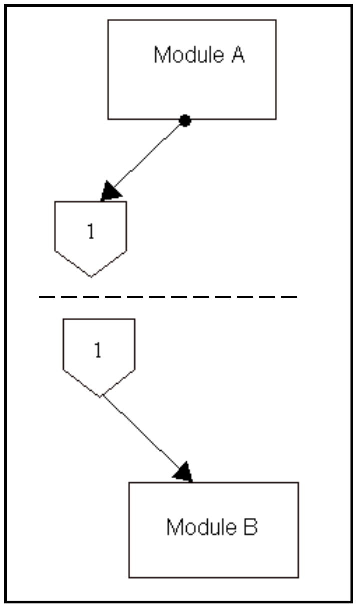


Figure 4-32

To resolve the error, first decide which connector type is the desired type. Then use the Change Item function to change the invalid connector to the desired type.

ERROR Matching page connector cannot be found for ‘x’.

The above error is the result of not using a page connector either on the calling or on the called page. Also, one or more page connectors used might be incorrectly labeled.

To resolve the error, locate the page connector without a match and use the Page function to access the paged diagram. Once on the connected diagram, review the diagram for the presence of the offending connector. If it isn't there, use the Symbols function to place one on the diagram and the Lines function to connect it to the module it should be connected to.

ERROR Off-page labeled 'x' appears as both input and output.

The above error indicates that a connector exists as both input and output. A page connector may be either input or output, but not both. Also, a page connector may not exist as separate input and output connectors on the same page.

To resolve the error, remove the connections to the flagged page connector so that it is either input or output.

WARNING Low fan-in for module 'x'.

The above warning message is the result of a module being called by too few other modules. This is an indication that the called module could, potentially, be moved into the body of the calling module. This warning falls into the same category as the next warning message in that it is a guideline, not a rule.

To resolve the warning, check the modules involved and verify whether upwards joining is needed or not needed. Once you've made the determination for yourself, you can safely ignore the warning message from then on.

WARNING High fan-out for module 'x'.

The above warning message is the result of using too many calls from the flagged module. The warning is based upon design principles and is meant to be a guideline indicating a potentially overly complex control structure within the calling module.

To resolve the warning, add levels of modules to break up control issues into less complex structures.[10]

ERROR Information cluster labeled 'x' has unnamed modules.

[10] Refer to Yourdon/Constantine, Chapter 9, for a full discussion of how to resolve high fan-out.

The above error is the result of not labeling one or more of the modules internal to an information cluster. See Figure 4-33.

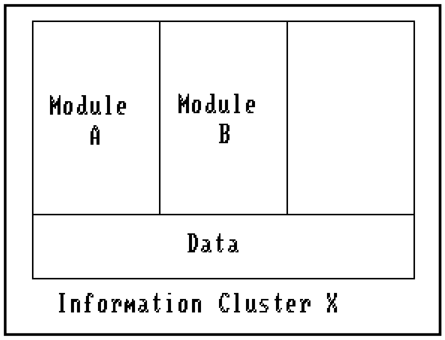


Figure 4-33

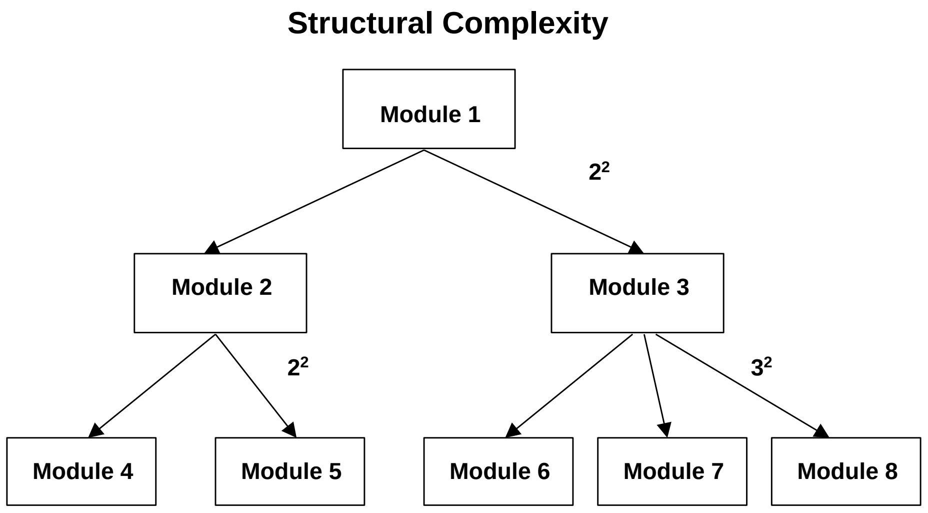
To resolve the error, label all of the modules in the information cluster.

Analysis Statistics

Analyze generates statistics to help you evaluate your design diagrams. The figures it displays are:

- Number of modules x
- Average module fan-out x
- Structural complexity x

Figure 4-34 explains how these statistics are calculated for the sample structure chart diagram shown.



$$S = \frac{\Sigma\ f^2}{n} \qquad \frac{2^2 + 2^2 + 3^2}{8} = 2.1$$

Note: Structural complexity is equal to the sum of the square of module fan-out (the number of invocation lines from a module) divided by the number of modules in the system.

S = structural (intermodule) complexity
f = fan-out of a module
n = number of modules in system
Statistics:

Number of modules	8
Module average fan-out	0.9
Structural complexity	2.1

Figure 4-34 Structural Complexity Information

ENTITY RELATIONSHIP DIAGRAMS

Data Modeling Overview

Entity relationship diagrams allow you to express the data model of your project diagrammatically. You can describe the entities (or, more properly, entity types) in the data you are modeling and the relationships between them by drawing them onto a diagram. Each diagram or view can show an arbitrarily large or small part of your data model. You can show multiple views of your data model by including different combinations of entities and relationships on various diagrams. However, the entire data model is retained in the repository. To show really large data models and still have them comprehensible, you can cluster groups of entities and show these clusters and the relationships between them on a view. See Drawing Diagrams for descriptions of this function.[11]

All information you place in a view is, of course, captured by the repository and is available to both your process model (data flow diagrams) and to your structure charts, where applicable, if you have an integrated tool set. The Analyze function can help you balance a data model against a process model and maintain consistency. Additionally, Key Analysis can help you set up a consistent relational database key structure by data elements (sometimes called attributes) which you designate as keys or parts of compound keys across relationships to associated entities and create foreign keys. Using associator element names in relationship repository entries makes this process work better.

Data Modeling Graphics

Entity relationship diagramming is designed to give you the ability to clearly show the characteristics of your data while still allowing a certain degree of flexibility in modeling style. By careful use of configuration settings and the line and symbol sets, you have the ability to use different entity symbols and various line terminators that show relationship cardinality.

Entity Symbols

There are three entity types available for you to use: fundamental, associative (sometimes called junction entities, concatenated entities, gerunds or correlation tables) and attributive, each with its own symbol in the symbol set. The Analyze function takes into account the type

[11] Although many of the methodological details are different from how it is done in Visible Analyst, a good introduction to the concepts of data modeling can be found in Shlaer, S., and Mellor, S. J., *Object-Oriented Systems Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

A practical, but more advanced, book is:
Fleming, C. C., and von Halle, B. *Handbook of Relational Database Design.* Reading: Addison-Wesley, 1989.

of entity you specify and helps you identify certain normalization errors, as well as assists in key migration. You can, if you wish, use a single symbol to represent entities of all types or you can alter your symbol set so that the different types of entities have any appearance you desire.

Entity symbols can be shown on an ERD at various levels of detail. The diagram can be set to show only the entity names, the entity name and attributes that are part of the primary key, or all attributes. When changing the detail level, entity symbols are automatically resized and the relationships are reconnected properly. To set the detail level, choose Entity Display Options from the View menu; or click one of the display level buttons on the control bar when the diagram tools tool bar is displayed.

Selecting “Physical Schema” from the Options menu allows you to display the entity attribute details, such as data type, alias name, null option, etc. See “Physical Schema Displayed on Data Model Diagrams” in Chapter 3 for additional details.

Relationship Lines

Relationships can be labeled in one or both directions. You set this value when you create the project. Relationship lines between entities usually have terminators on each end that show if the relationship cardinality is 0:1, 1:1, 0:many, 1:many or many:many. You can set default terminators to show relationship cardinality with Line Settings from the Options menu, as described in the Drawing Diagrams chapter. If Auto Label Lines is enabled, after you add a relationship to your view, a dialog box opens in which you can enter the labels for each direction of the relationship. At that time, you also have the opportunity to change the cardinality from the default.

View Objects

A view object is similar to an entity in that it has a composition but the items that appear in the composition of a view must belong to other entities or be expressed based on data elements used by another entity. View objects are described in detail in The Visible Repository.

Analysis for Entity Relationship Diagrams

For data modeling, after selecting Analyze from the Diagram menu you have an additional analysis choice available when analyzing either the current diagram or the entire project. You have the option of doing normalization as well as syntax analysis. Syntax analysis identifies unnamed entities and relationships. Normalization analysis does syntax analysis and also identifies certain possible relationship normalization errors. This warns you of many-to-many relationships, optional-optional relationships, mandatory one-to-one in both directions and the fact that the identifying relationships for associative and attributive relationships are not shown on the current view. (A relationship is identifying or defining if it comes from a fundamental entity where the cardinality is 1:1.)

From the Repository menu, you have the additional analysis choices Key Analysis, Key Synchronization and Model Balancing (balancing your data model against your process model).

Normalization Analysis

Normalization[12] errors at the entire project level warn you if an associative entity has fewer than two identifying relationships and if an attributive entity has other than one identifying relationship.

Key Analysis

Key analysis does syntax and normalization analysis, and also finds errors in primary and foreign key specification. It notes:

- An entity without a primary key.
- A primary key on one end of a relationship without a foreign key on the other end.
- An associative entity with fewer than two foreign keys making up the primary key.
- An attributive entity with other than exactly one foreign key as a primary key.
- A foreign key without a corresponding relationship.
- A foreign key that is part of the primary key of a fundamental entity.

As described in The Visible Repository, you have the option of adding the name of a special kind of data element called an associator to the repository entry of a relationship. When specified and automatically created, this is used by Analyze as the name of the foreign key for an entity at the other end of the relationship in the current direction.

Key Synchronization

Key synchronization migrates primary keys across normalized named relationships to make foreign keys in other entities and adds descriptive information about the relationship and the related entities. If you perform key migration on an entity pair whose relationship has no associator element name specified, Analyze uses whatever name you have already specified for the existing primary key (or just the abbreviations [PK] and [FK]). If you later add an associator element name for the relationship and rerun Key Synchronization, the key is updated with the proper associator element name. You can choose to do all of this key migration manually, if you wish. Alternate keys ([AK#]) are not migrated.

Process/Data Model Balancing

If at least one of the balancing options is set with ERD Balancing Rules from the Options menu, the Model Balancing function balances data elements and/or entities against existing data flow diagrams. The first option (All Fundamental Elements Must Be Used on a DFD)

[12] Normalization is a means of eliminating redundancy in data. It is a complex topic and is beyond the scope of this manual. Since understanding normalization is key to effective database design, you should consult a text on the subject, such as one written by C. J. Date.

tells Visible Analyst whether you want to be made aware of any data elements that aren't used. In other words, is an element listed as part of an entity but not used by at least one process on at least one data flow diagram? As for the second option (Every Entity Must Correspond to a Data Store), there is some degree of correspondence between the entities in a data model and the data stores in a process model. The nature of this correspondence is not generally agreed upon. You can specify that every entity must correspond to a data store with the same composition and Analyze notifies you if this is not the case.

Data Modeling Error Messages

ERROR Relationship 'x' is not normalized.

The above error indicates that the cardinality is not x:1 on either end (it is 0:many or many:many in both directions). (See Figure 4-35.)

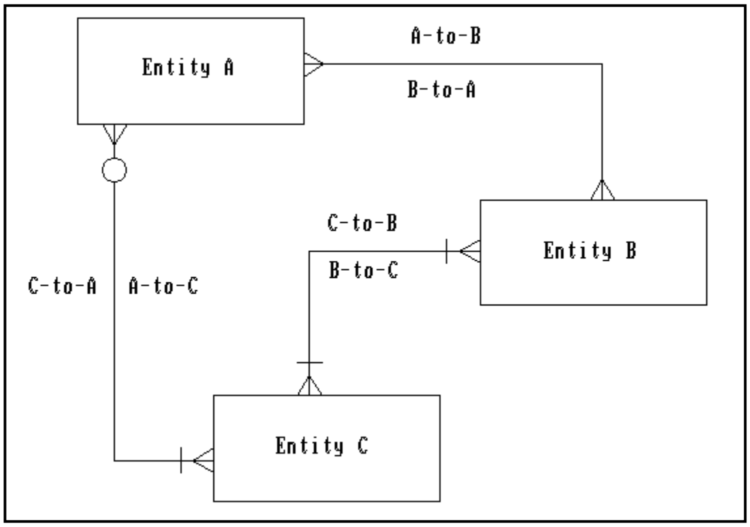


Figure 4-35

To resolve the error, change the cardinality on one end of the relationship so that the upper limit is one.

WARNING Reverse Relationship of 'x' is unnamed.

The above warning indicates that a relationship is labeled in only one direction, while the project has been configured for relationships to have two labels.

To resolve the warning, find the relationship and make sure that it is labeled in both directions.

WARNING Relationship ‘x’ is optional in both directions.

The above warning means that on both ends of a relationship, the cardinality is 0:x. Optional:optional relationships can be difficult to implement. (See Figure 4-36.)

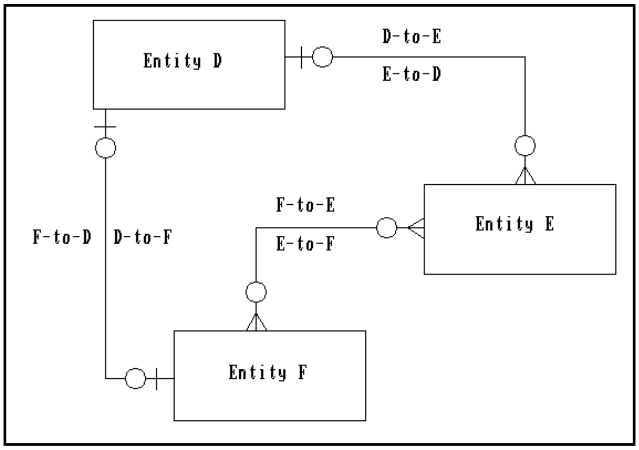


Figure 4-36

To resolve the warning, make the relationship cardinality mandatory on at least one end.

WARNING Relationship ‘x’ is one to one mandatory.

The above warning indicates that there is a possible normalization error for the entities at both ends might belong in the same table. (See Figure 4-37.)

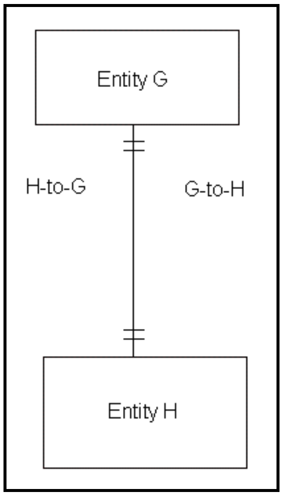


Figure 4-37

To resolve the warning, either combine the two entities into one or change the relationship cardinality. If the relationship is in fact correct, ignore the warning.

ERROR Unattached relationship ‘x’.

The above error should never happen in the normal course of things because Visible Analyst rules do not allow you to draw *and name* an unattached relationship. Also, if you detach a relationship from an entity or delete an attached entity, the name on the relationship is cleared. (If the relationship has locations on other views, only this location is cleared.) However, a corrupted file, such as from a power surge or system lockup at a critical time, could leave a relationship unattached.

To resolve the error, display the diagram on which the error occurred and delete the relationship. Then run the Rebuild function to restore synchronization between the repository and the diagram. Finally, add the properly attached relationship back to the diagram.

ERROR There are ‘x’ unnamed entity(s).
ERROR There are ‘x’ unnamed relationship(s).

These error messages are the result of not labeling entities or relationships on your ERD(s).

To resolve the error, find and label them.

ERROR Attributive entity ‘x’ has no identifying relationship attached.

The above error indicates that nowhere in the repository does the required identifying relationship for an attributive entity exist.

To resolve the error, be sure that this attributive entity has one identifying relationship.

ERROR Entity ‘x’ has no primary key defined.

The above error indicates that there is no data element listed in the composition field of this entity marked with [PK] to indicate that it is the primary key for the relationship.

To resolve the error for a fundamental entity, be sure that at least one data element is listed in the composition field of the entity marked with [PK]. To resolve the error for an associative entity, run key synchronization to migrate the keys from fundamental or other associative entities.

ERROR Associative entity ‘x’ has less than two identifying relationships attached.

The above error indicates that nowhere in the repository do the required two identifying relationships for an associative entity exist.

To resolve the error, be sure that this associative entity has two identifying relationships. A relationship is identifying if it comes from a fundamental entity where the cardinality is 1:1.

WARNING Associative entity ‘x’ has less than two identifying relationships
in this view.

The above warning alerts you that the identifying relationships for an associative entity might not exist. (This error only happens when you are analyzing a single diagram. If both identifying relationships exist in the repository, they are found by a whole-project analysis.) (See Figure 4-38.)

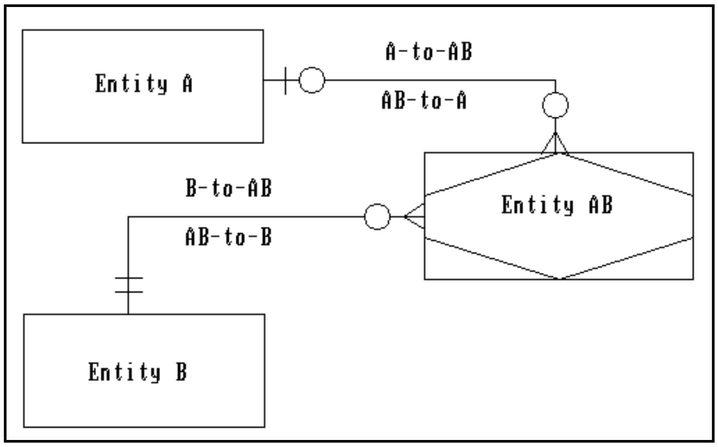


Figure 4-38

To resolve the warning, check to be sure that on some diagram this associative entity has two identifying relationships. If it does, then ignore this warning. It is good practice to include both identifying relationships on all views where an associative entity appears.

WARNING Attributive entity 'x' has no identifying relationship
in this view.

The above warning exists to warn you that the identifying relationship for an attributive entity might not exist. (This error only happens when you are analyzing a single diagram. If the identifying relationship exists in the repository, it is found by a whole-project analysis.) (See Figure 4-39.)

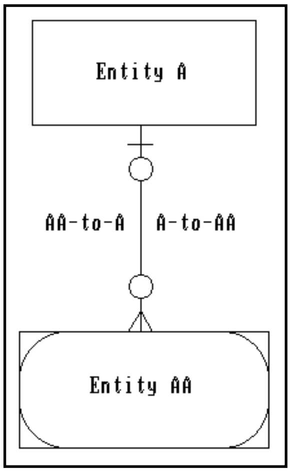


Figure 4-39

To resolve the warning, check to be sure that on some diagram this attributive entity has an identifying relationship. If it does, then ignore this warning. It good practice to include the identifying relationship on all views where an attributive entity appears.

ERROR Attributive entity ‘x’ has no unique primary key.

The above error indicates that either no primary key or one that is not unique has been defined for the named attributive entity. Attributive entities *must* be able to refer to only one fundamental entity.

To resolve the error, add a unique primary key.

ERROR Foreign key does not exist for relationship ‘x’.

The above error indicates that a primary key has not been migrated across a relationship to form a foreign key in another entity.

To resolve the error, you can either add a data element name preceded by [FK] to the composition field of the indicated entity or run key synchronization and let Visible Analyst do it for you.

ERROR Foreign key ‘x’ for entity ‘y’ already exists as associator element.

The above error indicates that both an associator element name and a foreign key with the name of the primary key of the related entity have been added to the composition field of an entity.

To resolve the error, run key synchronization; it leaves the name of the associator element as the only name for the foreign key name.

ERROR Duplicate foreign key ‘x’ for entity ‘y’.

The above error indicates that there are two foreign keys with the same name listed in the composition field of an entity.

To resolve the error, either eliminate the duplicate or run key synchronization again and Visible Analyst deletes the duplicate for you.

WARNING Foreign key ‘x’ for entity ‘y’ has associator element defined.

The above warning indicates that an associator name has been added to a relationship since a foreign key was added to the composition field of an entity.

To resolve the warning, run key synchronization; it substitutes the name of the associator element for the existing foreign key name.

ERROR Associative entity ‘x’ has less than two foreign keys as primary key.

The above error either means that there are fewer than two identifying relationships for an associative entity or the primary and/or foreign keys for existing identifying relationships have not been identified in the composition field of the associative entity.

To resolve the error, be sure that both identifying relationships exist, that each identifying entity has primary keys defined and that the keys have been migrated to form foreign keys in the associative entity. Key synchronization accomplishes at least this last step.

ERROR Entity ‘x’ has cyclic definition - primary key cannot be determined.

The above error indicates that an associative entity is the identifying entity for another associative entity that is, in turn, an identifying entity for this associative entity. There could possibly be a chain of associative entities between them, still resulting in a cyclic definition. (See Figure 4-40.)

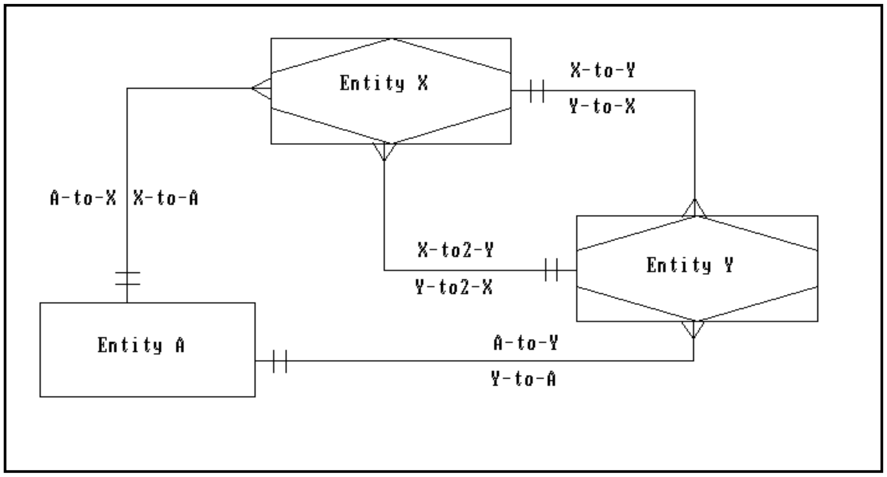


Figure 4-40

To resolve the error, make sure that there are no cyclic identifying relationships in your project.

ERROR Attributive entity ‘x’ has no foreign key as primary key.

The above error either means that there is no identifying relationship for an attributive entity or the primary and/or foreign key for an existing identifying relationship has not been identified in the composition field of the attributive entity.

To resolve the error, be sure that an identifying relationship exists, that the identifying entity has a primary key defined, and that the key has been migrated to form the foreign key in the attributive entity. Key synchronization accomplishes at least this last step.

ERROR Foreign key ‘x’ for entity ‘y’ does not have corresponding relationship.

The above error indicates that a foreign key has been identified in the repository entry for an entity, but that there is no relationship in a view that corresponds to this foreign key.
To resolve the error, add a relationship from another entity that has a primary key (or a relationship with an associator element name) to correspond to the foreign key in the current entity.

ERROR Primary key for fundamental entity ‘x’ contains foreign key.

The above error indicates that both a primary and a foreign key have been identified for a fundamental entity.

To resolve the error, either change the symbol for the entity so that it is no longer a fundamental entity or eliminate the [FK] notation in the composition field of the fundamental entity.

ERROR Old foreign key column ‘x’ in entity ‘y’ should be removed.

The above error message indicates a column that had been marked as a foreign key should be removed from the entity because it no longer appears as a key column in the parent entity, or the relationship to which the key belonged was deleted.

To resolve the error, remove the column from the composition field, or perform key synchronization.

ERROR Old foreign key column ‘x’ in entity ‘y’ should be removed from the primary key.

The above error message indicates a column that had been marked as a primary foreign key should be removed from the entity because it no longer appears as a key column in the parent entity, or the relationship to which the key belonged was deleted.

To resolve the error, remove the column from the composition field, or perform key synchronization.

ERROR Definition for column ‘x’ in entity ‘y’ should be updated for relationship ‘z’.

The above error message indicates that the type or physical characteristics of a foreign key column in the child are different from the parent.

To resolve the error, change the definition in the dependent entity, or perform key synchronization.

ERROR Extra primary key column ‘x’ in entity ‘y’ should be removed from the primary key because of singular relationship ‘z’.

The above message indicates that there are columns included in the primary key that should not be used. For supertype relationships and identifying relationships where the cardinality is singular, the primary key of the child must exactly match the primary key of the parent. This means all members of the primary key must be foreign key columns.

To resolve the error, remove the non-foreign key members of the primary key, or perform Key Synchronization which removes the extra column from the primary key. The column is not deleted from the entity.

ERROR Column ‘x’ in entity ‘y’ is used by overlapping foreign keys with conflicting types. Relationship ‘z’ should no longer use this column.

The above message indicates the same foreign key column was used by more than one relationship but the definition was different (physical information differs).

To resolve the error, add a new column for the conflicting key column or perform key synchronization.

KEY SYNC Primary foreign key created for relationship ‘x’.

The above message is not an error. It exists to notify you that key synchronization migrated a key across an identifying relationship.

KEY SYNC Foreign key created for relationship ‘x’.

The above message is not an error. It exists to notify you that key synchronization migrated a key across a non-identifying relationship.

KEY SYNC Old foreign key column ‘x’ in entity ‘y’ has been removed.

During the key synchronization, a column that had been marked as a foreign key was removed because it no longer appeared as a key column in the parent entity.

KEY SYNC Old foreign key column ‘x’ in entity ‘y’ has been removed from the primary key.

During the key synchronization process, a column that had been marked as a primary foreign key was removed because it no longer appeared as a key column in the parent entity.

KEY SYNC Column ‘x’ in entity ‘y’ is used by overlapping foreign keys with conflicting types. Relationship ‘z’ will no longer use this column.

The above message notifies you that key synchronization removed a foreign key from the child table because the same name was used by more than one relationship but the definition was different (physical information differs). A new column is added for the conflicting key column.

KEY SYNC Definition for column ‘x’ in entity ‘y’ has been updated for relationship ‘z’.

The above message is not an error. It notifies you that key synchronization changed the definition of a foreign key column because the definition in the child was different from the parent.

KEY SYNC Column ‘x’ in entity ‘y’ has been added to the foreign key of relationship ‘z’.

The above message is not an error. It notifies you that key synchronization migrated a key across a non-identifying relationship.

KEY SYNC Extra primary key column ‘x’ in entity ‘y’ has been removed from the primary key because of singular relationship ‘z’.

The above message is not an error. It notifies you that key synchronization removed a column from the primary key. For supertype relationships and identifying relationships where the cardinality is singular, the primary key of the child must *exactly* match the primary key of the parent. This means all members of the primary key must be foreign key columns. The extra column is removed from the primary key; the column is *not* deleted from the entity.

ERROR Foreign key for relationship ‘x’ is wholly dependent on the foreign key of relationship ‘y’ in entity ‘z’.

The above message indicates that one of the relationships attached to the indicated entity contains a subset of the foreign key columns used by the another relationship. This may indicate that the relationship may be drawn between the wrong pair of entities.

To resolve the error, either remove the relationship, choose a different set of key columns for the relationship, or redraw the relationship with a different parent entity.

WARNING Foreign key for relationship ‘x’ shares column(s) with the foreign key of relationship ‘y’ in entity ‘z’.

The above message is not an error. It notifies you that key synchronization migrated the same key column across multiple relationships because the same key column name was used in more than one entity. This happens as long as all columns have the same physical characteristics.

ERROR Foreign key for relationship ‘x’ should not be a component of the primary key of entity ‘y’.

The above message indicates that the columns of the foreign key are marked as members of the primary key for the dependent entity. This should only be true when the relationship is identifying and the dependent entity is either associative or attributive.

To resolve the error either remove the columns from the primary key list or select the indicated relationship, open the Change Item dialog box, and change the relationship type to identifying.

ERROR Definition for column ‘x’ in entity ‘y’ conflicts with the definition of the matching column on relationship ‘z’.

The above message indicates that a column that is part of the foreign key list has a different set of physical characteristics from the corresponding primary key in the parent table.

To resolve the error, change the physical characteristics of one of the columns to match the other.

ERROR Entity ‘x’ has multiple identifying relationships. This requires relationship ‘y’ to have a maximum cardinality greater than one.

The above message indicates that one of the identifying relationships attached to an associative entity violates second normal form, which states that each non-key attribute must be fully dependent on the entire primary key. By having a relationship that has a maximum cardinality of one, you are indicating the entity is only dependent on the primary key columns from this relationship.

To resolve the error, change the cardinality of the relationship by selecting the problem relationship, choose Change Item from the Diagram menu, and change the Maximum Cardinality field for the From entity.

ERROR Data Element ‘x’ is not used on data flow diagram.

The above error is generated when the process model/data model balancing criterion that all data elements must be used has been set. It means that the specified data element is not used by a process anywhere on any data flow diagram.

To resolve the error, be sure that the data element is listed in the composition field of some data flow that either enters or leaves a process on at least one data flow diagram.

ERROR Entity ‘x’ has no composition defined.

The above error indicates that no composition has been defined for the entity. Since balancing entities against data stores is based upon identifying identical compositions, balancing cannot be done.

To resolve the error, add composition information.

ERROR Primary key for table ‘x’ cannot be identified.

The above error indicates that either no primary key has been specified or that the data element named as the primary key has no physical information specified. Correct SQL definition statements cannot be produced.

To resolve the error, either specify a primary key or correct the one that is specified.

ERROR Physical information for data element ‘x’ does not specify length.

The above error indicates that the physical information for the named data element does not specify a length. Certain data types require a length for correct SQL definition statements to be produced.

To resolve the error, display the physical information section for the data element and enter a length.

ERROR Entity ‘x’ has no corresponding file.
ERROR Entity ‘x’ has no corresponding data store.

The above errors are generated when the process model/data model balancing criterion that every entity must correspond to a data store with the same composition has been configured. It either means that the specified entity does not correspond to a data store in the process model or that the compositions of the entity and the intended corresponding data store differ. The first message occurs for the Yourdon/DeMarco methodology, the second for Gane & Sarson.

To resolve the error, be sure that there is a data store in the process model with the same elemental composition as the named entity.

ERROR Columns for table ‘x’ cannot be identified.

The above error indicates that composition information for the entity that generates the identified table has not been supplied. Correct SQL definition statements cannot be produced.

To resolve the error, supply composition information for the entity.

ERROR Entity ‘x’ is involved in a cyclic supertype/subtype relationship.

The above error indicates that two supertype relationships exist between a pair of entities, where the supertype and subtype entities are different for each relationship. There could possibly be a chain of entities between them, still resulting in a cyclic definition. See Figure 4-41.

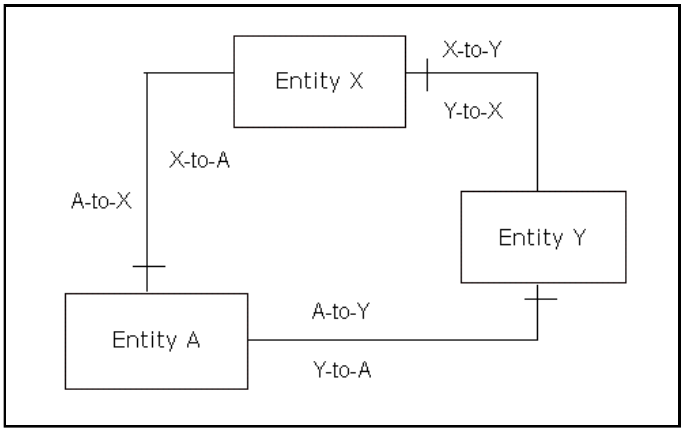


Figure 4-41

To resolve the error, make sure that there are no cyclic supertype relationships in your project.

WARNING Not all subtype group members for entity ‘x’ are shown in this view.

The above warning warns you that all the subtypes that have been defined for a supertype are not shown on a diagram. A supertype can have more than one subtype, and further, the subtypes can be grouped together. When supertype relationships have the same origin point on the supertype, they are considered members of the same group. There is no limit to the number of groups that can be defined for a supertype. To resolve the warning, check to be sure that all subtypes of a particular group are shown with all instances of the supertype and other members of the group.

WARNING A non-supertype/subtype relationship exists between ‘x’ and ‘y’.

The above warning alerts you that there is both a supertype and normal relationship between a pair of entities. If a supertype relationship is defined, there should not be another type of relationship between the parent entity and any of the children defined by the supertype structure.

To resolve the warning, remove the non-supertype/subtype relationship.

ERROR Syntax error in cardinality detail string (position ‘n’) for relationship ‘x’ between entities ‘y’ and ‘z’.

ERROR Syntax error in cardinality detail string (position ‘n’) for reverse relationship ‘x’ between entities ‘y’ and ‘z’.

In addition to specifying cardinality of a relationship using notation on the relationship line, you can specify a specific quantity in the detail field using numbers or a set of intervals, such as ‘1’, ‘1+’, ‘1-3’, or ‘2,4,6’. If this quantity does not conform to the correct syntax or it differs from the notation specified on the line, this error message is generated.

To resolve the error, change the detail to conform to the correct syntax and make sure it is consistent with the drawn cardinality.

ERROR Discriminator element ‘x’ targeted for entity ‘y’ is simultaneously used by more than one subtype group.

The above error message indicates that the indicated entity has more than one group of subtypes, but each group does not have a unique discriminator. If you create exclusive subtype groups (subtypes are in the same group if the relationship lines start from the same point on the supertype), each group must have a different discriminator if you want to collapse the subtypes into the supertype during SQL generation.

To resolve the error, make sure each subtype group that is being collapsed has a different discriminator. To change the discriminator, select the problem relationship and choose Change Item from the Diagram menu.

ERROR Recursive denormalizing cycle detected. Relationship ‘x’ completing cycle has been removed.

The above error message indicates that SQL for a group of entities involved in a relationship chain could not be produced because the denormalization options on the relationships are in conflict. In the figure, if all three relationships had the denormalization option set to Collapse Child, no tables could be produced, so one of the relationships would be dropped.

To resolve the error, either remove one of the relationships in the cycle, or change the denormalization options on the relationships. To change the denormalization option, select the problem relationship and choose Change Item from Diagram menu.

WARNING Discrepancy between specified cardinality of x and prefix/suffix count of y. Collapsing into parent table ‘z’ according to prefix/suffix count.

The above warning message indicates that the numerical cardinality specified for a relationship does not match the number of items in the prefix/suffix list.

To resolve the warning, change either the numerical cardinality for a relationship or the number of items in the prefix/suffix list. To change the cardinality, select the problem relationship, choose Change Item from the Diagram menu, and change the Cardinality Detail field for the From entity. To modify the prefix/suffix list, select the relationship, choose Define from the Repository menu, and select the Cardinality tab.

ERROR Specify at least x denormalizing prefix/suffix names to conform to cardinality of relationship 'y'.

The above error message indicates that the specified relationship has its denormalization option set to Collapse Child and the cardinality of the relationship has its maximum set to Many, but no prefix/suffix names have been specified. If one entity is collapsed into another, a prefix list is used to make unique column names. For example, if there was a relationship between customer and address, and the cardinality specified there was more than one address for each customer, if you choose to collapse the address table into the customer table, you would need at least two prefix names (such as Home_ and Business_) to uniquely identify the address columns in customer.

To resolve the error, change either the cardinality of the relationship or the number of items in the prefix/suffix list. To change the cardinality, select the problem relationship, choose Change Item from the Diagram menu, and change the Maximum Cardinality field for the From entity. To modify the prefix/suffix list, select the relationship, choose Define from the Repository menu and select the Cardinality tab.

CLASS DIAGRAMS

Object Modeling Overview

Class diagrams allow you to express the object model of your project diagrammatically.[13] You can describe the classes in the data you are modeling and the relationships between them by drawing them onto a diagram. Each diagram or view can show an arbitrarily large or small part of your object model. You can show multiple views of your object model by including different combinations of classes and relationships on various diagrams. However, the entire object model is retained in the repository.

[13] For a detailed discussion, refer to Rumbaugh, Blaha, Premerlani, Eddy, and Lorensen, *Object-Oriented Modeling and Design*, Englewood Cliffs: Prentice-Hall, 1991.

All information you place in a view is, of course, captured by the repository and is available to your process model (data flow diagrams), your structure charts, and your data model (entity relationship diagrams), where applicable, if you have an integrated tool set. The Analyze function can assist you in determining any syntax or definition problems with your object model.

Object Modeling Graphics

Class diagramming is designed to give you the ability to clearly show the characteristics of your classes while still allowing a certain degree of flexibility in modeling style. By careful use of configuration settings and the line and symbol sets, you have the ability to model different types of classes and various line terminators that show relationship cardinality.

Class Symbol

The class symbol is the primary graphic used when creating an object model. When a class is added to a diagram, an entry is created in the repository so that additional information can be specified regarding the class, in order to complete its definition. Each class can be associated with other classes through the use of relationship lines. Each class symbol can have a subtype classification assigned when modifying the definition in the repository that is used to provide additional information about how a class is to be used.

- *Standard* class (the default) indicates normal class.
- *Elemental* indicates the class contains no attributes and physical characteristics should be defined. (A data element and a class with an element subtype are equivalent except that methods cannot be specified for a data element.)
- *Structure* indicates a C style structure should be used instead of a class. Members (or attributes) of a structure are public by default while members of a class are private. *Union* indicates a C style union should be used instead of a class.
- *Entity*, *Associative*, *Attributive and View* indicate the class is persistent and can be used on an entity relationship diagram.

Class symbols can be shown on a class diagram at three levels of detail. The diagram can be set to show only the class names, the class name and attributes, or class names, attributes and methods. When changing the detail level, class symbols are automatically resized and the relationships are reconnected properly. See Class Attributes Displayed on Object Model Diagrams in the Drawing Diagrams chapter for details.

Relationship Lines

Relationships can be labeled in one or both directions. You set this value when you create the project. Relationship lines between classes usually have terminators on each end that show if the relationship cardinality is 0:1, 1:1, 0:many, 1:many or many:many. You can set default terminators to show relationship cardinality with Line Settings from the Options menu, as described in Default Line Selections. If Auto Label Lines is enabled, after you add a relationship to your view, a dialog box opens in which you can enter the labels for each direction of the relationship. At that time, you also have the opportunity to change the cardinality from the default to whatever you choose. You can also specify a textual phrase that describes the cardinality.

Different types of relationships can be specified on class diagrams, including inheritance and aggregation relationships. Role names and qualifiers can also be specified.

Analysis for Class Diagrams

Visible rules check for four classes of errors :

- Syntax errors. These are errors that would make your class diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate classes are improperly associated with other classes. Different rules apply depending on whether the relationship type is normal, inheritance or aggregation.
- Use errors. These indicate classes have not been used, either on a diagram or in the definition of another class.
- Definition errors. These indicate a class definition is incomplete, either attributes or methods have not be defined. Different rules apply depending upon the class subtype.

Object Modeling Error Messages

ERROR Class ‘x’ has no attributes defined.

The above error indicates that no attributes have been defined in the repository for the indicated class. Classes with a structure or union subtype must have attributes in order to have a valid definition. All other class subtypes are not required to have attributes, but in most cases a class without attributes should be defined as an elemental class. To resolve the error, specify at least one attribute for the class.

ERROR Class ‘x’ has no methods defined.

The above error (or in some cases warning) indicates that no methods have been defined in the repository for the indicated class. Classes with a standard subtype must have methods in order to have a valid definition, while those with an entity subtype should have methods defined but they are not required. To resolve the error, specify at least one method for the class.

ERROR Class ‘x’ is involved in cyclic inheritance relationship.

The above error indicates that two inheritance relationships exist between a pair of classes, where the base class and the derived class are different for each relationship. There could possibly be a chain of entities between them, still resulting in a cyclic definition. See Figure 4-42.

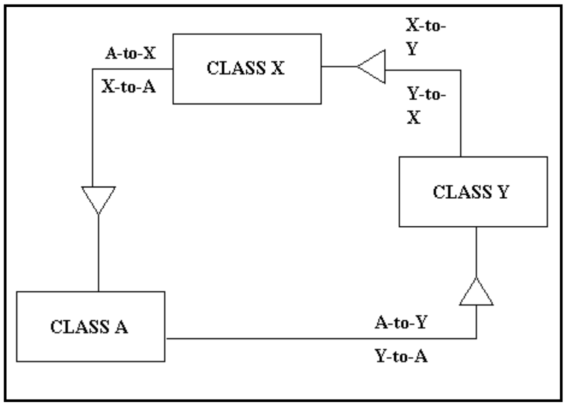


Figure 4-42

To resolve the error, make sure that there are no cyclic inheritance relationships in your project.

ERROR Class[Element] ‘x’ is undefined.

The above error indicates that the physical information for the named elemental class has not been specified. To resolve the error, bring up the physical information page for the class and open the Type box to select a data type.

ERROR Role names have not been assigned for relationship ‘x’ between ‘A and ‘B’.

When a non-inheritance relationship exists between a pair of classes, roles are used to create unique members to provide the linkage between the classes. The type and cardinality of the relationship determine whether roles are required or optional. In all cases, at least one role name must be specified, typically on the child class.

To resolve the error, open a diagram that contains the relationship, select it and open the Change Item dialog box. Add the required role name(s) based on the type of relationship.

ERROR There are ‘x’ unnamed classes.

This error message is the result of not labeling classes on your class diagram(s).

To resolve the error, find and label them.

WARNING A non-inheritance relationship exists between ‘x’ and ‘y’.

The above warning alerts you that there is both an inheritance and a normal relationship between a pair of classes. If an inheritance relationship is defined, there should not be another type of relationship between the base class and any of the derived classes defined by the inheritance structure.

To resolve the warning, remove the non-inheritance relationship.

ERROR Syntax error in cardinality detail string (position 'n') for relationship 'x' between classes 'y' and 'z'.

In addition to specifying cardinality of a relationship using notation on the relationship line, you can specify a specific quantity in the detail field using numbers or a set of intervals, such as 1, 1+. If this quantity does not conform to the syntax defined correct syntax or it differs from the notation specified on the line, this error message is generated.

To resolve the error, change the detail to conform to the correct syntax and make sure it is consistent with the drawn cardinality.

ERROR Syntax error in cardinality detail string (position 'n') for reverse relationship 'x' between classes 'y' and 'z'.

In addition to specifying cardinality of a relationship using notation on the relationship line, you can specify a specific quantity in the detail field 1-3, or 2,4,6. If this quantity does not conform to the correct syntax or it differs from the notation specified on the line, this error message is generated.

To resolve the error, change the detail to conform to the correct syntax and make sure it is consistent with the drawn cardinality.

ERROR Set of relationships for class 'x' is incorrect.

This error indicates that a class appearing on several diagrams, has both inheritance and aggregation relationships attached, but not all relationships are shown on all diagrams.

To resolve the error, check to be sure that all relationships for the class are shown on all diagrams where the class appears.

ERROR Class 'x' has no aggregation attributes defined.

When an aggregation relationship exists between a pair of classes, an instance of the child class should exist in the member list of the parent class to provide the linkage between the classes.

To resolve the error, you have two options: 1) open a diagram that contains the aggregation relationship for the class, select it, open the Change Item dialog box, and then add a role name to the child class; or 2) select the class, open the Define dialog box, and add the child class as a member.

STATE TRANSITION DIAGRAMS

Dynamic Modeling Overview

State transition diagrams (STDs) allow you to express the dynamic model of your project diagrammatically.[14] You can describe the states in the data you are modeling and the events between them by drawing them onto a diagram. Each state transition diagram that you create can should show the dynamic behavior of a single class. You can connect a class to a state transition diagram through the use of the Nest facility. Once connected, every state belongs to the class; and in the repository, the class name prefixes the state name. The Analyze function can assist you in determining any syntax or definition problems with your dynamic model.

Dynamic Modeling Graphics

State transition diagramming is designed to give you the ability to clearly show the dynamic characteristics of your classes while still allowing some degree of flexibility in modeling style. There are two basic elements to a state-transition diagram: states are represented by rectangles and events are represented by lines between states.

State Symbol

The state symbol is the primary graphic used when creating a dynamic model. When a state is added to a diagram, an entry is created in the repository so that additional information can be specified regarding the state in order to complete its definition. If the diagram has been connected to a class through the use of the Nest facility, each state on the connected diagram belongs to the exploded class, and the state name is prefixed by the class name. To represent a transition from one state to another, an event line can be drawn between states.

Events

Events are lines drawn between states to indicate a transition. They are given a name to describe the type of action that occurs to trigger the state change. Events are global objects that are not tied to a specific class. This means the same event can be used on different state

[14] For more detailed information on dynamic modeling, you can refer to the following books:
Rumbaugh, James et al, *Object-Oriented Modeling and Design*. Englewood Cliffs: Prentice-Hall, 1991.
Booch, Grady, *Object-Oriented Design with Applications*. Redwood City: Benjamin/Cummings Publishing Company, 1992.

transition diagrams. Event lines between states usually have a terminator on one end to indicate the direction of the transition. For example, the event Enter Password would cause a state change from Request Password to Verify Account.

Analysis for State Transition Diagrams

Visible rules checks for the following problems:

- Syntax errors. These are errors that would make your state transition diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate states are improperly connected to other states either because no events have been drawn, or because the state has only an input event or only an output event.
- Use errors. These indicate state transition diagrams have not been connected to a specific class.

Dynamic Modeling Error Messages

ERROR There are x unnamed state(s).

This error is the result of not labeling states on your state transition diagrams.

To resolve the error, find and label them using the Change Item function.

ERROR There are x unnamed event(s).

This error is the result of not labeling events on your state transition diagrams.

To resolve the error, find and label them using the Change Item function.

ERROR State labeled ‘x’ is a dangling state.

The above error message is the result of not connecting the indicated state with any event lines.

To resolve the error, move the events and/or the state to appropriately connect them, or use the Connect function.

ERROR Event labeled ‘x’ is a dangling event.

The above error message is the result of not connecting the indicated event to any states.

To resolve the error, move the event and/or the states to appropriately connect them, or use the Connect function.

ERROR State labeled ‘x’ is an input only state.

The above error message indicates incomplete analysis. For a state change to take place, an event must be processed. A state without both an input event and an output event is therefore incomplete.

To resolve the error, create or find the necessary output event and connect it to the state.

ERROR State labeled ‘x’ is an output only state.

The above error message indicates incomplete analysis. For a state change to take place, an event must be processed. A state without both an input event and an output event is therefore incomplete.

To resolve the error, create or find the necessary input event and connect it to the state.

WARNING Event labeled ‘x’ is an input only event.

The above warning message indicates incomplete analysis. An event signals a change from one state to another. In most cases, every event is attached to two states, the starting state before the event and the ending state after the event. If the event is not connected to a starting state, it could either signal an error or the initial event for the STD.

If an event is shown as an input only event on one state transition diagram, it may be shown on another diagram attached to a starting state. In this case, this warning message is not displayed.

To resolve the warning, create or find the necessary starting state and connect it to the event.

WARNING Event labeled ‘x’ is an output only event.

The above warning message indicates incomplete analysis. An event signals a change from one state to another. In most cases, every event is attached to two states, the starting state before the event and the ending state after the event. If the event is not connected to a ending state, it could either signal an error or the final event for the STD.

If an event is shown as an output-only event on one state transition diagram, it may be shown on another diagram attached to an ending state. In this case, this warning message is not displayed.

To resolve the warning, create or find the necessary ending state and connect it to the event.

WARNING State diagram ‘x’ is not associated with a class.

The above warning message indicates the diagram has not been attached to a class using the Nest function. When performing object modeling, a state transition diagram shows the dynamic behavior of a class, and as such should be tied directly to the class. When a class is exploded to a state transition diagram, all the states in the repository are members of that class and their names are of the form class::state.

To resolve the warning, open a class diagram, select an appropriate class, and choose Nest from the File menu. Choose Explode and select the indicated diagram. See Nested Decomposition for more information.

ENTITY LIFE HISTORY DIAGRAMS

Data Modeling Overview

The entity life history (ELH) diagram is a component of both the SSADM and Métrica methodologies and is similar to a structure chart. The ELH shows how events in a system affect data entities. An ELH applies to a single entity that originates on an entity-relationship diagram. There can be an ELH for each entity in the repository. An ELH can be exploded to or from an instance of an entity on an entity relationship diagram.

An ELH diagram is derived from the Jackson structure diagram and represents the processing performed on the data depicted by an entity. An ELH diagram should contain a single entity at the top of the diagram that originates on an entity-relationship diagram, with symbols arranged below the entity representing the events that affect an entity and the processing performed on it.

On an ELH diagram, symbols called structure boxes are displayed below a single entity that appears at the top of the diagram. Below the structure boxes are symbols called event boxes. Structure boxes are used when necessary to meet two requirements of ELH diagrams:

- Event types (selection, iteration) cannot be mixed.
- Box types cannot be mixed on a particular level.

Just like structure charts, ELH diagrams are intended to be read from left to right and top to bottom. Because an ELH diagram applies to an entity, the event processing deals with the creation, processing, and deletion of the entity.

Entity Life History Graphics

Entity life history diagrams use a set of seven symbols to represent the relationship between an entity, the events that affect the entity, and operations used as a result of an event.

Entity

The entity is the object of the diagram. The symbol originates on an entity-relationship diagram.

Structure Box

The structure box is a placeholder symbol used to prevent the mixing of different event types.

Sequence Event

A sequence event shows a particular event that affects the entity. It is neither a selection event nor an iteration event.

Iteration Event

An iteration event shows that an event may affect the occurrence of the entity more than once.

Operation

An operation symbol refers to the operation performed as the result of an event. An operation can also appear on a structure chart. An operation has both a name and a number. The number appears inside the symbol, while the name is listed in a table that is automatically maintained by Visible Analyst.

Parallel Life

A parallel life symbol indicates that events may occur within the life of an entity, but not in a prescribed order.

Analysis for Entity Life History Diagrams

Visible rules check for four classes of errors:

- Syntax errors. These are errors that would make your ELH diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate symbols are improperly associated with other symbols. Different rules apply depending on the type of symbols connected.
- Structure errors. At a minimum, there be at least event blocks defined on an ELH. In addition, event types cannot be mixed at different levels of the tree structure.

Analysis Error Messages

WARNING There are x unnamed structure box(es).

This warning is the result of not labeling structure boxes on your entity life history diagram. To resolve the warning, find and label them using the Change Item function. However, since structure boxes are simply placeholders, it is not necessary to label them.

ERROR There are ‘x’ unnamed sequence event(s).

This error is the result of not labeling sequence events on your entity life history diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed selection event(s).

This error is the result of not labeling selection events on your entity life history diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed iteration event(s).

This error is the result of not labeling iteration events on your entity life history diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed operation(s).

This error is the result of not labeling operations on your entity life history diagram. To resolve the error, find and label them using the Change Item function. Note that even though the name does not appear in the operation symbol, the name must exist and will be displayed in the operation table.

ERROR There should be only one entity on this diagram.

An entity life history diagram describes the events and operations of a single entity. Remove all but the desired entity for this diagram.

ERROR Entity labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated entity to any events on an entity life history diagram. To resolve the error, draw a line from the entity to any event type or structure box using the Connect function.

ERROR Structure box labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated structure box to an entity or any events on an entity life history diagram. To resolve the error, draw a line connecting the structure box to an entity or an event using the Connect function.

ERROR Sequence event labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated sequence event to an entity or any events on an entity life history diagram. To resolve the error, draw a line connecting the structure box to an entity or event using the Connect function.

ERROR Selection event labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated selection event box to an entity or a structure box on an entity life history diagram. To resolve the error, draw a line connecting the event to an entity or structure box using the Connect function.

ERROR Iteration event labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated iteration event to an entity or a structure box on an entity life history diagram. To resolve the error, draw a line connecting the event to an entity or structure box using the Connect function.

ERROR Operation labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated operation to an event on an entity life history diagram. To resolve the error, draw a line connecting the operation to an event using the connect function.

ERROR Operation labeled ‘x’ cannot be connected to an entity.

The above error message is the result of connecting the indicated operation to an entity. Operations can only be connected to events. To resolve the error, connect the operation to an event using the Connect function, or remove the connection to the entity.

ERROR There must be at least two event blocks defined on this diagram.

Event entity must have a creation event and a destruction event. In addition, there may be other events that take place in between. This error indicates there are not at least two connections from the defining entity to event symbols. To resolve the error, connect the entity to at least two events using the Connect function.

ERROR Event labeled ‘x’ must be of the same type as the other events at this level.

All symbols hanging from a single node must be of the same type, or be a structure box that is used as a placeholder. To resolve the error, change the symbol type of the indicated event using the Change Item function.

ERROR Event labeled ‘x’ has more than one input connection.

The above error message is the result of connecting the indicated event to more than one superior event/structure boxes. To resolve the error, move or delete all but one of the lines to connect the event to one and only one superior event/structure box.

ERROR Operation labeled ‘x’ has more than one input connection.

The above error message is the result of connecting the indicated operation to more than one event. To resolve the error, move or delete all but one of the lines to connect the operation to one and only one event.

ERROR Entity labeled ‘x’ should have no input connections.

The entity on an entity life history diagram must be at the top of the hierarchy. To resolve the error, remove all input connections to the entity.

WARNING Entity life history diagram ‘x’ is not associated with an entity.

This warning is the result of not exploding an entity on an entity relationship diagram to the indicated entity life history diagram. To resolve the warning, open an entity relationship diagram on which the entity appears, select the entity, and use the Explode function to attach to the indicated ELH diagram.

USE CASE DIAGRAMS

Use Case Modeling Overview

The use case diagram is a component of the Unified Modeling Language (UML) and is used to show the interaction between users and a system. A use case diagram is a graph of actors and use cases, together with their relationships.

Use Case Graphics

Use case diagrams use a set of three symbols: use case, actor and system boundary. In addition, lines can be used to indicate relationships among the symbols: communicates, extend, include, and generalization.

Use Case

The use case is represented by an ellipse where the name appears at the top.

Actor

An actor is represented by a ‘stick man’ figure. The name of the actor appears below the object.

Communicates

Communicates is a relationship line between actors and use cases. This relationship is drawn as a solid line with no arrowhead. Cardinality can be shown.

Extends

Extends is a relationship line between two use cases. It is drawn as a dashed line with a stick arrowhead at one end. The name of the relationship is <<extends>>.

Includes

Includes is a relationship line between two use cases. It is drawn as a dashed line with a stick arrowhead at one end. The name of the relationship is <<includes>>.

Generalization

Generalization is a relationship line between two use cases or between two actors. It is drawn as a solid line with an open arrowhead at one end. It is similar to a generalization relationship on a class diagram.

System Boundary

System boundary is a box with a tab (folder) drawn around use case symbols to indicate a complete system. The name of the system appears on the tab along with a stereotype name, unless it does not contain any objects, in which case the name is drawn inside the main part of the box. System boundary symbols can be related using both generalization and non-generalization relationships.

Analysis for Use Case Diagrams

Visible rules check for two classes of errors:

- Syntax errors. These are errors that would make your use case diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate symbols are improperly associated with other symbols. Different rules apply depending on the type of symbols connected.

Analysis Error Messages

ERROR There are ‘x’ unnamed note(s).

This error is the result of not labeling note symbols on one of several types of UML diagrams. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed actors(s).

This error is the result of not labeling actors on your use case diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed use case(s).

This error is the result of not labeling use cases on your use case diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed system boundary(s).

This error is the result of not labeling system boundaries on your use case diagram. To resolve the error, find and label them using the Change Item function.

WARNING There are ‘x’ unnamed relationship(s).

This warning is the result of not labeling relationships on your use case diagram. To resolve the warning, find and label them using the Change Item function.

ERROR Actor labeled ‘x’ is dangling.

This error message is the result of not connecting the indicated actor to another actor or use case on the use case diagram. To resolve the error, draw a line from the actor to any use case or actor using the Connect function.

WARNING Use case labeled ‘x’ is dangling.

This warning message is the result of not connecting the indicated use case to another use case on the use case diagram. To resolve the warning, draw a line from the use case to any use case using the Connect function.

ERROR System boundary labeled ‘x’ has no components.

This error message indicates the named system boundary does not have any use case object defined in its composition. A system boundary is used to group objects. To resolve the error, either delete the system boundary or add components using Define.

ERROR Stereotype not defined for relationship between use cases ‘x’ and ‘y’.

For relationships between a pair of use case symbols, a stereotype must be specified, indicating the type of relationship. A name is optional. To resolve the error, find the relationship and set the stereotype using the Change Item function.

SEQUENCE DIAGRAMS

Sequence Modeling Overview

A sequence diagram is a component of the Unified Modeling Language (UML). It is a type of iteration diagram that describes how objects collaborate in some behavior. It is drawn in a grid-like fashion where the vertical axis represents time, while the horizontal axis represents the participating objects.

Sequence Graphics

Sequence diagrams have one basic symbol, an object, represented by a box where the name is in the form ObjectName:ClassName. A dashed line, known as a lifeline, extends from the bottom of the object to the bottom of the sequence diagram. Adornments can appear along the lifeline:

Activation, represented as a thin rectangle, indicates when an object is active.
Deletion, represented by a large X at the end of an object lifeline, is used to indicate when an object is destroyed.

Each object must be based on a class. The object name is optional.

In addition, lines are used to connect object lifelines to indicate message passing. There are several types of messages that can be used:

- Procedure call, a solid line with filled arrowhead.
- Flat flow of control, a solid line with stick arrowhead.
- Asynchronous stimulus, a solid line with half-stick arrowhead.
- Return, a dashed line with stick arrowhead.
- Self-delegation, an arc line.

Each message must be based on a method in the target class.

Analysis for Sequence Diagrams

Visible rules check for two classes of errors:

- Syntax errors. These are errors that would make your use case diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate symbols are improperly associated with other symbols. Different rules apply depending on the type of symbols connected.

Analysis Error Messages

ERROR	There are ‘x’ unnamed note(s).

This error is the result of not labeling note symbols on one of several types of UML diagrams. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed object(s).

This error is the result of not labeling objects on your sequence diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed message(s).

This error is the result of not labeling messages on your sequence diagram. To resolve the error, find and label them using the Change Item function.

ERROR Object ‘x’ is not used. There are no messages attached.

This error is the result of not having messages attached to an object lifeline on a sequence diagram. To correct the error, connect two object lifelines with a message line.

WARNING Object ‘x’ is based on an abstract class.

An object used on a sequence diagram is based on an abstract class. Since abstract classes cannot be directly instantiated, a derived class should probably be used instead.

WARNING Message ‘x’ uses a virtual method.

A message used on a sequence diagram is based on a virtual method. Since virtual methods are meant to be overridden, a method in a derived class should probably be used instead.

ERROR Message ‘x’ uses a pure virtual method.

A message used on a sequence diagram is based on a pure virtual method. Since pure virtual methods cannot be executed, a method in a derived class should be used instead.

ERROR Message ‘x’ refers to a method that does not exist in the derivation tree of class Y.

The indicated message is based on a method that does not exist in the indicated class or one of its base classes. To correct the problem, select the item and use the Change Item function.

COLLABORATION DIAGRAMS

Collaboration Modeling Overview

The collaboration diagram is a component of the Unified Modeling Language (UML). It is a second form of iteration diagram that shows an interaction organized around the objects in the interaction and their links to each other. Unlike a sequence diagram, a collaboration diagram shows the relationships among the object roles. On the other hand, a collaboration diagram does not show time as a separate dimension; so the sequence of messages is specified using sequence numbers.

Collaboration Graphics

Collaboration diagrams have one basic symbol, an object, represented by a box where the name is in the form ObjectName:ClassName. Each object must be based on a class. The object name is optional.

In addition, lines are used to connect objects to indicate message passing. Each message is represented by a shore line along side the object link line. There are several types of messages that can be used:

- Procedure call, a solid line with filled arrowhead.
- Flat flow of control, a solid line with stick arrowhead.
- Asynchronous stimulus, a solid line with half-stick arrowhead.
- Return, a dashed line with stick arrowhead.
- Self-delegation, an arc line.

Each message must be based on a method in the target class.

Analysis for Collaboration Diagrams

Visible rules check for two classes of errors:

- Syntax errors. These are errors that would make your use case diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate symbols are improperly associated with other symbols. Different rules apply depending on the type of symbols connected.

Analysis Error Messages

ERROR There are ‘x’ unnamed note(s).

This error is the result of not labeling note symbols on one of several types of UML diagrams. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed object(s).

This error is the result of not labeling objects on your collaboration diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘x’ unnamed message(s).

This error is the result of not labeling messages on your collaboration diagram. To resolve the error, find and label them using the Change Item function.

ERROR Object ‘x’ is not used. There are no messages attached.

This error is the result of not having messages attached to an object link on a collaboration diagram. To correct the error, add a message to an object link line.

WARNING An object used on a collaboration diagram is based on an abstract class.

Since abstract classes cannot be directly instantiated, a derived class should probably be used instead.

WARNING Message ‘x’ uses a virtual method.

A message used on a collaboration diagram is based on a virtual method. Since virtual methods are meant to be overridden, a method in a derived class should probably be used instead.

ERROR Message ‘x’ uses a pure virtual method.

A message used on a collaboration diagram is based on a pure virtual method. Since pure virtual methods cannot be executed, a method in a derived class should be used instead.
ERROR Message ‘x’ refers to a method that does not exist in the derivation tree of class Y.

The indicated message is based on a method that does not exist in the indicated class or one of its base classes. To correct the problem, select the item and use the Change Item function.

ACTIVITY DIAGRAMS

Activity Modeling Overview

An activity diagram is a component of the Unified Modeling Language (UML). It is a special form of a state diagram in which the states represent the performance of actions or subactivities. Transitions are triggered by the completion of the actions or subactivities. It represents a state machine of a procedure itself.

Activity Graphics

Activity diagrams use a set of six symbols.

Action State

An action state is represented by a rectangle with round corners.

Decision/Merge

A decision is represented by a diamond with one incoming transition and two or more outgoing transitions.

Synchronization Bar

A synchronization bar is represented by a black bar with one or more input transitions and one or more output transitions. A synchronization bar is used to signal parallel activities.

Start

The start is represented by a filled circle. It is used to indicate the starting point of the activity.

End

The end is represented by a filled circle inside a hollow circle. It is used to indicate the end point of the activity.

Swimlane

A swimlane is represented by a box containing action states. Swimlanes are used to organize responsibilities for actions. The name appears at the top of the box.

In addition, lines are used to connect action states to indicate a transition, represented by a solid line with a stick arrowhead. It is labeled by a transition string of the form ‘event signature [guard condition]/action expression’.All components of the transition string are optional.

Analysis for Activity Diagrams

Visible rules check for two classes of errors:

- Syntax errors. These are errors that would make your use case diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate symbols are improperly associated with other symbols. Different rules apply depending on the type of symbols connected.

Analysis Error Messages

ERROR	There are ‘x’ unnamed note(s).

This error is the result of not labeling note symbols on one of several types of UML diagrams. To resolve the error, find and label them using the Change Item function.

ERROR	There are ‘x’ unnamed swimlane(s).

This error is the result of not labeling swimlanes on your activity diagram. To resolve the error, find and label them using the Change Item function.

WARNING	Activity diagram ‘x’ is not associated with a class, use case, or activity.

This warning is the result of not exploding a class, use case or activity to the indicated activity diagram. To resolve the warning, open a class, use case, or activity diagram, select an appropriate object, and use the explode feature to attach to the indicated activity diagram.

BUSINESS PROCESS MODELING DIAGRAMMING (BPMN)

Business Process Modeling Overview

Visible Analyst provides support for Business Process Modeling Notation (BPMN) diagrams based on the Business Process Modeling Initiative developed by the Object Management Group (OMG). [2]

The primary goal of BPMN is to provide a modeling notation is an effective communication medium across all the constituencies in computing technology supported organizations. BPMN is specifically designed to communicate process behavior information in a manner easily understood by business end users while providing supporting technology organizations with sufficient information about process execution, flow and dependencies to understand the workings of the business processes being modeled. BPMN notation is designed therefore to support the needs of not only business end users, but business analysts who develop models and technical analysts who implement the model processes.

[2] The complete specification is available for download from the OMG website, www.omg.org.

Future releases of the Visible Analyst Workbench are planned that will use defined BPMN models to generate execution languages such as BPEL4WS (Business Process Execution Language for Web Services).

BPMN models describe business process behavior and as a result use an event based paradigm. Both parallel and conditional behavior is supported in the modeling notation and also in the Visible Analyst's implementation of BPMN. A number of symbols are used to describe process flows, events and decisions and allow the viewer to easily differentiate between sections of the BPMN diagram.

BPMN Model types

There are three types of sub-models that may be used within an end-to-end BPMN model:

Functional Private (internal) business processes - Private business processes internal to an organization, such as the Sales Department. If swimlanes (described below) are used, then a private business process will be contained within a single pool, with all Sequence Flows contained within the Pool and no Message Flows crossing the boundaries of the Pool.

Abstract (public) processes - These processes represent the interactions between a private business process and another process or participant. Only those activities that are used to communicate outside the private business process, plus the appropriate flow control mechanisms are included in the abstract process. The "internal" activities of the private business process are not shown in the abstract process. Thus, the abstract process shows to the outside world the sequence of messages that are required to interact with that business process. An example of this would be the interaction between the Customer and the company's Sales Department.

Collaboration (global) processes - A collaboration process depicts the interactions between two or more business entities, and is defined as a sequence of activities that represent the message exchange patterns between the entities involved. A collaboration process can be shown as two or more abstract processes communicating with each other. The interaction of the company's Accounting Department and Sales Department and the processes performed by both to complete the sales cycle is an example of a collaboration process.

Flow Objects

Flow Objects are the main graphical elements of the diagram and consist of Event, Activity (process) and Gateway symbols.

Event

An event is something that “happens” during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). There are three types of Events based on the way they affect the flow: Start, Intermediate and End. Each event trigger has a unique appearance to indicate the trigger type.

A **Start Event** indicates when a particular Process will start. In terms of Sequence Flow, the Start Event starts the flow of the Process and will not have any incoming Sequence Flow. A Start Event can have a Trigger that indicates how the Process Starts: Message, Timer, Rule, Link and Multiple.

An **Intermediate Event** is an event that occurs after a process has been started and can affect the flow of the activity, but not Start or (directly) terminate the activity. An Intermediate Event will show where messages or delays are expected within the process, disrupt the flow through exception handling, or show the extra flow required compensating a transaction. Intermediate events can be added to the boundary of the activity symbol to indicate an error. Intermediate events can be defined as the following types: Cancel, Compensation, Error, Link, Message, Multiple, None, Rule or Timer. An Intermediate Event symbol is a circle drawn with a thin double line.

The **End Event** indicates when a process will end, and no Sequence Flows would emanate from an End event. End Events may have certain Results as specified on the Change Event dialog. The End Event Results types available are: Cancel, Compensation, Error, Link, Message, Multiple, None and Terminate. An End Event symbol is a circle drawn with a single bolded line.

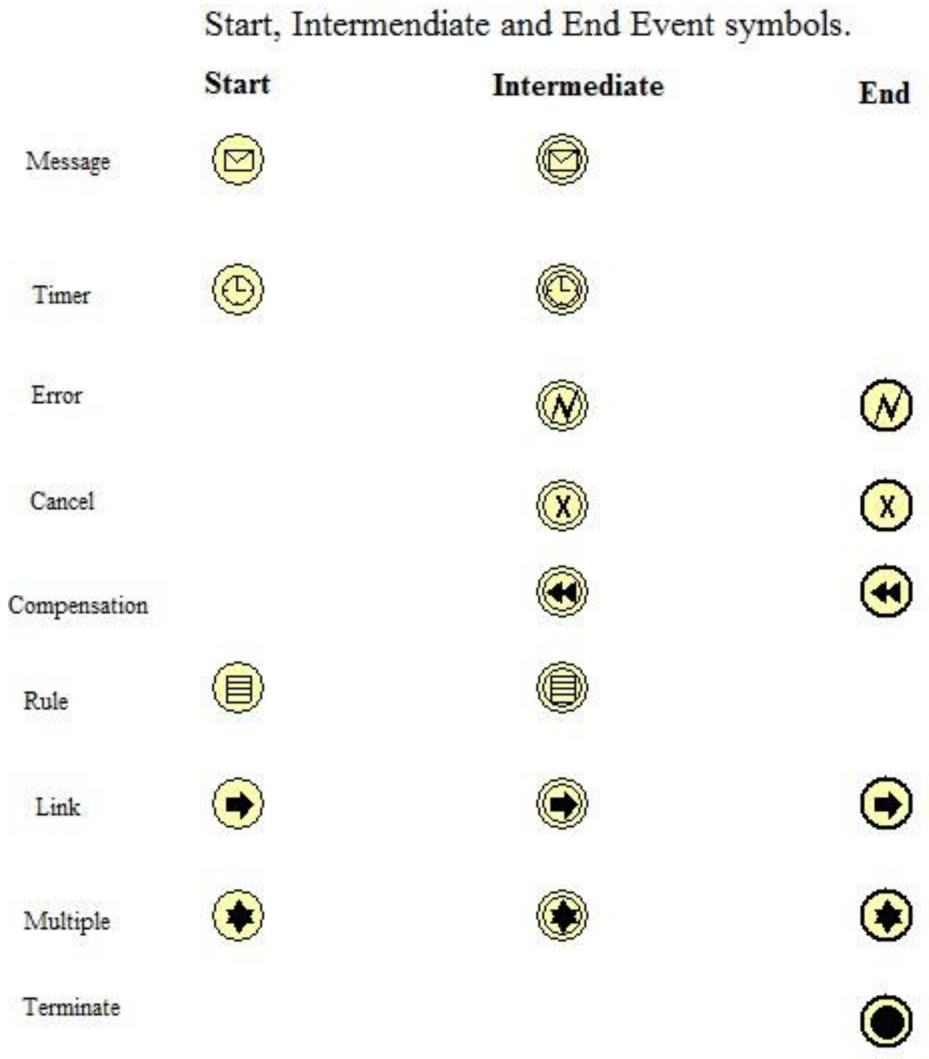


Figure 4-43 Event Symbols

Message Event

A Message Start Event indicates that a message arrives from a participant and triggers the start of the Process.

A Message Intermediate Event indicates that a message arrives from a participant and triggers theEvent. This causes the Process to continue if it was waiting for the message, or changes the flow for exception handling. In Normal Flow, Message Intermediate Events can be used for sending a message to a participant. If used for exception handling it will change the Normal Flow into an Exception Flow.

A Message End Event indicates that a message is sent to a participant at the conclusion of the Process.

Timer Event

A Timer Start Event indicates a specific time-date or a specific cycle (e.g. every Monday at 9 AM) that can be set to trigger the start of a Process.

A Timer Intermediate Event indicates a specific time-date or a specific cycle (e.g. every Monday at 9 AM) that can be set to trigger the Event. If used within the main flow it acts as a delay mechanism. If used for exception handling it will change the Normal Flow into an Exception Flow.

Error Event

An Error End Event indicates that a named error should be generated. This Error will be caught by an Intermediate event within the Event Context. The Error text is entered into the Error Code field on the End Event’s Trigger tab in the repository.

An Error Intermediate Event is used for error handling both to set (throw) and react to (catch) errors. It sets (throws) an error if the Event is part of a Normal flow. It reacts to (catches) a named error, or to any error if a name is not specified, when attached to the boundary of an activity. The Error text is entered into the Error Code field on the Intermediate Event’s Trigger tab in the repository.

Cancel Event

A Cancel Intermediate Event symbol must be attached to the boundary of the sub- process. It SHALL be triggered if a Cancel End Event is reached within the Transaction sub-process. It also SHALL be triggered if a Transaction Protocol “Cancel” message has been received while the Transaction is being performed.

A Cancel End Event designation indicates that a transaction should be cancelled, and it will trigger a Cancel Intermediate Event attached to a sub-process boundary. Additionally, a Transaction Protocol Message will be sent to any entities (participants) involved in the Transaction.

Compensation Event

A Compensation Intermediate Event is used for compensation handling - - both setting and performing compensation. It calls for compensation if the Event is part of a Normal Flow. It reacts to a named compensation call when attached to the boundary of an activity.

A Compensation End Event indicates that a Compensation is necessary. The Compensation identifier will trigger an Intermediate Event when the Process is rolling back.

Rule Event

A Rule Start Event is triggered when the rule condition becomes true, such as "S&P changes by more than 10%". Enter the rule expression in the Rule field on the Event's Trigger tab.

A Rule Intermediate Event is only used for exception handling and is triggered when a Rule becomes True. Enter the rule expression in the Rule field on the Event's Trigger tab.

Link Event

A Link Start Event is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two sub-processes within the same parent process.

A Link End Event is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two sub-processes within the same parent process. A Token arriving at the Link End Event will immediately jump to its corresponding target Start or Intermediate Event.

A Link Intermediate Event is a mechanism for connecting an End Event (Result) of one process to an Intermediate Event (Trigger) in another process. Paired Intermediate Events can also be used as "Go To" objects within a process.

Multiple Event

A Multiple Start Event means that there are multiple ways of triggering the Process. Only one of them will be required to start the Process. The attributes of the Start Event will define which of the other types of Triggers apply.

A Multiple Intermediate Event means that there are multiple ways of triggering the Event, but only one of them will be required. The attributes of the Intermediate Event will define which of the other types of Triggers apply.

A Multiple End Event means that there are multiple consequences of ending the Process. All of them will occur (e.g. there might be multiple messages sent). The attributes of the End Event will define which of the other types of Results apply.

Terminate Event

A Terminate End Event indicates that all activities in the Process should be immediately ended. This includes all instances of Multi-Instances. The Process is ended without compensation or event handling.

Activity

An activity (process) is a generic term that represents the work that a company or organization performs. Activities can be atomic or non-atomic (compound) and are represented as a rounded rectangle. Activity types include Process, Sub-Process (which can be defined as a Transaction), and Task.

A **Process** is an activity performed within a company or organization. In BPMN, a Process is depicted as a network of Flow Objects (Events, Activities, Gateways), which are a set of other activities and the controls that sequence them.

A **Sub-Process** is a compound activity that is included within another process. It is compound in that it can be broken down into a finer level of detail (a Process) through a set of sub-activities.
There are 5 standard markers to indicate a sub-process in the Visible Analyst, i.e. a process that shows additional detail on the child diagram. More than one of these markers can be associated with a process.
The plus sign + within an activity symbol indicates a collapsed sub-process(s) with additional detail on the child diagram.
The Loop marker is a small line with an arrowhead that curls back upon itself.
A pair of parallel vertical lines indicates a Multiple Instance sub-process.
An Ad Hoc sub-process marker is indicated by the tilde symbol, ~.
The Compensation sub-process marker is a pair of left facing triangles, similar to the symbol for a tape rewind button.

A **Task** is an atomic activity that is included within a process. A task is used when the work in a process is not broken down to a finer level of Process Model detail. Generally an end-user or an application is used to perform the Task when it is executed. There are 3 BPMN Task types, with each type indicated by a specific marker with the Task activity symbol:
Loop: The marker is a small line with an arrowhead that curls back upon itself.
Multiple Instance: The marker is a pair of vertical lines in parallel.
Compensation: The marker is a pair of left facing triangles.

Gateway

A Gateway is used to control the divergence and convergence of multiple Sequence Flows. It will determine branching, forking, merging, and joining of paths.

The Gateway, sometimes called a “fork in the road”, is used to indicate decisions, where the Sequence Flow can take two or more alternative paths. For a given performance (or instance) only one of the paths can be taken. A Decision is not an activity from the business perspective, but is a type of Gateway that controls the Sequence Flow between activities. It can be thought of as a question that is asked at that point in the process, and the question has a defined set of alternative answers (Gates). Each Decision Gate is associated with a condition expression found within an outgoing Sequence Flow. The conditions are evaluated in a specific order and first condition that evaluates to “True” determines the Sequence Flow that will be taken. One of the Gates may be marked as “default”, and is the last Gate considered. Choosing a particular Gate chooses the corresponding Sequence Flow.

Exclusive Gateway (XOR)

An Exclusive Decision (Gateway XOR) has two or more outgoing sequences, but only one of them may be taken. The conditions are evaluated in a specific order and first condition that evaluates to “True” determines the Sequence Flow that will be taken. One of the Gates may be marked as “default”, and is the last Gate considered. Exclusive Gateways can be used as a merge, for alternative Sequence Flow, though rarely used in this way. There are two types of Exclusive Decisions: Data-Based and Event-Based.

Inclusive Gateway (OR)

The Inclusive Gateway OR can be used to model Inclusive Decisions or it can be used as a Merge. An Inclusive Decision Gateway symbol has a Bolded circle within the diamond Gateway symbol.

When used to model Inclusive Decisions, each of the Sequence Flows (at least two) would be attached to the Gates of the Gateway. The Condition Type of the Sequence Flow is set to Expression, and the modeler would add the expression to be evaluated in the Condition Expression field of the flow. A default Sequence Flow may be used.

When the Inclusive Gateway is used as a Merge, it will wait for (synchronize) all Tokens that have been produced upstream. It does not require that all incoming Sequence flow produce a Token, (as the Parallel Gateway does). It requires that all Sequence Flow that were actually produced by an upstream (by an Inclusive OR situation, for example). If an upstream Inclusive OR produces two out of a possible three Tokens, then the downstream Inclusive OR will synchronize those two Tokens and not wait for another Token, even though there are three incoming Sequence flows.

NOTES:

If there is only one Gate, the Gateway is acting as a Merge, then the outgoing Sequence Flow MUST have its Condition set to “None”.

If there is a Default Gate, it MUST have an associated Sequence Flow, and the Sequence Flow Must have its Condition set to “Default”.

Complex Gateway

Complex Gateways are included to handle situations that are not easily handled through the other types of Gateways. Complex Gateways can also be used to combine a set of linked simple Gateways into a single, more compact situation. Modelers can provide complex expressions that determine the merging/splitting behavior of the Gateway. The symbol for a Complex Gateway contains the asterisk symbol within the Gateway diamond symbol

When the Gateway is used as a Decision, then the expression determines which of the outgoing Sequence Flow will be chosen for the Process to continue. The expression may refer to process data and the status of the incoming Sequence Flow. For example, the expression may evaluate the Process data and then select different sets of outgoing Sequence Flow, based on the results of the evaluation. The expression should be designed so that at least one of the outgoing Sequence Flows will be chosen.

When the Gateway is used as a Merge, then there will be an expression that will determine which of the incoming Sequence Flow will be chosen for the Process to continue. The expression may refer to process data and the status of the incoming Sequence Flow. For example, the expression may specify that any 3 out of 5 incoming Tokens will continue the Process. Another example would be an expression that specifies that a Token is required from Sequence Flow “a”, and that a Token from either Sequence Flow “b” or “c” is acceptable. The expression should be designed so that the Process is not stalled at that location.

Parallel Gateways

Parallel Gateways provide a mechanism to synchronize parallel flow and to create Parallel. Flow These Gateways are not required to create parallel flow, but they can be used to clarify the behavior of complex situations where a string of Gateways are used and parallel flow is required.

There MAY be zero or more Gates. Gates are allowed if the Gateway is the last object in a Process flow, and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow (i.e. the Gateway is acting as a fork), then there MUST be at least two Gates. Each Gate must have an associated Sequence Flow. The Sequence Flow MUST have its Condition attribute set to “None”. The Parallel Gateway symbol contains a plus sign within the Gateway diamond symbol to indicate a Parallel Gateway.

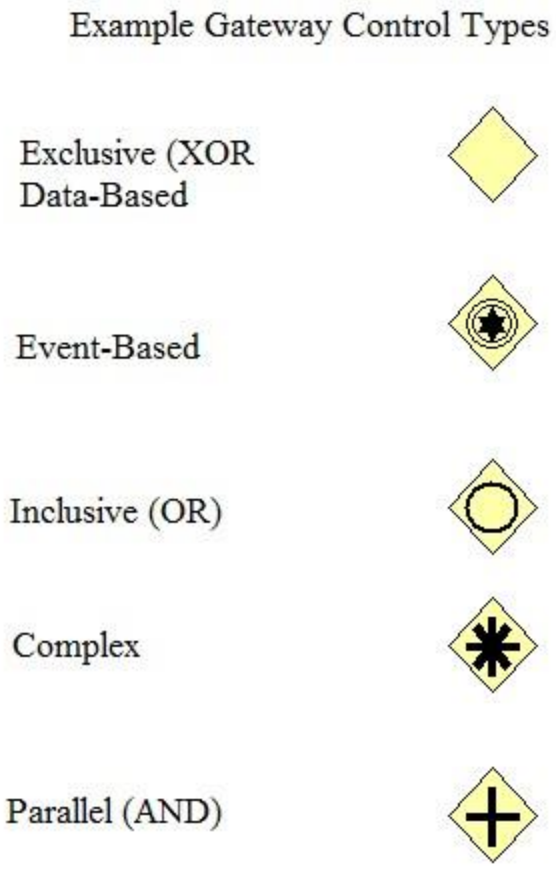
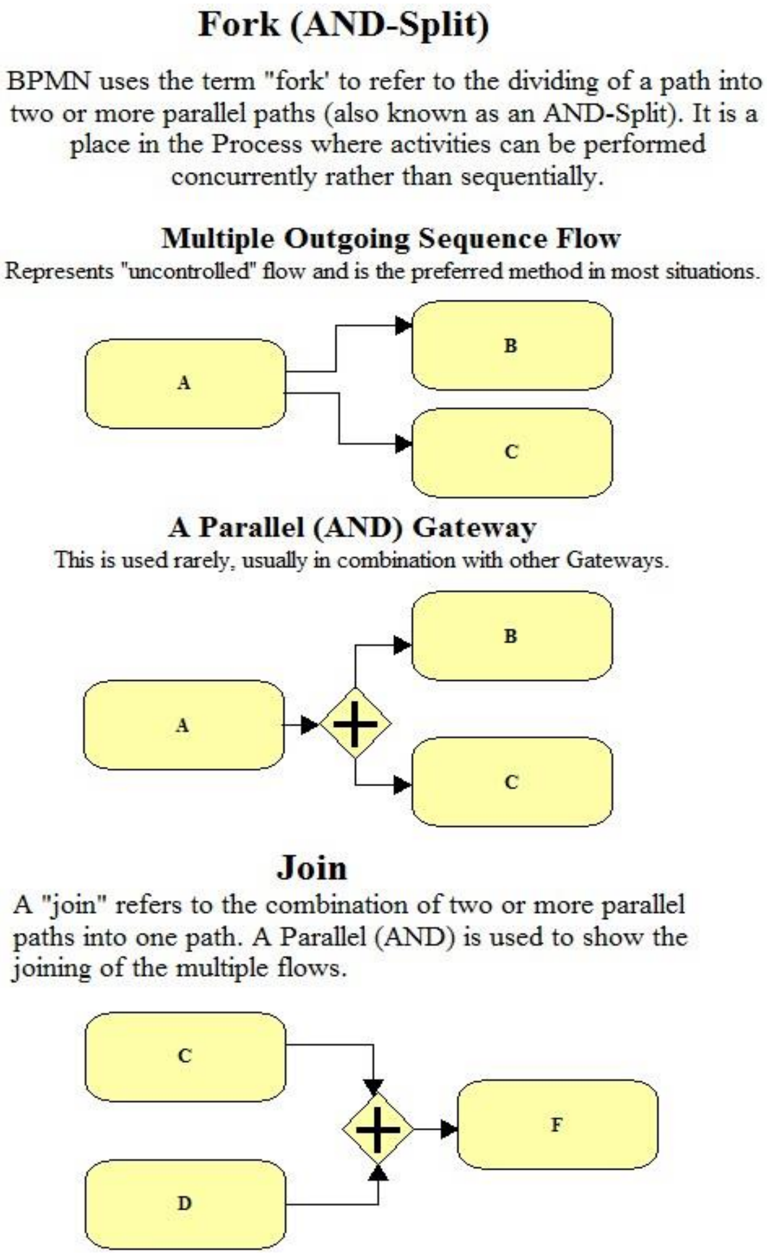


Figure 4-44 Gateway Symbols

Fork, Split and Join

A **Fork** (also known as a Split) is a point in a process where a single flow is divided into 2 separate flows, and allows activities to be performed concurrently instead of sequentially.

A **Join** is a point in the process where two or more parallel Sequence Flows are combined into one Sequence Flow. A Parallel (AND) Gateway is used to show the joining of the multiple Flows. Also known as an AND-Join or synchronization.



Fork (AND-Split)

BPMN uses the term "fork' to refer to the dividing of a path into two or more parallel paths (also known as an AND-Split). It is a place in the Process where activities can be performed concurrently rather than sequentially.

Multiple Outgoing Sequence Flow

Represents "uncontrolled" flow and is the preferred method in most situations.

A Parallel (AND) Gateway

This is used rarely, usually in combination with other Gateways.

Join

A "join" refers to the combination of two or more parallel paths into one path. A Parallel (AND) is used to show the joining of the multiple flows.

Figure 4-45 Fork, Split and Join Examples

Business Process

A Business Process is defined in the Visible Analyst repository and describes the interactions between the business units. A Business Process contains one or more Processes (Activities) as displayed on a Business Process Model Diagram.

Process Type field
This field provides information about which lower-level language the Pool will be mapped. By default, Type is “None” (or undefined).
A *Private* process type MAY be mapped to an executable BPEL4WS process.
An *Abstract* process type is also called the public interface of a process (or other web services) and MAY be mapped to an abstract BPEL4WS process.
A *Collaboration* process type is also considered a “global” process and MAY be mapped to languages such as ebXML or WS Choreography.
Status
The Status of a Process is determined when the Process is being executed by a process engine. The Status of a Process can be used within Assignment Expressions.
Ad Hoc
Specifies whether the Process is Ad Hoc or not. The activities within an Ad Hoc Process are not controlled or sequenced in a particular order, their performance is determined by the performers of the activities.
Ordering
Defines if the activities within the process can be performed in Parallel or must be performed sequentially. The default setting is Parallel and the setting of Sequential is a restriction on the performance that may be required due to shared resources.
Completion Condition
Defines the conditions when the process will end. If the process is Ad Hoc, this attribute MUST be included.
Suppress Join Failure
Specifies whether or not BPEL4WS joinFailure fault will be suppressed for all activities in the BPEL4WS process. This option is located on a Business Process’s Type / Status tab in the repository
Enable Instance Compensation
This check is included on a Business Process repository entry, and specifies whether or not compensation can be performed after a process has completed normally.

When an activity is defined as “Independent” or “Reference”, users can link the activity to a Business Process on the Type / Status tab of the activity’s repository entry. If the Business Process does not exist, it will be added to the repository once the activity’s entry is saved.

Connecting Objects

Flow Objects are connected on the diagram using Sequence Flows, Message Flows and Associations.

Sequence Flow A Sequence Flow is a solid graphical line that is used to show the order that the activities will be performed in a Process. Each flow has only one source and one target.

Conditional Flow A Sequence Flow can have condition expressions that are evaluated at runtime to determine whether or not the flow will be used. If the conditional flow is outgoing from an activity, then the sequence flow will have a mini-diamond at the beginning of the line.

Default Flow For Data-Based Exclusive Decisions or Inclusive Decisions, one type of flow is the Default condition flow. This flow will be used only if all of the other outgoing condition flows are not true at runtime. The Default Sequence Flow will have a diagonal slash at the beginning of the line.

Exception Flow An Exception Flow occurs outside the Normal Flow of the Process and it is based upon an Intermediate Event that occurs during the performance of the Process. The Exception Flow begins at the Error Intermediate or End Event attached to the boundary of the activity, signifying an interruption in the activity.

Compensation Flow A Compensation Flow defines the set of activities that are performed during the rollback of a transaction to compensate for the activities that were performed during the normal flow of the process. Compensation can also be called from a Compensate End or Intermediate Event

Message Flow A Message Flow is drawn using a dashed line with a small circle at the beginning of the line. This line type is used to show the flow of messages between two entities that are prepared to send and receive them. In BPMN, two separate Pools on the diagram will represent the two entities

Association An Association Flow is used to connect one or more Data Objects (Artifacts) to an activity, and to show that the Data Object is either an input or output to the Activity.

Pools and Lanes

BPMN models utilize two important conceptual modeling devices to distinguish process flow characteristics, pools and lanes.

Pool

A Pool (defined as a System Boundary entry type in the repository) represents a participant in a process, and it also acts as a "swimlane" and graphical container for portioning a set of activities from other pools.

When a Pool is added to a diagram, only the name of the Pool is displayed. The swimlane labels will be displayed once a BPMN item is moved within the pool boundary. If the orientation of the Pool symbol is vertical, the name of the Pool and lanes are displayed at the top of the Pool, otherwise the labels are drawn on the left-hand side of the pool. Pools whose width is greater than their height are vertical. To change the orientation of the Pool, highlight the pool symbol, select one of the “handles” (the green boxes surrounding the Pool symbol) a drag the handle to a new location on the diagram, expanding the Pool symbol. You can also right mouse click on the Pool symbol and choose the Stylize option to modify the size and orientation of the Pool.

The Attributes field on the Define Item tab in the repository will display the items contained within the Pool and its lanes.

Lane

A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either horizontally or vertically. Lanes are used to categorize the activities in the Pool. If there is only one lane, then the lane shares the name of the Pool. While BPMN does not specify the usage of the Lanes, they are often used to identify internal roles (Manager, Associate), systems (enterprise application) and internal departments (shipping, finance sales).

Up to 10 Lane names can be assigned for each Pool.

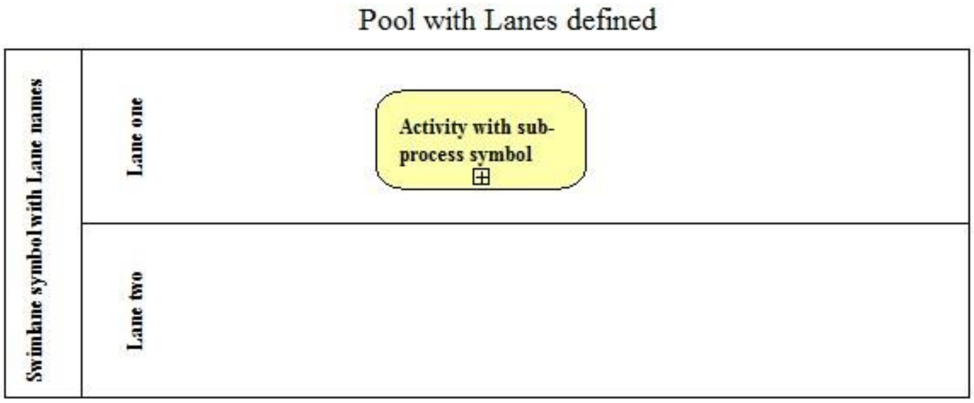

Pool with Lanes defined
Swimlane symbol with Lane names
Lane one
Activity with sub-process symbol
Lane two

Figure 4-46 Pool with Lanes Defined

Note

The lanes are only displayed in the pool when a BPMN symbol is added within the boundary of the pool. Drag the symbol to position it within the appropriate lane.

Changing Lane Sizes

To change the size of a lane, click on the pool and hover the cursor over the lane divider. The cursor will change to ⇹ or ⇳ depending on the pool orientation. Depress the left mouse button and drag the lane boundary to its new position.

Artifacts

Artifacts are used to provide additional information about the Process, and are represented by a Data Object, a Group, and an Annotation.

Data Object

A Data Object is represented by a square symbol with the top right corner of the symbol turned down. Artifacts are used to provide information about the process or elements defined or produced within the process, but do not affect the process. An Association Flow connects data Objects to other BPMN symbols.

Group

A Group is a visual representation around a number of activities. Groups are represented in the Visible Analyst through the use of symbol coloring, or the user can draw a flow around the diagram objects to represent a group.

Annotation

An Annotation (referred to and implemented as a Note symbol in the Visible Analyst) is a Text Annotation, used as a mechanism for the modeler to provide additional information for the reader of a BPMN diagram. A Note object (and a Note Link) in the Visible Analyst does not maintain a repository entry; all information is written on the diagram as the Note label. Notes are connected to a specific object on the diagram with a Note Link line, but neither the Note nor the Note Link line affect the flow of the Process.

Business Process Diagram Attributes

A number of attributes are associated with BPMN Diagrams. Some of these attributes include, Diagram Id, Name, Version, Author, Language (for code generation), Creation Date, Modification Date, etc. Use the

Boilerplate and Boilerplate Keywords feature in the Visible Analyst to display these attributes on a BPMN diagram. See Boilerplates and Boilerplate Keywords.

Note

Boilerplates and Boilerplate keywords are not available in the Educational versions of Visible Analyst.

Analysis for BPMN Diagrams

Visible rules check for two classes of errors:

- Syntax errors. These are errors that would make your BPM diagram impossible for Visible Analyst to understand.
- Connection errors. These indicate symbols are improperly associated with other symbols. Different rules apply depending on the type of symbols connected.

Analysis for BPMN Diagrams

ERROR There are ‘X’ unnamed Activities.

This error is the result of not labeling activities on your BPM diagram(s). To resolve the error, find and label them using the Change Item function.

ERROR There are ‘X’ unnamed Data Object(s).

This error is the result of not labeling data objects on your BPM diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘X’ unnamed Event(s).

This warning message is the result of not labeling events on your BPM diagram(s). In order for events to have an entry in the repository, they must be named. If you don't want the names to appear on a diagram use the «unnamed» keyword.

To resolve the error, find and label them.

ERROR There are ‘X’ unnamed Gateway(s).

This error message is the result of not labeling gateways on your BPM diagram(s). In order for gateways to have an entry in the repository, they must be named. If you don't want the names to appear on a diagram use the «unnamed» keyword.

To resolve the error, find and label them.

ERROR There are ‘X’ unnamed Message Flow(s).

This error is the result of not labeling message flows on your BPM diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘X’ unnamed Pool(s).

This error is the result of not labeling pools on your BPM diagram. To resolve the error, find and label them using the Change Item function.

ERROR There are ‘X’ unnamed Message Flow(s).

This error message is the result of not labeling sequence flows on your BPM diagram(s). In order for sequence flows to have an entry in the repository, they must be named. If you don't want the names to appear on a diagram use the «unnamed» keyword.

To resolve the error, find and label them.

ERROR Sequence Flow labeled 'X' is dangling.

This error is the result of not connecting at least one end of the indicated sequence flow to appropriate flow objects on a BPM diagram. To resolve the error, connect the sequence flow to an activity, event or gateway using the Connect function.

WARNING Sequence Flow labeled ‘X' is only connected to one flow object.

This error is the result of not connecting both ends of the indicated sequence flow to appropriate flow objects on a BPM diagram. This warning can be ignored if the flow is an input flow to an activity and it is a starting "event" or it is an output flow from an activity and it is and ending "event". To resolve the warning, connect both ends of the sequence flow to an activity, event or gateway using the Connect function.

ERROR Activity labeled 'X' is dangling.

This error is the result of not connecting the indicated activity to another activity, event, or gateway on a BPM diagram using a sequence flow. To resolve the error, draw a sequence flow between the activity and another activity, event, or gateway using the Connect function.

ERROR Gateway labeled 'X' is dangling.

This error is the result of not connecting the indicated gateway to another flow object on the BPM diagram. To resolve the error, draw a sequence flow from the gateway to an activity or event using the Connect function.

ERROR Object labeled 'X' is dangling.

This error is the result of not connecting the indicated object to another flow object on the BPM diagram. To resolve the error, draw a sequence flow from the object to another diagram object using the Connect function.

ERROR Event labeled 'X' is dangling.

This error is the result of not connecting the indicated event to an activity or gateway on a BPM diagram using a sequence flow. To resolve the error, draw a sequence flow from the event to an appropriate flow object using the Connect function.

ERROR Data Object labeled 'X' is dangling.

This error is the result of not connecting the indicated data object to an activity on a BPM diagram using an association line. To resolve the error, draw a line from the data object to any activity using the Connect function.

ERROR Flow Object labeled 'X' does not have both an input and output sequence flow.

Activities, gateways, and intermediate events must have both an input sequence flow and an output sequence flow. To resolve the error, connect the end point of one sequence flow and the start point of another sequence flow to the indicated flow object using the Connect function.

ERROR Timer event labeled 'X' does not have a time date or time cycle set.

An event with a trigger type of 'Timer' must specify either a date and time when the event will fire or a time cycle such as 'Every Monday morning at 5:00 AM. To resolve the error, double-click on the indicated event to view its repository entry, then click on the **Trigger** tab and enter the correct information in the **Trigger Time** box.

ERROR Rule event labeled 'X' does not have a rule set.

An event with a trigger type of 'Rule' must specify a rule expression to be used for triggering the event. An example of a rule is "“S&P 500 changes by more than 10% since opening”. To resolve the error, double-click on the indicated event to view its repository entry, then click on the **Trigger** tab and enter the correct information in the **Rule** box.

ERROR Message event labeled 'X' does not have a related message set.

An event with a trigger type of 'Message' must specify a message that arrives from a participant and triggers the start of the process. To resolve the error, double-click on the indicated event to view its repository entry, then click on the **Trigger** tab and enter the correct information in the **Message** box. You can either enter a new message or use the browse button to the right of the control to search for an existing message object.

ERROR Error result event labeled 'X' does not have an error code set.

An 'End Event' with a result type of 'Error' must specify an error code that will be generated when the event completes. To resolve the error, double-click on the indicated event to view its repository entry, then click on the **Trigger** tab and enter the correct information in the **Error Code** box.

ERROR Independent Sub-Process labeled 'X' does not reference a business process.

An independent sub-process must reference a business process to complete its definition. To resolve the error, double-click on the indicated sub-process to view its repository entry, then click on the **Type / Status** tab and enter the name of a business process in the **Bus. Proc.** box. You can also use the browse button to the right of the box to search for an existing business process.

ERROR Reference task labeled 'X' does not specify a related task.

A reference task must reference another task to complete its definition. To resolve the error, double-click on the indicated task to view its repository entry, then click on the **Type / Status** tab and enter the name of a task in the **Task** box. You can also use the browse button to the right of the box to search for an existing task.

ERROR Reference Sub-Process labeled '%s' does not specify a sub-process.

A reference sub-process must reference another sub-process to complete its definition. To resolve the error, double-click on the indicated process to view its repository entry, then click on the **Type / Status** tab and enter the name of a task in the **Task** box. You can also use the browse button to the right of the box to search for an existing task.

ERROR Task labeled 'X' does not specify an input message.

When an activity type is set to Task and the task type is Service, Receive, Send, or User, an input message is required to indicate that a Message will be sent at the start of the Task. To resolve the error, double-click on the indicated task to view its repository entry, then click on the **Type / Status** tab and enter the name of a message object in the **In Message** box. You can also use the browse button to the right of the box to search for an existing message.

ERROR Task labeled 'X' does not specify an output message.

When an activity type is set to Task and the task type is Service or User, an output message is required the arrival of which indicates the completion of the task. To resolve the error, double-click on the indicated task to view its repository entry, then click on the **Type / Status** tab and enter the name of a message object in the **Out Message** box. You can also use the browse button to the right of the box to search for an existing message.

ERROR Activity labeled 'X' has components defined but no data objects are connected.

In order for an activity to have components (properties), these components must be feed from a data object. This error indicates that there are no data objects connected to the activity by a reference line. To resolve the error, draw a reference line from one or more data objects that contain the properties of the activity in its **Attributes** field to the indicated activity using the Connect function.

ERROR Activity labeled 'X' has components defined that are not contained in any source data objects.

In order for an activity to have components (attributes), these components must be feed from a data object. This error indicates that while there are data objects connected, none of the attributes of the activity are contained in any of the connected data objects. To resolve the error, double-click on one of the connected data objects to view its repository entry, then click on the **Description** tab and enter the attributes in the **Attributes** box.

Chapter 5

The Visible Repository

AN ORGANIZED PRESENTATION OF PROJECT DATA

The Visible repository adds database management and data definition capabilities to Visible Analyst. The Repository functions interactively with diagramming and rules to automatically create an organized database of project information for each project as it is developed. A project repository is automatically updated and maintained during all of your work sessions, and it can be accessed at any time to manually add, delete, edit or review entries. Reports showing repository entry information can also be printed in a variety of formats.

The repository is accessible by selecting Define from the Repository menu. The following highlight some of the functions of the repository:

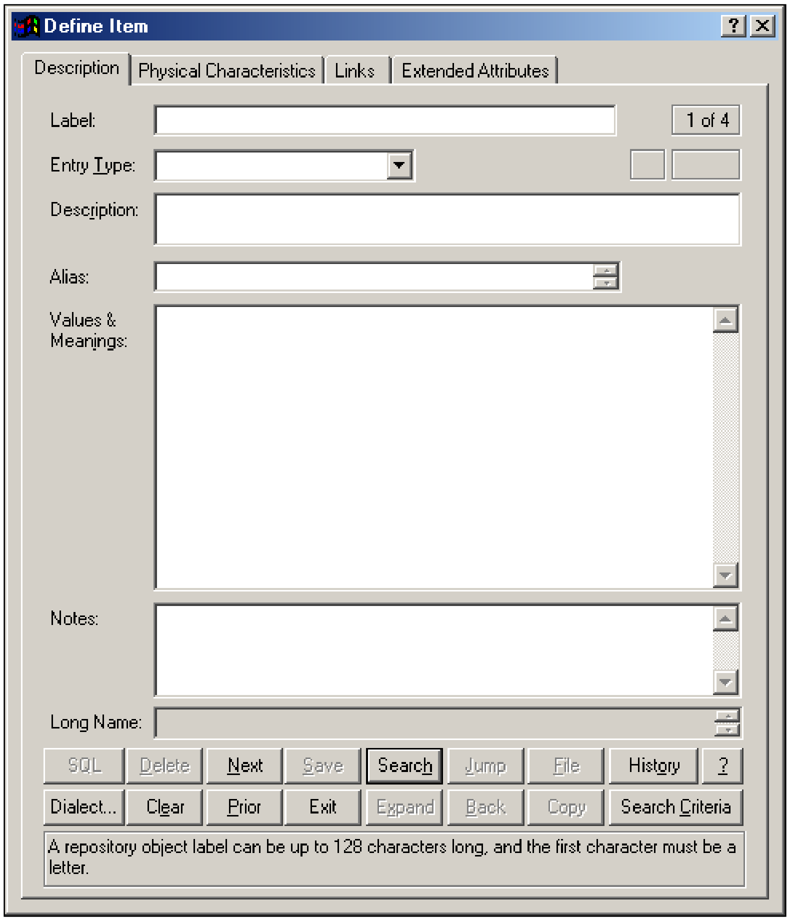
- All objects drawn on diagrams of methodology projects, or added to the planning hierarchy, are automatically entered into the repository and are addressable using the label field at the top of the dialog box.
- You can manually make additional entries into the repository by assigning a label and entering information into a blank repository dialog box.
- You can page through the repository dialog boxes to display all database entries for the selected project.
- You can edit most fields in the repository dialog boxes.
- You can define new object types and fields for objects.
- You can define a view specification for an entity.
- You can set up search criteria and search through a selected portion of the repository.
- You can enter part of a label and perform a “wildcard select” to locate the first matching label in the repository, or a “wildcard find” to display all repository entries that have labels that begin with the characters you entered.
- You can write some repository fields to a specified DOS file, or read data from a DOS file into those fields.
- You can read and write repository information to and from the Windows Clipboard.
- You can set up report criteria and print only the selected group of repository records.
- You can create hyperlinks to external items such as Word documents, World Wide Web pages, etc.

REPOSITORY BASICS

Accessing the Repository

The repository can be accessed in four ways:

- You can access the Repository menu by selecting the Define function. If no diagram object is active, Visible Analyst presents a blank repository dialog box, as shown in Figure 5-1. You have several options at the blank dialog box; these are explained in the pages that follow. If a diagram object is selected, you are presented with the repository entry for that object.
- You can double-click on a named diagram object to display its repository entry.
- You can right-click on a diagram object and select Define from the pop-up menu to display the repository entry for the object.
- You can double-click on an object in the object browser to display the repository entry for the object.



A repository object label can be up to 128 characters long, and the first character must be a letter.

Figure 5-1 Blank Repository Dialog Box

Hyperlinks

In all the descriptive-type fields in the repository, such as Values & Meanings, Process Description, Notes, Scenario, etc. you can add a link to a document or other item outside the Visible Repository using a hyperlink. To insert a hyperlink, simply enter a URL or file name with its full path specification (both UNC paths and drive letters can be used). Examples are www.visible.com and \\VS\Develop\planning.doc. If the hyperlink is recognized as having a program association in the registry, it is displayed in blue and underlined. When the cursor is moved over the link, it will change to a pointer; clinking on the link loads the appropriate document.

Understanding the Repository Dialog Box

The repository dialog box for each entry provides you with information and functional selections, as shown in Figure 5-1. Detailed descriptions of these repository dialog box items are provided in the following pages. All entries contain multiple pages of information, as described below.

The Repository Dialog Box Buttons

The repository dialog box buttons (see Figure 5-2) are always displayed at the bottom of the box, although not all buttons are enabled at all times. You can use them to control activities within the Visible Analyst repository for your project. As always in Windows, you can use the keyboard shortcut by holding down the ALT key and pressing the underlined letter to execute the button function. The button functions are shown below.

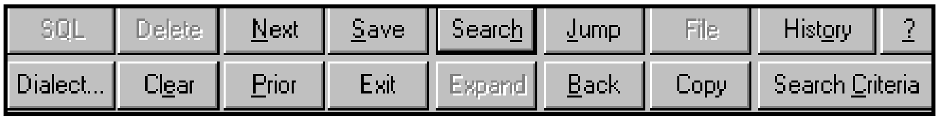


Figure 5-2 Repository Dialog Box Control Buttons

SQL — For view objects, this activates the Generated SQL for View window. This window displays the SQL that will be generated for the current view when you click the SQL button on the Repository Define dialog box. The button is only active when the entry type is a view.

Dialect — This activates the RDBMS SQL Dialect dialog box. From there, you can change the current SQL dialect.

Delete — This deletes the currently displayed entry from the database. Whenever you delete a repository entry, all associated entries are automatically updated. For example, if you delete a data flow, Visible Analyst deletes any references made by other entries to the deleted flow. Note: Any entry may be deleted from the repository *provided that it has no diagram locations.* (An entry has locations when it appears on diagrams or when it is referenced in the composition field of another entry; only diagram locations are relevant here.) If an entry has diagram locations, these must be eliminated before the entry can be deleted.

Clear — This clears the display of an entry and displays a blank dialog box or eliminates all edits you performed that have not yet been saved. If you edited the entry prior to clicking Clear, you are prompted to confirm clearing your changes without saving them. Clearing the dialog box allows you to Search for an existing entry or to add a new entry.

Next — This displays the next sequential repository entry that meets the repository search criteria you set. The ordering of repository entries is alphanumeric by label.

Prior — This displays the previous sequential repository entry that meets the repository search criteria you set.

Save — This saves all changes to an entry; the dialog box entry remains displayed. The date created and date last altered information is updated as applicable.

Exit — This or the ESC key exits the repository and clears the dialog box from the screen. If edits were made, you are prompted to save the changes or ignore them.

Search — This initiates a search of the repository for another item. It is explained in detail below.

Expand
Contract — This allows you to expand the displayed size of certain fields. Fields normally displaying four lines are expanded to display 15 lines so that you can work with a larger section of the field at one time. When the field is expanded, the button reads Contract. You can move around within the edit box in the usual Windows manner. The Alias, Composition, Notes, Values & Meanings, Process Description, Module Description, Related to, and Location fields may be expanded and contracted like this.

Jump — This allows you to immediately jump to another repository entry that is referred to in the current one. When the cursor is positioned on the line containing the name of another repository entry, clicking Jump allows you to quickly jump to the repository entry for the first item listed on that line. You can also execute the jump by double-clicking on that line. On lines that contain the names of more than one repository item, moving to an item beyond the first requires that you highlight that item before clicking Jump. If edits were made, you are prompted to save the changes or ignore them before the jump is executed. From the destination entry, you may continue to jump wherever you wish; you are not obligated to return to your original location in the repository. To load a diagram referenced in the location field, highlight the diagram and click Jump.

Back — This button provides a means of moving to the previous repository entry.

You can then continue to move backward displaying previous repository entries.

File
This allows you to insert text from a DOS file at the cursor position or to copy highlighted text to a DOS file. It is explained in detail below.

Copy
This button provides a means of copying the current object.

History
This provides a means of moving back to a previously displayed repository object. A list is kept of every object definition that has been displayed. If you click this button, the history dialog box appears and you can move between entries by double-clicking an entry or by highlighting the entry and pressing the ENTER key. The maximum is 500 objects.

Help (?)
This displays context-sensitive help about the repository. You can also press F1 to activate the help system.

Search Criteria
This allows you to specify how the repository is to be searched. It is explained in detail below.

Other buttons that may be displayed on the Define dialog box are:

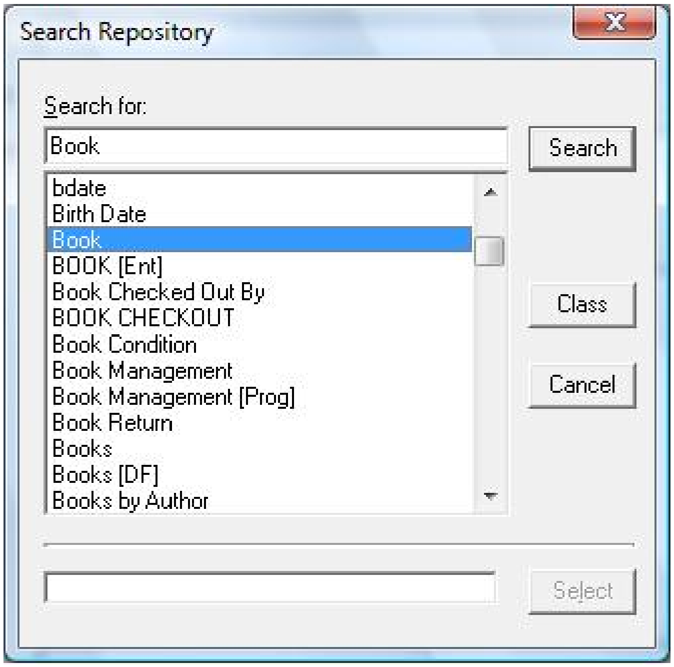
Primary Key
If the current object being examined is an entity type, the primary key button is displayed to the left of the Composition/Attributes field. The procedure for adding keys is explained in Defining a Primary Key later in this chapter.

Attributes Details
This button, displayed to the left of the Composition/Attributes field, provides a means of populating the composition of a repository entry with components and physical information. If the current object is a class, it also provides a means of adding attributes and methods.

Add
When the Entry Type is a Class, or when the Classic User Interface is turned off, an Add button is displayed beneath the Attributes Details button. You can use this button to quickly add details. When you begin typing in the field next to the Add button, the button is enabled and the repository is checked for an object that matches the name you typed. If a match is found, the type, length, and null field are set. Click the Add button to add the attributes to the Attributes field. If the current entry is a class, the object is added as a local element if it does not already exist. Otherwise, it is added as a global object. A local object is a blue icon; a global object is yellow.

Repository Searches

You can search a project repository in several ways. The easiest way is to click the Search button when the repository dialog box is displayed. You can then enter the name (or part of the name) of the desired item into the Search box (see Figure 5-3). Visible Analyst incrementally searches as you type. When Visible Analyst finds your item, click Select to display the repository entry for that item.



Search for: Book

bdate
Birth Date
Book
BOOK [Ent]
Book Checked Out By
BOOK CHECKOUT
Book Condition
Book Management
Book Management [Prog]
Book Return
Books
Books [DF]
Books by Author

Figure 5-3 Repository Search Dialog Box

Note

- If your search key (the full or partial name) is very broad, it can take a long time to fill the search list box with all of the matching repository entry names. In this case, Visible Analyst does as much as it can in ten seconds and then returns control to you. Periodically, Visible Analyst continues to add matching items to the list box when it senses idle time; you see the hourglass cursor when this takes place. If you want to select an item near the end of the alphabet in such a full list box, there might be a delay before the item displays.

Another way to search is to enter a full or partial entry name in the dialog box label field and click the Search button or press ENTER. The search box displays with an incremental search already completed on the full or partial name you entered.

Yet another way to search is to enter a full or partial item name and affix an “*” to the end of the name. Click the Search button or press ENTER to search for the entered text and accept and display the first item found that matches the name. This is called a “wildcard select.”

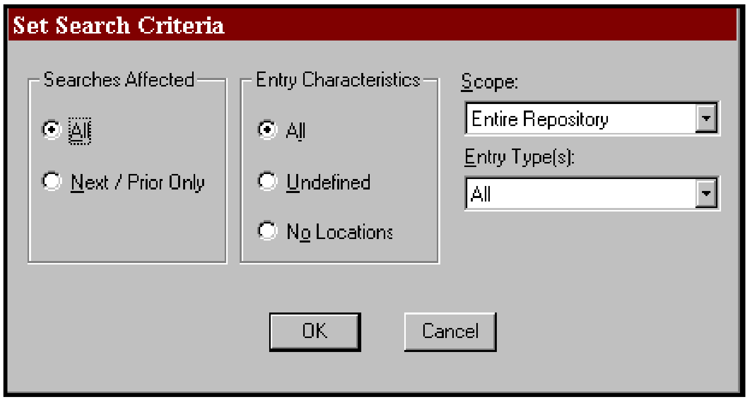
If the search list box contains objects that belong to classes (either modules or local data elements), the class of the item is displayed in parenthesis following the item. If you want to change the sort order so that all members of a class are grouped together, click the Class button. The list is resorted in class order. To revert to member order, click the Member button. Although less obvious, both the Next and Prior buttons execute searches. They don’t search on an entered name, but on a name alphanumerically greater or lesser, respectively, than the currently displayed item.

There are three more specialized uses of the Search function. These are mentioned in the sections entitled Composition, Creating Clusters, and Domains, below.

All of the above searches are subject to the search criteria you enter.

Repository Search Criteria

You can set up criteria to be applied when searching for repository entries. Upon entering the repository, you can click Search Criteria and the window shown in Figure 5-4 is displayed. Three sets of criteria may be selected, all of which operate concurrently to determine the subset of the repository to be accessed. The criteria currently in effect are displayed on the title line of the dialog box. Note that until you exit Visible Analyst and return to Windows, the search criteria are maintained and are present the next time the repository is accessed.



Set Search Criteria

Searches Affected: ● All ○ Next / Prior Only

Entry Characteristics: ● All ○ Undefined ○ No Locations

Scope: Entire Repository

Entry Type(s): All

Figure 5-4 Search Criteria Dialog Box

The Searches Affected category determines which searches are to be implemented. If you select All, then all searches are limited by the criteria, including wildcard selects, manually entered complete labels, and next and prior searches. If you select Next/Prior Only, then the search criteria applies only to next and prior searches.

The Entry Characteristics category of search criteria allows the repository to present All entries (effectively disabling this set of criteria), Undefined entries (those for which you have not yet entered any descriptive information), or entries that have No Location references.

The Scope category allows you to limit searches by diagram type. Searches can be conducted on the repository entries specific for data flow, functional decomposition, structure chart, entity relationship, class, or state transition entries, or on all entries in the repository regardless of diagram type. The list of entry types changes when this is done, displaying the entry types that correspond to the selected diagram type.

The Entry Types category defines the type of the entries that are presented during searches. You can specify All entries, a compound type like All Entity Types, or you can specify a specific entry type to be searched for.

Repository Search Criteria Default
When working in the Repository, the search criteria defaults to the current diagram type.

Copying Data To and From DOS Files

This function allows you to copy information to or from the Values & Meanings, Composition, Notes, Process Description, Macro Description, Modules Contained or Module Description fields. You can copy the fields to a DOS ASCII file; you can then read the information from the DOS file into another field. This is an easy way to copy information into multiple entries without retyping it. To copy data in a repository field to a file, highlight the information you want to copy and click the File button. The dialog box asks you the name of the file and where you want it stored.

To copy data from a file to a repository field, place the cursor where you want the data inserted and click the File button. The dialog box asks you the filename and where it is stored. If you perform the insert from a file immediately after copying a repository field to DOS file, Visible Analyst defaults to that DOS file name.

Copying Data To and From the Windows Clipboard

Visible Analyst allows you to copy repository information to or from the Windows Clipboard. To copy to the Clipboard, highlight text in any edit control in the repository dialog boxes and press the keyboard shortcut for Copy displayed in the Visible Analyst Edit menu.

To enter data from the Clipboard, position the cursor where you want the text to go and press the keyboard shortcut for Paste displayed in the Visible Analyst Edit menu.

Repository Object Menu

In addition to the buttons defined on the Repository dialog box, you can use the right mouse button to activate the repository object menu for each of the repository fields. When you click the right mouse button over a field, the menu gives you a list of options that can be performed on the current field. The SHIFT+F10 keyboard shortcut can be used to activate the repository object menu.

Enterprise Links

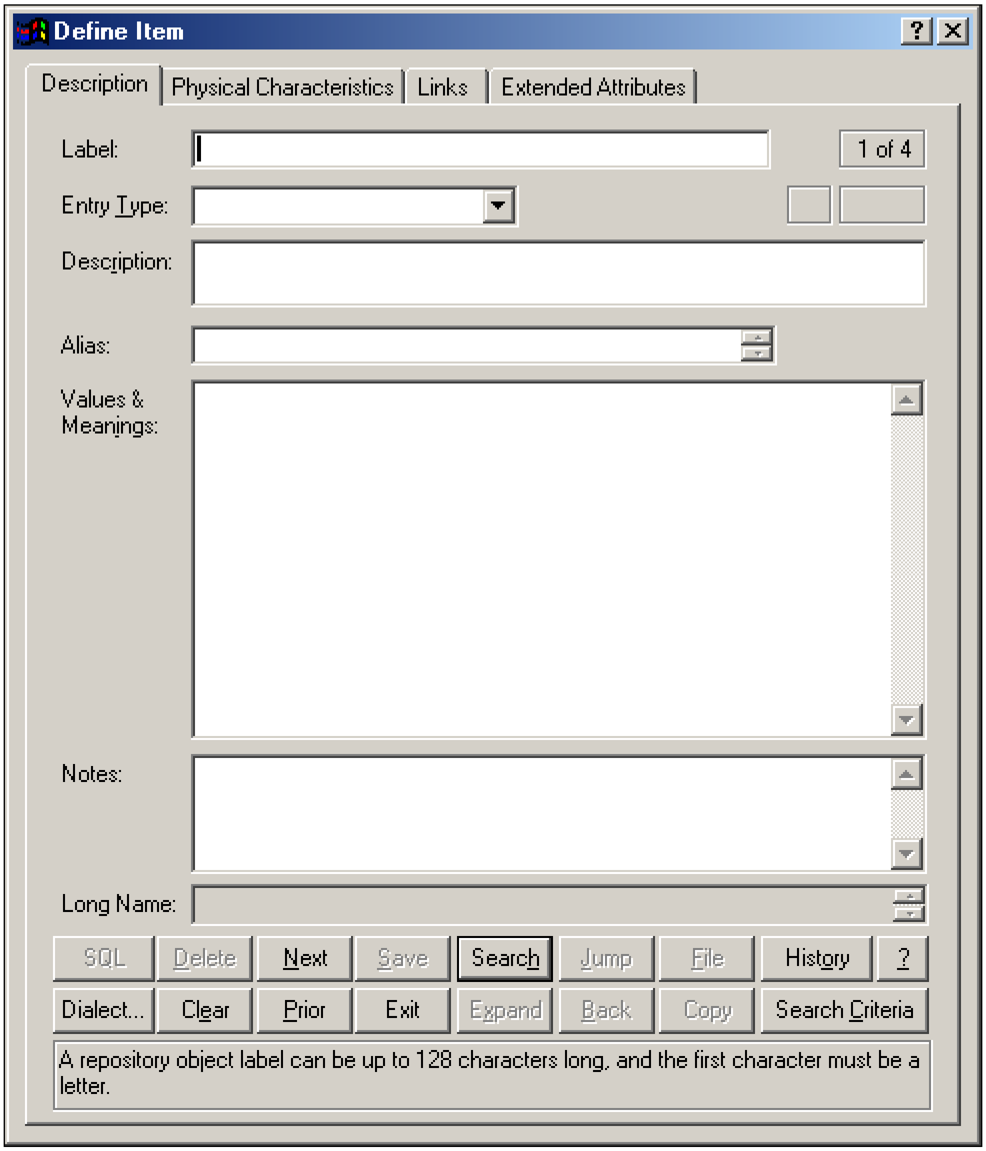
If the current object being examined has enterprise links to other projects, a marker is displayed to the right of the object name. If you double-click on the marker, a list of enterprise links is displayed. In addition, if the current project is a satellite project, the name of the enterprise project is displayed to the right of the Entry Type field. Refer to the Enterprise Modeling chapter for more information.

Read-Only Indicator

If you do not have rights to make changes to the current object, RO (Read Only) is displayed to the right of the Entry Type field. Read-only access is established by adding an object to a division and revoking the Modify Items right. Refer to Enterprise Modeling for more information.

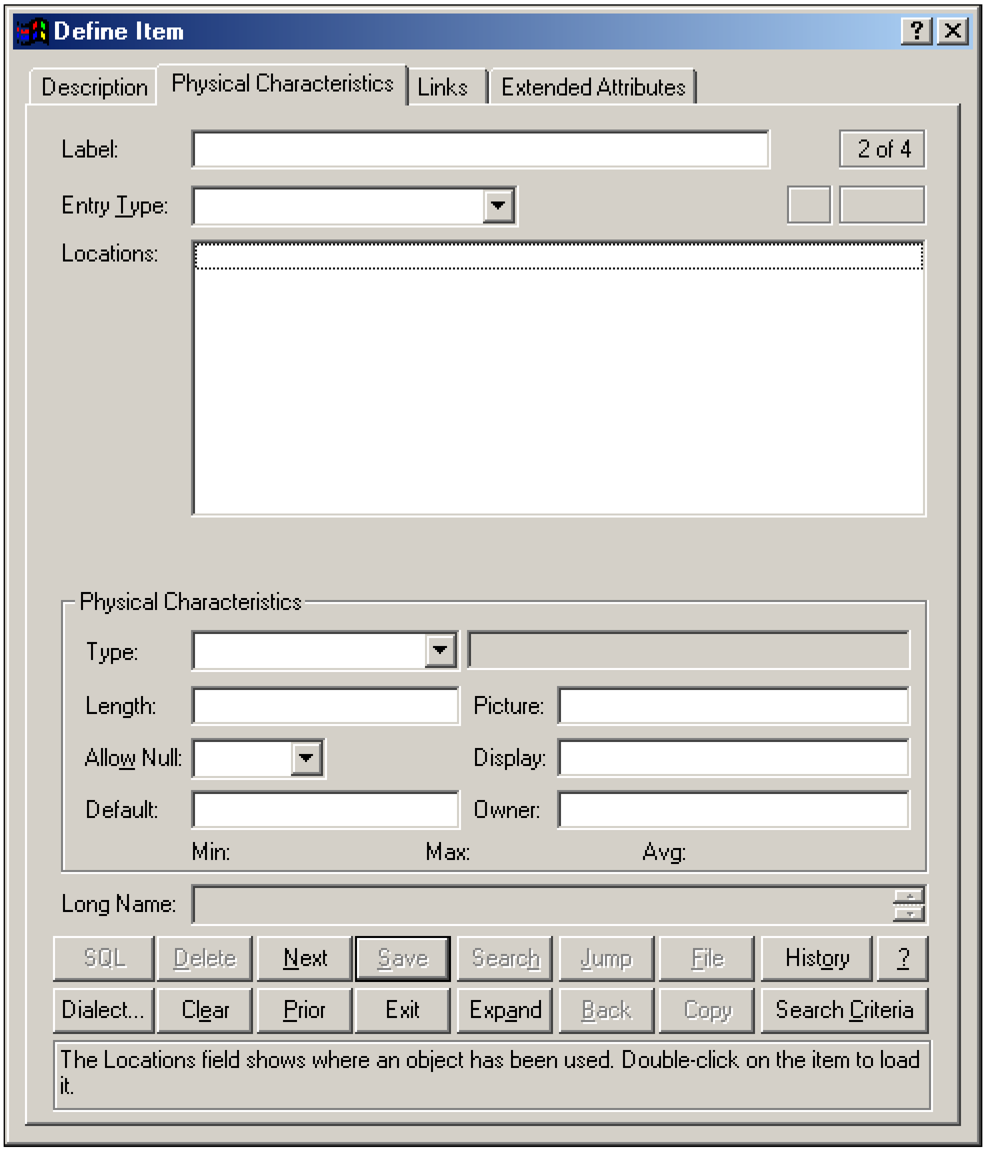
Understanding Repository Fields

The names of fields that identify and describe each repository entry are listed on the pages of the Define dialog box. The fields include a combination of data that is entered automatically when a diagram is saved and data that is entered manually during repository editing sessions.



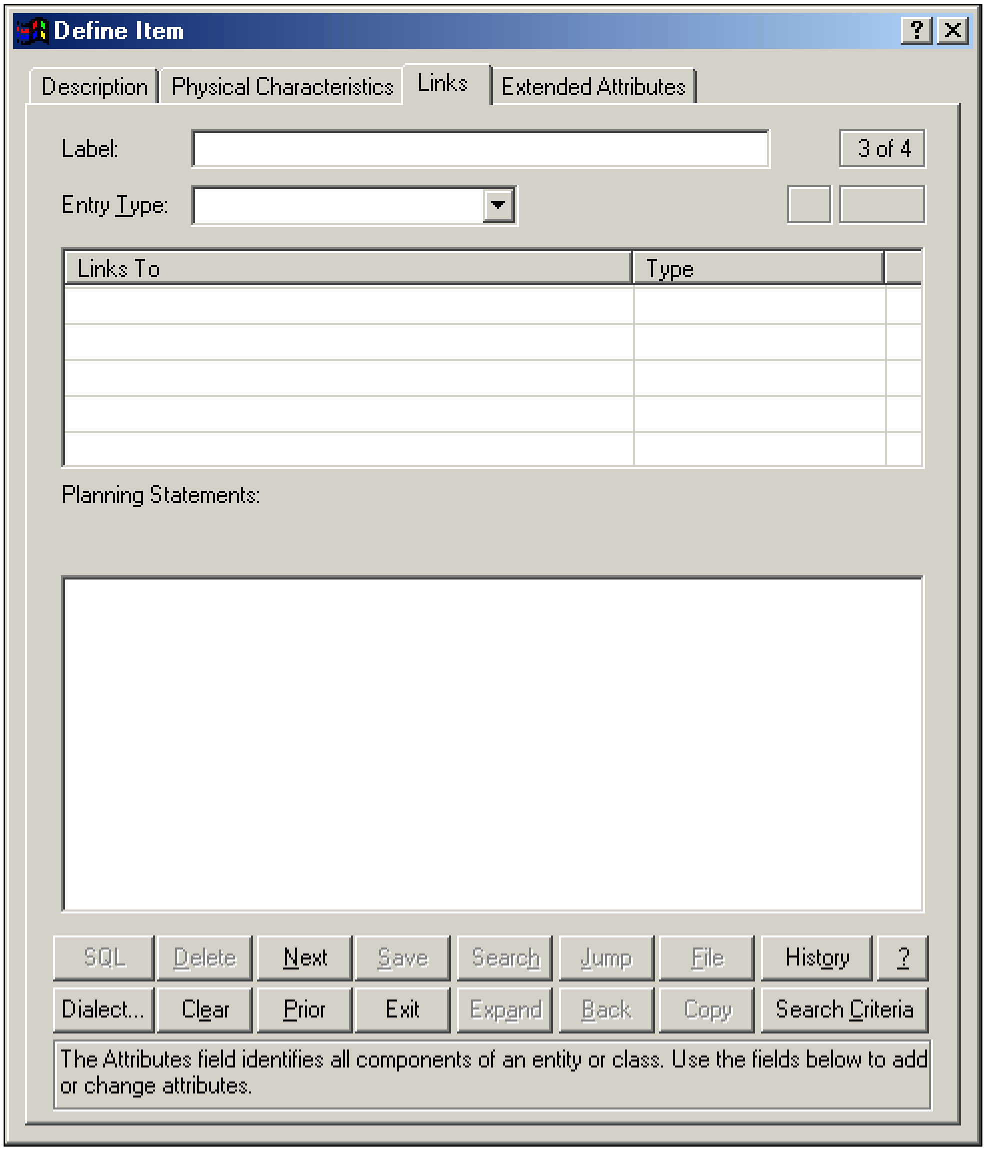
A repository object label can be up to 128 characters long, and the first character must be a letter.

Figure 5-5 Repository Dialog Box, Page 1, Description Tab



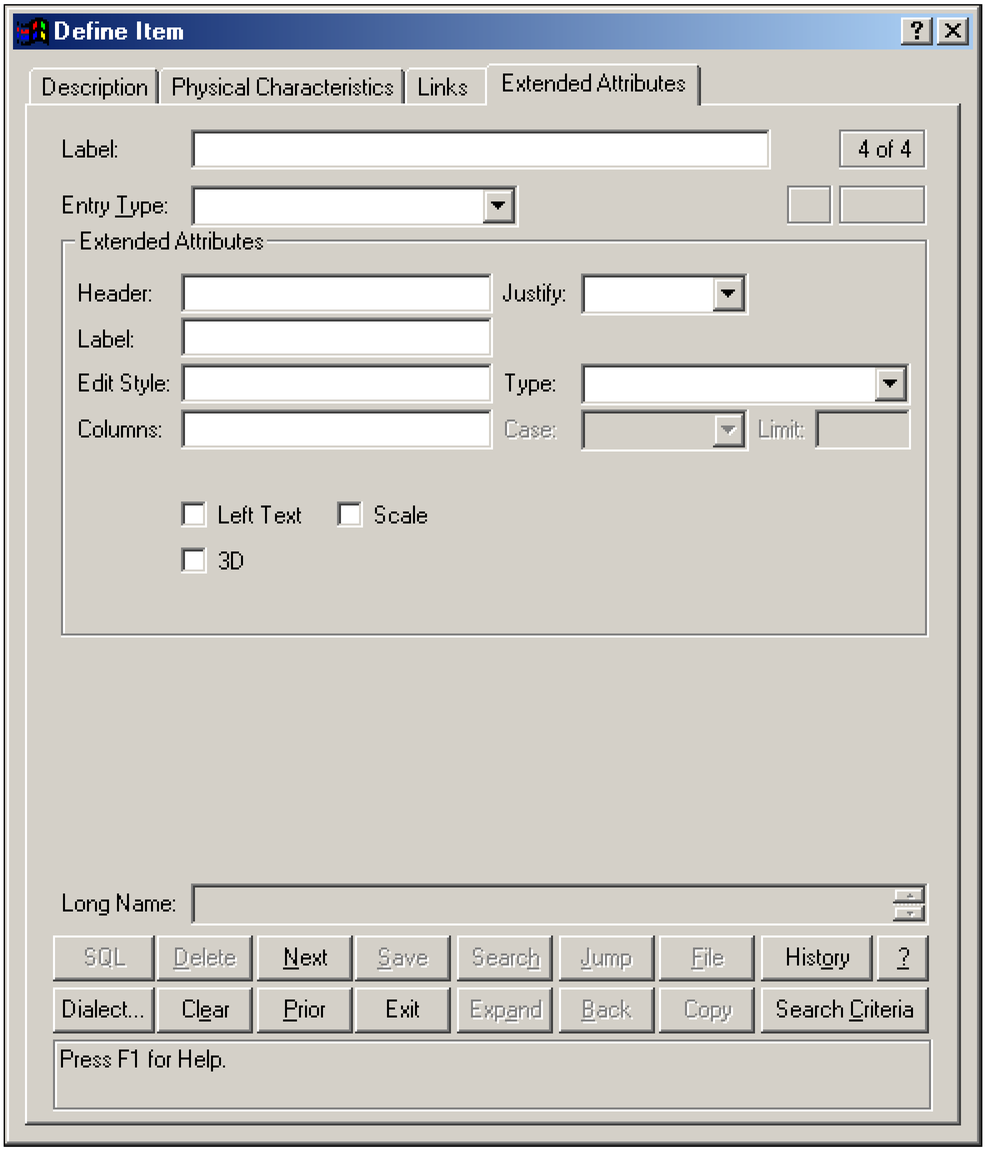
The Locations field shows where an object has been used. Double-click on the item to load it.

Figure 5-6 Repository Dialog Box, Page 2, Physical Characteristics Tab



The Attributes field identifies all components of an entity or class. Use the fields below to add or change attributes.

Figure 5-7 Repository Dialog Box, Page 3, Links Tab



Press F1 for Help.

Figure 5-8 Repository Dialog Box, Page 4, Extended Attributes Tab

Different repository entry types have different combinations of dialog box tabs and fields in which data and descriptive information can be entered and stored. For example, pages 2 and 4 of the basic repository form provide location and relationship information and specifications for PowerBuilder/VISION extended attributes. These two pages are similar for most entry types. For some entry types, additional pages can be displayed:

- When the entry type is an entity, the next five pages contain keys, foreign keys, triggers, check constraints, and physical information.
- For views, the next five pages provide table, column, join, clauses, and option information.
- When the entry type is a relationship, there are additional pages that contain foreign key and cardinality information.
- When the entry type is a tablespace, an additional page contains property information.
- When the entry type is a BPM activity or business process, additional pages contain type / status and assignment information.
- When the entry type is a data object, an additional page contains state information.
- When the entry type is an event, the additional page contains trigger information.
- When the entry type is a gateway, the additional page contains condition information.
- When the entry type is a message or a system boundary (BPM), the additional page contains participant information.

Table 5-1 lists the repository information fields for all entry types except view objects. View objects are discussed in detail later in this chapter. The main Business Process Modeling fields are listed in table 5-1, while the sub-fields within the field are explained in the main field explanation.

Table 5-1
Possible Repository Information Fields

Ad Hoc	From (Role)	Produced At Completion
Ad Hoc Properties	Function Description	Referential Integrity*
Alias	Implementation	Required For Start
Assignments	Incoming	Related To
Associator Element	In Message	Relations
Attached Entities/Classes	Instantiate	Relationship
Attributes	Label	Rule
Cardinality*	Links To	Scenario
Check Constraints*	Locations	Start Quantity
Class Characteristics	Long Name	State
Column in Key	Looping	Statement Type
Composition	Macro Description	Status
Concurrency	Methods	Suppress Join Failyre
Data Mod(s)	Message	Tablespace
Description	Module Description	To (Property)
Discriminator Values & Meanings	Modules Contained	To (Role)
Entry Type	Notes	Trigger
Enable Instance Compensation	Outgoing	Trigger Time
Extended Attributes	Particiapant (Role)	Type
Error Code	Physical Characteristics*	Values & Meanings
Foreign Keys*	Priority	Web Service Properties
Friends	Process #	
From (Expression)	Process Description	

The following paragraphs provide a detailed description of these fields, except for those marked by an asterisk (*). Asterisked fields are described later in this chapter.

Label

This is the key field by which every repository entry is accessed. It can be up to 128 characters long, and the first character must be a letter. An entry that refers to an object on a diagram maintains the identical label that appears on the diagram. Entries that are manually entered into the repository can be assigned any label that does not already appear in the repository. The label for any repository item (including entities and relationships) can be changed *so long as the item does not appear on any diagram and does not appear in the composition field of another item.*

If you want an entry to begin with a numeric value, edit the VAW.INI file and add this line to the file: Allow Numeric Prefix=Yes and then restart the Visible Analyst to enable the change.

You can access any repository item using its label as described under Repository Searches and from a diagram. If the entry searched for is not in the repository, Visible Analyst assumes that you wish to add a new entry and your cursor is moved into the Entry Type field.

Entry Type
The content of this field indicates the type of repository entry. Normally, this field cannot be changed if the entry being displayed was added to the repository automatically by Visible Analyst (in order to guarantee consistency between the project diagrams and the database). Entry types that are automatically added to the repository include all data flow types except miscellaneous and all other rules-recognized entry types. All entry types except alias, relationship and information cluster can also be manually keyed into the repository.

Also, if the entry type is defined as a data element, data structure, generic couple, control couple, or data couple, it can be changed until you enter additional descriptive information in the Composition or Values & Meanings fields or until you enter physical information for a data element. This lets you change either a data element to a data structure and vice versa, or any type of couple to any other type of couple or to an interface table row before the entry is saved.

Table 5-2
Repository Entry Types by Diagram Type

Data Flow Types	**Use Case Types**
Data Structure	Actor
Data Flow	System Boundary
Process	Use Case
Source/Sink	**Structure Chart Types**
External Entity	Module
Miscellaneous	Library Module
Data Element	Macro
Domain	Library Macro
Data Store	Data-only Module
File	Control Couple
Entity Relationship Types	Data Couple
Relationship	Generic Couple
Data Element	Information Cluster
Cluster	Data ITR
Domain	Control ITR
Entity	Generic ITR
Associative Entity	Program
Attributive Entity	**Activity Types**

Table 5-2
Repository Entry Types by Diagram Type

Tablespace	State
View	Event
Business Process Model	**Functional Decomposition Types**
Activity	Function
Business Process	Process
Gateway	**Class**
Data Object (Artifact)	Class
Start, Intermediate, End Events	Data Element
System Boundary (Pool / Swimlane)	Relationship
Note	**Entity Life History Types**
Message Flow	All entity types
Sequence Flow	Event
State Transition Types	Module
State	**Collaboration Types**
Event	Object
	Sequence Types
	Object

Note

- Each variety of module can have a subtype classification. These module subtypes are used to store SQL procedural code to be generated with SQL schemas.
- Each class can also have a subtype classification that is used to provide additional information about how a class is to be used. A *standard* class (the default) indicates a normal class. *Elemental* indicates the class contains no attributes and physical characteristics should be defined. (A data element and a class with an element subtype are equivalent except that methods cannot be specified for a Data Element.) *Structure* indicates a C style structure should be used instead of a class. Members (or attributes) of a structure are public by default while members of a class are private. *Union* indicates a C style union should be used instead of a class. *Entity*, *associative*, and *attributive* indicate the class is persistent and can be used on an entity relationship diagram. You can also specify a stereotype for a class, with the following stereotype values: <<implemtationClass>>, <<interface>>, <<metaclass>>, <<powertype>>, <<process>>, <<structure>>, <<thread>>, <<type>>, <<union>>, <<utility>>.

- Each planning statement must have a statement type that is displayed to the right of the Entry Type field. See manual section Defining Planning Statement Types for how to modify the list of available types.

Description

This field provides a convenient place for you to enter a brief, high-level description of the entry being displayed. You can enter a maximum of two lines of text. When you generate comments on columns or tables in SQL Schema Generation, this field is used. Comments are also imported from and exported to PowerBuilder using this field.

Alias

This field is displayed for all data flow diagram-based repository entries except processes and a number of entry types for other diagram types. It allows you to enter labels that refer to the same entry/object, but that are different than the label at the top of the dialog box. For each alias that you enter, Visible Analyst automatically creates a new repository entry of type alias. That new entry provides a reference to the same repository entry from its label as well as its alias. Note that any alias entry automatically refers to the original entry using its own alias field.

Only one alias is visible in the dialog box at any one time, but you can scroll through all alias names in the usual Windows manner. There can be up to 10 aliases for each repository entry that has an alias field. Each line may contain one alias of up to 128 characters. The first alias in the list is used during SQL schema generation if the Use Alias Name option is selected.

Note

- The alias capability is not enabled for Planning Statements.

Associator Element

Appearing for relationship entries only, this field gives you the opportunity to name a foreign key element. When the entry is saved, the repository creates an entry for this special variety of data element. Analyze can then use this name for key analysis and synchronization.

Related To

This field is displayed for most structure chart diagram entry types. Different entry types are allowed to relate to different data flow diagram entry types. The following list shows you the structure chart entry types and the corresponding data flow entry types to which they may be related.

Module	Up to ten processes
Library Module	Up to ten processes
Macro	Up to ten processes
Library Macro	Up to ten processes

Data Only Module	One data flow, data structure, file, or data element
Data Couple	One data flow, data element
Control Couple	Any one data element
Generic Couple	Same as data couple
Event	A module

Only one Related To entry is presented in the dialog box at any one time unless you expand the field. There are up to 10 Related To lines available for every repository entry that has a Related To field. Each line may contain one entry of up to 128 characters.

Process #

Whenever a process entry is displayed by the repository, this field appears. It cannot be edited from within the repository; all changes must be made on a diagram. The process number is displayed as it appears on the project diagram.

Data Mod(s)

The Data Mod field is a specialized field that appears only for describing data only modules included in information clusters. It cannot be edited from within the repository; all changes must be made on a diagram. An entry for this field is automatically generated from a structure chart diagram, and its content is the name of a data only module that has its own repository entry.

Values & Meanings

This field lists the possible values and meanings of data elements, miscellaneous entries and couples. Information entered into this field may be in any form that is relevant to the application being analyzed or designed. The field may contain up to 64K characters, and you can scroll through the lines in the usual Windows manner.

Discriminator Values & Meanings

If the current object is a data element that is used as a discriminator, this field contains a list of values to identify the subtype entities. For each subtype, a value can be entered that will uniquely identify it. By default, these values are numbers starting with 0 for the supertype. To change the value, click on the value until an edit control appears, make your changes, then press Enter.

Process Description

This field can be used to enter a description for a process. The description should ideally be in some form of structured English or the equivalent, so that data element references can be found within the field. The references provide a means for checking process incoming and outgoing data flows. Good structured analysis techniques dictate that a process description should exist only for the lowest level processes in a project. The field may contain up to 64K characters. Note that in the current release of Visible Analyst, data element references are not searched for in this field.

Module Description

This field can be used to enter a description for a module or a library module or one of the subtypes used for SQL schema generation: triggers, check conditions and stored procedures. The description should either be source code for code generation or some form of structured English or equivalent, so that procedures are documented for beginning processing, for calling other modules, and for the control and data parameters being passed to and from the module. The field may contain up to 64K characters. Refer to the Shell Code Generation section, below, for more uses for this field.

Macro Description

This field can be used to enter a description for a macro or library or one of the subtypes used for SQL schema generation: triggers, check conditions and stored procedures. The description should either be source code for code generation or some form of structured English or equivalent, so that procedures are documented for beginning processing, for calling other modules, and for the control and data parameters being passed to and from the module. The field may contain up to 64K characters. Refer to the Shell Code Generation section, below, for more uses for this field.

Modules Contained

The Modules Contained field is a specialized field that appears only for information clusters. It is automatically generated from structure chart diagrams and may contain up to seven module names. It is not editable from within the repository; all changes must be made on a diagram. Any module name contained in this field also has its own entry in the repository.

Composition/Attributes

The Composition field replaces the Values & Meanings field whenever the repository dialog box displays a file, data store, data flow, data structure, data only, interface table row, entity, class or BPM activity, data object or system boundary (pool). The field can be used to identify all of the components of the repository entry, such as: data elements, data flows, data structures, files, data stores, and couples (all types). For some uses, the composition field does not allow all of the above entry types. It is automatically populated by the Split Data Flow function for DFDs, and the key synchronization function for ERDs. Entry types with a specialized composition field are:

- Data Only Modules. This entry type only allows the composition field to contain data elements and couples (data, control, and generic).
- Interface Table Row. This entry type allows the composition field to contain a group of couples, but only couples of one type; for example, it may contain control couples but not generic or data couples, or it may contain data couples but not the other types.

For every item entered into the composition field, Visible Analyst creates a repository entry (if one with the same name does not already exist) and updates that entry’s location field. If an item is removed from this field, that entry’s location field is updated to reflect this. These repository entries are generally created as data elements and may be changed to data structures. However, some entry composition fields deal only with specific component entry types (see the previous paragraph).

If you click the Composition field or click the Attributes Details button, a dialog box opens, allowing you to define up to 12 components and some of their properties. In addition to the component name, you can specify its physical storage type, length and whether nulls are allowed. As you enter items, the dialog box automatically scrolls as necessary to allow you to enter more items until you reach 12. When you complete the entries, click OK to add them to the Composition field. If you need to add more than 12 components, click the Attributes Details button again and a new dialog box opens so that you can add more. This speeds up the process of generating a table from an entity if you want to generate it without adding full descriptive information for all of the data elements involved.

Item names entered into the composition field may contain up to 128 characters each and may consist of upper or lower case letters, numbers, spaces, periods, underscore characters and hyphens; but the first character must always be a letter. See the Label explanation previously explained in this chapter if you want an item to begin with a number. Asterisks may be used to denote comments; all characters following the asterisk on a line are ignored. *All other characters in the field are considered delimiters and are used to separate items and components entered into this field.* A maximum of 10 items may appear in one line; any additional items are ignored. Item names may not wrap from one line to the next.

For entities, (and classes defined as an entity sub-type) the composition field may carry descriptors of items indicating that they are keys or parts of keys. If the Classic User Interface is selected from the Options menu, you can manually enter the descriptors as explained below. If the Classic User Interface is not selected, use the Keys tab or Key button on the Define dialog to indicate a composition item is part of the primary key, an alternate key, or a performance index. To indicate that a composition item is a primary key, prefix it with [PK]. To indicate a foreign key, prefix with [FK]. An alternate key is indicated by [AK#], and a performance index is indicated by [PI#], where # is a number that annotates the alternate key/performance index this element is a part of. Alternate key and performance index indication is strictly for your own reference; it is not used in any way by Analyze (they are used, however, to generate indexes in some SQL dialects). Up to 64K characters are available in this field. Keys can also be specified by clicking the Keys or the Foreign Keys tab on the Define dialog box when the Classic User Interface is on or off.

The Add button, displayed below the Attributes field when the Classic User Interface is not selected, allows you to add individual components to the entry. If the component already the repository is checked for objects that match the name you typed. Click the Add button to add the entry to Attributes field. Placing the cursor in the Name field enables the Search button, allowing you to search for an existing repository entry as described previously in this chapter.

Populating a Composition

The repository Search function can be used to populate the Composition field of a repository item. To do this, place the cursor in the item's Composition field and click the Search button to display the repository search dialog box. In the Search list box, locate each repository item you want to place in the Composition field and click the Search button; the item is added to the Select list box at the bottom. When you have found all of the entries you want, click the Select button to enter them into the Composition field.

Defining a Primary Key

In addition to the method described above, you can define a primary key for an entity by clicking the primary key button at the repository dialog box. Select the desired columns from the Columns in Table list to add them to the Column in Key list. The order the columns are selected determines the key order. To remove an item from the key, simply deselect the column. Click OK when you are finished. Depending on the type of object being modified, the composition field or attributes field (described below) is updated with a [PK] prefix to indicate the primary key components. (Keys can also be defined by clicking the Keys or Foreign Keys tab on the Define dialog box.)

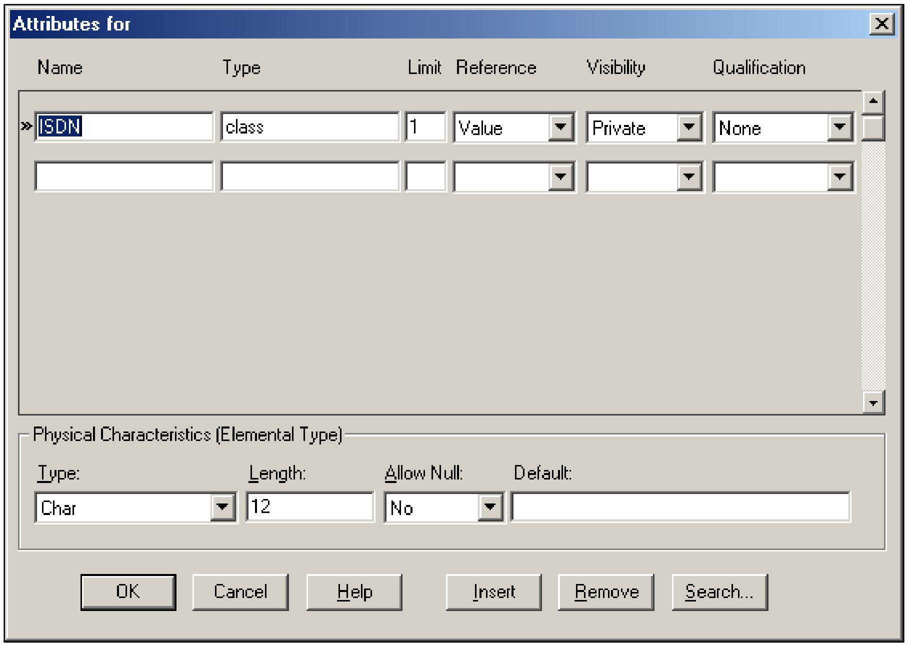
Attributes

The Attributes field replaces the Values & Meanings field whenever the repository dialog box displays a. It also replaces the Composition field if the Classic User Interface option is turned off. The field contains a list of the data members for the class showing the local data element and type. To add, change, or remove members, click Attributes Details button, select Add/Change from the Repository Object menu, double-click in the Composition field while holding down the CTRL key, or click the Add button (the Add button becomes active when you begin typing in the Add field). For each attribute, the following information can be defined:

- **Name.** The name of the attribute. Each attribute of a class has a separate entry in the repository with a type of local data element. This is an optional field. The Search button can be used to find other local data elements in the repository.
- **Type.** The attribute type can be class, data element, or data structure. If the type does not exist in the repository, a new class is created. The location field of the attribute type contains a reference to the current class. This is a mandatory field. The Search button can be used to display a list of valid types. If the attribute type is data element or elemental class, its physical characteristics are displayed.
- **Limit.** The number of occurrences of the attribute. If this field is blank, the attribute occurs once.
- **Reference.** A qualifier to indicate the access method for an attribute. *Value* indicates the object defined in the Type field is used; *address* indicates a pointer to the object is to be used; and *reference* indicates a reference to the object is to be used. The default is value.
- **Visibility.** *Public* members have global visibility. *Private* members are only accessible to member functions and friends. Protected members are accessible to derived classes and

friends. *Implementation* members are only accessible to the class itself. The default is private.

- **Qualification.** *Constant* indicates the member value cannot be changed. *Volatile* indicates the member can be modified by something other than the program, the operating system or hardware. *Static* indicates there is only one instance of the member regardless of the number of times a class is instantiated. The default is none.
- **Physical Characteristics.** If the attribute type is elemental, the physical characteristics can be set. See the section Data Element Physical Information later in this chapter for details.



Attributes for

Name	Type	Limit	Reference	Visibility	Qualification
ISDN	class	1	Value	Private	None

Physical Characteristics (Elemental Type)

Type:	Length:	Allow Null:	Default:
Char	12	No	

Figure 5-9 Class Attributes

For every item entered into the Type field, Visible Analyst creates a repository entry (if one with the same name does not already exist) and updates that entry's location field. If an item is removed from this field, that entry's location field is updated to reflect this. These repository entries are generally created as classes unless a data element already exists with the same name, or the physical characteristics are defined.

As you enter items, the dialog box automatically scrolls as necessary to allow you to enter more items until you have finished. Insert is used to insert a new attribute into the list at the current position, while Delete removes the current attribute (the current position is indicated by ➢➢). To reorder the list, click the left mouse button on the current position indicator and drag the row to its new position. When you have completed the entries, click OK to add them to the Attributes field.

Item names entered into this field may contain up to 128 characters each and may consist of upper or lower case letters, numbers, spaces, periods, underscore characters and hyphens; but the first character must always be a letter. If you want an entry to begin with a numeric value, edit the VAW.INI file and add this line to the file: Allow Numeric Prefix=Yes and then restart the Visible Analyst to enable the change.

Attached Entities/Classes

The attached entities/classes for the currently displayed relationship are listed in this field. When an inheritance relationship is displayed, the characteristics of that relationship can be changed (see Changing Inheritance Characteristics later in this chapter), otherwise, it cannot be edited from within the repository; all changes must be made on a diagram. The field lists the two entities or classes attached to this relationship. Below the second entity name is listed the reverse of the current relationship. If either direction of the relationship has not been named, the name of the relationship in the reverse direction is displayed as "reverse of (opposite relationship name)." This field allows you to jump to the repository entries for any of these entities or relationships.

Relations

For an entity or class, the Relations field displays, for each relationship attached to this entry, the relationship name followed by the name of the entity or class on the other end of this relationship. These sets are ordered alphabetically by the opposite entry name. When an inheritance relationship is displayed, the characteristics of that relationship can be changed (see Changing Inheritance Characteristics later in this chapter), otherwise the information cannot be edited from within the repository; all changes must be made on a diagram.

This field allows you to jump to the repository entries for any of these entities, classes, or relationships by positioning the cursor on the line containing an entity, class, or relationship name and clicking the Jump button.

Notes

This field allows you to enter information in free form fashion for any repository entry. Up to 64K characters are available in this field.

Locations

This field displays an entry's *locations*; it cannot be edited. Three types of locations may be present in this field for any entry. The first type shows the label of the diagram on which the entry is located, followed by its diagram number in parentheses (DFDs only). The second type shows any other entries that reference the currently displayed entry in their composition fields. The third type of location displays the name of the satellite project if this entry was used in an Enterprise Copy. See the chapter on Enterprise Modeling for additional information. The Repository Report function lists all locations.

Long Name

When a repository entry, either a local data element or a module, belongs to a class, the full name of the entry includes the class name. The Long Name field displays this name, and in the case of modules, includes the argument list (the argument list is required to differentiate overloaded member functions). If you want to change the argument list for a class method, click the right mouse button on the Long Name field and select Change (see the Methods section later in this chapter for details). If you want to change the class to which the method belongs, select Class from the Repository Object menu. To display the class definition, click the Jump button.

Class Characteristics

Concurrency is a class property that distinguishes an active object from inactive object. An *active* object may represent a separate thread of control. A *sequential* object is a passive object whose semantics are guaranteed only in the presence of a single thread of control. A *guarded* object is a passive object whose semantics are guaranteed in the presence of multiple threads of control.

A *persistent* class exists beyond the lifetime of an executable program. This means it must be stored on a non-transitory storage device. If the subtype of a class is set to associative or attributive entity and the class is used on an entity relationship diagram, this field cannot be changed.

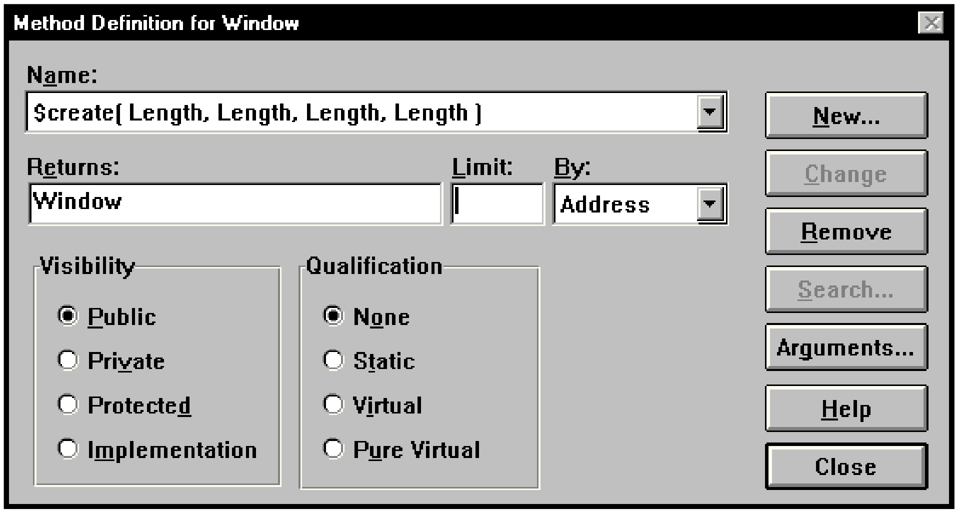
An *abstract* (or virtual) class cannot be instantiated because it contains pure virtual methods. If pure virtual methods exist for a class, Abstract is checked. If you attempt to uncheck this field, all pure virtual methods are reset to virtual. If you attempt to check it and virtual methods exist, they are converted to pure virtual methods.

Methods

Methods (or Member Functions) are the operations that are defined for accessing a class. The Methods field contains a list of the functions for a class showing the name, return value, argument list, and flags to indicate its visibility. To add, change, or remove methods, click on the Methods field and click the Attributes Details button, select Add/Change from the Repository Object menu, or double-click the Methods field while pressing the CTRL key.

To add a new method for a class, click the New button and type the name of the method you wish to add. To search for methods that have already been defined in the repository, navigate from the Name field and click the Search button. The list contains all modules that have previously been defined in the repository. If the module already belongs to a class, the class name is displayed. Note that when you select a module that already exists, the complete definition for that module is used including return value and argument list. Click OK to add the method name to the list of methods for the current class. For each method, the following information can be defined:

- **Returns.** The return type can be a class or data element. If the type does not exist in the repository, a new class is created. The location field of the attribute type contains a reference to the method. This is an optional field. The Search button can be used to display a list of valid types.
- **Limit.** The number or size of the parameter. If this field is blank, it occurs once.
- **By.** A qualifier to indicate how the return value is passed. *Value* indicates a copy of the parameter is passed; *address* indicates a pointer to the object is to be used; and *reference* indicates a reference to an object is to be used.
- **Visibility.** *Public* methods have global visibility. *Private* methods are only accessible to other member functions within the same class and friends. *Protected* methods are accessible to derived classes and friends. *Implementation* methods are only accessible to the class itself. The default is public.
- **Qualification.** *Static* indicates a method can be used without a specific instance of an object (it can only be used with static attributes (data members)). A *virtual* method is one that you expect to be redefined in a derived class. A *pure virtual* method has no definition and must be defined in a derived class. A class with pure virtual functions is an abstract (or virtual) class. The default is none.
- **Arguments.** A list of parameters to be used by the method. This is an optional field. If a method appears more than once with the same name within a class, it must have a different argument list for each definition. This is known as function overloading. See the next section for defining arguments.



Method Definition for Window

Name: $create(Length, Length, Length, Length)

Returns: Window

Limit:

By: Address

Visibility

● Public

○ Private

○ Protected

○ Implementation

Qualification

● None

○ Static

○ Virtual

○ Pure Virtual

Figure 5-10 Class Methods

When a method is added to a class definition, a module entry is created in the repository. The long name includes the class name and the argument list. The argument list is needed to differentiate between overloaded functions.

Note

- Because the same name can be used for more than one method, there may be duplicate module entries in the repository, each belonging to a different class.

Arguments

When defining methods (member functions) for a class, the parameters to the function must be specified. To add, change, or remove arguments, click on the Arguments button on the Method Definition dialog box. For each argument, the following information can be defined:

- **Name.** The name of the parameter. This is an optional field.
- **Type.** The parameter type can be a class or data element. If the type does not exist in the repository, a new class is created. This is a mandatory field. The Search button can be used to display a list of valid types. If the parameter type is a data element or elemental class, its physical characteristics are displayed.
- **Limit.** The number or size of the parameter. If this field is blank, it occurs once.
- **Pass By.** A qualifier to indicate how the parameter is passed. *Value* indicates a copy of the parameter is passed; *address* indicates a pointer to the object is to be used; and *reference* indicates a reference to an object is to be used. The default is value.

- **Qualification.** *Constant* indicates a parameter value cannot be changed. *Volatile* indicates the parameter can be modified by something other than the program, either the operating system or hardware. The default is none.
- **Physical Characteristics.** If the parameter type is elemental, the physical characteristics can be set. See the section Data Element Physical Information later in this chapter for details.

For every item entered into the Type field, Visible Analyst creates a repository entry (if one with the same name does not already exist). These repository entries are generally created as classes unless a data element already exists with the same name or the physical characteristics are defined.

As you enter items, the dialog box automatically scrolls as necessary to allow you to enter more items until you have finished. Insert is used to insert a new parameter into the list at the current position, while Delete removes the current parameter (the current position is indicated by ➢➢). To reorder the list, click the left mouse button on the current position indicator and drag the row to its new position. When you have completed the entries, click OK to update the method name field.

Item names entered into this field may contain up to 128 characters each and may consist of upper or lower case letters, numbers, spaces, periods, underscore characters and hyphens; but the first character must always be a letter.

Friends

The Friends field displays a list of both friend classes and methods (or functions). A friend is allowed access to the private data members of a class. To add friends, click on the Friends field and click the Search button, select Add from the Repository Object menu, or double-click on the Friends field while pressing the CTRL key. A list of classes and member functions is displayed in the Search list box. Locate each repository item you want to place in the Friends field and click the Search button; the item is added to the Select list box at the bottom. When you have found all of the entries you want, click the Select button to enter them in the Friends field.

To remove a friend, highlight the desired item and press the DELETE key, or select Cut or Delete from the Repository Object menu.

Scenario

When the entry type is use case, this field provides a place for you to enter a description of the scenario.

Types of Activity

Specifies the type of activity on the Type/Status tab.

None: Indicates a normal Task. See None.

Service: The Task provides some type of service. See Service

Receive: The Task is designed to wait for a message to arrive from an external participant (relative to the Business Process). See Receive

Send: A Task that is designed to send a message to an external Participant (relative to the Business Process). A Send Task is a simple Task that is designed to send a message to an external Participant (relative to the Business Process). Once the message has been sent, the task is completed. The task Type is specified on the activity's Type / Status tab in the repository with the following fields enabled: A Message MUST be entered into the In Message field to indicate the message to be sent by the Task. The Message in this context is equivalent to an out-only message pattern. The Message Flow may be shown on the diagram, but is not required.

Implementation: This field specifies the technology that will be used to send the message, and a Web service is the default.

User: A typical "workflow" task where a human performer performs the Task with the assistance of a software application and is scheduled through a task manager of some sort.

Script: The script written in the Script field of the Task is executed by a business process engine. The modeler or implementer defines a script in a language that the engine can interpret. When the Task is ready to start the engine will execute the script. When the script is completed, the Task will also be completed. The task Type is specified on the activity's Type / Status tab in the repository with the Script field enabled:

Abstract: This Task represents the interactions between private business processes and another process or participant.

Manual: A Task that is expected to be performed without the aid of any business process execution engine or any application

Reference: If the two (or more) activities share the exact same behavior, then by one referencing the other, the attributes that define the behavior only have to be created once and maintained in only one location. When Reference is specified as the type for a Transaction activity, the activity will reference the Business Process. Enter the Business Process name in the Process field on the activity's Type/Status tab.

Independent: An Independent sub-process activity is an activity within a process that calls to another process within a Business Process Diagram that may have Pools and swimlanes. The Independent activity type can only be specified for a Transaction activity.

Embedded: An Embedded (or Nested) activity is a sub-process of an activity that contains other sub-processes and is dependant on the parent activity for instigation. The Embedded activity type can only be specified for a Transaction activity.

Service

A Service Task is a Task that provides some type of service, which could be a Web service or an automated application. The task Type is specified on the activity's Type / Status tab in the repository, with the following fields enabled:

In Message: An In Message MUST be entered, and the message indicates that the message will be sent at the start of the Task, after the availability of any defined InputSets.
Out Message: An Out Message MUST be entered and the arrival of the message indicates the completion of the Task, which may cause the production of an OutputSet.
Implementation: This field specifies the technology that will be used by the Performers to perform the Task. A Web Service is the default technology.

Receive

A Receive Task is a simple Task that is designed to wait for a message to arrive from an external participant (relative to the Business Process). Once the message has been received, the task is completed. A Receive task is often used to start a Process. In order for the Task to Instantiate the Process, it must meet one of the following criteria:

- The Process does not have a Start Event and the Receive Task has no incoming Sequence Flow
- The Incoming Sequence flow for the Receive Task has a source of a Start event. Note that no other incoming Sequence Flow are allowed for the Receive Task (in particular, a loop connection from a downstream object)

The task Type Receive is specified on the activity's Type / Status tab in the repository with the following fields enabled:
A Message MUST be entered into the In Message field to indicate the message to be received by the Task. The Message in this context is equivalent to an in-only message pattern (Web service).
Instantiate: Receive tasks can be defined as the instantiation mechanism for the Process with the Instantiate checkbox. The attribute may be selected (checked to indicate true) if the Task is the first activity after the Start Event or a starting Task if there is no Start event (i.e. there are no incoming Sequence Flow). Multiple Tasks MAY have this attribute set to "True".
Implementation: This field specifies the technology that will be used to send the message; a Web service is the default.

Send

A Send Task is a simple Task that is designed to send a message to an external Participant (relative to the Business Process). Once the message has been sent, the task is completed. The task Type is specified on the activity's Type / Status tab in the repository with the following fields enabled:
A Message MUST be entered into the In Message field to indicate the message to be sent by the Task. The Message in this context is equivalent to an out-only message pattern. The Message Flow may be shown on the diagram, but is not required.
Implementation: This field specifies the technology that will be used to send the message, and a Web service is the default.

User

The User Task Type is a typical “workflow” task where a human performer performs the Task with the assistance of a software application and is scheduled through a task manager of some sort.
The task Type is specified on the activity’s Type / Status tab in the repository. This type of Task maps to an invoke activity in BPEL4WS.

An activity of type “User” has the following attributes enabled:
Performers: The human resource that will perform the task, i.e. an individual, a group, or an organization.
In Message: An In Message MUST be entered, and the message indicates that the message will be sent at the start of the Task, after the availability of any defined InputSets.
Out Message: An Out Message MUST be entered and the arrival of the message indicates the completion of the Task, which may cause the production of an OutputSet.
Implementation: This field specifies the technology that will be used by the Performers to perform the Task. A Web Service is the default technology.

Script

A Script Task is a Task executed by a business process engine. The modeler or implementer defines a script in a language that the engine can interpret. When the Task is ready to start the engine will execute the script. When the script is completed, the Task will also be completed. The task Type is specified on the activity’s Type / Status tab in the repository with the following field enabled:
Script: The script to be run when the Task is performed. If a script is not included, then the task will act equivalent to a task type of “None”.

Abstract

An abstract process represents the interactions between private business processes and another process or participant. Use the Type field on the Type / Status tab on the activity’s repository entry to select this entry type.

Manual

A Manual Task is a Task that is expected to be performed without the aid of any business process execution engine or any application. An example would be a technician installing a telephone at a customer location. The task Type is specified on the activity’s Type / Status tab in the repository with the following field enabled:
Performers: The human resource that will perform the task, i.e. an individual, a group, or an organization.

Reference

There may be times where a modeler may want to the reference another activity that has been defined. If the two (or more) activities share the exact same behavior, then by one referencing the other, the attributes that define the behavior only have to be created once and maintained in only one location. The task Type is specified on the activity’s Type / Status tab in the repository with the following field enabled:

Task: The field where the Task name is entered. Once the entry is saved, a new Activity will be created in the repository with the Task name, if the activity does not exist. Use the search button (…) to the right of the Task field to search for existing activities.

Independent

An Independent sub-process activity is an activity within a process that calls to another process within a Business Process Diagram that may have Pools and swimlanes. The process that is called is not dependent on the parent process for data or instantiation, and can be instantiated by other sub-processes or by a message from an external source. The Independent sub-process may pass data to or from the called process. The Independent activity type is selected from the Type field on the Type / Status tab for an activity's repository entry.

Embedded

An Embedded (or Nested) activity is a sub-process of an activity that contains other sub-processes and is dependant on the parent activity for instigation. An embedded activity diagram since, it is dependant on the parent activity, does not make use of the Pools and Swimlanes, and would only contain Flow Objects, Connecting Objects and Artifacts. The sub-processes share the same data as the parent activity. The + sign symbol is added to an embedded activity symbol to signify it has sub-processes.

Business Process Type

Provides information about which lower-level language the Pool will be mapped. By default, Type is None (or undefined).
A **Private** process type MAY be mapped to an executable BPEL4WS process.
An **Abstract** process type is also called the public interface of a process (or other web services) and MAY be mapped to an abstract BPEL4WS process.
A **Collaboration** process type is also considered a "global" process and MAY be mapped to languages such as ebXML or WS Choreography.

Suppress Join Failure

This field for a Business Process specifies whether or not BPEL4WS joinFailure fault will be suppressed for all activities in the BPEL4WS process.

Enable Instance Compensation

This check is included on a Business Process repository entry, and specifies whether or not compensation can be performed after a process has completed normally.

Activity Process Type

Sub-Process:
Defines whether the Sub-Process details are embedded within the higher level Process or refers to another, re-usable Process.
Task:
The “Task Type” will be impacted by the Message Flow to and/or from the Task, if Message Flow are used. A task type of “Receive” MUST NOT have an outgoing Message Flow. A task type of “Send” MUST NOT have an incoming Message Flow. A task type of “Script”, “Manual”, or “None” MUST NOT have an incoming or an outgoing Message Flow.

Compensation (activity)

A Compensation Activity compensates for work done in the source activity. It is special in that it does not follow the normal Sequence Flow rules; it is outside the Normal Flow of the Process. There can be only one target activity for compensation. If compensation requires more than one activity, these activities are added as sub-processes of the target activity, and shown on the attached child diagram. The Compensation marker is two backward pointing triangles, similar to the rewind button for a tape player.

Only activities that have been completed can have compensation, and the compensation can be triggered in two ways:

- The activity inside a Transaction Sub-Process is cancelled. In this situation the whole Sub-Process would be “rewound” or rolled back - - the Process flow will go backwards and any activity that requires compensation will be compensated. This is why the Compensation marker for events looks like a “rewind” symbol for a tape player. After the compensation has been completed, the Process will continue its rollback.
- A downstream Intermediate Event of type Compensation “throws’ a compensation identifier that is “caught” by the Intermediate Event attached to the boundary of the activity.

InputSets

The attributes that make up an InputSet that allows the activity to be performed, and one or more inputs must be defined for each InputSet. An Input is an artifact, usually a Data Object that may be displayed on the BPMN diagram. An Association Flow is used to link the Data Object Symbol to the activity. These attributes are added to the activity on the activity’s Description tab.
Note: Global data elements defined in the Visible Analyst repository as part of the entity, class, and data flow diagram object definitions are available for use in these activities and data objects. See the on-line Help topic Search for an explanation how to use these previously defined attributes as part of the activity’s definition.

OutputSets

OutputSets are the data requirements for output from the activity. Zero or more OutputSets may be defined, but at the completion of the activity only one OutputSet would be produced. An Output is an Artifact, usually a Document Object.

Participant

A Participant is a business entity, (i.e. company, company division, customer, etc.) or a business Role (a buyer or seller for example), which controls or is responsible for a business process. If Pools are used, then a Participant would be associated with one Pool. The modeler MUST define the participant for the Pool, either a Role or Entity.

The Participant field is included on an activity's Type/Status tab and on event's Trigger tab when the Web Service Property fields are enabled for these items. The two options available for the Participant field type are **Role** or **Entity**, while the text field to the right of this option contains the name of the Role or Entity participant.

Role

The person involved in the BPMN model (buyer, seller, etc.), as opposed to an Entity, the company department (sales, Accounting, IT, etc.) involved in the process. The Role / Entity attribute can be assigned for the following BPMN items: Pool, Events, and Activities.

Entity

An Entity identifies the Participants involved in the BPMN activities. The Entity Participant field value can be specified on the following BPMN objects as part of the Web Service Properties: an Activity, End Event, Intermediate Event, Start Event, and on the Participant Tab of a Pool (System Boundary).

Interface

This is a Web Service field property for an activity or an event. This field allows you to specify an interface (also known as a port type) for a web service. This is a required field.

From (Expression)

An activity Assignment property, the user would enter an expression that is made up of a combination of values, properties, and attributes, which are separated by operators such as add or multiply.

From (Message)

Specifies the role type for a target participant with the name of the sending participant, either a Role or an Entity.

Message

A Message is the object that is transmitted through a Message Flow. The Message will have an identity that can be used for alternative branching of a Process through the Event-Based Exclusive Gateway.

In Message

This activity field specifies the In Message, which indicates that the defined message will be sent at the start of the task. Enter the name of the message or use the browse button to the right.

Out Message

This activity field specifies the Out Message, which indicates that the arrival of the defined message marks the completion of the task. Enter the name of the message or use the browse button to the right.

Performers

A task attribute of a Manual Task, the Performers attribute defines the human resources that will be performing the Manual Task. The Performers entry could be in the form of a specific individual, a group, an organization role or position, or an organization.

Implementation

Specifies the technology that will be used to send and receive the messages for a service task. A Web Service is the default, but the values of "Other" and "Unspecified" can be selected.

Bus. Proc.

This field is displayed on the Type/Status tab for a process with the Type set to Independent (an independent sub-process) since the process must have a related business process. Enter the name of the business process into this Bus. Proc. field or use the browse button indicated by "…" to the right of the field to Search for an existing business process.

Start Quantity

This attribute defines the number of Tokens that must arrive from a single Sequence Flow before the activity can begin. The default value is 1, and the value MUST NOT be less than 1.

Status

The Status of a Process is determined when the Process is being executed by a Process Engine, and the Status of a Process can be used within Assignment Expressions. The status values include: None, Ready, Active, Cancelled, Aborting, Aborted, Completing, Completed.
The Status field is located on the Type / Status tab of the activity's repository entry.

Task (Reference)

The Task field for an activity specifies a referenced task. Users can enter the name of the task into this field or use the browse button (…) to the right of the field to Search for an existing activity.

Assignments

Assignments are defined on the Assignments tab of the activity's repository entry, and use the Attributes (data elements) listed in the Attributes field on the Description tab. The assignment information is used when generating the BPEL code (Business Process Execution Language) based on the BPMN diagram information. Each assignment consists of the name of the data element in the "To" field, an expression operation such as add or multiply in the "From" field, and a determination when the expression is implemented, either at the "Start" or the "End" of the activity. To add a new assignment, use the controls on the activity's Assignments tab. To change an existing assignment, use the same controls. To remove an assignment, click on the desired attribute and then press the Delete key on the keyboard

Ad Hoc

Specifies whether the Process is Ad Hoc or not. The activities within an Ad Hoc Process are not controlled or sequenced in a particular order, their performance is determined by the performers of the activities.

Ad Hoc Process Properties

Ordering: Defines if the activities within the process can be performed in Parallel or must be performed sequentially. The default setting is Parallel and the setting of Sequential is a restriction on the performance that may be required due to shared resources.
Completion Condition: Defines the conditions when the process will end. If the process is Ad Hoc, this attribute MUST be included.

Operation

This is a Web Service field property for an activity or an event, and is used to specify one or more operations for a web service, receive, reply, or invoke, as detailed in the BPEL4WS documentation. This is a required field.

Looping

BPMN provides two mechanisms for looping, i.e. continuing a process until some threshold is reached.
Standard: A Standard loop activity will have a Boolean expression that is evaluated after each cycle of the loop. If the expression is still "True", then the loop will continue.
Multi-Instance: The loop expression for a Multi-Instance loop is a numeric expression evaluated only once before the activity is performed. The result of the expression evaluation will be an integer that will specify the number of times the activity will be repeated.

The **Condition** field
For a Standard loop activity, this is a Boolean expression that is evaluated after each cycle of the loop. If the expression is still True, then the loop will continue. For a Multi-Instance loop, this is a numeric expression evaluated only once before the activity is performed. The result of the expression evaluation will be an integer that will specify the number of times that the activity will be repeated.

The **Counter** field
Used at runtime to count the number of loops and is automatically updated by the process engine. This attribute MUST be incremented at the start of a loop.

The **Maximum** field
Defines the maximum number of times the loop will execute.

The **Test Time** field
Determines when the loop condition is to be evaluated. Expressions that are evaluated before the activity begins are equivalent to a programming while function, while expressions that are evaluated after the activity finishes are equivalent to a programming until function.

The **Ordering** field
Defines whether the loop instances will be performed sequentially or in parallel. Sequential is the more traditional looping order, as opposed to parallel.

The **Flow Condition** field
This attribute is equivalent to using a Gateway to control the flow past a set of parallel paths:
None - The same as uncontrolled flow (no Gateway) and means that all activity instances SHALL generate a token that will continue when that instance is completed.
One - The same as an Exclusive Gateway and means that the token SHALL continue past the activity after only one of the activity instances has completed. The activity will continue its other instances, but additional Tokens MUST NOT be passed from the activity.
All - The same as a Parallel Gateway and means that the token SHALL continue past the activity after all of the activity instances have completed.
Complex - The same as a Complex Gateway. The Complex Flow Condition attribute will determine the token flow.

Activity looping is shown on the Activity symbol with a small line with an arrowhead curling back on itself. The attributes of the Tasks and Sub-Processes determine if they are repeated or performed once.

Sequence Flow looping is created by connecting a Sequence Flow to an “upstream” object causing the process to be repeated.

None

Many different BPMN objects have a field(s) that can be set to “None”.
None, when selected as an Event type means that the modeler does not display the type of event.
Additional uses of None are:
As a Start Event, it is also used for a sub-process that starts when the flow is triggered by its parent process.
As an Intermediate Event, this None value is only valid for events that are in the main flow of the Process. It is also used for modeling methodologies that use Events to indicate some change in the state of the Process.
As an End Event it is also used to show the end of the sub-process that ends, which causes the flow to go back to its Parent Process
As a Condition Type for a Sequence Flow, “None” indicates that no runtime evaluation is performed and the flow is used as the normal uncontrolled flow between activities.
For a Business Process, the value of “None” in the Type field indicates that no lower-level language will be mapped to the Pool.
For a Task (activity) of type “None”, None indicates a normal task.
For the Looping field of an activity, “None” indicates no looping.
None also identifies the Status of a process when being executed by a process engine

Trigger Time

The specific time-date or a specific cycle an event will occur to trigger a process. Either a Date and /or Time can be specified, or a Cycle such as daily, weekly, monthly, etc. If a Cycle is not specified, a Date and time must be specified.

Rule

A Rule is an expression that evaluates some Process data. The type “Rule” can be specified for a Start or Intermediate Event that is evaluated and triggered when the Rule evaluates to “True”.
For a **Start Event** of type Rule, the event is triggered when the rule condition becomes true, such as “S&P changes by more than 10%”.
An **Intermediate Event** of type Rule is only used for exception handling and is triggered when a Rule becomes True. Enter the rule expression in the Rule field on the Event’s Trigger tab.

Error Code

Enter an error code for an Intermediate or End Event Error result.
For an Intermediate Event within Normal Flow:
- If the Trigger is an Error, then the error code MUST be entered. This “throws” the error.

For an Intermediate Event attached to the boundary of an Activity:
- If the Trigger is an Error, then the error code MAY be entered. This “catches” the error.

- If there is no error code, then any Error SHALL trigger the Event.
- If there is an error code, then only an Error that matches the error code SHALL trigger the Event.

Incoming

This field is located on the Conditions tab for a Gateway object. If there are multiple incoming sequence flows, an incoming condition expression MUST be set. This will consist of an expression that can reference sequence flow names and or process properties.

Outgoing

This field is located on the Conditions tab for a Gateway object. If there are multiple outgoing sequence flows, an outgoing condition expression MUST be set.

State

Located on a Data Object's State tab as a text field in the repository, this field indicates the impact the process has had on the data object. Multiple data objects with the same name MAY share the same state within one process.

Required For Start

If this checkbox on a Data Object's State tab is checked, this means that the input is required for the activity to start. If not checked, then the activity MAY start without the input, but MAY accept the input (more than once) after the activity has started.

Produced At Completion

If this checkbox on a Data Object's State tab is checked, this means that the output will be produced when the activity has been completed. If not checked, then the activity MAY produce the output (more than once) before it has completed.

The following fields are located on the various Label item dialogs. These field values are included in the reports but are not editable directly in the repository.

Note and Note Link

The Note symbol is used on A BPMN diagram in the Visible Analyst to represent an Annotation (or Text Annotation). The Note is used as a mechanism for the modeler to provide additional information for the reader of a BPMN diagram. A Note object (and a Note Link) in the Visible Analyst does not maintain a repository entry; all information is written on the diagram as the Note label. Notes are connected to a specific object on the diagram with a Note Link line, but neither the Note nor the Note Link line affect the flow of the Process.

The Note Link line type is used in the Visible Analyst to connect a Note (also known as an Annotation or Text Annotation) to another BPMN diagram object. The Note Link does not maintain a repository entry.

Quantity

This attribute, specified on the Label Sequence Flow dialog, defines the number of Tokens that must arrive from a single Sequence Flow before the activity can begin. The value MUST NOT be less than one.

Conditional Expression

A valid expression evaluated at runtime. If the result of the evaluation is True, then a Token will be generated and traverse the Sequence—Subject to any constraints imposed by a Source that is a Gateway. This expression field can be populated when the Condition Type of the Sequence Flow is set to Expression. NOTE: This field is displayed on the "Label Sequence Flow" dialog. Use Change Item to edit the flow and enter the expression.

Condition Type

A Sequence Flow attribute, as determined on the Label Sequence Flow dialog, this field indicates the runtime evaluation whether or not the Sequence Flow will be used.
The default value is "**None**", indicating no runtime evaluation and is used as the normal uncontrolled flow between activities.
A value of "**Expression**" enables the Conditional Expression that will be evaluated at runtime. A diamond symbol is added to the beginning of the line to graphically indicate that the flow is a flow of type "Expression".
A value of "**Default**" indicates a default value will be evaluated at runtime. A slash is added to the beginning of the line to graphically indicate that the flow is a flow of type "Default".
NOTE: The Condition Type of the Sequence Flow may be automatically determined in the Visible Analyst based on symbol type in which the flow originates. A "None" Condition Type MUST NOT be used if the source of the flow is an Exclusive Data-Based or Inclusive Gateway, so the NONE selection is unavailable when a Sequence line originates from this symbol type.

Planning Statement Links

In order to track the software development process from the planning stages through analysis, design, and implementation, it is important to be able to link planning statements to model objects to help you determine the significance of each object and to ensure that each object is essential in supporting the organization's business plan. The Links To field on the Links tab of the Define dialog box allows you to maintain these relationships.

There are three methods for creating a link between a planning statement and any other object. Either right-click the Links To field and choose Add from the Properties menu, or drag a planning statement from the Planning Hierarchy window using the mouse. You can also set the focus in the Links To field and press the Insert key. If you use the Add method, a list of repository entries is presented; and you can select the desired object. If the current object is not a planning statement, only planning statements are listed.

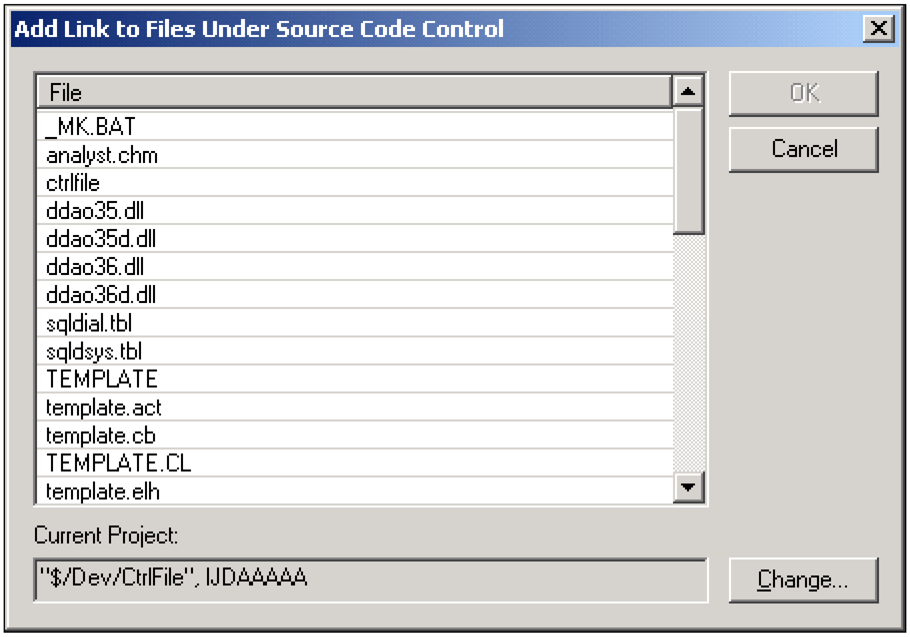
Once the link has been added, its name and type are displayed in the field. The link is visible from both directions. If you are looking at a planning statement, you see the linked object. If you are defining any other type of object, the planning statement is displayed. The linkage rules for a statement type dictate how statements of that type can be linked to other object. See manual section Defining Planning Statement Types for details.

To remove a link, highlight the item and then press the Delete key.

External Links

Because the software development process does not end when your models have been completed, it is important to be able to link your model elements to the source code that actually implements them. Using the external links feature of Visible Analyst, you can link any model element to a file under source code control by a Microsoft SCC-compliant provider, such as Razor.

To add an external link, right-click on the Links To field and choose Add External Link. (You can also press the Insert key while holding down the Ctrl key.) If you have access to an SCC-compliant provider, you are prompted to log in and provided a list of source files under version control to choose from. An example of a list of source files is shown below.



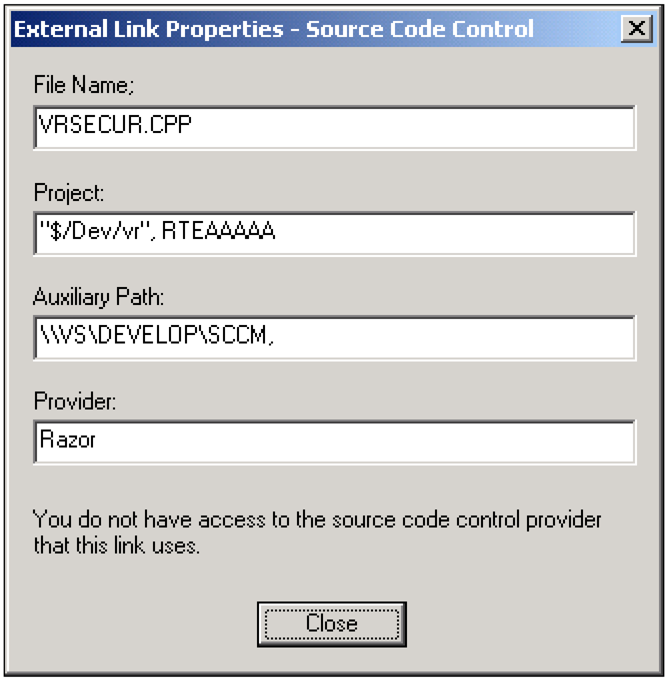
File
_MK.BAT
analyst.chm
ctrlfile
ddao35.dll
ddao35d.dll
ddao36.dll
ddao36d.dll
sqldial.tbl
sqldsys.tbl
TEMPLATE
template.act
template.cb
TEMPLATE.CL
template.elh

Current Project:
"$/Dev/CtrlFile", IJDAAAAA

Figure 5-11 Add Link to Files Under Source Code Control Dialog Box

Select the files to link, and then click OK. If you want to change the current process, click the Change button. Note that the information in the Current Project field is unique to the provider, and may not have meaning to you.

Once an external link has been added, you can view the properties and revisions history of the file by right-clicking on the link and selecting the appropriate option if the provider supports these features. If you do not have access to a provider, selecting Properties displays a dialog box similar to the one shown below.



External Link Properties - Source Code Control

File Name: VRSECUR.CPP

Project: "$/Dev/vr", RTEAAAAA

Auxiliary Path: \\VS\DEVELOP\SCCM,

Provider: Razor

You do not have access to the source code control provider that this link uses.

Figure 5-12 External Link Properties, Source Code Control Window

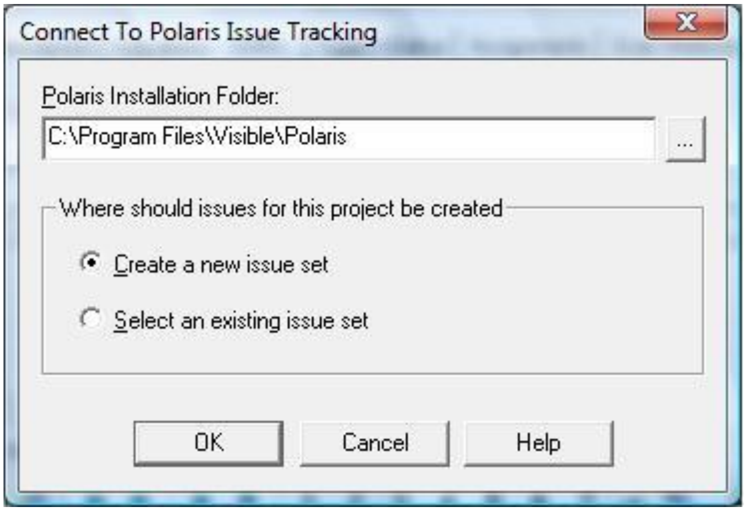
To remove an external link, highlight the link and press the Delete key.

Polaris Integration

Polaris is a customizable issue tracking and defect management tool that you can use to improve your software development process. By using Polaris in combination with Visible Analyst, you can associate issues with objects in the VA repository to provide a rich environment to monitor the development of your models. To lean more about Polaris, visit our web site at www.visible.com.

Linking a Project to an Issue Set

Before issues can be associated with model objects, the project manager must establish a link between the Visible Analyst project and a Polaris issue set. The first time a project manager attempts to add an issue to an object, the **Connect To Polaris** dialog is displayed.



Connect To Polaris Issue Tracking

Polaris Installation Folder: C:\Program Files\Visible\Polaris

Where should issues for this project be created

● Create a new issue set

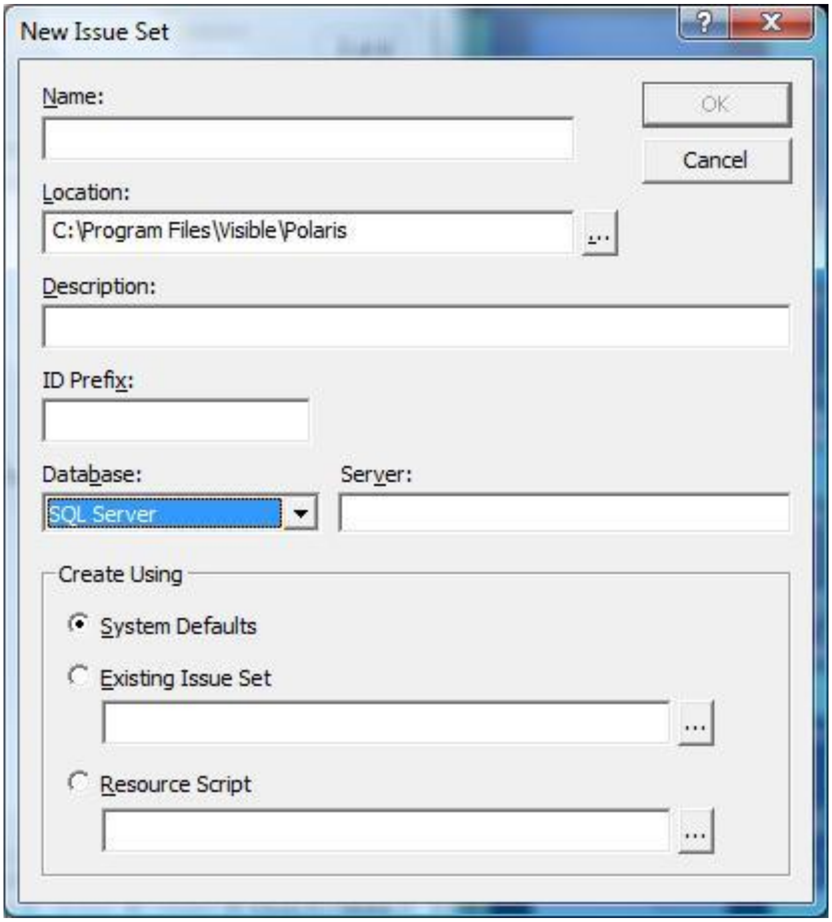
○ Select an existing issue set

Figure 5-13 Connect To Polaris Dialog

Select Polaris Location: 1 — Enter the name of the folder where Polaris is installed or click on the browse button (...). This should only have to be done the first time a project is connected to Polaris. On any subsequent attempt, Visible Analyst remembers the location.

Select Issue Set: 2 — Choose whether issues for this project should be stored in a new or existing issue set. Only experienced Polaris users should choose to create an issue set. Contact your Polaris administrator for help.

Press OK after selecting or creating a new issue set. An attempt is made to connect to Polaris, and if successful, the New Issue Set or New Issue dialog is displayed.



New Issue Set

Name:

Location: C:\Program Files\Visible\Polaris

Description:

ID Prefix:

Database: SQL Server

Server:

Create Using

● System Defaults

○ Existing Issue Set

○ Resource Script

Figure 5-14 New Polaris Issue Set Dialog

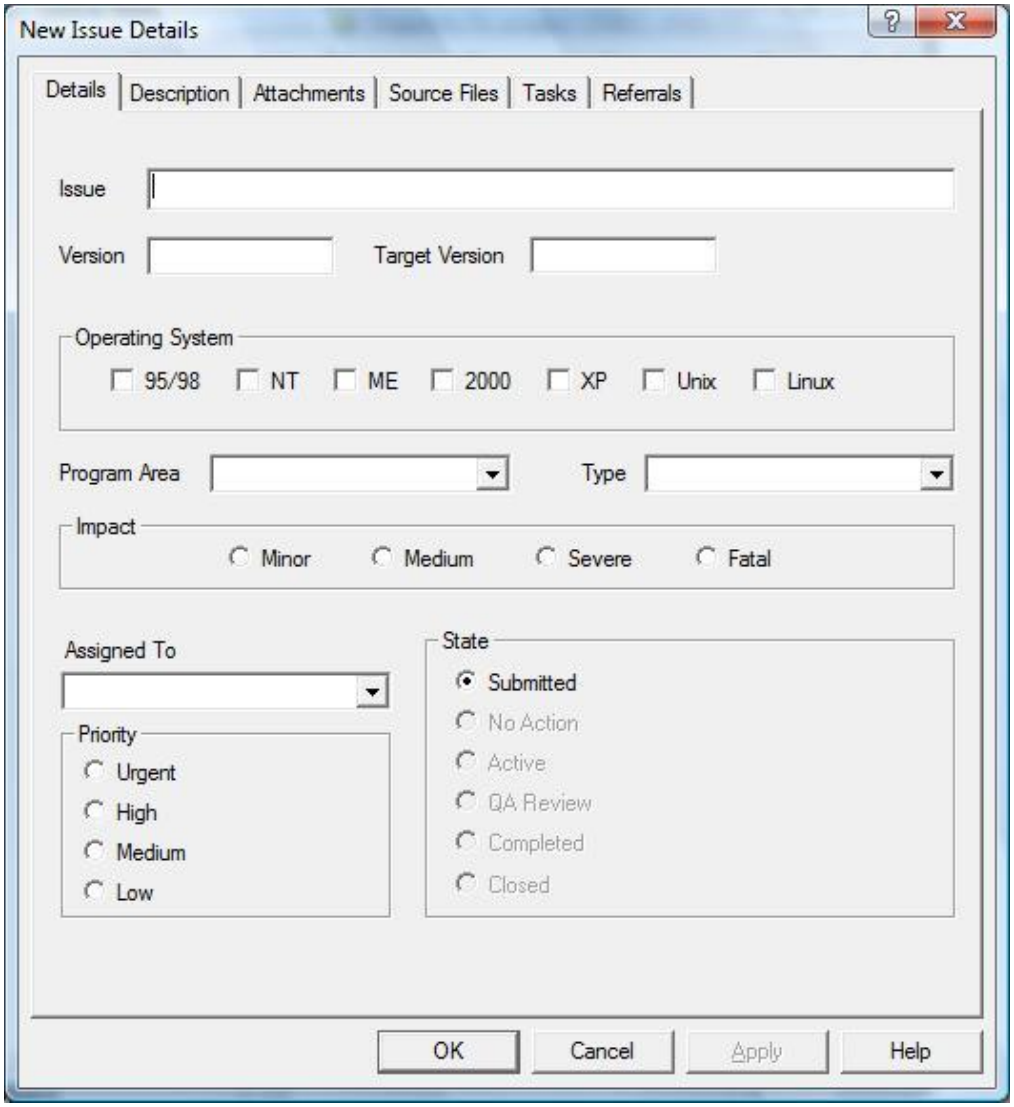


Figure 5-15 New Issue Dialog

Link to A New Polaris Issue

To create an issue and associate it with an object in your model:

Select the Project: 1 Open a model and then double-click on an object to open the **Define** dialog.

Select Links Tab: 2 Click on the Links tab of the selected object.

Select Links To Field: 3 Right-click on the **Links To** field and select **Add Issue** to open the New Issue Details dialog in Polaris. An attempt will be made to logon to Polaris using your Visible Analyst user name. If this fails, you will be prompted to enter valid Polaris credentials.

Enter the Issue Details: 4 Enter the appropriate issue information. See the Polaris documentation for details. Once you are finished, press **OK**. The new issue will be created and a link will be added to the Links To field. Click **Save** to save the changes made in the object's repository entry.

Planning Statements

See the manual chapter Strategic Planning for a description of the planning window.

SQL View Support (IntelliViews™)

Visible Analyst supports the concept of an SQL view, which can be thought of as a derived or virtual table. A view is similar to an entity in that it has a composition but the items that appear in the composition of a view must belong to other entities or be expressed based on data elements used by another entity.

An SQL view is made up of two major components: a list of column names and a Select statement that is used to filter information from the tables in the view. For each view, there is one primary select clause that can refer to any number of sub-select clauses and an optional union select clause. Each of these clauses can refer to other subselects or to unions. When building a select clause, choose the tables, columns, and joins appropriate for only that clause. Joins can be defined using a combination of key-driven joins using relationship keys and expression-based joins using pseudo SQL. All expressions and clauses are entered and displayed in a standard pseudo SQL syntax that is a functional superset of all supported SQL syntax. An expression builder is available to help in this process. When pseudo SQL is translated into a specific dialect's syntax, complex operators that are not directly supported are expanded into layers of simpler syntax to accomplish the same task. As needed, translation generates slave views to overcome dialect limitations, such as one outer join per view.

Note

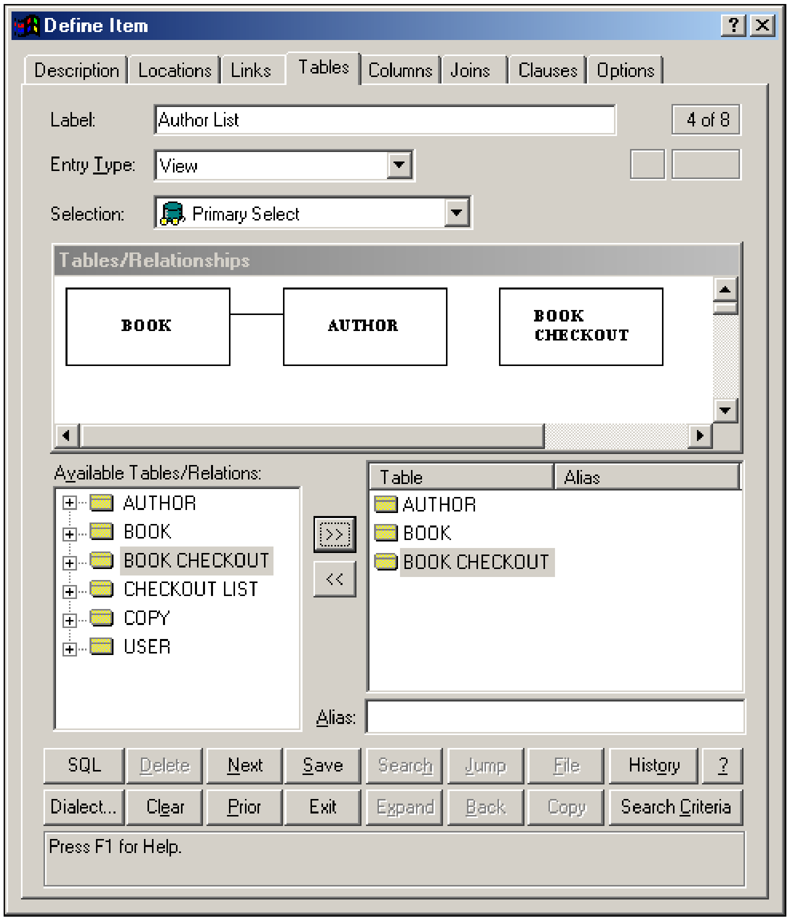
- You can also generate the view as a Create Table statement to facilitate data warehouse/data mart modeling efforts.

To add a view specification to an entity, select Define from the Repository menu. At the Define dialog box, select View from the Entity Type drop-down list to display the five additional tabs unique to view objects:

- On the Tables tab, select the tables that are to be used by the view.
- On the Columns tab, select the columns to be used by the view.
- On the Joins tab, define the join relationships to be used by the view.
- On the Clauses tab, define where, group by, having, start with, and connect by clauses.
- On the Options tab, set flags such as restrict and with check.

Each of these tabs is described in detail below. (The Description, Locations, and Links tabs for view objects are similar to those for other entry types.)

View Tables



Label: Author List

4 of 8

Entry Type: View

Selection: Primary Select

Available Tables/Relations:
- AUTHOR
- BOOK
- BOOK CHECKOUT
- CHECKOUT LIST
- COPY
- USER

Table	Alias
AUTHOR	
BOOK	
BOOK CHECKOUT	

Alias:

Press F1 for Help.

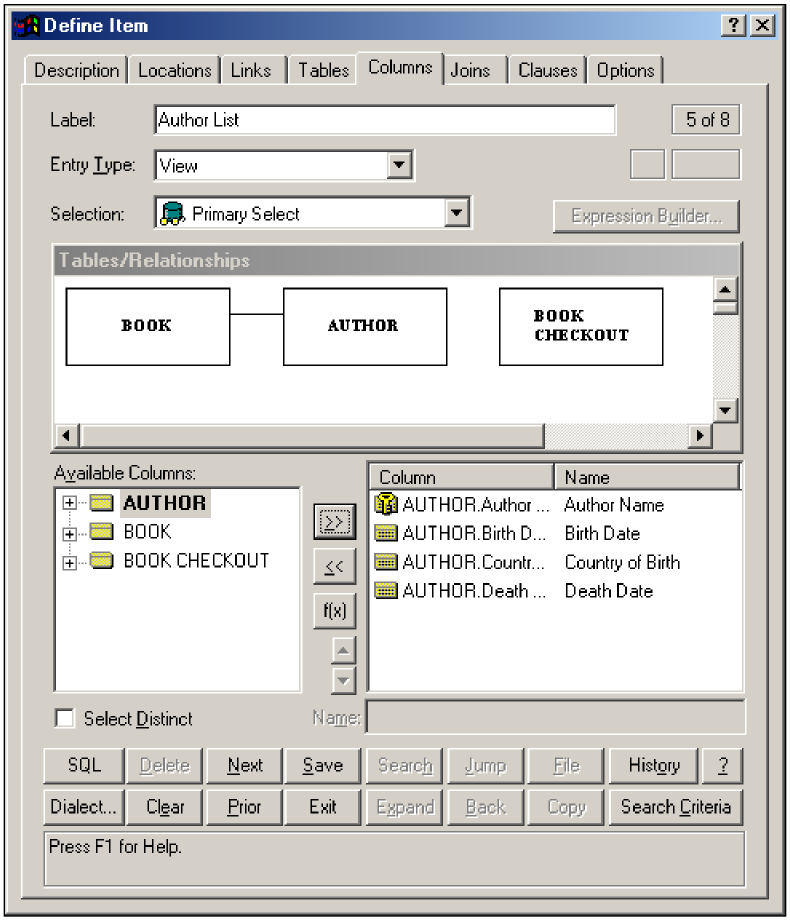
Figure 5-16 Define Dialog Box, View Tables Tab

- **Selection.** Specifies the name of the select clause that uses the tables. Each view must have one primary selection. All other selections are referred to by the Where clause of the primary select, by a union select, or by other non-primary selects (sub-select).
- **Tables/Relationships.** A graphical representation of the tables and join relationships used by the current select clause. This window is updated each time a new entity or relationship is added to the view. Once a table has been added, its position in the window can be changed by clicking the object with the mouse and dragging it to a new position.

Move (or undock) the window itself by clicking on the title bar and moving it. Once moved, the rest of the controls on the page are resized to use the available space. To dock the window, move it back to its original position on the page. To prevent docking, press the CTRL key while moving.

- **Available Tables.** A list of tables and their associated relationships that exist in the current project. Highlight a table and click the ➢➢ button to add the table to the list of tables used by the view. A table can be added more than once, but each instance must have a unique alias. If a relationship is selected, and neither table exists in the view list, both tables and a join relationship are added. If either table exists, a specific instance of the table must be specified. You can also drag-and-drop with the mouse to add tables.
- **Selected Tables.** A list of tables and their correlation names (alias) that are used by the view. To remove a table from the selected list, highlight it and click the ≺≺ button. You can also use drag-and-drop with the mouse to remove tables.
- **Alias.** Correlation name for a table. To change the alias, select a table and type the new alias in the Alias field.

View Columns



Label: Author List

5 of 8

Entry Type: View

Selection: Primary Select

Available Columns:
- AUTHOR
- BOOK
- BOOK CHECKOUT

Column	Name
AUTHOR.Author ...	Author Name
AUTHOR.Birth D...	Birth Date
AUTHOR.Countr...	Country of Birth
AUTHOR.Death ...	Death Date

☐ Select Distinct

Name:

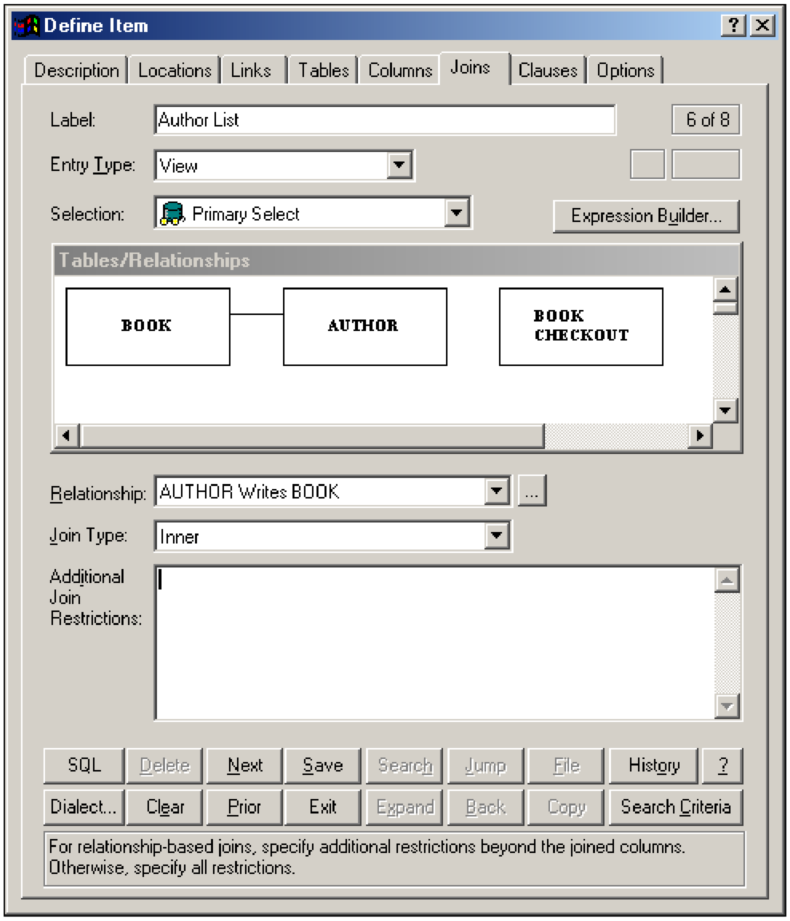
Press F1 for Help.

Figure 5-17 Define Dialog Box, View Columns Tab

- **Available Columns.** A list of tables and columns that belong to the current selection. To add a column to the list of columns used by the view, highlight it and click the ➢➢ button. To add all the columns in a table, highlight the table name and click ➢➢. A table may be added more than once, but each instance must have a unique alias. To add an expression, click the f(x) button. If the current selection is a sub-select, only one column is allowed; if it is a union, the number of columns must match the primary selection. You can also drag-and-drop with the mouse in two ways to add columns. Select either a table name or column name from the Available Columns list and drag it to the Selected Columns list, or select a column or group of columns from an entity in the Tables/Relationships window and drag to the selected columns list.

- **Selected Columns.** A list of columns and their names that are used in the view. To remove a column, highlight it and click the ≺≺ button. The change the name used by the view, highlight a column and type the changes in the Name field. To change the order of the columns, click the up and down arrows or highlight and drag to the new position. You can also drag-and-drop with the mouse to remove columns.
- **Tables/Relationships.** A graphical representation of the tables and join relationships used by the current column. When the Columns tab is displayed, you can select a column or group of columns from an entity in this window and drag it to the Selected Columns list.
- **Select Distinct.** Click this check box to indicate that duplicate rows are not to be returned by the select clause.
- **Expression Builder.** Click this button to display the Expression Builder to help you create expressions to be used in the Filter, Group By Having, Start With, Connect by, or Join Expression controls.

View Joins



Define Item

Label: Author List

6 of 8

Entry Type: View

Selection: Primary Select

Tables/Relationships

Relationship: AUTHOR Writes BOOK

Join Type: Inner

Additional Join Restrictions:

For relationship-based joins, specify additional restrictions beyond the joined columns. Otherwise, specify all restrictions.

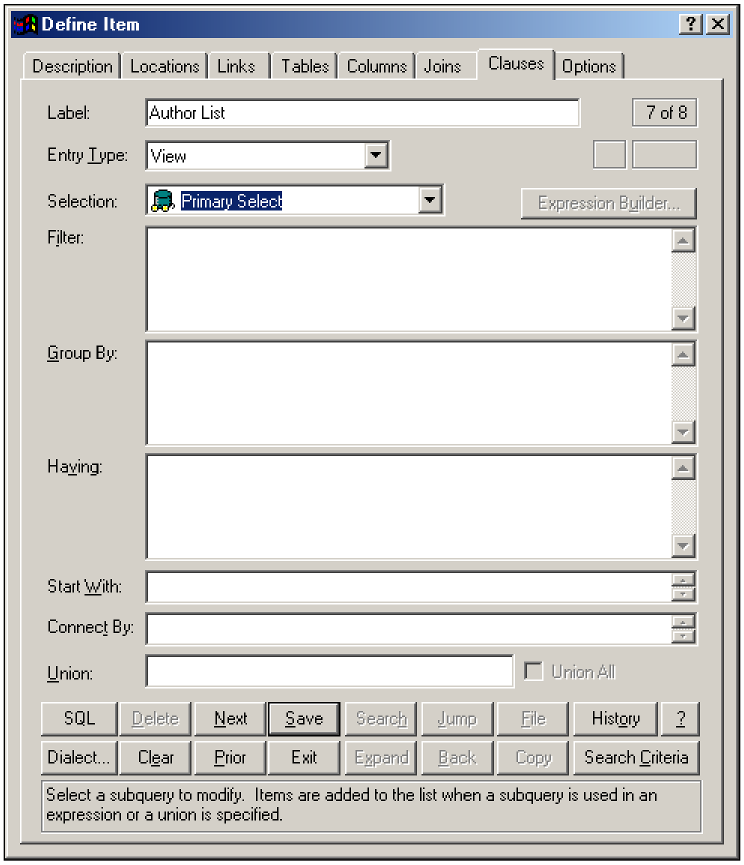
Figure 5-18 Define Dialog Box, View Joins Tab

Two types of join relationships can be defined. Key-driven joins are based on relationship keys created on an entity-relationship diagram. Pseudo SQL relationships are created simply by selecting join columns in a pair of tables.

- **Relationship**. This field lists all the join relationships for the current selection. To change the join type or expression, choose a relationship from the list or highlight it in the Table/Relationships box.

- **… Button.** Click this button to display the Join Relationships dialog box. Use the Join Relationships dialog box (described later in this chapter) to add or delete relationship-based and non-relationship-based joins.
- **Join Type**. Join types available are inner, left outer, right outer, or full outer.
- **Join Expression**. If the join relationship is not key-based, use this field to specify the join expression. If the join is based on a relationship, the Join Expression edit control is replaced by Additional Join Restrictions used to specify filtering clauses beyond the basic expression used to define the join. Click the Expression Builder button to help you define any additional restrictions. Note: This field should not be used to define a basic join expression for relationship-based joins. This information can be determined from the repository. In most cases, this field should be blank for relationship-based joins.
- **Tables/Relationships.** A graphical representation of the tables and join relationships used by the current relationship. When a join relationship is added to the view, its position can be changed by clicking the object and dragging it to a new position.
- **Expression Builder.** Click to display the Expression Builder to create expressions to be used in the Join Expression control.

View Clauses



Label: Author List

7 of 8

Entry Type: View

Selection: Primary Select

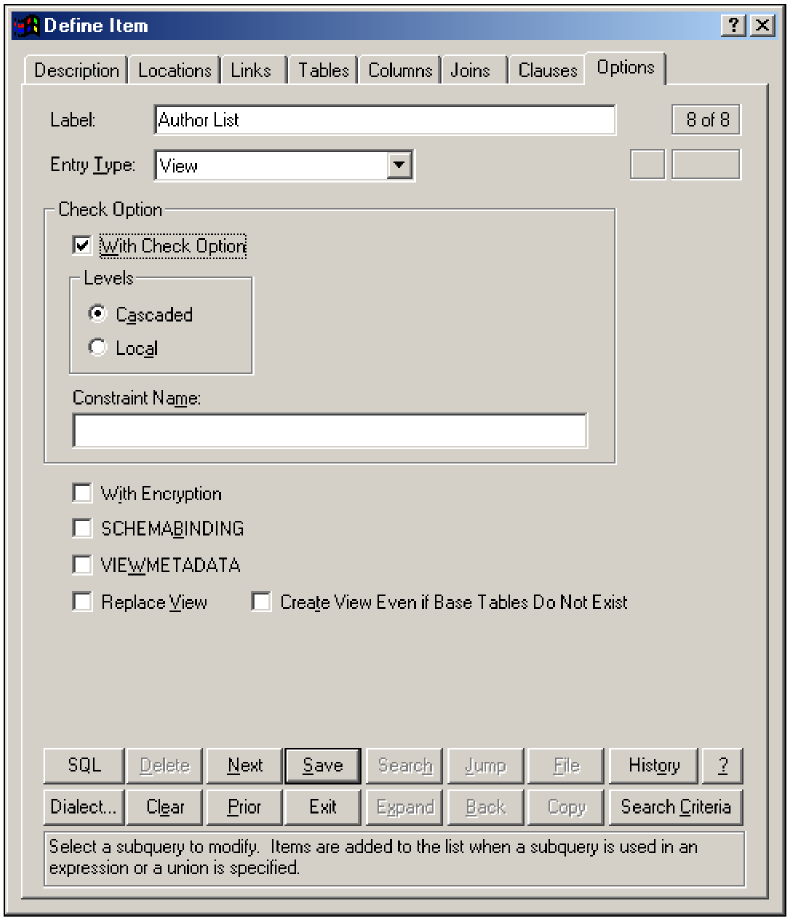
Select a subquery to modify. Items are added to the list when a subquery is used in an expression or a union is specified.

Figure 5-19 Define Dialog Box, View Clauses Tab

- **Filter.** Type a filtering condition to restrict the number of rows returned by the select. If the where clause is a sub-select, specify the name of a new selection item in the Selection field. Use the Expression Builder to create a filter.
- **Group By.** Specify grouping criteria.
- **Having.** Specify a having clause.
- **Start With.** Specify a starting column in a hierarchical search.
- **Connect By.** Specify the connect by column in a hierarchical search.

- **Union.** Use this field to indicate that the current selection has a union relationship with another selection clause. This field is valid only for top-level select clauses (primary select or other union selects). When a union is specified, it is added to the Selection list control
- **Union All.** If a union select is specified, click this check box to indicate if it should be a normal union or a union all relationship.
- **Expression Builder.** Click this button to display the Expression Builder to help you create expressions to be used in the Filter, Group By Having, Start With, Connect by, or Join Expression controls. In addition to the Expression Builder, each of the clauses control supports an "Attribute Completion" facility. If you type a table name followed by a period (.), a list of columns is displayed. To select a column, use the arrow keys to highlight an entry and then press the Tab key. If you type a period preceded by a space, a list of tables is displayed, followed by a list of columns.

View Options



Label: Author List

8 of 8

Entry Type: View

Check Option

☑ With Check Option

Levels

● Cascaded

○ Local

Constraint Name:

□ With Encryption

□ SCHEMABINDING

□ VIEWMETADATA

□ Replace View □ Create View Even if Base Tables Do Not Exist

Select a subquery to modify. Items are added to the list when a subquery is used in an expression or a union is specified.

Figure 5-20 Define Dialog Box, View Options Tab

- **Check Option.** Use the Check Options to ensure that data modification statements adhere to the criteria set within the select statement defining the view. When a row is modified through a view, the With Check Option guarantees that the data remains visible through the view after the modification.
- **Drop Behavior.** This option determines where a Drop View statement is created during SQL generation. None means do not generate the drop, Cascade means delete the view and all dependent views, and Restrict means do not drop if there are dependent views.

- **With Encryption.** Click this check box to store the view select statement in an encrypted format in the target RDBMS.
- **Create View Even If Base Tables Do Not Exist.** Base tables that a view references usually must exist before the view can be created. Click this check box to allow the view to be created regardless of whether the base tables exist or not. However, the view cannot be used until the base tables are created.
- **Expression Builder.** Click this button to display the Expression Builder to help you create expressions to be used in the Filter, Group By Having, Start With, Connect by, or Join Expression controls. In addition to the Expression Builder, each of the clauses control supports an "Attribute Completion" facility. If you type a table name followed by a period (.), a list of columns is displayed. To select a column, use the arrow keys to highlight an entry and then press the Tab key. If you type a period preceded by a space, a list of tables is displayed, followed by a list of columns.

Expression Builder
When creating expressions to be used in the Filter, Group By, Having, Start With, Connect By, or Join Expression controls, you can either enter the information manually or use the Expression Builder to help you.

To start the Expression Builder, click the Expression Builder button when it is enabled.

An expression is made up of a set of criteria used to filter information. To build the expressions, use the following controls:

- **Prefix Op.** A prefix operator used to join the criteria being defined with the rest of the expression. The options available are And, Or, or no operator. (Optional)
- **Function.** The name of a function to be used. If Exists or Not Exists is specified, a subquery name must be specified in the Subquery field. (Optional)
- **Field/Subquery.** The name of a column or subquery to be used in the expression. The drop down list contains a list of the valid column names that can be used. (Mandatory)
- **Operator.** The relational operator used to compare the Field against a Value. If a Function is not specified, this item is mandatory; otherwise, it depends on the function selected. If you want to perform a comparison with any or all members in a list or subquery, choose All or Any from the second operator control.
- **Value.** The value to compare Field against or the name of a subquery if All or Any were specified in the second operator field.
- **Add** - Once the criteria for the subexpression has been defined, click Add to update the entire expression. The Expression control contains all the criteria that have been defined so far.
- **Clear.** Clear the current criteria controls.

- **Expression.** Displays the expression-in-progress. The caret (÷) indicates where the new criteria will be added.

When the expression is complete, click OK to save it and return to the Define dialog box.

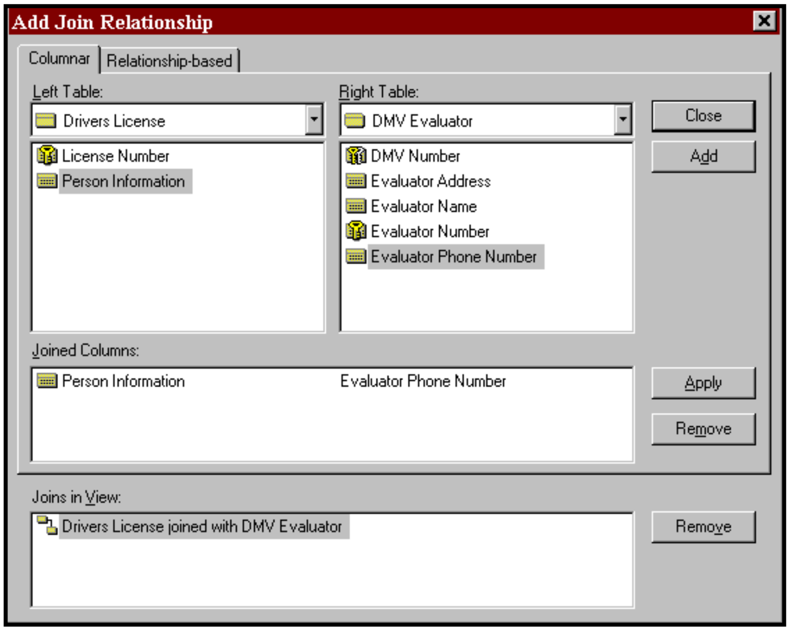
Join Relationships Dialog Box

The Join Relationships dialog box allows you to add and delete joins. To display the Join Relationships dialog box, click the button next to the Relationships drop-down button on the Define dialog box.

The Join Relationships dialog box has two tabs:

Columnar Tab

The Columnar tab allows you to add non-relationship-based joins by joining columns together. You can also delete joins from this tab.



Left Table: Drivers License

License Number
Person Information

Right Table: DMV Evaluator

DMV Number
Evaluator Address
Evaluator Name
Evaluator Number
Evaluator Phone Number

Joined Columns:

Person Information	Evaluator Phone Number

Joins in View:

Drivers License joined with DMV Evaluator

Figure 5-21 Join Relationships Dialog Box, Columnar Tab

- **Left Table.** Select a table from the drop-down list to display the columns in the selected table. Select a column from the list. You can drag-and-drop a column from one list to the other to add a Joined Columns pair.

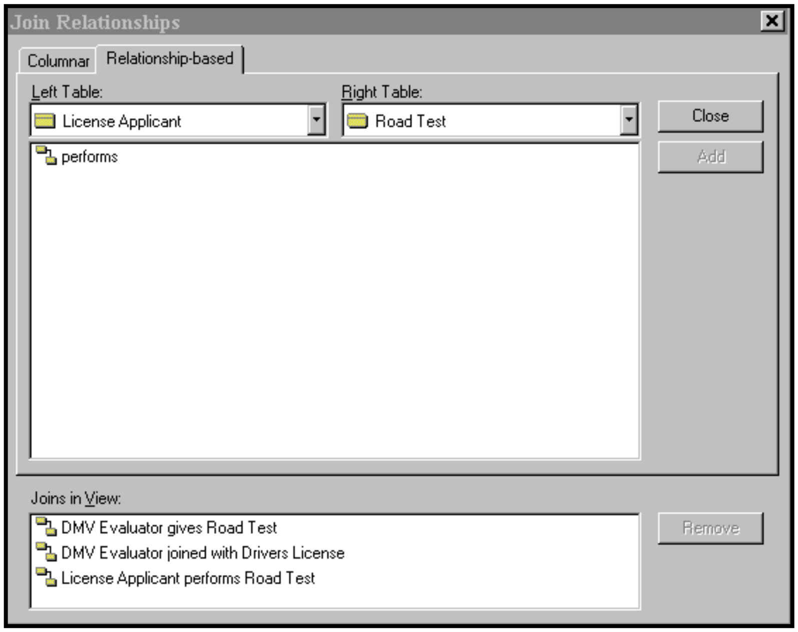
- **Right Table.** Select a table from the drop-down list to display the columns in the selected table. Select a column from the list. You can drag-and-drop a column from one list to the other to add a Joined Columns pair.
- **Add.** Click this button when you have selected a column from the left and right table list boxes to add it to the view. You can drag-and-drop a relationship from the available list to the Joins in View list.
- **Joined Columns.** The joined columns are displayed in this list box. Select a joined column pair. Click Remove to delete the joined column pair, or click Apply to display it in the Joins in View list box.
- **Joins in View.** The list of joins in the view are displayed here. Select a join and click Remove to remove the join from the view.

Note

📋 Double-clicking on either column list adds a pair of joined columns if the same name exists in the other column list.

Relationship-based Tab

The Relationship-based tab allows you to add relationship-based joins. You can also delete joins from this tab.



Left Table: License Applicant

Right Table: Road Test

performs

Joins in View:

DMV Evaluator gives Road Test
DMV Evaluator joined with Drivers License
License Applicant performs Road Test

Figure 5-22 Join Relationships Dialog Box, Relationship-based Tab

- **Left Table.** Select a table from the drop-down list.
- **Right Table.** Select a table from the drop-down list.
- **Add.** Relationships between the selected tables are displayed in the list box. Select a relationship and click Add to add it to the view. You may also double-click on a relationship or drag it to the Joins in View control.
- **Joins in View.** The list of joins in the view are displayed here. Select a join and click Remove to remove the join from the view.

Join Tables Dialog Box

The Join Tables dialog box is displayed when you add the same relationship to a view twice.

When adding a join relationship to a view, if either one of the entities involved in the join already belongs to the view, you must decide if the join is to use an existing instance of the entity or create a new instance. The Join Tables dialog box is used to supply the appropriate information.

The dialog box is divided into two parts: the upper part deals with the parent entity in the relationship, while the lower part deals with the child entity. In each case, you must decide whether an existing instance of the entity is to be used, or a new instance should be created. If the entity is not already used, you must create a new instance. If the entity already exists but you want to create a new instance, the name given must be unique.

Once you have selected the appropriate information, click OK to add the new join relationship.

Data Element Physical Information

For a data element (see description, below), physical information like data storage type and field length may be added. The repository supports the following data types for data elements and domains.

Table 5-3
Valid Storage Types

Storage Type Name	Internal #	Storage Type Name	Internal #
Binary	8	Serial	29
Bit	1	SmallDateTime	27
Char	3	SmallFloat	23
Date	24	SmallInteger	16
DateTime	26	SmallMoney	19

Table 5-3
Valid Storage Types

Decimal	20	Sysname	32
Float	22	Time	25
Integer	15	TinyInteger	17
Interval	28	Unicode Char	11
LargeInteger	14	Unicode Long VarChar	13
Long VarBinary	10	Unicode VarChar	12
Long VarChar	7	UpdateStamp	30
Money	18	VarBinary	9
National Char	4	VarBit	2
National VarChar	6	VarChar	5
Rowid	31	Zoned Decimial	21

Notes:
The column headed by *Internal#* indicates the Visible Analyst internal code that corresponds to a storage type name. If you are accessing the repository data files from a third-party tool such as Microsoft Access, this number is offset by 53.

A number following a data element type is the number of bytes used. For example:

Integer 4 uses 4 bytes
LargeInteger 8 uses 8 bytes
TinyInteger 1 uses 1 bytes

Physical Characteristics Repository Fields

Storage Type
Physical data storage types for elements can be selected from a list of valid logical types that cover all the data types used by the supported SQL dialects. During SQL generation, the logical type is translated to the native data type supported by the targeted RDBMS. If an item in the list is marked with a ^{1}, this means the logical-to-physical mapping for the current SQL dialect is wasteful. For example, if a dialect does not support a tiny integer type, it is mapped to a small integer. If an item in the list is marked with a ^{2}, this means there is not a good match between the logical type and the physical type. If you do not like the logical-to-physical mappings created, you may change them by customizing the SQL dialect. See Appendix B for details.

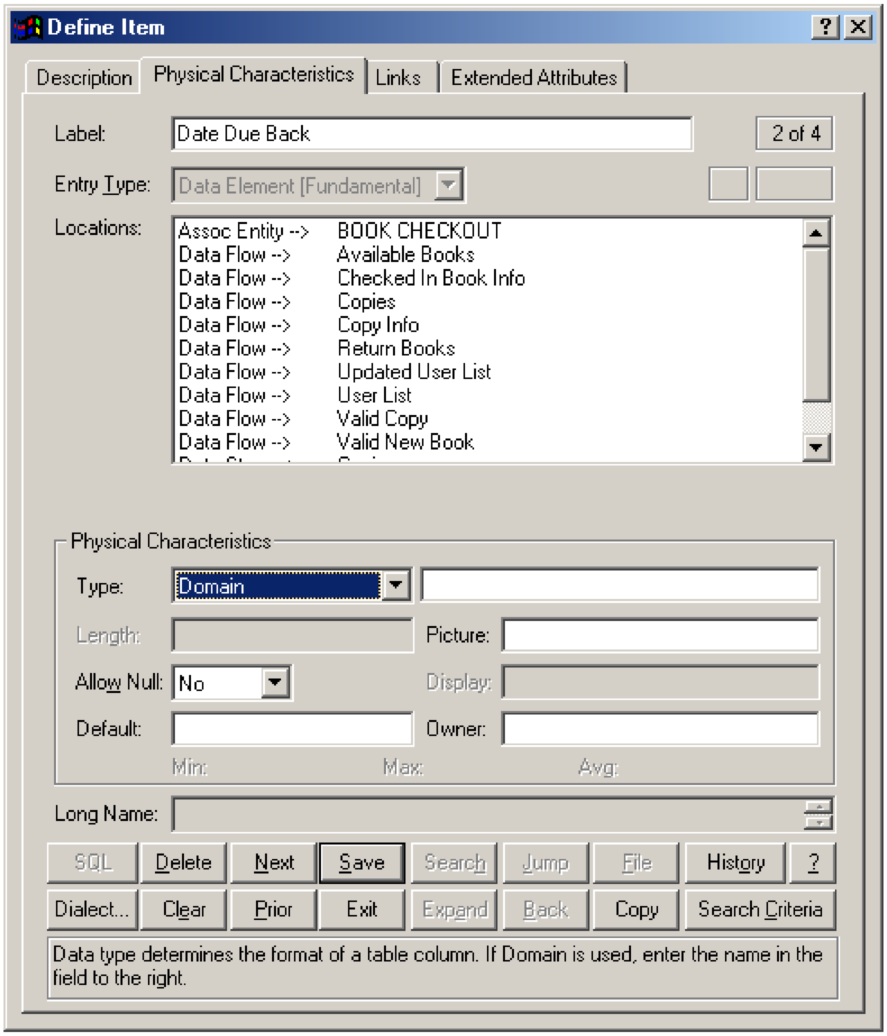
To display both the logical and physical data types in the Define dialog or printed on reports, edit the VAW.INI file and add this line: ShowPhysicalDataTypes=Yes
The display will be changed after the Visible Analyst is restarted.

Length
The length field accepts the format *digits*, *decimal digits*. If the SQL dialect type is Informix and storage type is Interval or DateTime, you can specify the range in the format *largest* to *smallest*.

Display
This field allows you to customize the display format for the data element you are defining. This field accepts either the actual mask representation of the display format or a previously defined display format object. You can define a display format as a separate object in the repository and then reference that display format in the Display field. If you do not know the name of a display format you have already defined, you can use the Search button by placing your cursor in the Display field and then clicking Search on the bottom of the repository screen. This searches the repository for all display formats defined matching the text you entered.

Picture, Default and Owner
The Picture, Default and Owner fields accept whatever text you enter into them.

Allow Null
The Allow Null field is used to indicate if an attribute is required. If yes, the attribute is optional. If Default is selected, the system-defined default value is supplied. If Identity is selected, this indicates the column has an identity constraint. You can also specify the seed and increment value in the seed control.



Label: Date Due Back 2 of 4

Entry Type: Data Element [Fundamental]

Locations:

Assoc Entity -->	BOOK CHECKOUT
Data Flow -->	Available Books
Data Flow -->	Checked In Book Info
Data Flow -->	Copies
Data Flow -->	Copy Info
Data Flow -->	Return Books
Data Flow -->	Updated User List
Data Flow -->	User List
Data Flow -->	Valid Copy
Data Flow -->	Valid New Book

Physical Characteristics

Type: Domain

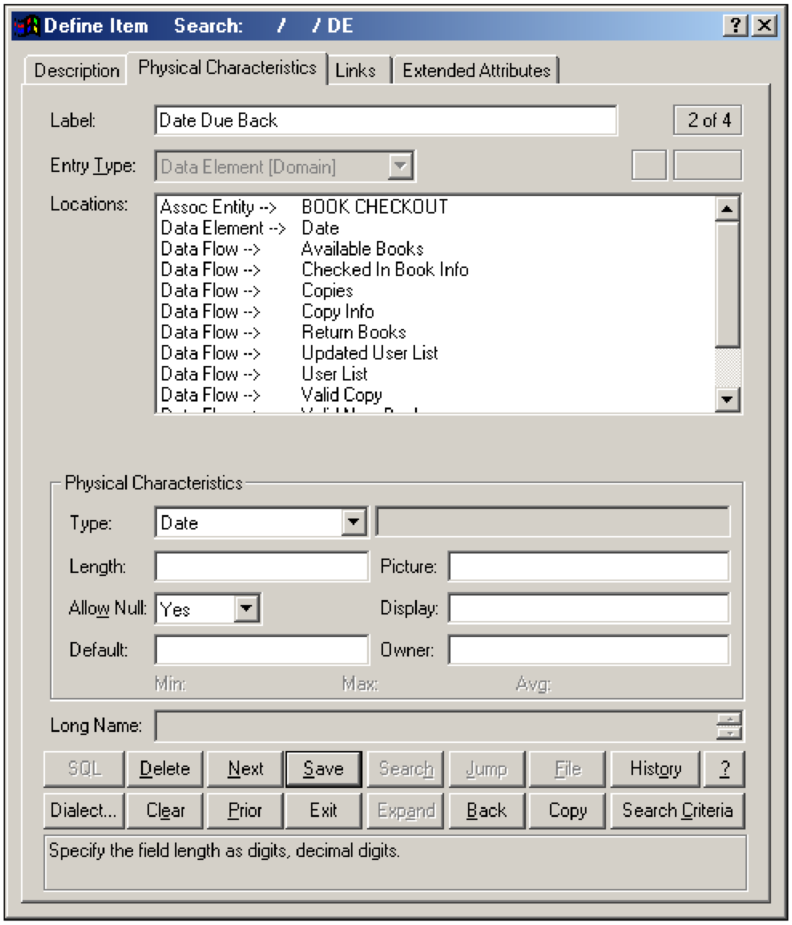
Allow Null: No

Data type determines the format of a table column. If Domain is used, enter the name in the field to the right.

Figure 5-23 Data Element Entry with a Domain

Domains

A data element can maintain physical characteristics by specifying them on page two of the element's repository entry. (See Figure 5-23.) An alternative method of specifying physical information for an element is to list the name of a domain in its storage type field. A domain indicates the physical characteristics of a class of items. If the domain named does not yet exist, an entry in the repository is created. (If you like, you can now Jump to the new domain and enter information about it.) The details of the physical information can be specified on page two of the domain entry. (See Figure 5-24.) Multiple data elements can reference a domain. If a domain is used, the storage type in the domain's repository entry can be selected from a list of items that are already defined in the repository. If a data element references a domain, the Length field is not accessible; it should be set in the domain entry.



Label: Date Due Back   2 of 4

Entry Type: Data Element [Domain]

Locations:
Assoc Entity --> BOOK CHECKOUT
Data Element --> Date
Data Flow --> Available Books
Data Flow --> Checked In Book Info
Data Flow --> Copies
Data Flow --> Copy Info
Data Flow --> Return Books
Data Flow --> Updated User List
Data Flow --> User List
Data Flow --> Valid Copy

Physical Characteristics
Type: Date
Length:
Picture:
Allow Null: Yes
Display:
Default:
Owner:
Min: Max: Avg:

Long Name:

Specify the field length as digits, decimal digits.

Figure 5-24 Repository Domain Entry

Selecting a Domain

The repository Search function can be used to select a domain for a data element. To do this, display page two of the element's repository entry. In the Type box, select Domain. Place the cursor in the domain name edit box to the right of the Type box and click the Search button to display the repository search dialog box. In the Search list box, locate the domain you wish to use and click the Search button; the item is added to the Select list box at the bottom. Then click the Select button and the domain label is added to the element's repository entry.

Local Physical Characteristics
If a data element is derived from a domain, you can override certain characteristics, excluding type and length. If the value is changed, the text appears blue to indicate this value is different from the base domain. When SQL is generated, the local value is used instead of the value defined in the domain.

Other Physical Characteristics Information

If physical information for a data element is defined, its Entry Type field cannot be changed.

Extended Attributes

A data element can maintain extended physical characteristics by specifying them on page three of the element's repository entry. You can then transfer these extended attributes and all data model information you have defined in the repository to Powersoft's PowerBuilder application development system or Unify's VISION or Visual Basic. Conversely, data model information can be extracted from a PowerBuilder database thus populating the Visible Analyst repository with all appropriate tables, columns, physical characteristics, extended attributes, primary and foreign keys and validation rules.

There are several repository fields used to define a column's extended attribute information. These extended attributes are PowerBuilder/VISION-specific information that enhance the definition of a column. The extended attribute screen is located on page three of an element definition in the repository.

Header and Label
These fields contain the text to describe and identify columns and are used in the DataWindow object. You can also specify the justification for the header field using the drop down list box provided.

Edit Style and Type
These fields specify how column data is presented in the DataWindow objects. The Edit Style field contains the name of the style. The type of edit style for the column is defined in the Type drop-down list box. Currently nine edit style types are supported. For each style there are appropriate flag settings that appear at the bottom of the extended attribute screen, as well as other edit fields depending on the type of edit style you selected. Refer to the appropriate documentation for more information.

If you have specific display values that the user would see and data values that are stored in the database, these can be defined in the Values & Meanings field of the data element. The list of values should always start with BEGIN PBLIST and end with END PBLIST. Following

BEGIN PBLIST is the actual list of display values in quotes and the data values in quotes, separated by a comma. For example if you had a list of names defined as a list box and wanted to add the specific values seen by the user and the values stored by the database, the entry in the Values & Meanings field for the element would be as follows:

```
BEGIN PBLIST
        ‘Paul Newman’,’PN’
        ‘Bing Crosby’,’BC’
        ‘Sylvester Stallone’,’SS’
END PBLIST
```

User-Defined Fields

The Visible Analyst repository is extensible. New fields, called user-defined, can be defined in a project repository allowing you to extend the information that can be stored with an object. The field information that can be stored includes name, type and length.

Adding New Repository Fields to a Project

To add fields to a project:

Select Define User Attributes: 1 Select Define User Attributes on the Options menu. The Define Repository Attributes dialog box appears.

Select New or Existing Project: 2 Click New Project Default or Current Project.

- New Project Default. An attribute is defined for all subsequent project creation operations.
- Current Project. An attribute is added only to the current project. If a project has not been selected, this option is disabled.

Set Field Name: 3 Type the name given to the field. The name can be up to 128 characters, but must begin with a letter. If you want a user object to begin with a number, edit the VAW.INI file and add this line: AllowNumericPrefix=Yes. The change will be implemented when you restart the Visible Analyst.

Set Field Type: 4 Any valid Visible Analyst-supported field can be used. Move your mouse to the drop-down arrow button at the right and click with the left mouse button for a list of supported types.

Set Field Length: 5 Type the size of the field length and scale, where scale is optional indicating the number of digits to the right of the decimal point.

Save the Field: 6 Click Add to save your changes. If New Project Defaults was selected, every new project that is created contains the field. If Current Project was selected, only the current project is updated.

After the attributes are defined and a new project created, you can access the field(s) from the repository definition for each object by choosing one of the two paging buttons located on the repository dialog box at the bottom right of the screen. Each user-defined attribute is represented in Visible Analyst as an entry in the User-Defined Attributes list on page three.

Note

- Once these new fields are defined, you must create a new project for access to these fields if you selected New Project Defaults. They ARE NOT accessible with the currently selected project unless you select Add to Current Project or you choose the Current Project option in Step 2. Another important note: if using a network version of Visible Analyst, every user-defined object is accessible with any subsequently created project. If you create a 'date created' attribute field, anyone else with access to Visible Analyst also sees that 'date created' attribute field for objects in any project they create.

Changing or Removing Repository Fields

To change or remove fields:

Select Define User Attributes: 1 Select Define User Attributes on the Options menu. The Define Repository Attributes dialog box appears.

Select New or Existing Project: 2 Click New Project Default or Current Project.

- New Project Default. An attribute can be changed or removed from the list of attributes created when a new project is created.
- Current Project. An attribute can be removed from the current project, but the definition cannot be changed.

Select the Field: 3 Select the attribute to be changed from the list.

Change or Remove Field: 4 If the attribute is to be changed, make the appropriate changes and then click Change. If the attribute is to be deleted, click Remove. If you choose to remove an attribute, the repository of the current project is modified. This may take several minutes depending upon the size of the project.

User-Defined Objects

You can define your own object types, making the Visible Analyst repository fully extensible. This feature allows user-defined objects to be linked to any repository object and reference other objects within the composition field of the user-defined object. You can then produce an association matrix report depicting the link between objects. An applicable example would be a requirement entry type. The requirement objects you define could then be linked to a process(es) that satisfies the requirement. Other examples would be critical success factors, business rules, goals or test plans.

To create a user-defined object:

Select Define User Objects: 1 Select Define User Objects from the Options menu. The Define Repository Objects dialog box appears.

Select New or Existing Project: 2 Click either New Project Defaults or Current Project.

- New Project Defaults. Choose this option to modify the list of objects added to all new projects.
- Current Project. Choose this option to modify the list of objects defined for the current project. If this option is selected, objects can either be added or removed, but the definition of existing objects cannot be changed.

Set Object Name: 3 This is the name of the new entry type being created. Whenever you define an object, this name appears in the list of entry types on the initial repository definition dialog box.

Set Field Name: 4 This name is associated with the description field of the new entry type in the repository. Visible Analyst automatically enters the name of the new entry type as defined above, followed by the word Description, in this dialog box field. For example, if you defined a new object type called Requirement, as you enter the new object type in the Name field, you notice the name Requirement Description is automatically entered in the Field Name.

Now when you open the repository and define an object of type requirement, you see the content of Field Name listed as an actual dialog box field within the repository. You can override this naming convention and type your own name if you choose.

Set Composite Type: 5 If you choose this option by clicking your left mouse button on the check box labeled Composite Type, you allow any newly defined objects of this entry type to reference other repository entry types in the Field Name field of the repository, as described above. For every item entered into this field a new repository entry is created, if one does not already exist, and the location reference for this entry is updated. See the Composition section earlier in this chapter for more details. If Composite Type is not checked, the defined Field Name field is considered a free form text field with a 64K character limit.

Select the Composition Types Contained: 6 If you defined an object as a composite type, you are given the choice of the entry types you want to include. You may choose a single type, group type, or all types. Click the drop-down arrow button for a list of types. If you choose all, the default creation type for an item entered into the composition field is a data element.

For example, if you define a user-defined object called database, and you would like to reference any entity objects the various database objects include in the composition, choose Composite Type by clicking the check box provided and choose All entity types in Contains. You can now reference all the entities that comprise a particular database object in the Field Name field of the object repository definition. This new field is located on the initial definition screen for the user-defined object you are defining.

Select the Link Option: 7 This option allows you to establish a link or series of links between repository objects and user-defined objects. To choose this option, click the left mouse button on the check box.

Choose the Link Field Name: 8 This field is used to enter a descriptive name for the link field in the repository. This descriptive name appears as a field in the repository for any object type you define in the Link To: box below. In this field you choose the user-defined object(s) that the object you are defining is linked to.

Set the Link Cardinality: 9 This list box describes the cardinality between any linked objects. The default setting is 1:1. This means that only one user-defined object can be linked to one repository object. There are four cardinalities allowed. The entry to the left of the colon always refers to the user-defined object. The entry to the right always refers to the linked repository object.

Set the Link To: 10 This field describes the repository types that can be linked to the user-defined object. The default is All. You can click the drop-down arrow for a list of options.

Save the Object: 11 Click Add to save the object.

Note

- If you modify the New Project Defaults list, you must create a new project in order to implement newly defined User-Defined Objects. They ARE NOT valid with the currently selected project unless you select Add to Current Project or you choose the Current Project option in Step 2. Another important note: if using a network version of Visible Analyst, every user-defined object is accessible with any subsequently created project. If you create a requirement object, anyone with access to Visible Analyst also has that requirement object type in any project they may create.

Other Repository Information

On-page and Off-page Connectors

Structure chart on-page and off-page connectors are in the repository, although they are invisible to you.

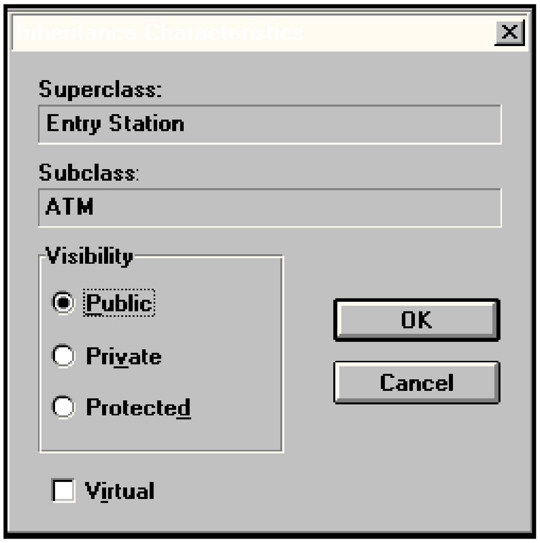
Creation and Modification Dates

Creation and modification dates are maintained for all items in the repository. As items are modified, the latest-change date reflects these modifications. Note that all of these dates are reset to the current date whenever a project is rebuilt.

Inheritance Relationship Characteristics

When an inheritance relationship exists between two classes, you can identify the characteristics of the relationship by highlighting the desired relationship in either the Attached Classes or Relations field and selecting Change from the repository object menu (activated by clicking the right mouse button over the appropriate relationship). For each relationship, the following information can be defined (see Figure 5-25).

- An inherited class with a *public* derivation means that the public members of the base class are public members of the derived class, and protected members of the base class are protected members of the derived class. *Private* derivation means both the public and protected members of the base class are private members of the derived class. *Protected* derivation means both the public and protected members of the base class are protected members of the derived class. The default is public.
- **Virtual.** A virtual base class can act as an indirect base class more than once without duplication of data members. This is useful for classes that are derived from multiple base classes.



Inheritance Characteristics

Superclass: Entry Station

Subclass: ATM

Visibility

● Public

○ Private

○ Protected

□ Virtual

Figure 5-25 Inheritance Characteristics

CLUSTERS

A cluster is an object composed of entities. Its sole purpose is to be displayed on a data model view instead of a group of entities in order to eliminate detail and bring clarity to an otherwise complex view. This section explains the techniques for working with clusters within the repository. The Drawing Diagrams chapter explains the diagramming aspects of working with clusters.

Creating Clusters

A cluster can be composed of entities already existing within the current project repository. To create a new cluster, display a blank repository screen and enter the name of the cluster in the Label field and set the Type field to Cluster.

In the Composition field, you can manually enter the names of entities you know are not members of any other clusters, or you can use the repository search function to help you populate the cluster. To do this, place the cursor in the Composition field and click the Search button. The repository is scanned and the search box opens displaying all entities that are in the free entity pool (not currently a member of a cluster). Move through the list and select an entity to be a member of the cluster. You can select multiple entities by double-clicking on them; they appear in the selection list at the bottom of the dialog box. You can eliminate an entity from the select list by highlighting it and clicking the Delete button. When you are finished, click the Select button to add them to the cluster Composition field.

After adding any other description information for the cluster, click Save and the cluster is created.

Modifying Clusters

Entities may be added to or deleted from a cluster. To delete them, highlight them in the cluster Composition field and press the DELETE key. Entities deleted from the cluster are returned to the free entity pool and are available to become members of other clusters. To add entities to a cluster, follow the procedure outlined in Creating Clusters above. You may modify the name of a cluster at any time by changing the Label field of the repository entry for the cluster. Clicking Save makes your changes permanent.

Deleting Clusters

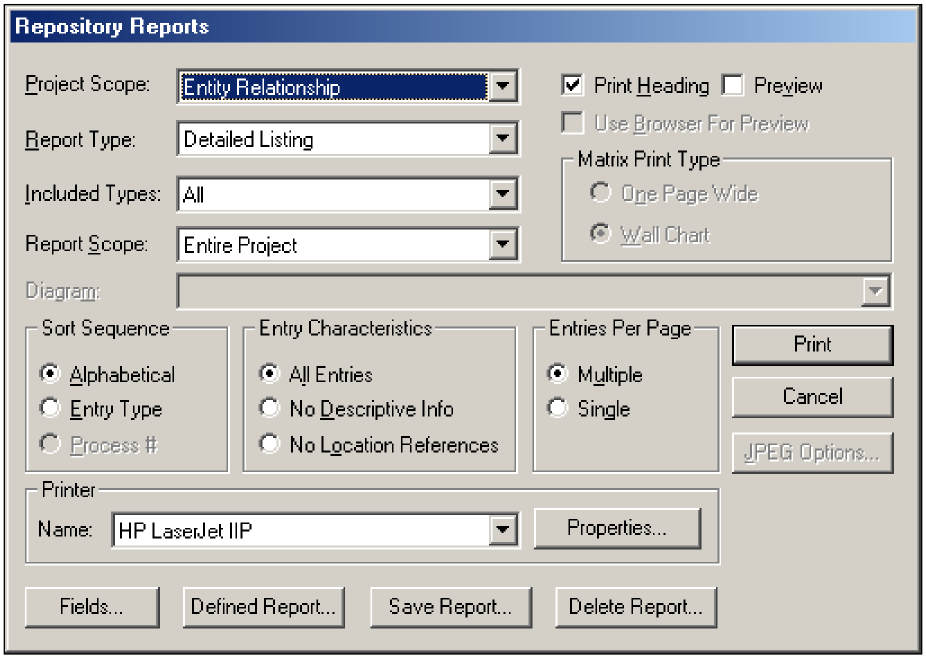
A cluster is deleted like any other repository item, by clicking the Delete button in the repository dialog box when the cluster entry is displayed. When you delete a cluster, all of its entities are returned to the free entity pool and are available to become members of other clusters.

REPOSITORY REPORTS

A variety of reports on repository data can be generated. The output may be sent to a printer or to a file, whichever you have configured in your Windows Control Panel. If you are running Visible Analyst over a relational database engine, such a Centura SQLBase, third-party report writers can be used.

Report Functions

When you select Reports from the Repository menu the repository report dialog box is displayed, allowing you to perform either or both of the following:



Repository Reports

Project Scope: Entity Relationship

Report Type: Detailed Listing

Included Types: All

Report Scope: Entire Project

Diagram:

☑ Print Heading ☐ Preview

☐ Use Browser For Preview

Matrix Print Type
○ One Page Wide
● Wall Chart

Sort Sequence
● Alphabetical
○ Entry Type
○ Process #

Entry Characteristics
● All Entries
○ No Descriptive Info
○ No Location References

Entries Per Page
● Multiple
○ Single

Printer
Name: HP LaserJet IIP

Figure 5-26 Repository Report Dialog Box

Print or Define a New Report

You can define a report's content and format. The report definition (or format) is created by selecting the various criteria in the reports dialog box. The content may be defined as a detailed listing, summary listing, cross reference, single entry listing, related to listing, split flows listing, or one of several matrix reports. The different report types are more fully described in the sections below.

Reports can be defined to include information for the entire project, a project branch (a diagram and all of its child and grandchild, etc., data flow diagrams), or for a single diagram. The report may also be defined to print from only one diagram type. Reports may be printed in alphanumeric order, in entry type order, or in process number order. You can specify that all entries, only entries with no descriptive information, or only entries with no location references be included in the report. In addition, reports can be made on specific entry types or entry types with certain characteristics (undefined, for example).

A report may also include all standard entries (all those other than miscellaneous), all miscellaneous entries, or be limited to the specific entry types defined for the given diagram type.

Repository Report Preview

The content of a repository report can be previewed before the report is actually printed. In the Repository Reports dialog box, check the Preview box. After you click Print, a box appears containing the report, and you can view the report on screen. You can also print it, save it to a file, or abandon it. If you want the printed report to be formatted, you must return to the Reports dialog and turn off Preview.

You can generate the report in HTML format so that it can be viewed in a browser. If you have a browser installed, when you click Preview, the Use Browser For Preview option is available. If you select this option, the report is displayed in your browser when it is generated. You can do this for both standard reports and matrix reports. In the generated HTML report, multiple entry type listings include hot links that allow you to "jump" around the report from entity to element, back to entity, etc. When you select the Locations-Diagrams field from the Field in the Reports dialog box, the HTML report generated includes diagrams in JPEG format. You can click the diagram listing hot links in the HTML report to display the JPEG diagram file in your browser.

Selecting Fields to Include in Reports

You can select the repository fields that are included in a detailed or single-entry report. Click the Fields button to display a list of fields to include. By default, all fields are included so the entire list is highlighted. To remove a field from the list, click the right mouse button on it. The Invert button selects all fields that are not highlighted and deselects all those that are highlighted.

Printer Setup for Reports

You can select the printer to use by clicking the down arrow by the Printer Name field and selecting a printer from the list. You can also change settings for the printer driver, such as page orientation (portrait or landscape), print to file, etc., by clicking the Properties button on the Reports dialog box.

Predefined Reports

Saving a Predefined Report

After you define a report format, you can save the report definition by clicking the Save Report button. If you choose to save it, you must name it. The name allows you to subsequently print repository data according to your predefined report content. For this and the other predefined report functions, you can scan your predefined reports and display the report details that are defined for each in the predefined reports dialog box.

Selecting a Predefined Report to Print

Clicking Defined Report allows you to print any repository report according to your predefined format. After you make this selection, you can select the name of the predefined report from the list. After you choose the report, you must indicate the specific branch (data flow diagrams only), or single diagram that is to be used as the scope of the report, unless the predefined report has a scope of the entire repository. The branch or diagram name is not saved with the other report criteria. It is done this way to give you the flexibility to save one format usable for numerous specific diagrams or branches.

Deleting a Predefined Report

You can also delete predefined reports by clicking Delete Reports, choosing the name of the predefined report that you wish to delete, and clicking OK.

Repository Report Formats

Detailed Listing

Detailed Listings prints repository data from all information fields in the repository, as well as certain information captured by Visible Analyst but not displayed in the repository dialog box. In addition to the information from the dialog box, Detailed Listings provides the following information:

For Data Flows	Source and destination information for each occurrence of the data flow (entry type and label).
For Processes	Both input and output data flows to and from the process listed by name.
For Structure Chart Symbols	Called-by and calls information, including passed and returned couples.
For Structure Chart Couples	Returned-from and Returned-to information.
For Gateways	The Gateway type.
For Events	The Event type.
For Sequence Flows	The Condition Type, Condition Expression and Quantity.
For All Items	Date created and date last altered.

Summary Listing
This report type prints only the repository data in the label, entry type, and description fields. A summary listing may include all the same report contents that a detailed listing report has, including process, file, module, data element, etc.

Cross Reference
For each data element, couple, data structure or entity, this report lists all entries that reference it in a composition field and the referencing item locations.

Single Entry Listing
This selection allows printing a detailed report for a single repository entry. After making this selection, you must choose the name of the desired repository entry for which the detailed report will be printed. The Search button displays a Search Repository dialog box and works in the same way described earlier for displaying and editing repository items.

Related To Listing
This listing reports on all structure chart items in a project repository that have been related back to data flow diagram items and the items to which they have been related.

Entity/Attributes Detailed Listing
This report is similar to a detailed listing, except the composition field includes physical characteristics of the component items. Each component appears on a separate line and includes type, length, allow null, and key information.

Entity/Attributes Summary Listing
This report is similar to a summary listing except that entity types include the composition field with physical characteristics of the component items. Each component appears on a separate line and includes type, length, allow null, and key information. If only entity types are included in the report, the description of each component follows the physical information.

Business Rules Report
This report lists instances of an entity, the relationship name, and cardinality when the Project Scope is set to Entity Relationship and you select Business Rules as the Report Type.

Split Flow Listing Generation
This report lists all parent and subflows for a data flow diagram, a branch (a diagram and its children/grandchildren), or a complete DFD portion of a project.

Statement Hierarchy
This report lists all the planning statements in the repository in hierarchical order.

Statement Outline

This report lists all the planning statements in the repository in an outline format listing the hierarchical order of the statements and the statement type..

Matrix Reports

Matrix reports generates information about one class of items with respect to those in another class, laid out as a matrix. For example, you can produce a report of what data stores occur on which diagrams or what data elements are listed in the composition fields of which entities.

There are three types of matrix reports.

- Diagram location matrices show on which diagram or view certain objects appear.
- Composition matrices reveal what items are listed in the composition fields of which other items. The item content is similar to what appears in a cross reference report, but the layout is directed more specifically toward item correspondences, without additional information about each item.
- Association matrices are more general and display the class of objects that has a certain type of association with objects of another class. For example, you can produce a report showing which entities in the data model of your project are accessed by which processes in the process model.

The content selection for each report type is dependent upon the diagram type currently in effect (set in the report Project Scope box). For certain association matrices, inverted matrices with identical contents can be generated by producing the same report under a different diagram type. When entire repository reports are in effect, some subclasses of items of objects (e.g., library modules) cannot be selected, only the general type of object (e.g., all modules).

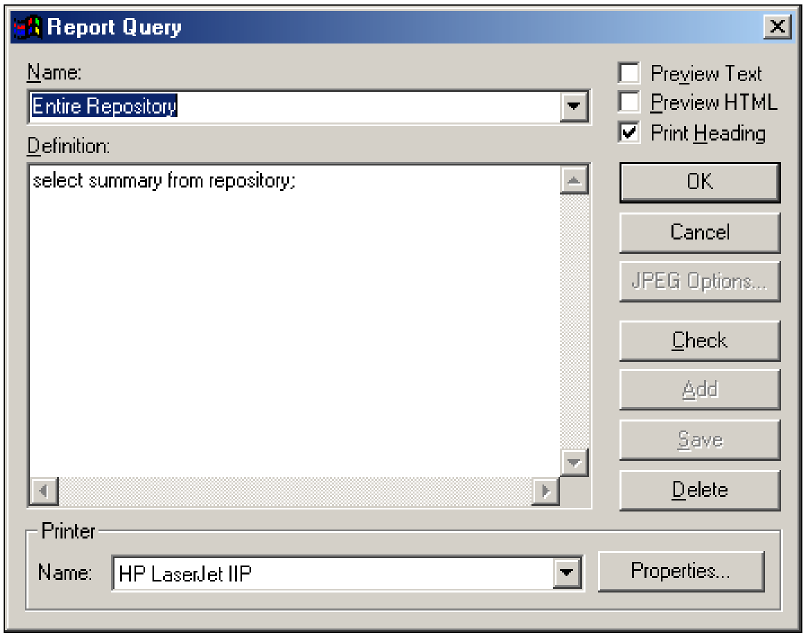
There are two matrix report formats that can be chosen from the reports dialog box:

- A wall chart matrix is physically as large as necessary to accommodate the data in the report. Visible Analyst automatically segments it horizontally and vertically. You must cut and paste the pages to produce the final displayable report chart.
- A one-page wide matrix is more suitable for binding. All rows of the report are printed over as many pages as necessary, but each page displays only as many columns as fit on one page. After all rows are printed, the report starts over, printing all rows and the next group of unprinted columns. This process repeats until all columns are printed.

You can preview a matrix report by clicking the Preview check box. You can also generate the report in HTML format so that it can be viewed in a browser. If you have a browser installed, when you click Preview, the Use Browser For Preview option is available. If you select this option, the report is displayed in your browser when it is generated. You can do this for matrix reports and standard reports.

REPORT QUERIES — CUSTOM REPORTS

The custom report facility allows you greater flexibility in defining reports. On the Repository menu, select Report Query. You enter the name of a report, select a report from the list of the available custom reports in the custom report file “REPORTS.TBL,” or manually enter the specification for the custom report in the Report Query Definition dialog box. When you finish defining the new report, click the Add button to enter it into the custom reports file. You can change an existing report query by selecting it from the list, editing it and clicking the Save button to put the changed version into the custom reports file. If you select an existing report query from the list and click the Delete button, it is removed from the custom reports file. When you complete editing, click OK to generate the selected report.



Name: Entire Repository

Definition:

```
select summary from repository;
```

□ Preview Text
□ Preview HTML
☑ Print Heading

Name: HP LaserJet IIP

Figure 5-27 Repository Custom Report Dialog Box

Custom Report File Format

The custom report file is an ASCII file that contains custom report definitions. The file is in a free format. The text is made up of “tokens;” that is, keywords separated by spaces and tabs. Carriage returns count as spaces. The keywords are not case sensitive. Comments are introduced by two dashes. The format somewhat resembles SQL.

After adding a new report or changing one, there is a check function, described below, that checks the syntax of the custom report file. Examples are given later in this section. Additional examples are in the “REPORTS.TBL” file included with Visible Analyst.

Custom Report Definition Syntax

The notations used in the report definition syntax are:

- Items marked in **bold face** are required.
- Items within square brackets (“[ ]”) are optional.
- Curly braces (“{ }”) enclose a list of items separated by the “|” character (logical OR); you must choose one.
- Parameters are delimited by “%” characters.
- Italicized items are explained later.
- Extra spaces and tabs may be added as you wish to improve legibility.

Each report definition in the custom report file has the following syntax:

Syntax	**Comments**
report "*report name"*	
params %*parameter name*% [**...**]	clause is optional
select { * | **detail** | **summary** | column name list } [from repository]	• is the same as **detail** (default)
where *restriction*	clause is optional
order by { **name** | **type** [(*entry type list*)] }	clause is optional, default is **order by name**
format { **multiple** | **single** }	clause is optional, default is **multiple**
;	

The report definition ends with a semicolon or with the end of file. The *report name* you enter appears in the custom report Name box as a selection. For each *parameter name,* the report program prompts you for the parameter value before the report executes. The *parameter name* can be used in the where clause text as a string, integer or date constant.

The select clause is used to specify the report type – detail or summary; the default is detail. The optional from repository clause is for decoration purposes only.

The optional order by clause allows you to select the sort order; sorting by name is the default. When sorting by type is selected, an *entry type list* can be specified in parentheses. The *entry type list* allows you to define a custom sorting sequence of entry types. It also serves as an additional restriction – only entries of types on the list go into the report. The list of acceptable types is the same as the valid entry types listed later in this chapter.

The optional format clause gives you the choice of printing multiple entries per page or a single entry on each page; the default is multiple.

The default *restriction* is no restrictions – the entire repository is used. The *restriction*, if entered, is a set of logical conditions concatenated using the logical operators NOT, AND, OR, and parentheses. Each logical condition applies to one repository "column." The available columns are shown in Table 5-4.

Table 5-4
Repository Column Information

Col. #	**Column Name**	**Column Type**	**Comments**
1	name	String	
2	type	Set	
3	description	String	
4	notes	String	
5	composition	String	not for data elements, domains, couples, etc.
6	location	Location	
7	creation date	Date	
8	modification date	Date	
9	process number	String	for processes only
10	data store number	String	for G&S data stores only
11	storage type	Set	for data elements and domains
12	domain	String	for data elements only
13	length	Integer	for data elements and domains
14	picture	String	for data elements and domains
15	default	String	for data elements and domains
16	allow null	String	for data elements only
17	owner	String	for data elements only
18	from entity	String	for relationships only
19	to entity	String	for relationships only
20	reverse relationship	String	for relationships only

Table 5-4
Repository Column Information

21	parent flow	String	for data flows only
22	alias	String	for non-structure chart items
23	associator element	String	for relationships only
24	values & meanings	String	for data elements, domains, couples
25	related to	String	for structure chart items
26	IC data module	String	for info clusters only
27	IC modules	String	for info clusters only
28	module description	String	for modules only
29	process description	String	for processes only
30	function description	String	for functions only
31	scenario	String	for use cases only
32	guard condition	String	for messages and events only
33	message details	String	for messages only
34	action expression	String	for events only

The logical conditions allowed for a column depend on the *column type*. For columns of type *string*, the allowed conditions are:

column **is** **null**
column **is not** **null**
column **=** “constant”
column **begins with** “constant”
column **contains** “constant”

where the constant is a string and **null** means an empty string.

For columns of type *date*, the allowed conditions are:

column = constant | **current**
column > constant | **current**
column < constant | **current**
column <= constant | **current**
column >= constant | **current**
column != constant | **current**

The date constant must be in the Windows date format as defined in Control Panel/Regional Settings/Date.

For columns of type *integer* the allowed conditions are:

column	**is**	**null**
c*olumn*	**is not**	**null**
column	=	constant
column	>	constant
column	<	constant
column	<=	constant
column	>=	constant
column	!=	constant

where the constant is a number. **Null** means that the value was not entered in Visible Analyst.

For columns of type *set*, the allowed conditions are:

column	**is**	**null**
column	**is not**	**null**
column	=	set_element
column	**in**	(set_element [, …])

For column type *Set*, set_element is anything valid for that column name. So for column name "type," set_element is any valid Visible Analyst entry type; a **null** value for *set* type columns never occurs. Valid entry types are the entry types listed or any of the following composite types:

any entity type
any module type
any couple type
any dfd type
any sc type
any erd type
any fdd type
any cld type
any sttd type
any use type
any seq type
any col type
any act type

For the column name "storage type," set_element is one of the Visible Analyst storage types, listed in Table 5-4.

For the *location* column type, the allowed conditions are:

exists location

exists location of entry_type [“name_constant”]
exists location on diagram [“diagram_name”]
exists location on branch of “diagram_name”
exists location with source of “name_constant”
exists location with destination of “name_constant”

Entry_type is one of Analyst’s repository types (see above). *Name_constant* is a name of a valid object with a storage type of *entry_type* in the repository. If name_constant is not specified, the condition becomes true if a location of any object with *entry_type* exists in the repository.

Examples of Custom Report Specifications
Several examples of custom report specifications are shown in Figure 5-28.

Syntax Error Checking
The Check button in the Report Query dialog box performs a syntax check of the current report. If it is stored in the report definition file “REPORTS.TBL,” it checks the syntax there.

Check reads the report, parses it and displays various error messages (see Figure 5-29). In those error message lines, *n* is the line number in REPORTS.TBL.

Entire repository

```
  report “Entire Repository”
  select * from repository;
```

All entries with no diagram locations

```
  report “No Diagram Locations”
  select er from repository where not
    exists location on diagram;
```

All unidirectional relationships

```
  report “Unidirectional Relationships
  select * from repository
    where type = relationship and
      reverse relationship is null;
```

Dictionary-only undefined relationships

```
  report “Repository-only Relationships”
  select * from repository
    where type = relationship and
       not exists location on
       diagram and composition is
       null and description is null
       and notes is null;
```

Dictionary-only entities and relationships

```
  report “Repository-only Entities and
    Relationships”
select * from repository
    where type in (all entity types,
       relationship) and not exists
       location on diagram
    order by type;
```

All fundamental data elements with an integer storage type

```
  report “Integer Fundamental Elements”
  select * from repository
    where type = data element and
       exists location of all entity
       types and storage type in
       (integer, smallint, tinyint);
```

Entities with description or notes but without composition

```
  report “Defied Entities Without
    Composition”
  select * from repository
    where type = all entity types and
       (description is not null or notes is not
       null) and
       composition is null
```

All objects on a diagram

```
  report “Diagram Contents”
    params %Diagram Label%
  select * from repository
    where exists location on diagram
  %Diagram Label%
    order by type;
```

All processes created after a given date

```
  report “New Processes”
  params %Starting Date%
  select * from repository
    where type = process and
    creation date >=%Starting Date%;
```

Figure 5-28 Examples of Custom Report Specifications

Errors decoding the report clause
REPORTS.TBL(n)
‘report’ expected.
Report name missing.
Invalid report name.

Errors decoding the params clause
REPORTS.TBL(n)
Duplicate parameter ‘xxx’.
Too many parameters.
Parameters missing.

Errors decoding the select clause
REPORTS.TBL(n)
Missing select value.
Invalid select value ‘xxx’.

Errors decoding the where clause
REPORTS.TBL(n)
Unknown parameter %xxx%.
Unknown Column ‘xxx’.
Unexpected ‘xxx’.
‘(‘ expected.
Entry type expected.
Storage type expected.
String constant expected.
Integer constant expected
Date constant expected.
Invalid entry types list.
Invalid storage types list.
Invalid condition ‘xxx’.
Missing restriction condition.
Restriction too long.
Unbalanced parentheses.
Invalid where expression.

Errors decoding the order by clause:
REPORTS.TBL(n)
Missing order by value.
Invalid order by value ‘xxx’.

Errors decoding the format clause:
REPORTS.TBL(n)
Missing format value.
Invalid format value ‘xxx’.
Warning: ‘format single’ ignored for summary reports

Miscellaneous errors
REPORTS.TBL(*)
Fatal error: cannot open custom report file

REPORTS.TBL(n)
Fatal error: too many errors.
No reports found.
Unexpected end of file.

Final Messages from Check
When done with parsing REPORTS.TBL file, Check displays statistics:

Total *nnn* lines, *nnn* reports (*nnn* correct), *nnn* errors, *nnn* warnings.

Figure 5-29 Report Syntax Errors Reported by Check

Repository Report Preview

The content of a custom report can be previewed before the report is actually printed. In the Report Query dialog box, check the Preview Text box. After you click Print, a box appears containing the report. You can view the report on screen. You can also print it, save it to a file, or abandon it.

You can also generate the report in HTML format so that it can be viewed in a browser. If you have a browser installed, when you click the Preview HTML box the report is displayed in your browser when it is generated.

Printer Setup for Custom Reports

You can select the printer to use by clicking the down arrow by the Printer Name field and selecting a printer from the list. You can also change settings for the printer driver, such as page orientation (portrait or landscape), print to file, etc., by clicking the Properties button on the Report Query dialog box.

SHELL CODE GENERATION

Overview

This Visible Analyst function allows you to generate code for the skeletons of programs in the C and COBOL languages. The shell code generation process is substantially user configurable.

Although the export of Visible Analyst data to IBM External Source Format™ (ESF), used in their Cross System Product™, is done using Export on the Tools menu, it is so similar to shell code generation that the two are covered together here. Note that since ESF was deliberately designed to be similar to COBOL in many ways, much of what is said here about COBOL also applies to ESF.

Scope

Shell code generation encompasses the sequence of functions or paragraphs that make up a program, including global data definitions, descriptive comments, function calls/PERFORM statements, and passed parameters. Information entered in text fields in the Visible Analyst repository for a program item and for structure chart modules of various kinds produces comments that describe these items within the generated code.

Optionally, actual source code may be entered in the module description field of a module or macro, and this code is placed in-line with the function calls or PERFORM statements that are generated by invocation lines. Couples or interface table rows (ITRs) used with invocation lines generate parameters for C code.

Code Generation Speed and Code Size

The code generation speed is quite rapid. The actual time it takes and the volume of data descriptions and code produced, of course, depends on the amount of information entered into the project and upon the configuration and customization options chosen.

Target Languages

COBOL, C and ESF are currently supported. What you have available depends upon the product configuration you purchased.

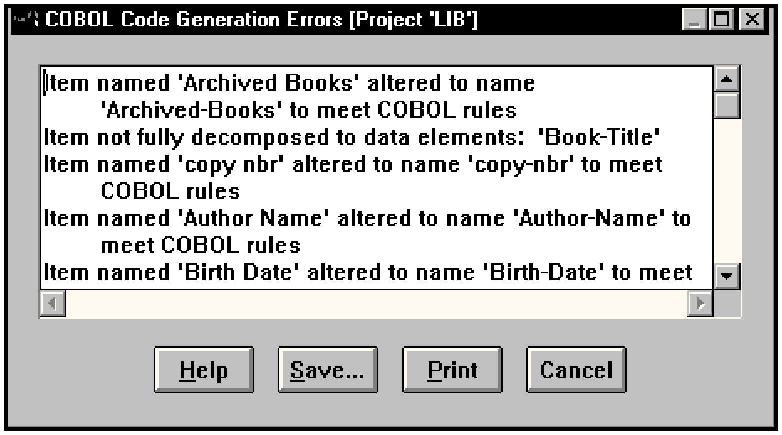
Flexibility of Use

There are numerous code generation configuration options available by selecting Code Generation Options from the Options menu or by clicking the Options button in the Generate Code dialog box accessed from the Repository menu. Those options are described in detail in this section.

There are also ways to customize the code you generate for C and COBOL. This is done by editing the files “VATYPES.H” (for C) or “COBOLTYP.TBL” (for COBOL) that are in your Visible Analyst program directory.

Error Checking

The code generator notifies you of a number of types of errors that would cause bad or incomplete code to be generated and notifies you of certain inconsistencies in the way you specified information within the project repository. You are also told of other items that affect program code, such as the way item names within the project repository may have been changed in the generated code to conform to the rules of the target language. No attempt is made to substitute for the target language compiler. (See Figure 5-30.)



```
Item named 'Archived Books' altered to name
        'Archived-Books' to meet COBOL rules
Item not fully decomposed to data elements:  'Book-Title'
Item named 'copy nbr' altered to name 'copy-nbr' to meet
        COBOL rules
Item named 'Author Name' altered to name 'Author-Name' to
        meet COBOL rules
Item named 'Birth Date' altered to name 'Birth-Date' to meet
```

Figure 5-30 COBOL Code Generation Errors

Code Generation Requirements

There are things you can do within Visible Analyst that enhance and make code you generate more complete. The most important of these are listed below.

Full Decomposition

The most fundamental requirement for good generated data descriptions is that all structural data items be decomposed fully to data elements within the project repository.

Data Element

Data elements should have physical information entered about them on page two of their repository entries or in domain items to which they refer.

Storage Type

The storage type entered is translated using one of the customization files mentioned above to a USAGE (for COBOL), a data type (for ESF) or a native data type, typedef (for C).

Length

Length values translate to a display length for COBOL or ESF, or to one of several values for C. The number of decimal places entered is also used.

Picture

This applies to COBOL only. Any picture you enter for a data element is used verbatim in generated data definitions. It is your responsibility to ensure its correctness under COBOL rules.

Default Value

Any value you have entered in the default field is translated into the code of the language you have chosen. It is made to conform, in so far as is possible, to the data type you specified. If there is a total conflict, however, the default value is ignored.

Structure Chart

If you do not have a structure chart already prepared in your project, neither procedure shell code nor function/paragraph calling and parameter passing is generated. Only data definitions are produced.

Program Item in the Repository

You should have at least one item in the project repository of type program. This gives you a place to enter text information describing your program. In its composition field, you must enter the name of the top module in your structure chart hierarchy. Only one module can be so indicated for a given program. You may have more than one program in a project, each indicating a different top structure chart module. If you have no program item, the entire project is used as if it were one program and modules are ordered alphabetically.

Descriptive Text

Text entered in the various description and notes fields of repository items is used to add comments to generated code (if you have configured the code generator to do so).

In-Line Code

If you exercise this configuration option, you can enter actual target-language code in a module and macro description fields. When the procedure code for your program is generated, this entered code is dropped in-line, verbatim. You are responsible for the accuracy and "compilability" of this code.

File/Data Store Record Layouts (COBOL Only)

Files/data stores and/or entities, depending upon configuration options, generate SELECT and FD statements and the accompanying record descriptions. The names used for these items is taken from the file/data store or entity as entered in the repository (with adjustments made for target-language naming rules) and appropriate suffixes added to distinguish them.

Processing and Storage Redefinition

The code generator recognizes syntax in the composition fields of data items that specify arrays and generates COBOL OCCURS and REDEFINES clauses and their equivalent for C. A number preceding an item or a group of items enclosed in curly braces ('{', '}') constitutes the OCCURS value. If a vertical bar ('|') precedes an item, that item is a redefinition of the previous item in the composition field. These notations do not allow nested constructs (nested sets of braces or redefines of groups of items); separate data structures should be used instead. A REDEFINES item can have an OCCURS value.

For example, if the Composition field of item "str" is:

2 x1 + 3 { x2 + x3 } x4 | x5 | x6 + x7 + x8 | 5 x9

The resulting generated code is shown in Figure 5-31.

The Code Generation Process

To initiate shell code generation, select Generate Code from the Repository menu and then choose COBOL or C from the dialog box. For ESF, select Export from the Tools menu and then ESF from the dialog box.

First, the code generator expands all structural data items as far as they have been decomposed in Visible Analyst. If a structural item is used as a part of more than one higher-level item, it is expanded in each instance where it is used. Next, it makes a program header from comments in the program repository item. Then, for COBOL, Visible Analyst generates file SELECT statements and FILE SECTION data declarations. After that, it generates global (for C) WORKING STORAGE (for COBOL) data definitions.

C

```
struct str_t
        {
            char x1[2];
            char x2[3];
            char x3[3];
            union
            {
                char x4;
                char x5;
                char x6;
            };
            char x7;
            union
            {
                char x8;
                char x9[5];
            };
        };
```

COBOL

```
01    str.
      05 x1                     pic x(1) occurs 2 times.
      05 x2                     pic x(1) occurs 3 times.
      05 x3                     pic x(1) occurs 3 times.
      05 x4                     pic x(1).
      05 x5 redefines  x4       pic x(1).
      05 x6 redefines  x4       pic x(1).
      05 x7                     pic x(1).
      05 x8                     pic x(1).
      05 x9 redefines  x8       pic x(1) occurs 5 times.
```

Figure 5-31 C Array and COBOL OCCURS Examples

C

```
struct name_t
{
        char first-name[ ,,, ]l
        char middle-init[ … ];
        char last-name[ … ];
}
struct address_t
{
        char street[ … ];
        char city[ … ];
        char state[ … ];
        char zip-code[ … ];
}
struct customer_t
{
        struct name_t name;
        struct address_t;
        char phone_number[ … ];
}
struct customer_t customer;
char report-date[ … ];
```

COBOL

```
01   customer.
     05 name.
        10 first-name      PIC. . . .
        10 middle-init     PIC. . . .
        10 last-name       PIC. . . .
     05 address.
        10 street          PIC. . . .
        10 city            PIC. . . .
        10 state           PIC. . . .
        10 zip-code        PIC. . . .
     05 phone-number       PIC. . . .

01   report-date           PIC. . . .
```

Figure 5-32 Generated C and COBOL Code Examples

Following that, Visible Analyst (for C code) produces function prototypes and then it generates code for executable modules (paragraphs for COBOL, functions for C), one for each module in your structure chart set. PERFORM or function call statements are produced for each invocation line and, for C, calling parameters and function return types are produced for couples and ITRs. Finally, if so configured, it drops in-line code entered in module and macro description fields and creates debug statements.

All along, the code generator produces error and information statements, some in the generated-code listing file and the rest in a file that you can review on the screen and, optionally, print.

Code Generation Output

For C and COBOL, the shell code generator leaves you with a compiler-ready ASCII file in your transient file directory for each program in the project. (This is the TRANS directory under the Visible Analyst program directory unless you changed it.) The file is named with the first eight characters of the program name in the project repository plus the extension .c or

.cbl, respectively. For ESF, there is a file in the same directory with the project root as the file name plus the extension .esf. It contains project information and code, including an :appl tag for each program for ESF. You also have the above-mentioned error and information statements, if you chose to print them or save them to a file. If your copy of Visible Analyst has prototyping capability, you can choose to export Visible Prototyper panel files in ESF format. You can either export a panel file in conjunction with project data or export it by itself.

Configuration Options

There are two kinds of code generation configuration. The first is done in the Code Generation Options dialog box. The second, really a form of customization, is done outside of Visible Analyst. You should be careful how you enter the customizations described later. You should change only the sections of the customization files indicated. Making other, indiscriminate changes could make these files unreadable to Visible Analyst. If that were to happen, you must fix them manually or copy their original, unaltered versions from your installation media.

General Configuration

The general configuration options for shell code generation are set using Code Generation Options on the Options menu. These selections, listed below with their default settings, give you alternatives for the Visible Analyst items that appear in generated code and where and how these items appear in generated output. They are distinct from the language-specific customizations discussed later. A checked box is equivalent to Yes and an unchecked one to No.

All Data Elements	**Yes**
All Data Structures	**Yes**
All Data Flows	**Yes**
All Files/Data Stores	**Yes**
All Entities	**Yes**

The first six configuration options give you the opportunity to pick the repository items that appear in your generated code. You can decide whether all items of each type are used or just those listed in the composition fields of other items. The choices available are:

- Yes means list all items of the selected type in their places in the composition hierarchies of superior items *and* create top-level data description code for any item not part of the composition of any other item. In Figure 5-32 for example, the data element “zip-code” that is part of the composition of the data structure “address” is at the bottom level of that data structure. The independent element “report-date” has its own top-level data definition. (Note that in generated COBOL, level 77 entries are not used; level 01 entries are used instead.)

- No means that items not part of the composition of other items do not have data description code generated. In the Figure 5-32 example, the data element “report-date” would not appear.

Note

- If top-level superior items (files/data stores, entities, data structures, data flows) do not appear because items of that kind are set to No, the next level of subsidiary items (elements, flows, structures) that are set to Yes appear in data description code as top-level items.

All Data-Only Modules **Yes**

The meaning of No is slightly different for data only modules in the project repository. It means only use data-only modules that are referenced by some module in the structure chart of the current program.

Data Only Module Usage **Structure**

Data only modules can be interpreted one of two ways:

- The data only module is interpreted as a structure, with the data only module name itself being the top level of the structure; and the data elements and data structures named in its composition field are all subsidiary items at the next lower hierarchical level. To use this interpretation, select Structure.
- The data-only module is interpreted as a collection of independent, top-level data elements or data structures. The name of the data only module itself does not appear anywhere in any data definition. To use this interpretation, select Elements.

Description Field in Module Comment **Yes**

Notes Field in Module Comment **Yes**

Module Description in Module Comment **Yes:** **Comment**

The text fields of programs and modules of all kinds (modules, macros, library modules, etc.) can be selectively used or not used at all.

- Description Field – Yes means use this field as a comment to describe modules in generated code. No means don’t use the field for anything.
- Notes Field – This is identical to the above.
- Module Description Field – If the box is checked, you can choose Comment or Code. Comment means the same as Yes, above. Code means that you have entered source code for the target language in this repository field. The code generator puts it verbatim into the syntactically correct place in the generated code for this module. Function calls or PERFORM statements generated by structure chart invocation lines appear as comments in the code of any module where your own code is used. This is to help you keep your code in sync with your structure chart diagrams.

Generate Debug Yes

Choosing Yes causes the code generator to produce a source code statement that displays the name of the module on the system console screen. This allows you to compile and run the generated code and watch it execute, even if the modules do not yet contain any other code.

COBOL-Specific Configuration Items

These items apply only to COBOL and they can all be selected independently. They specify whether you want to generate COBOL FD statements for:

Files Yes

Yes means that SELECT and FD statements are generated for all files/data stores in the project. (This presumes that the All Files/Data Stores is also set to Yes.)

Entities Yes

Yes means that SELECT and FD statements are generated for all entities in the project.

COBOL File Data Structure Location (FD)

This selection is enabled if one of the previous two items is checked. Some people like to have full record data descriptions in the FILE SECTION. Others prefer an 01-level statement declaring a maximum record size and placing the full data description in the WORKING STORAGE SECTION. If Working Storage is chosen, there is a record data structure declaration under the 01-level in the FILE SECTION, with all names below the 01-level set to FILLER, so the COBOL compiler can compute the record length.

Language-Specific Customizations

C

There is a file in the Visible Analyst program directory named VATYPES.H that the code generator uses to translate Visible Analyst values for storage type, length, etc., into C data types, structures, and so forth. You can customize the way some Visible Analyst storage types are translated by making changes to this file. If you edit VATYPES.H using an ASCII text editor, you see that there are five categories of Visible Analyst storage types described in the file.

- The first and third sections are fixed (not customizable).
- The second and fourth sections contain the C typedef statements that allow you to specify the native C data types that correspond to some Visible Analyst storage types. You can alter which native C types are generated for each Visible Analyst type. Note that you can also replace the native C type with a structure, as described in the section below.
- The fifth section allows you to customize the C structure that is generated for some Visible Analyst storage types. You can either change what appears between the curly braces or you can replace the structure itself with a native C data type.

COBOL

There is a file in the Visible Analyst program directory named COBOLTYP.TBL that the code generator uses to translate Visible Analyst values for storage types into COBOL USAGE types, with appropriate picture clauses. This file comes configured with COBOL-85 USAGE types, but you can change this.

You can customize which COBOL USAGE type corresponds to which Visible Analyst storage type by making changes to this file. If you edit COBOLTYP.TBL using an ASCII text editor, you can see and change the correspondences between the COBOL and the Visible Analyst types in the file. By assigning a picture class identifier to each, generated PICTURE clauses (produced only when you do *not* specify a picture yourself) are constructed to your preference.

There are a number of other customizations that you can make to adjust the appearance of COBOL code to your own shop standards. You can also set default values for display length and number of decimal places, to be used when you *don't* enter these values into the project repository. All of these customization options are explained in detail in the COBOLTYP.TBL file. You should examine the file before generating code to be sure that the values currently set meet your needs.

ESF

There are no customizable options for ESF. The ESF data types that correspond to Visible Analyst storage types are listed in the table below. The BYTES value for each data type is also in the table. Note that the BYTES value for some data types depends on the length value entered in the physical characteristics of a data element entry in the Visible Analyst repository. Note also that if a length is not entered, the default BYTES value is used. Further note that ESF data types marked as n/a are translated to the ESF type HEX.

The conversion from Visible Analyst length to ESF BYTES value for some data types is as follows:

DECIMAL:	BYTES = length / 2 + 1<
FLOAT:	BYTES = if length <= 21, 4 else 8
CHARACTER:	BYTES = length<
VARCHAR:	BYTES = length
GRAPHIC:	BYTES = length * 2
VARGRAPHIC:	BYTES = length * 2

Table 5-5
ESF Data Types

Analyst		ESF		
#	**ANALYST Type**	**ESF Type**	**BYTES**	**Default BYTES Value**
0	Undefined	n/a		
1	Integer	BIN	4	4
2	Decimal	PACK	*,*	8
3	Float	HEX	*	8
4	Character	CHA	*	1
5	Date	CHA	10	10
6	Time	CHA	8	8
7	Binary	n/a		
8	Bit	n/a		
9	DateTime	n/a		
10	Autoinc	n/a		
11	Real	HEX	4	4
12	Image	n/a		
13	Bfloat	n/a		
14	Money	n/a		
15	Note	n/a		
16	SmallInt	BIN	2	2
17	Sysname	n/a		
18	Text	n/a		
19	TimeStamp	CHA	26	26
20	TinyInt	n/a		
21	VarBinary	n/a		
22	VarChar	CHA	*	1
23	Zstring	n/a		
24	Lstring	n/a		
25	Lvar	n/a		
26	Logical	n/a		
27	Graphic	DBCS	*	2
28	VarGraphic	DBCS	*	2
29	Number	n/a		
30	Numeric	n/a		
31	Double Precision	HEX	8	8
32	Raw	n/a		
33	Long Raw	n/a		
34	Long VarChar	CHA	32767	32767
35	Long VarGraphic	DBCS	32766	32766

Table 5-5
ESF Data Types

36	SmallFloat	n/a		
37	Serial	n/a		
* means that length is derived from the Visible Analyst length (if it exists) n/a (not applicable) types are exported as HEX				

Exporting Visible Prototyper Panels

When exporting Visible Analyst project data in ESF format, you can choose to export Visible Prototyper panel files. You can either export a panel file in conjunction with project data or export it by itself. If your copy of Visible Analyst has prototyping capability, there is an active Panel File button in the Export dialog box. Clicking it displays a dialog box in which you can select a panel file and choose whether you want to export a panel file, repository data, or both. Note that if you don't select a panel file for the current export, no panel file is dumped, regardless of what you may have done in the past.

Parameter

Currently, parameter passing in generated code is only supported for the C language. COBOL supports parameter passing in multi-program systems, but these are not yet supported by Visible Analyst.

You should attach couples to the invocation lines on your structure charts when you want to indicate parameters being passed. The ways they should be used to properly interface with the code generator are:

- Downward-pointing (along the invocation line) couples represent parameters passed *to* a function.
- An upward-pointing (opposite to the invocation line) couple indicates the return value *from* a function. Only one of these is allowed by the code generator and its type becomes the type of the function. If one of these return couples is not present, the function is of type "void."
- To represent a parameter that is passed by reference and can be changed in the calling function by the called function (for example, a passed address), use a bi-directional couple.
- An ITR can be used to represent a group of couples. Using ITRs is encouraged. It not only cuts down on the volume of detail that appears on your structure charts, but also maintains a consistent parameter order and number in all calls to a given function.

The first invocation of a function that has the maximum number of couples of all invocations of that function is used to produce a C function prototype for the function. If other calls to that function are incompatible with this prototype, error messages are produced to call this to your attention.

SQL SCHEMA GENERATION

SQL DDL (Data Definition Language) syntax can be generated for all entities in the repository for a project. Dialects supported include:

Access 97, 2000
ANSI 92
CA Datacom 8x
CA Open Ingress II 2x
Centura SQL Base 5x
DB 2/2, 2x, 5x, 6x, 7x, 8x
DBASE IV
Informix 7x
Ingres 6x
InterBase 4x, 5x

MS SQL Server 4x, 6x, 7x, 2000, 2005
Netware 1x

Oracle 7x, 8x, 9x, 10x
Paradox 7x, 8x
Progress 7x, 8x, Native 7x, Native 8x
SYBASE SQL Anywhere 5x
SYBASE SQL Server 4x, 10x
Teradata SQL V2 2.1.0
Unify 2000
User Defined
Vax RDB 6x
Watcom 3x

xdb 1x
XML DCD, Visible Developer 3x, 4x

Statements supported include:

- CREATE TABLE with options indicating primary, foreign and unique keys, tablespace, and column characteristics including data type, default value, and nullability.
- CREATE UNIQUE INDEX for primary and unique constraints, when required by the selected dialect including additional dialect-specific characteristics.
- CREATE INDEX for performance indexes.
- COMMENT ON for tables and columns.
- CREATE TABLESPACE with options indicating SQL dialect-dependent physical characteristics.
- CREATE STORED PROCEDURE for dialects such as Rdb, Oracle Server, Informix and SQLServer. Any text stored in the module description field for an entry in the repository comprises the body of the stored procedure during SQL generation.
- CREATE TRIGGER to generate SQL triggers corresponding to those created in the project repository and ones generated to enforce referential integrity.
- CREATE SYNONYM to pass on an entity alias to the SQL engine as an alternate name for a table.
- CREATE SCHEMA or authorization ID to define a name space within a given set of tables. You are prompted for the name you wish to use.
- ALTER TABLE statement is used if a foreign key references a table that has yet to be created in the DDL script.

- CONSTRAINTS statements are generated if this option is selected from the options dialog box. A constraint name is generated for primary and foreign keys and/or check constraints with the following syntax:

primary keys	PKC_<TableName>hex number
foreign keys	FKC_<RelationshipName>hex number
alternate keys	AKC_<TableName>hex number
column constraint	<ConstraintModuleName>hex number

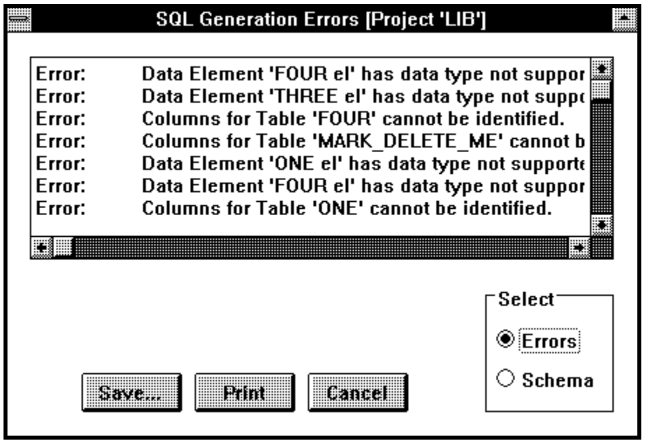
- CREATE VIEW, with options for a column list, select statement. Under certain conditions, generation produces multiple CREATE VIEW statements for a single view object in the repository. Most conditions causing spawned views are related to deficiencies in the expressiveness of a DBMS select statement. Deficiencies such as restricting an outer member[3] from participating as an inner member of an inner join or prohibiting multiple outer joins within a given select statement cause the generation of supporting views. Another potential source of supporting views occurs when bridging differences between the logical data model and its physical implementation. The options controlling denormalization can cause tables to be folded within other tables. When table folding occurs, the contents of a logically-specified view must be converted to account for the redirection and renaming of logical columns from one table to another. Views are used to ease this transition.

The dialect you wish to use can be selected from the SQL Dialect from Options menu or by clicking the Dialect button on the Repository Define dialog box.

You can generate an SQL schema with the Generate SQL function list under the Repository menu. Visible Analyst analyzes the repository and produces the schema, displaying the names of items it is examining on the status line.

If errors are found, they are displayed on the screen (see Figure 5-33); you can save them to a file, print them or ignore them. The errors may be entities without keys, entities with improperly specified keys, an invalid physical data type for a composing data element, or a number of other errors. If you have previously run the Analyze functions for key analysis and key synchronization and have resolved all errors found, you are much less likely to get errors at the schema generation stage.

[3] Outer member refers to the table of an outer join that only contributes values to the resulting relationship when a row in it matches the criteria of the join expression. The table appearing on the right-hand side of a left outer join is the outer member.

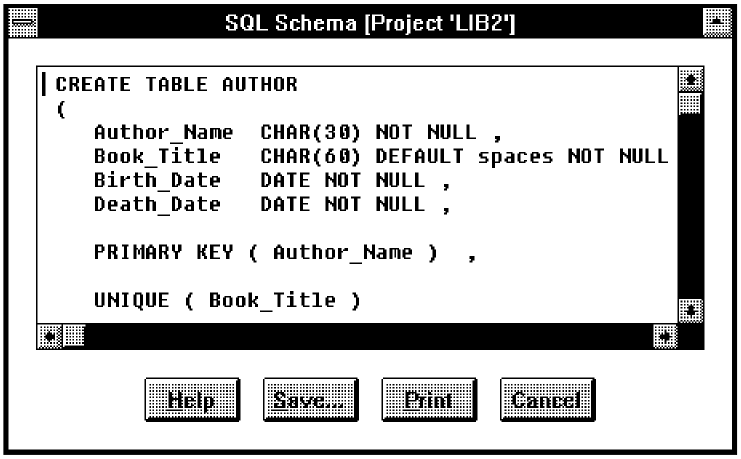


SQL Generation Errors [Project 'LIB']

```
Error:     Data Element 'FOUR el' has data type not suppor
Error:     Data Element 'THREE el' has data type not suppo
Error:     Columns for Table 'FOUR' cannot be identified.
Error:     Columns for Table 'MARK_DELETE_ME' cannot b
Error:     Data Element 'ONE el' has data type not support
Error:     Data Element 'FOUR el' has data type not suppor
Error:     Columns for Table 'ONE' cannot be identified.
```

Figure 5-33 SQL Schema Generation Errors

Visible Analyst creates as much of the schema as it can, leaving out items for which it finds errors. The schema displays on the screen (see Figure 5-34); you can save it to a file, print it or cancel it. If errors are found, you can cancel it, correct the errors, and generate a corrected schema at a later time.



SQL Schema [Project 'LIB2']

```
CREATE TABLE AUTHOR
(
   Author_Name  CHAR(30) NOT NULL ,
   Book_Title   CHAR(60) DEFAULT spaces NOT NULL
   Birth_Date   DATE NOT NULL ,
   Death_Date   DATE NOT NULL ,

   PRIMARY KEY ( Author_Name )  ,

   UNIQUE ( Book_Title )
```

Figure 5-34 Generated SQL Schema

Other features of SQL schema generation include:

- The ability to generate a schema for only a portion of a data model.
- Generation of key constraint names.
- An option to replace an attribute name with an alias, for those who wish to maintain alternate (physical) names and use them in schemas.
- An option to generate tablespaces and tablespace references from tables and indexes.
- Storing check constraints, triggers, stored procedures (all three are various ways of enforcing validation rules) and enhanced referential integrity information in the repository, and including them in generated schemas.
- Generating UNIQUE constraints for alternate keys.
- Ascending and descending information for primary and unique key column ordering and created indexes.
- Generation of the IDENTITY clause for SQLServer 6.x, 7.x, 2000, 2005 and System 10/11 when the repository “Allow Null” property specifies the identity constraint.
- An option to place quotes around column and table names (this may be necessary if your name contains characters that conflict with the target RDBMS, or the name is a keyword).
- An option to ignore the denormalization settings for the relationships included in the schema.
- An option to choose the type of referential integrity generated, either Declarative that uses the Foreign Key clause of the Create Table or Alter Table statements or Trigger Wizard that generates an appropriate trigger for each dependent table. (If you set the Referential Operation to No Check for a relationship, no referential integrity is generated for the relationship.)

When generating XML, a Document Content Description (DCD) is generated that can be read by eXcelon from Object Design. You can also generate XML for Visible Developer, Visible’s software component design and code generator.

Users now have the additional option to generate XML Schema based on the W3C standard for the entities and (optionally) classes developed in their project. Simply use the classes on an entity relationship diagram and the classes will be included in the XML Schema generation. This generation XML Schema Generation option is available on the Tools | Export menu.

When generating a Uniface schema, the following options are available:

- If you want to generate domains, you should use the Template option for version 6.x of Uniface, and the v5.2 Domains option for all other versions.
- If you generate a schema, you have the option to Overwrite, Replace, or Clean an existing Uniface Conceptual Schema. Overwrite modifies only the objects that appear in the generated schema. Replace deletes the entire schema and replaces it with the generated schema. Clean overwrites existing objects and deletes those items that are not referenced

in the generated schema. For more information, refer to the publication Uniface CASE Bridge Cookbook.

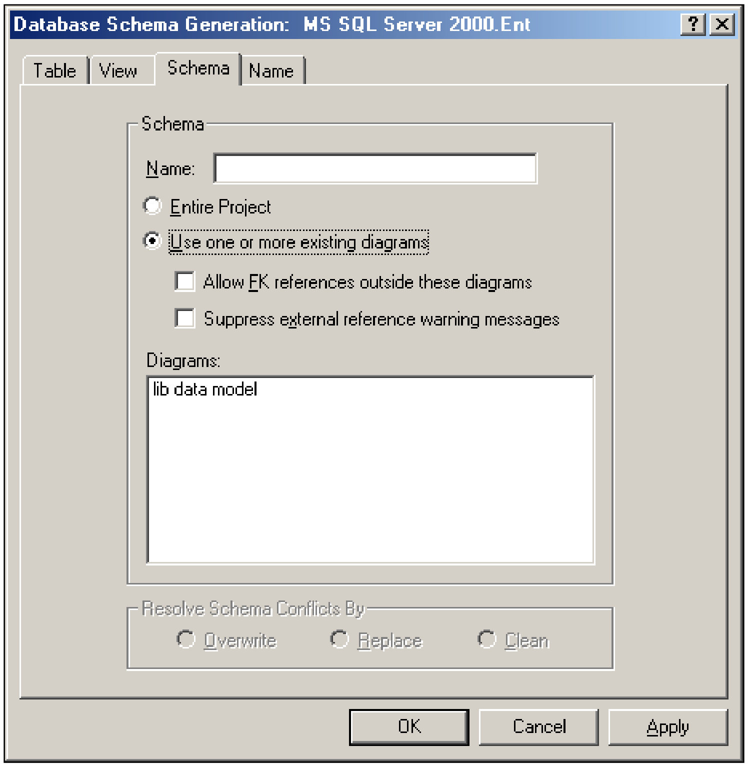
Note

Uniface schema is generated from the Export option on the Tools menu.

Generating a Schema for Part of a Project

To generate a subset of a project's complete schema:

Open Dialog Box: 1 Select Generate Database Schema from the Repository menu to display the Database Schema Generation dialog box (Figure 5-35).



Schema

Name:

○ Entire Project

● Use one or more existing diagrams

□ Allow FK references outside these diagrams

□ Suppress external reference warning messages

Diagrams:

lib data model

Resolve Schema Conflicts By

○ Overwrite ○ Replace ○ Clean

Figure 5-35 SQL Schema Generation Information

Choose (Scope of 2 On the Schema tab, select "Use one or more existing

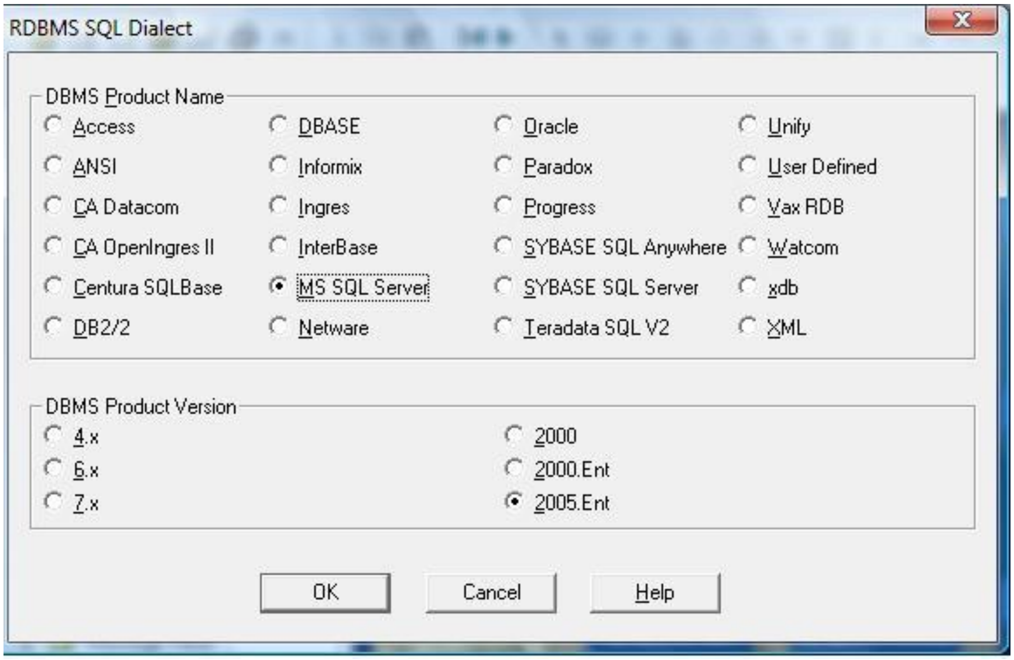
Schema):		diagrams" and then select the diagram options that is then enabled. Select one or more diagrams to include in the schema.
Select Options:	3	Select table, statement and index options on the Table tab; then complete the Name tab.
Click OK:	4	Click OK to generate the SQL.

You may want to check the "Suppress external reference warning messages" box. Items within the view(s) selected might reference entities not included in the schema. The generator is not able to produce DDL if it encounters a key reference outside the scope of the selected views and has not been directed to ignore it. If you *do* want foreign keys within the schema to be able to reference items not included in the schemas, you should check the "Allow FK references outside of these diagrams" box. The schema generates properly, but issues warnings to you when those references occur unless you suppress them.

Adding SQL Schema Generation Information for Entities and Relationships

This information is attached to the repository entry of an entity (table) or relationship. To add to it, the Repository Define dialog box for the entity/relationship must be open on your screen.

- To determine the SQL dialect currently in use, click the Dialect button on the Define dialog box to display the RDBMS SQL Dialect dialog box. To change the dialect, click on a dialect in the list, select the version if necessary, and click OK.



RDBMS SQL Dialect

DBMS Product Name

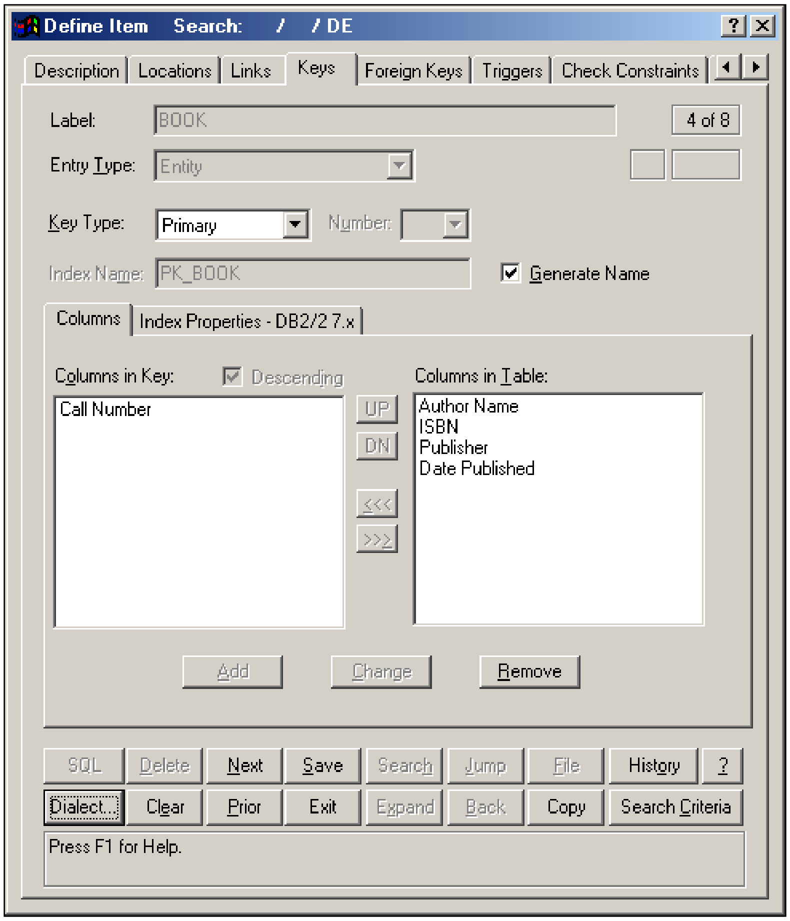
○ Access ○ DBASE ○ Oracle ○ Unify
○ ANSI ○ Informix ○ Paradox ○ User Defined
○ CA Datacom ○ Ingres ○ Progress ○ Vax RDB
○ CA OpenIngres II ○ InterBase ○ SYBASE SQL Anywhere ○ Watcom
○ Centura SQLBase ● MS SQL Server ○ SYBASE SQL Server ○ xdb
○ DB2/2 ○ Netware ○ Teradata SQL V2 ○ XML

DBMS Product Version

○ 4.x ○ 2000
○ 6.x ○ 2000.Ent
○ 7.x ● 2005.Ent

Figure 5-36 RDBMS SQL Dialect Dialog Box

For entities, Repository Define dialog box tabs display key, foreign key, trigger, table and column check constraint, and physical information for the entity. For relationships, Repository Define dialog box tabs display referential integrity and cardinality information for the foreign key. This information is displayed by clicking the appropriate tab at the top of the Repository Define dialog box (see Figure 5-37 below).



Label: BOOK 4 of 8

Entry Type: Entity

Key Type: Primary Number:

Index Name: PK_BOOK ☑ Generate Name

Columns in Key: ☑ Descending

Call Number

Columns in Table:

Author Name
ISBN
Publisher
Date Published

Press F1 for Help.

Figure 5-37 Define Dialog Box SQL Schema Generation Information Tabs

Table and Column Check Constraints

Any table may have an arbitrary number of check constraints attached to it. All of these check constraints are generated, along with their associated search conditions, with the schema. The check constraints are stored as separate entries in the repository and can be referenced from multiple entities.

Attaching a Check Constraint to a Table

- Click the Check Constraint tab.
- To attach a constraint, type the name of the constraint in the Table Check Constraints box and click Add. If the constraint exists in the project repository, the search condition for it

displays in the box. If it is not currently in the repository, a new constraint is added to the repository; you can edit it later and add the search condition to its Module Description field and any other information you desire.

You can also add, edit, and delete check constraints (as well as triggers and stored procedures) directly to the repository.

- Bring up a blank repository dialog box using Define from the Repository menu. Name the check constraint and select Module as the entry type. Click the Save button to add the module to the repository. At this time, the module subtype box appears to the right of the Entry Type box, allowing you to define the module as a check condition, trigger or stored procedure. Add any other descriptive information you wish, such as SQL procedure code in the Module Description field, and click Save again.
- You can also use the repository search function to attach one or more existing check constraints to the table. You can view the search condition for each of these by highlighting them in turn in the Select Check Constraints box.
- To detach a constraint from the entity, highlight it in the Check Constraints box and click Delete. As in the Composition field in the Define dialog box, if you delete a check constraint and there is no other descriptive information added for it, it is deleted from the repository.
- To edit information in the repository entry for the selected check constraint, click the Jump button.

When you are finished:

- Click Save to save your changes. Changes you made are reflected in the SQL Schema Generation Information dialog box. (To abandon your changes, click Clear.)
- Check constraints for columns are added identically, except that only one constraint per column is allowed.

Adding Key Information for a Table

To specify key information for a table:

- Click the Key tab. Information can be entered here to tell the schema generator what it should do with repository primary and alternate key data and performance indexes.
- Highlight different keys in the Key Number box to display or enter information for the various alternate keys and performance indexes that exist for the current entity.

If your SQL dialect allows it, you can name the indexes that are generated from primary and alternate keys and performance indexes.

- Enter the name in the Index Name boxes. The primary key index can have a name, as well as each alternate key and performance index. If you want Visible Analyst to generate a name for you, check the Generate Name box. If you leave the Index Name box empty and the Generate Name box unchecked, no index is included in the schema.

You can change the columns that are part of the primary or alternate keys or a performance index.

- To add a component of the key, highlight a column in the Columns in Table list and use the ≺≺ button to make it part of the current key.
- To remove a component of the key, highlight a column in the Columns in Key list and use the ➤➤ button to move it back to the list of available columns.
- To change the position of a key column, highlight a column in the Columns in Key list and use the up or down arrows to change its position.

You can add ascending and descending information for primary and unique key column ordering and created indexes, if your SQL dialect allows it.

- Highlight a column in the key for which you want to add it.
- Click Descending if the column should be sorted in descending order, otherwise it is sorted in ascending order. If your SQL dialect doesn't allow such a specification, the button is disabled.

You can add physical storage and index type information about an index by clicking the Physical Characteristics tab.

When you are finished:

- Click Save to save the key definition, or click Clear to abandon your changes.

Adding Foreign Key Information for a Table

To specify foreign key information:

- Click the Foreign Key tab if the current entry is an entity or a relationship. Information can be entered here to tell the schema generator what it should do with repository foreign key data.
- Highlight different relationships in the Relationship box to display or enter information for the various relationships that exist for the current entity.

Referential integrity information, telling the SQL engine what to do when database information is deleted or updated, can be added for each foreign key.

- Click the buttons next to On Delete and On Update to indicate the action you want taken for each. Any actions your SQL dialect does not allow are disabled. If you select No Check, a referential integrity constraint is not generated for the relationship. These options are from the parent perspective (if you delete a row from the parent table, how are the rows in the child table affected).
- If you use the Trigger Wizard to generate referential integrity constraints, click the buttons next to On Insert (Or Update) Of Child to indicate the action you want taken when the child table is modified. The only valid options are Restrict and No Check. If your SQL dialect does not support triggers, these options are disabled.

Key Reference information, telling the SQL engine how columns from the child table correspond to key columns in the parent table, can be added for each foreign key.

- Click Primary Key to indicate that the foreign key columns in the child should correspond to the primary key in the parent, or click Alternate Key to indicate that they should be associated with an alternate key. If an alternate key is selected, choose the appropriate number.
- Highlight a column from the Available Columns list and then click on the corresponding parent key column. Repeat this procedure for each component of the foreign key.
- Click Migrate to add a key column from the parent to the child.

Cardinality information instructs the SQL engine how to resolve column name conflicts for supertype relationships that have any form of denormalization specified.

- Syntax displays the numerical cardinality that was specified when the relationship was added to a diagram.
- Click Suffix Instead of Prefix to indicate column names should be made unique by appending text to the column name.
- Click Prefix/Suffix Names to modify the list of prefixes/suffixes that should be used when generating SQL to uniquely identify columns. The number of entries in the list should match the numerical cardinality. If no cardinality is specified, the number of columns generated corresponds to the number of entries in the list.

When you are finished:

- Click Save to save your changes. Any changes you made are reflected in the composition field of the child entity. If you select a different relationship, any pending changes are automatically saved. Click Clear to abandon your changes without saving.

Adding Trigger Information for a Table

A table can have an arbitrary number of triggers attached to it. The triggers are stored as separate entries in the repository and can be referenced from multiple entities. Any actions your SQL dialect does not allow are disabled. Triggers need not be specified in the repository if their only purpose is to implement referential integrity (RI). The Trigger Wizard will automatically generate complete trigger definitions to enforce RI. However, if additional integrity constraints, other than RI, must coexist with RI triggers, refer to Combine Generated Trigger Code with User Triggers in Appendix C.

To specify trigger information:

- Click the Triggers tab.
- Attach and detach triggers to an entity as with check constraints, above.

Note

- This only functions if the SQL dialect in effect supports triggers.

- Highlight an attached trigger in the Trigger box to view the information specified for that trigger, or to change it. The action text entered in the Module Description field of the trigger's repository entry displays in the Trigger Text box.
- Click Update and/or Insert and/or Delete to specify the action that causes the trigger to fire. Your SQL dialect may allow multiple actions to fire a trigger or it may require a different trigger for each action.
- Click Before or After to tell the SQL engine when to fire the trigger relative to the action(s) specified.
- In the Scope box, click Table or Row to indicate whether the trigger is to fire once for an update (or insert or delete) of the table or once for each row that is updated.

If the Update button is checked, you can specify for which columns in the table an update fires the trigger. Initially, the names of all columns in the table are listed in the No-Update Columns box.

- Highlight one or more column names in the No-Update Columns box and click the ≺≺ button to move them into the Update Columns box.
- Highlight one or more column names in the Update Columns box and click the ➢➢ button to move them into the No-Update Columns box.

If you want to edit information in the repository entry for the selected trigger:

- Click the Jump button. Changes you made are saved, as if you clicked OK.

Note

- When defining the body of a trigger, you are free to use built-in macros supplied by the Trigger Wizard, or you may create your own. See Appendix C, Customizing the Trigger Wizard, for details.

When you are finished:

- Click Save to save your changes. Click Clear to abandon your changes.

Adding SQL Schema Generation Tablespace Information

Some SQL dialects support the concept of a tablespace, which is an allocation of space on a physical media for the storage of tables and indexes (SQLServer 7.x uses the term file group, while other versions of SQLServer use the term segment).

Adding Physical Storage Information for a Table

To specify the physical storage characteristics of a table, including the tablespace to which a table belongs, click the Physical tab on the Repository Define dialog box.

For each table or index, the following information can be maintained (not all SQL dialects support all options):

- **Tablespace.** The name of the tablespace where the table or index is to be stored. If the name specified does not exist in the repository, a new tablespace object is created. You can use the search button to find tablespace objects that have already been defined. (SQLServer 7.x uses the term file group, while other versions of SQLServer use the term segment.)

- **PctFree.** The percentage of space reserved for future updates.
- **PctUsed.** The minimum percentage of used space.
- **IniTrans**. The initial number of transaction entries allocated within each data block.
- **NoSort.** Indicates that the rows are stored in ascending order and therefore the rows do not have to be sorted when creating the index.
- **Initial.** The size in bytes of the tablespace's first extent.
- **MaxTrans.** The maximum number of concurrent transactions that can update a data block allocated to the table.
- **Next**. The size of the next extent to be allocated to the tablespace.
- **MinExtents.** The total number of extents allocated when the tablespace is created.
- **MaxExtents.** The maximum number of extents that can be allocated for the tablespace.
- **PctIncrease.** The percent by which each extent after the second grows over the previous one.
- **Freelists**. The number of free lists for each of the free list groups.
- **Freelist Groups.** The number of groups of free lists.

DB2 2 Properties

- **Primary Tablespace.** Identifies the tablespace in which the table will be created. The tablespace must exist and be a REGULAR tablespace. If no other tablespace is specified, all table parts will be stored in this tablespace.
- **Long Column Tablespace.** Identifies the tablespace in which the values of any long columns (LONG VARCHAR, LONG VARGRAPHIC, LOB data types, or distinct types with any of these as source types) will be stored.
- **Index Tablespace.** Identifies the tablespace in which any indexes on the table will be created. This option is allowed only when the primary tablespace specified in the IN clause is a DMS tablespace. The specified tablespace must exist and be a REGULAR DMS tablespace.
- **Data Capture.** Indicates whether extra information for data replication is to be written to the log. *None* indicates that no extra information will be logged, while *Changes* indicates that extra information regarding SQL changes to this table will be written to the log. This option is required if this table will be replicated and the Capture program is used to capture changes for this table from the log.

DB2 2, 5, 6, 7, 8 Properties

- **Partitioning Key.** Specifies a key used when distributing the rows of a table across physical media. Visible Analyst lists the performance keys assigned to the table as possible choices. Only performance keys whose columns represent a subset of the primary key should be selected as a partitioning key. Please refer to the section "Adding Key Information for a Table" for directions on building a Performance Key. A column must not appear more than once in the key. No LOB, LONG VARCHAR, or LONG VARGRAPHIC column may be used as part of a partitioning key. If not specified, and this table resides in a multiple partition nodegroup, then the partitioning key is defined as follows:

  - If a primary key is specified, the first column of the primary key is the partitioning key.
  - Otherwise, the first non-long column (LOB or long column type) is the partitioning key.
  - If none of the columns satisfies the requirement of the default partitioning key, the table is created without one. Such tables are allowed only in tablespaces defined on single-partition nodegroups.

- **Using Hashing.** Specifies the method applied to partitioning keys to yield the insertion location for a newly added row. Hashing is the only method currently available.
- **Not Initially Logged.** This option is useful for situations where a large result set needs to be created with data from an alternate source (another table or a file) and recovery of the table is not necessary. Using this option will save the overhead of logging the data.

SQLServer Storage Options for Indexes

When the SQL engine generates DDL for SQLServer (either 4.x, 6.x, 7.x, 2000, 2005 or System 10/11), physical characteristics can be specified for each index that is created. To specify the physical storage characteristics of an index, click the Physical tab.

For each index, the following information can be maintained:

- **Segment**. The name of the segment where the index is to be stored. If the name specified does not exist in the repository, a new tablespace object is created. You can use the search button to find tablespace objects that have already been defined. (Visible Analyst uses the generic term tablespace to refer to a physical storage object. SQLServer 7.x uses the term file group, while other versions of SQLServer use the term segment.)
- **Clustered.** Create a clustered index where the physical order of the rows is the same as the indexed order of the rows. Only one clustered index is allowed per table.
- **Ignore Duplicate Keys**. If an insert or update operation would result in a duplicate key, do not perform the operation and discard the data. If this option is not set, the operation fails.
- **Sorted Data**. Indicates that the rows are stored in sorted order. Not valid for SQLServer 7.x.
- **Fill Factor**. A percentage that specifies how full each data page is made when creating a new index on existing data.
- **Duplicate Rows Check.** Specifies whether duplicate rows should be allowed when creating the index. This option is only valid for clustered indexes. Not valid for SQLServer 7.x.
- **Pad Index.** Specifies the space to leave open on each page (node) in the intermediate levels of the index. This option is useful only when Fill Factor is specified because it uses the percentage specified by Fill Factor.

- **No Recompute Statistics.** Specifies that out-of-date index statistics are not automatically recomputed.

Oracle 7 Tablespace Properties

Visible Analyst generates CREATE TABLESPACE statements during SQL generation if the current dialect is Oracle and any tables or indexes reference a tablespace object in the repository. For each tablespace, the following pieces of information can be maintained:

- **Datafile Name(s).** The name(s) of the datafile(s) that comprise the tablespace.
- **Datafile Size.** The size of the data file. If this option is omitted, the data file must already exist.
- **Reuse.** Reuse an existing file. If the file does not exist, it is created. If this option is not checked, the file must not already exist. This option is only significant when used with the size option. If you omit the size option, the file must already exist.
- **Initial.** The size of the tablespace's first extent.
- **Next.** The size of the next extent to be allocated to the tablespace.
- **Minextents.** The total number of extents allocated when the tablespace is created.
- **Maxextents.** The maximum number of extents that can be allocated for the tablespace.
- **Freelists.** The number of free lists for each of the free list groups.
- **Freelist Groups.** The number of groups of free lists.
- **Offline.** Do not allow access to the tablespace after it is has been created.

Oracle 8

Oracle 8 includes all the Oracle 7 properties plus the following:

- **Logging.** Specifies whether creation of the index will be recorded (LOGGING) or not recorded (NOLOGGING) in the redo log file. Table and index objects inherit the value assigned to the tablespace when it is unspecified. The system default for this property when unspecified for the tablespace is LOGGING. It affects whether the direct loading of data using SQL * Loader writes insert operations to the redo log file.
- **Cache.** (Table/tablespace only) CACHE specifies that the data will be accessed frequently, therefore the blocks retrieved for this table are placed at the most recently used end of the LRU list in the buffer cache when a full table scan is performed. This option is useful for small lookup tables. NOCACHE specifies that data will not be accessed frequently; therefore, the blocks retrieved for this table are placed at the least recently used end of the LRU list. The default is NOCACHE.

Although Oracle 8 does not support the CACHE property for tablespaces, it may be specified on the repository tablespace object. When specified as a property of a tablespace, any table associated to this tablespace will derive its default cache value from the tablespace.

SQLServer Segment Properties

Visible Analyst generates SP_ADDSEGMENT and SP_EXTENDSEGMENT statements during SQL generation if any tables or indexes reference a tablespace object in the repository.

For each segment, the following information can be maintained:

- **Logical Device Name.** The database devices where the segment is to be located.

DB2 Tablespace Properties

For each tablespace, the following information can be maintained:

- **Tablespace Type.** *Regular* indicates the tablespace is to be used for all data except temporary tables. *Long* is for storing long or LOB columns. *Temporary* indicates the tablespace is used for storing temporary tables.
- **Managed By.** The tablespace is to be managed either by the *System* (SMS) or the *Database* (DMS).
- **Nodegroup.** Specifies the nodegroup for the tablespace. The nodegroup must exist. The only nodegroup that can be specified when creating a TEMPORARY tablespace is IBMTEMPGROUP. The NODEGROUP keyword is optional. If the nodegroup is not specified, the default nodegroup (IBMDEFAULTGROUP) is used unless TEMPORARY is specified and then IBMTEMPGROUP is used.

Note

> If the Tablespace Type or the Managed By button is disabled, this means that an entity definition references the tablespace and any changes that you make would cause a conflict. For example, if an entity uses the tablespace as a *Long* tablespace, you cannot change the type to *Regular.*

- **Container Name(s).** The name(s) of the container(s) that comprise the tablespace.
- **Pages.** The size of the container in 4K pages. This option is only valid for DMS tablespaces.
- **Container Type.** *File* indicates the container is an absolute or relative filename while *Device* indicates it is a device. This option is only valid for DMS tablespaces.
- **Extentsize.** The number of 4K pages that are written to a container before skipping to the next container.
- **Prefetchsize.** The number of 4K pages that are read from a table when data prefetching is being performed.
- **Overhead.** The I/O controller overhead and disk seek and latency time, in milliseconds. (Used to calculate the cost of I/O during query optimization.)
- **Transferrate.** The time to read one 4K page into memory, in milliseconds. (Used to calculate the cost of I/O during query optimization.)
- **Bufferpool.** The name of the bufferpool used for tables in this tablespace. The bufferpool must exist. If not specified, the default bufferpool (IBMDEFAULTBP) is used. This nodegroup of the tablespace must be defined for the bufferpool.

Informix Tablespace (dbspace) Properties

Informix supports the concept of a dbspace, an allocation of space on a physical media for the storage of tables and indexes. Visible Analyst, which calls these objects tablespaces, adds dbspace and storage information to CREATE TABLE statements during SQL generation if any tables or indexes reference a tablespace object in the repository.

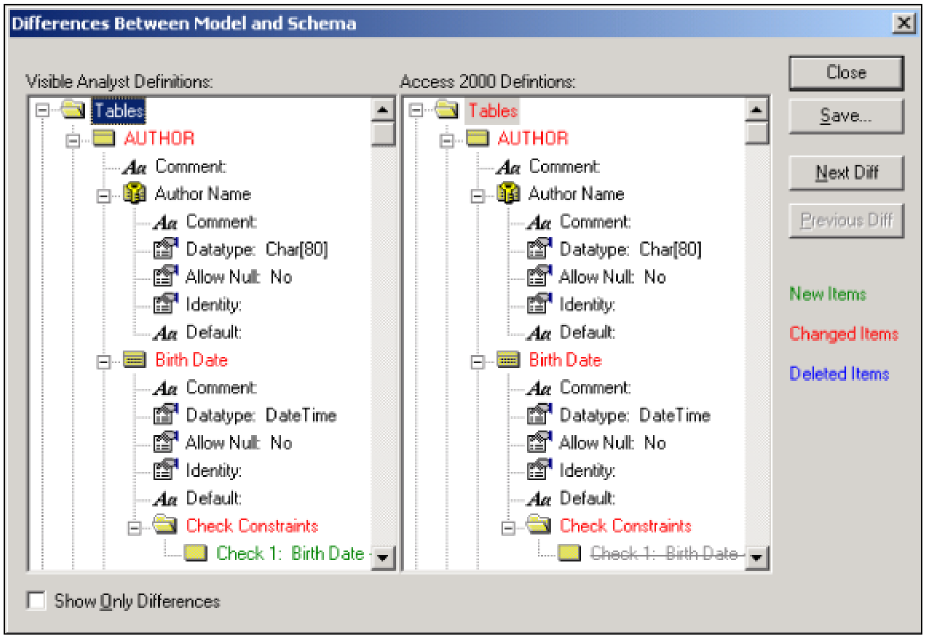
For each tablespace, the following information can be maintained:

- **Pathname.** If Informix SE is checked, the pathname contains the fully qualified filename in which you want to store the database table.
- **Extent.** The length in kilobytes of the first extent for the table.
- **Next.** The length in kilobytes for each subsequent extent.
- **Lock Mode.** The granularity used for locking a table, either *Page* or *Row*.
- **Fill Factor.** A percentage that specifies how full each data page is made when creating a new index on existing data. This value is used as a default for all indexes on the table; you can specify a different fill factor for each index.

COMPARE MODEL AGAINST SCHEMA

When generating a database schema, it is sometimes useful to be able to compare a model to an existing database to examine the changes that have been made. To perform the comparison, select Compare Model Against Schema on the Repository menu. You can use the current settings, or change your generation options including scope and content.

Once the comparison is complete, the Differences dialog box is displayed.



Differences Between Model and Schema

Visible Analyst Definitions:

```
Tables
  AUTHOR
    Comment:
    Author Name
      Comment:
      Datatype: Char[80]
      Allow Null: No
      Identity:
      Default:
    Birth Date
      Comment:
      Datatype: DateTime
      Allow Null: No
      Identity:
      Default:
      Check Constraints
        Check 1: Birth Date
```

Access 2000 Defintions:

```
Tables
  AUTHOR
    Comment:
    Author Name
      Comment:
      Datatype: Char[80]
      Allow Null: No
      Identity:
      Default:
    Birth Date
      Comment:
      Datatype: DateTime
      Allow Null: No
      Identity:
      Default:
      Check Constraints
        ~~Check 1: Birth Date~~
```

Figure 5-38 Differences Between Model and Schema Dialog Box

Objects are displayed in a tree structure with colors used to indicate differences. Items in green are new to the model and do not exist in the database. Items in blue exist only in the database and have been deleted in the model. Items in red are different. You can use the Next and Previous buttons to scroll through the differences. Click Show Only Differences to eliminate items that are the same in both the model and the database. To save the comparison information to an HTML file, click Save.

Note

📋 A number of different factors determine the results of the comparison. One of the most important is the set of generation options that are used when the comparison is started. If the options are changed between the time the original model was generated and when the comparison occurs, you could get unexpected results. For example, if the database schema was created with the option set to replace spaces with underscores and then you perform the comparison with the option turned off, table names and columns may not be identified properly.

DATA DESCRIPTION SPECIFICATION GENERATION

Visible Analyst supports the generation of Data Description Specifications (DDS) for the IBM AS/400. DDS is generated for all entities in the repository and is similar to SQL schema generation (see above). To generate DDS, select Generate DDS from the Repository menu. Visible Analyst scans your project repository and generates the DDS. You can view both the generation errors and the generated DDS, with the option to save or ignore either.

The DDS Name Translation selection available on the Options menu allows you to tailor how object names are generated during AS400 DDS generation and how they are mapped to Visible Analyst object names during DDS import. There are three mapping schemes available:

- Logical to Alias. A Visible Analyst object name is mapped to the ALIAS( ) field.
- Logical to Colhdr. A Visible Analyst object name is mapped to the COLHDR( ) field.
- Logical to Text. A Visible Analyst object name is mapped to the TEXT( ) field.

At this time, generation of views in DDS syntax is not supported.

GENERATING STRUCTURE CHARTS FROM THE REPOSITORY

COBOL program data can be imported from the Application Browser product and stored in the Visible Analyst repository. Structure charts can be created from that imported and stored information.

To use this feature:

- Import Application Browser information using Import.
- Choose Draw VIRTUAL Chart from the File menu.
- Name the diagram when the dialog box requests it.

Visible Analyst scans the repository for structure chart information on a virtual diagram and adds it to a new diagram. The new chart is displayed.

Regenerating Existing Structure Charts

- Open a structure chart diagram.
- Choose Regenerate Chart from the Diagram menu.
- The structure chart is redrawn from the information on the open diagram and any other relevant information in the repository. Save it to make it permanent.

Chapter 6

Local Area Networks

MULTI-USER ACCESS TO PROJECTS

The LAN version of the Visible Analyst adds powerful multi-user capabilities to the basic functionality of Visible Analyst. It offers program access to a potentially unlimited number of users operating over a local area network, while creating little or no noticeable system degradation. With it, projects may be created in such a way that many project team members can concurrently work on different parts of a system design and/or data model, while maintaining a common set of project files, including a common repository.

There are two network versions of Visible Analyst available. The Novell LAN version uses the file and record locking mechanisms within Novell NetWare. The Generic DOS LAN version supports networks other than Novell networks. Any network that performs standard DOS file locking is supported.

Typically, program data and executable files are stored in the \VA directory or its subdirectories on the network file server. There they can be protected against unauthorized access by limiting Visible Analyst privileges to only those users who have been:

- Granted access to the network.
- Granted access (All Rights are necessary) to the \VA directory.
- Granted access to each individual Visible Analyst project.
- Granted rights to modify a project, as opposed to the right to view only.

It is available for all Visible Analyst tool set configurations.

LAN OPERATING CHARACTERISTICS

The operating characteristics of the LAN version of Visible Analyst are virtually identical to those of the single user version, except that you must be logged onto the network prior to accessing the program and starting a work session. Once logged onto the LAN, you have all of the same capabilities that apply to each Visible Analyst tool described throughout this manual. You also have other features and capabilities described below.

LAN Visible Analyst Features and Capabilities

Unlimited Access to Program and Project Files
The number of network users that may access Visible Analyst is limited only by the number of nodes purchased. A “node” is defined as “concurrent access by a user.” For example, three nodes equals the ability to have three users concurrently accessing Visible Analyst. You may assign as many users as you wish to access Visible Analyst, but only three users may run it at any given time.

Limited Access to Diagrams and Repository Entries
Only one user may access a particular diagram or repository entry at any one time, to prevent conflicting edits being entered for a diagram or its project repository entries. The only exception to this rule is if a user opens a diagram in read-only mode. In this case, one user can modify a diagram, and any number of users can open the same diagram for viewing only.

Security
Access to program and project files is limited to users defined as having access to the network, access to the \VA directory, access to each particular project and granted rights to modify a project, as opposed to the right to view only.

Messaging System
The Visible Analyst messaging system allows all users to communicate easily over the network. The messaging feature gives access to NetWare’s messaging while using Visible Analyst and is invoked with function key F5. The Generic LAN version does not support the Visible Analyst messaging system.

Current Activity and Error Message Help Windows
The Current Activity function on the File menu allows you to review the activities of Visible Analyst users working on the same project. It also provides a description of the activity each user is performing within the project. The Error Details window allows you to identify users who are working with project files you are attempting to access.

Novell NetWare Compatibility

The LAN version of Visible Analyst is fully compatible with the network configuration and protocol requirements of Novell Advanced NetWare v1.0 and higher. Workstations that access Visible Analyst over the LAN must be running DOS 3.1 or higher.

All descriptions and instructions provided in this chapter assume that NetWare is already installed, networked workstations or PCs are already connected, users are already knowledgeable in the procedures required to log on and off the network, and the LAN Visible Analyst program is already installed (refer to Getting Started). Additional information and procedures describing how to setup the \VA directory, how to define user security at the project level, and how to use Visible Analyst's networking features are provided in the following pages.

NetWare MAP ROOT Command

The LAN version of Visible Analyst supports the NetWare MAP ROOT command. This means that if you have used this command to map a drive letter to a server directory, Visible Analyst recognizes it and acts accordingly.

Windows Networking

The LAN version of the Visible Analyst should be installed from a client PC onto a common folder on the Windows server. Once installed on the server, map a drive to the server from each client PC and then create an icon on the local PC pointing to the VA32.EXE file on the server. Besides mapping the drive and creating the icon, no additional software is loaded on the client PC's. All users should be assigned the following installation folder rights: Modify, Read & Execute, List Folder Contents, Read, Write. It is not necessary to assign Full Control rights to the users. The same folder rights should be granted to all of the Visible Analyst sub-folders. **Note:** After mapping the drive to the server, install the Visible Analyst to the VA folder, not to the root of the mapped drive.

The newer versions of the Windows operating systems, such as Windows 2000, Windows, XP, Windows Vista, etc. implemented user level security which affect users access to the Visible Analyst in both the single user and LAN versions.

"Product Serialization Failure" Error Mesage

When a user first accesses the Visible Analyst, a Btrieve database key is written to the registry of the users PC. This key is written on the client PC if the user is running the single user or multi-user version of the Visible Analyst. If the Visible Analyst user is not an Admin level user of the PC, the key is not written, and the user is unable to access the Visible Analyst. The error message generated is "Product Serialization Failure" error. To resolve the error, an admin level user of the PC should log onto the PC, access and then exit the Visible Analyst to create the key. The admin level user can assign the non-admin level user logon ID full access rights to the registry key, and the user will be able to access the Visible Analyst.

The Btrieve database engine shipped with the Visible Analyst also writes 2 temporary files to the local PC's TEMP folder. If the non-admin user does not have Full Control rights to the folder, the "Product Serialization Failure" error is generated. To resolve the error, edit the Home and Trace file keys in HKEY_LOCAL_MACHINE\SOFTWARE\Btrieve Technologies\Microkernal Workstation Engine\Version 6.15\Settings.
Change the path listed in these keys to a folder such as C:\temp, where the user can be assigned full control rights to write these temporary files.

If the error message is still generated, contact Visible Systems support at support@visible.com.

Preventing Editing Collisions

There are no inherent restrictions on the number of network users that may access Visible Analyst or Visible Analyst projects concurrently. However, there are restrictions that allow only one user to access a specific diagram or repository entry at any one time. This restriction prevents editing collisions; that is, prevents conflicting edits from being entered into the same diagram or repository entry by different users.

If a potential conflict is detected, Visible Analyst prevents the function from being initiated. You are alerted to the nature of the conflict and the user(s) involved (as shown in Figure 6-4 later in this chapter). For example, global changes that are normally carried throughout multiple diagrams in the same project are inhibited in some instances, such as when a global change impacts multiple diagrams and one or more of those diagrams is currently being accessed by others.

Local Operation

Projects may be stored on a local workstation drive instead of the network drive by specifying the appropriate data path when the project is created. Note, though, that any project stored on a local workstation drive can never be accessed by another workstation; the project becomes a single-user project.

RUNNING ON THE NETWORK

Before proceeding, be sure that the initial steps in Visible Analyst installation process have been completed as described in Getting Started. If you are using the DOS Networks version, you must define users who access Visible Analyst. (See Assigning Access Rights later in this chapter.) The Netware version of Visible Analyst is accessible to all users defined under NetWare; you need not redefine NetWare users, but you must decide how you wish to implement NetWare-level security and the process of creating projects. Appropriate steps for addressing each of these LAN considerations is provided in the following pages.

Defining Default Path Settings

For the LAN version of Visible Analyst, default settings can be accessed from one centralized location so that all users start with the same setup information. These defaults can later be overridden by users. This is most useful to set a network drive and path preferences for projects and temporary files generated by Visible Analyst.

- Edit the VAW.INI file, located in the Visible Analyst installation directory.
- Insert the statement “Default Data Path=” for project files, and insert the appropriate drive and directory.
- Insert the statement “Transient Data Path=” for temporary files, and insert the appropriate drive and directory.

When a user accesses Visible Analyst from a node on the network for the first time, Visible Analyst creates a new initiation file named VAW#.INI, where # is the number the network assigns to the user's node, and copies the contents of VAW.INI into the new file.

Implementing Security

There are two levels of project security available to network users:

- The lower level of NetWare security is the easiest to implement. It allows any user assigned to Visible Analyst to create projects at will.
- The second level of NetWare security requires that the supervisor, or a user with equivalent status, create a subdirectory and assign trustees to any new project before it is created. For details, refer to Creating Projects on the Network later in this chapter. We recommend that you choose between the two levels, though multiple variations are possible. If the two levels described don't meet your needs, please contact Visible Systems technical support.

As shipped, Visible Analyst has as the default data path for project data <path>\VA\DATA\<root>, where <path> is any path you entered for the installation of Visible Analyst and <root> is the project root. In this way each project resides in its own subdirectory. For example, project LIB would reside in <path>\VA\DATA\LIB. We strongly recommended that you leave this default data path setting as shipped as it makes security much simpler to implement and facilitates switching between levels of security. However, it can be changed when you create a project if you so desire. (For the remainder of the chapter, any directory levels above the VA directory are ignored, but you must consider them when you create directories, etc.)

Implementing the Lower Level of NetWare Security

Selecting this level of security allows any user who becomes a trustee of the VA directory the ability to create a project. Any user who creates a project, in addition to the supervisor, may then assign other users access to that project (see Project-Level Security – Assigning Project Access Rights later in this chapter). Note that this security is operable only within Visible Analyst; control is not implemented at the DOS command level. As a result, any user with knowledge of how to access files under DOS may modify or delete any Visible Analyst projects.

To implement the lower level of NetWare security, simply make each user accessing Visible Analyst a trustee with all rights to the VA subdirectory on the install-to drive. By default, assigning a user as a trustee gives that user all rights to that directory and all lower-level directories.

Implementing the Second Level of Security

This level of security provides all of the security of the lower level and adds security at the DOS command level. This means that there is no way for an unauthorized user to modify or delete any project information. When implemented, the supervisor or someone with equivalent rights must assign users to each project before the project is created.

For NetWare Versions Earlier Than 3.0

To implement the second level of security, be sure that all of the following are applied:

- For each user who is to access Visible Analyst, make that user a trustee of the VA subdirectory on the install-to drive. By default, assigning a user as a trustee gives that user all rights to that directory.
- Revoke the Modify and Parental rights for each user for the VA subdirectory.
- For each user who is to access Visible Analyst, make that user a trustee of the VA\DATA subdirectory on the install-to drive.
- Revoke *all* rights for each user for the VA\DATA subdirectory.

For NetWare Versions 3.0 or Later

These NetWare versions implement inheritance rights. To implement the second level of security, be sure that all of the following are applied:

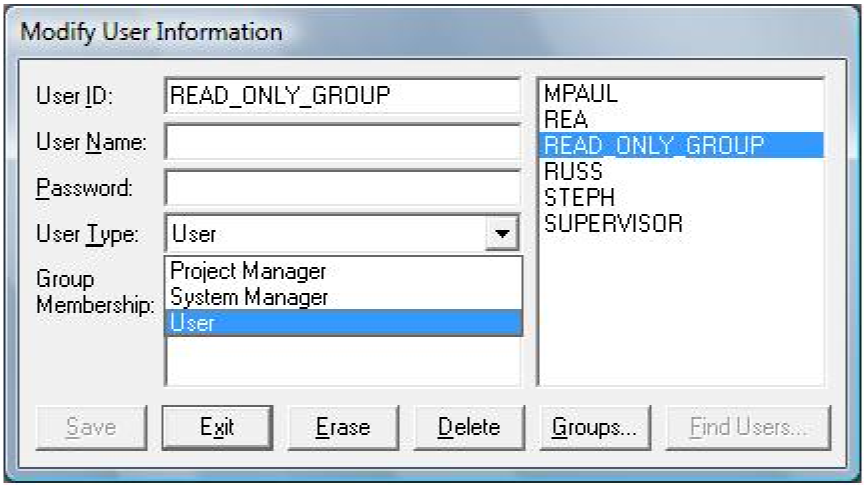
- For each user who is to access Visible Analyst, make that user a trustee of the VA subdirectory on the install-to drive. By default, assigning a user as a trustee gives that user all rights to that directory.
- Revoke the Access Control, Modify and Supervisory rights for each user for the VA subdirectory. (If you want a user to be able to rename a project, do not revoke the Modify rights.)
- Revoke all Inheritance rights except Supervisory for the VA\DATA subdirectory on the install-to drive.

Assigning Access Rights

Users who access Visible Analyst can be assigned one of three levels of security: system manager, project manager, or user. A *system* m*anager* has full rights to all projects in Visible Analyst, can maintain user information, and assign project rights to users. A *project* m*anager* can create projects, and assign users to the projects they create. A *user* can access Visible Analyst and be granted rights to projects, but cannot create new projects. If you installed Visible Analyst using the Novell version (Visible Analyst automatically installs the Novell version if your network is listed in the Windows Control Panel as Novell Netware OS at installation time), initially you must have the supervisor (or someone with security equivalent to the supervisor) log into Visible Analyst and assign at least one system manager. By default any user who can access the Novell server can start Visible Analyst without having been granted specific access rights; but they can only be a user and do not have rights to any existing projects or to be able to create new projects. If you are using the DOS Networks version, you must create user IDs and passwords to secure Visible Analyst access to only those users who have been defined. To access Visible Analyst for the first time, log in as supervisor with no password.

To assign user access privileges to Visible Analyst:

Select Users: 1 Select Users from Tools menu to display the Add User dialog box (see Figure 6-1).



Modify User Information

User ID: READ_ONLY_GROUP

User Name:

Password:

User Type: User

Group Membership:

Figure 6-1 Add User Dialog Box

2 If you want to change the information about a previously defined user, highlight the desired user ID in the list.

Enter the User ID: 3 Enter the user name to be used to log on to Visible Analyst. If you are using Netware, this name should be the same as the user name used for login. This field can only be changed by a system manager.

Enter the User Name: 4 This is an optional field used to further describe the user.

Enter the Password: 5 The password is used by the DOS Networks version to further enhance security. If a password is defined, both the user ID and password must be provided when starting Visible Analyst.

Select the User Type: 6 Select the user type to be assigned to this user. This field can only be changed by a system manager.

Save Your Changes: 7 Click Save to record your changes, Erase to cancel the changes and clear the selection, or Delete to remove the user from the list.

Creating User Groups

Users who access Visible Analyst can be assigned membership in up to 34 groups. A group is simply another user, which provides security equivalence. For example, if user A has security equivalence to user B by adding user B to user A's group membership list, when user B is assigned to a project, user A has the same rights to that project, even if user A was not assigned directly to the project.

Groups can also be used to assign a group of users specific access rights to individual projects. All members of the group inherit the group’s project access rights as explained in the Project Level Security section of this chapter. However, if a user is granted individual access to a project with more assigned rights than the rights assigned to the group, the users rights to the project override the groups assigned access rights. The rights assigned to the group are on a project-by-project basis; so one project may be Read Only for members of the group, while another project may be assigned full project access rights.

Groups can contain other groups or users. Visible Analyst expands the group list to contain all subgroups when checking security access to a project.

The System Managers maintain the Group Membership list.

Note

If you are using the Novell Netware driver, both NetWare groups and Visible Analyst groups can be used together to provide access to projects.

To create a group and add users to the group:

Add the User Group: 1 Select Users from Tools menu to display the Add User dialog box (see Figure 6-1).

Enter the Group ID: 2 Enter the name of the group in the User Id field.

Enter the Password: 3 The password is used by the DOS Networks version to further enhance security. If a password is defined, both the user ID and password must be provided when starting Visible Analyst. Users would not normally use the group ID to access a project, but their own user ID.

Select the User Type: 4 Select the user type to be assigned to this group. This field can only be changed by a system manager.

Save Your Changes : 5 Click Save to record your changes.

Add Group Users: 6 Select the user (not the Group ID) from the user list and click the Group button to display the Update Group List dialog as shown in Figure 6-2. Use the << or >> buttons to add or remove a user from the group.

Update User Group *7* *Click the Update button to save the group membership. Use*

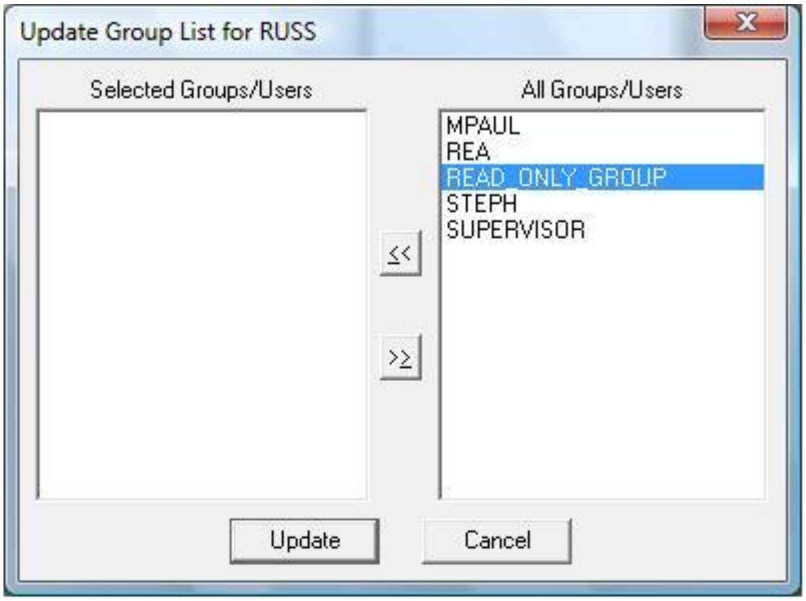


Figure 6-2 Add User Dialog Box

Creating Projects on the Network

The steps involved in creating projects on the network depend on the security level you have chosen, as described earlier in this section. With the lower level of NetWare security, any user with access to Visible Analyst may create projects. With the second level of security, two steps must be taken prior to a project being created with Visible Analyst, as described below:

- The supervisor, or a user with equivalence, must create a subdirectory for the project. Under the recommended data path setting this would be VA\DATA\<root>, where <root> is the project root of the project to be created. For example, if a user wants to create a project with the root LIB, the supervisor must create the subdirectory VA\DATA\LIB on the file server.
- Any user who is to be assigned to the project must be assigned as a trustee to the subdirectory created in the step above. Note that even after doing this, users must still be assigned to projects from within Visible Analyst.

Note

Only system managers and project managers can create projects.

Project-Level Security

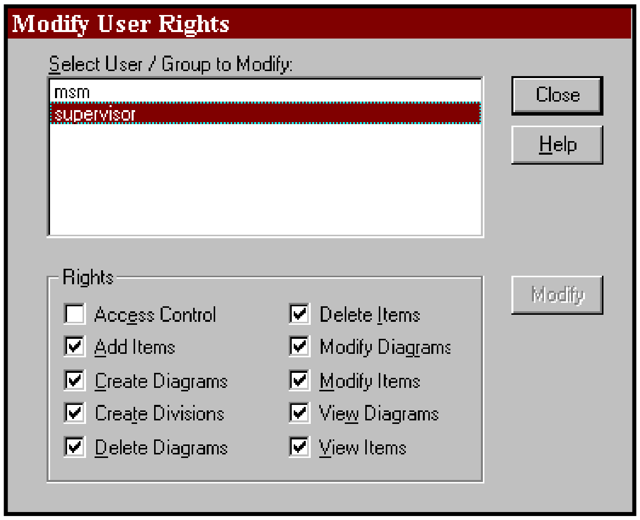
The LAN version of Visible Analyst gives only the network supervisor and the user who creates a new project the capability to assign network access privileges to the project. Access may be assigned to any users having access to the directory where the project is stored. Access can be limited to viewing project information.

To assign user access privileges to any Visible Analyst project that you create:

Select Modify User List: 1 Select Modify User List from the File menu.

Select the Access Function: 2 Select Add, Delete or Modify Rights from the submenu.

- If you are adding users, Visible Analyst displays a dialog box listing the IDs of network users and user groups *not* currently assigned access to the project.
- If you are deleting users, Visible Analyst displays a dialog box listing the IDs of network users who *are* currently assigned access to the project.
- If you are modifying user rights, Visible Analyst displays a dialog box listing the IDs of network users who *are* currently assigned access to the project and the rights each has to access the project. (See Figure 6-3.)



Modify User Rights

Select User / Group to Modify:

msm
supervisor

Rights

☐ Access Control
☑ Add Items
☑ Create Diagrams
☑ Create Divisions
☑ Delete Diagrams
☑ Delete Items
☑ Modify Diagrams
☑ Modify Items
☑ View Diagrams
☑ View Items

Figure 6-3 Modify User Rights Dialog Box

Review the Displayed List of Users and Make Desired Changes: 3 Add or delete network users or user groups as desired. Note that until users have been added to a project, these functions are disabled. Note further that initially, users have all rights to a project.

Modify Users Rights to the Project: 4 To modify the rights of any user who should have less than full access to the project, select Modify User Rights and click the ID of a user or group. The rights currently available to that user or group are checked in the list at the bottom of the dialog box. (See Figure 6-3.) Review the rights of the project users and toggle individual rights on or off by clicking on them. For example, if the only item checked on the list is View Diagram, that user can look at a diagram but cannot make changes. These rights can be changed later if necessary.

Note

- To restrict user access to individual objects or diagrams, you must use divisions. See Creating Divisions in the Enterprise Modeling chapter for details.

MESSAGING

Visible Analyst provides access to NetWare's messaging capability. The feature allows you to communicate with other network users from anywhere within Visible Analyst. Note that messaging is available to all network users, regardless of whether or not they currently have Visible Analyst active; messaging is a function of NetWare rather than Visible Analyst. You can therefore send a message from your workstation once logged onto the network, or you can send a message from anywhere in Visible Analyst. The Generic LAN version does not support the Visible Analyst messaging system.

To send or receive a message while in Visible Analyst:

Initiate Message Sending with the F5 Key: 1 Press the F5 function key at your workstation to initiate a message. Visible Analyst responds by displaying the Send A Message dialog box. See Figure 6-4.

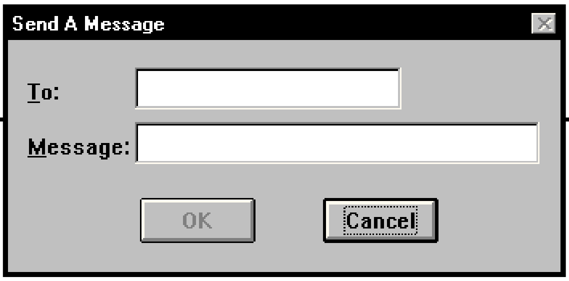


Figure 6-4 Send Network Message Dialog Box

Enter the ID of the User To Receive Your Message:

2 Type the user ID of the recipient network user in the To box. Then type your message in the Message box. Each message is limited to 40 characters. Click OK to send the message.

If a Network Message is Received:

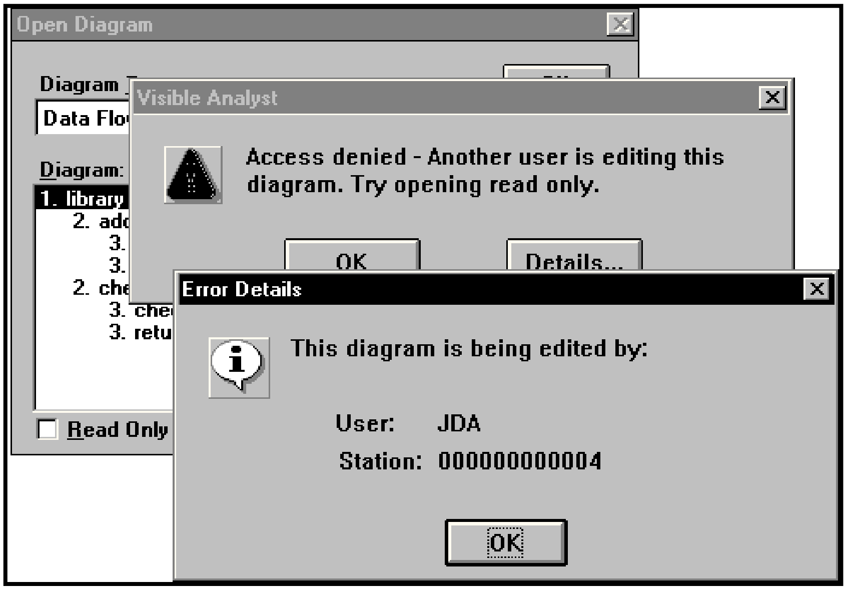
3 Messages sent to your ID from another user automatically appear in a network message box on your screen if the program NWPOPUP.EXE is running. You must instruct Windows to start this program when Windows starts running. A message does not interfere with work you are doing. Note that messages sent to you appear on all workstations where you are logged in.

REVIEWING PROJECT ACTIVITY

Whenever a Visible Analyst project that is stored on the network is selected at a workstation, you can display the current activity and the project history. The Current Activity function on the File menu displays who is currently accessing the project. The current activity function presents a list of all users who currently have the project selected and what functions they are performing. The current activity is updated by Visible Analyst every 10 seconds. The Project History function, also on the File menu, presents a list of diagrams and the users who last edited each of them.

LAN HELP FUNCTION

Occasionally while using the LAN version of Visible Analyst, you may attempt to access a diagram or repository entry that is already selected by another LAN user. Since only one user may access any diagram or repository entry at any one time, you receive an error message that explains that the diagram or repository entry is currently being edited; and you are prompted to click OK to continue. You also have the opportunity to access the LAN help function by clicking the Details button. Visible Analyst responds by displaying a window titled Error Details (see Figure 6-4). The Error Details window displays the ID of the network user who currently has the item selected, as well as the network station (the physical address that the workstation occupies on the network). Note that if a conflict exists with more than one other user, the Error Details window appears only for the first user who accessed the diagram or entry. Once you know the user accessing the item you want, you can send a message to that user. The Generic LAN version does not support the Visible Analyst messaging system.



Access denied - Another user is editing this diagram. Try opening read only.

This diagram is being edited by:

User: JDA

Station: 000000000004

Figure 6-4 Typical Network Help Details Window

While performing various Visible Analyst activities, you may be interrupted by messages indicating that you cannot perform a selected function due to a potential editing collision. In these instances, you can access the LAN help function described above and determine the user currently working with the function you tried to select. Again, you can send a message to that user if you wish.

Chapter 7

Visible Analyst Tools

GENERAL INFORMATION

Visible Analyst includes a Tools menu (see Figure 7-1) that allows you to perform data conversion and copying functions at the project level. These functions are an important part of the overall "housekeeping" tasks that should be used to backup and maintain the organization of your project files. Not all of these functions are available in the single user Visible Analyst Student Edition.

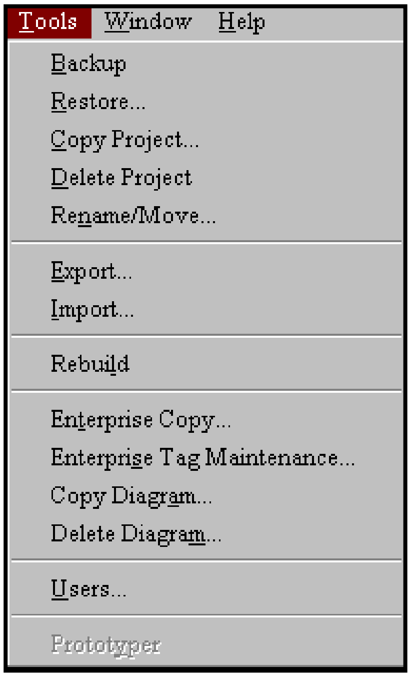


Figure 7-1 Tools Menu

- **Backup** allows you to back up a Visible Analyst project to one or more floppy disks, to a local hard disk or, for the LAN version, to a directory on a file server.

- **Restore** allows you to restore a Visible Analyst project from one or more floppy disks, from a local hard disk, or from a file server.
- **Copy Project** allows you to duplicate a Visible Analyst project.
- **Delete Project** allows you to eliminate a Visible Analyst project.
- **Rename/Move** allows you to change the name (root) that was previously defined for a project, or to rename the directory where its files are stored, or to move the project files to a different directory.
- **Export** allows you to convert data repository files for a project into ASCII format for use in other programs such as a database management system or into one of a number of other export formats.
- **Import** allows you to load data into a project data from an ASCII file or from one of a number of other import formats.
- **Rebuild** allows you to convert a project from an earlier Visible Analyst version format to the current format or to recreate a project that may be corrupted due to a power failure while Visible Analyst is in use, etc.
- **Enterprise Copy** Allows you to copy a division between an enterprise project and a satellite project.
- **Enterprise Tag Maintenance** allows you to remove the link between an enterprise project and a satellite project.
- **Copy Diagram** allows you to copy a diagram or a branch (a data flow diagram and all of its children/grandchildren). This command allows copying similar-methodology diagrams and branches from one project to another. All data repository information associated with the diagram(s) is also copied into the new project. Copy Diagram can be used to change the symbol methodology when copying data flow diagrams into a project using a different methodology symbol set. Copy Diagram can also be used to change the ERD Cardinality Notation to the notation defined for the Copied To project.
- **Delete Diagram** allows you to delete a diagram or a branch.
- **Users** allows you to define user security information, including user type. For the single-user and generic DOS network versions of Visible Analyst, you can define user names and passwords.
- **Prototyper** allows you to start the Visible Prototyper. If you purchased this tool, you can work with the prototyping and simulation capabilities.

BACKING UP A PROJECT

The Backup selection on the Tools menu creates backup copies of all files for the currently selected project. If the drive selected in the dialog box is a floppy drive, a DOS-formatted disk must be inserted into the drive. The formatted disk does not have to be blank; it may contain unrelated data, or it may contain data for the same project or other projects being backed up. If it does contain data for the same project, you are prompted to indicate whether or not you want the project data on the disk to be overwritten. Also, if the disk you are copying to becomes full, you are prompted to insert additional disks to complete the backup.

If the backup destination is a hard disk, such as a file server, any valid drive and path specification can be entered. If the destination drive becomes full, Visible Analyst asks you to specify a new location for the backup. When you enter one, the backup continues from where it left off, with the remaining data going to the new location.

Notes

- It is ***very important*** to remember that, if a backup is split between two disks or directories, as in the previous paragraph, you *must not* manually consolidate the files into one directory. Project files that have to be split between the two locations have the *same name* and manually consolidating all of the backup files into a single directory *overwrites* one of the files, thus losing backed up data *permanently*.
- Anytime a project is backed up, it should be restored *only* by the Visible Analyst Restore tool. Do not attempt to restore Visible Analyst projects using DOS commands such as COPY. The Visible Analyst project master file is not updated correctly.
- If you run the Backup tool while one or more diagrams are being edited, the diagrams are closed and you are prompted to save any change before the backup procedure begins.

RESTORING A PROJECT

The Restore selection on the Tools menu allows you to restore a project that has been previously backed up. When the project restore is performed, all diagrams and repository entries for the defined project are copied from the backup drive or directory.

If multiple floppy disks were used to backup the project, you are prompted to insert each disk in the order that the backup was performed to restore the project. If the backup was split over multiple hard disk directories, you are prompted to enter the path where the next group of backed up files resides. If the hard disk being copied to already contains data for the project being restored, the data read from the backup disk overwrites the existing hard disk project files.

Note

- In the multi-user version, anyone may restore a project. However, only those who have been granted rights can access it thereafter. If a project was created by a single-user version of Visible Analyst with security turned off, Visible Analyst does not know who created it. Thus, when restored into a LAN version, Visible Analyst doesn’t have creator information. As a result, only a system manager, supervisor, or another user assigned supervisor equivalence is able to access it until access rights have been granted to others.

COPYING A PROJECT

The Copy Project selection on the Tools menu allows you to duplicate an existing project, providing a means to recreate a project without having to redraw the diagrams. It therefore provides a convenient and time-saving method for starting new projects that are similar to existing ones. For example, you can select a project and copy it to a new project root, then revise and edit the duplicated files to develop unique project entities. The new project files become another stored Visible Analyst project that can be manipulated in the same manner as any other Visible Analyst project.

Before any copy is initiated, you are prompted to define a new project name for the duplicate project, a new description and a location in which to store. The new root name provides a calling label for the duplicated project files. Note that all project files are copied to the new project root, including any existing repository files.

DELETING A PROJECT

The Delete Project selection on the Tools menu allows you to eliminate the current project from Visible Analyst. It is useful as a housekeeping function to erase project files that are no longer needed. Note that any project that you delete is erased completely, including all repository entries. Before any deletion is executed, Visible Analyst prompts you to confirm the project being deleted. This helps prevent erasing the wrong project; however, it is recommended that you back up all project files using the Backup tool described above, including those that you are about to delete. This allows you to maintain a project archive, and lets you restore any project at any time.

Deleting Projects with No Project Files

It sometimes happens that someone deletes the data files for a project from DOS instead of doing it from within Visible Analyst. The result is that the project is still listed in the project master file, but cannot be accessed because the data files are gone. To delete a dangling project from the master file, open the Select Project dialog box from the File menu. Select the project name you want to delete from the list or type its name into the Project Name box, then click Remove. The project name is removed from the project master file after confirmation.

The Remove feature is not available in the Visible Analyst student edition. To remove the project from the project list, select the project in the Project Name dialog, hold down the Alt key and press the R key on the keyboard. When prompted to remove the project from the project master file, click OK.

Note

- This function is not a substitute for the Delete Project function. It should only be used in the circumstances described above.

RENAMING OR MOVING A PROJECT

The Rename/Move selection at the Tools menu allows you to change the name or description of an existing project and/or to move it to a different directory. This is another tool that can be helpful when performing housekeeping functions for your Visible Analyst files. When renaming a project, only the one-to-four character root is changed. The project and diagram labels remain the same. If you have decided that the project root should also be the name of the subdirectory in which project files are to be placed, this directory name is also changed.

REBUILDING A PROJECT

The Rebuild selection on the Tools menu allows you to recreate project files. This may become necessary for either of the following situations:

- If your Visible Analyst program software has been updated from an earlier version, you may have to rebuild existing project files to convert them to a format that is compatible with the new version. Visible Analyst prompts you to perform the rebuild whenever you access a project for which a rebuild is required.

Note

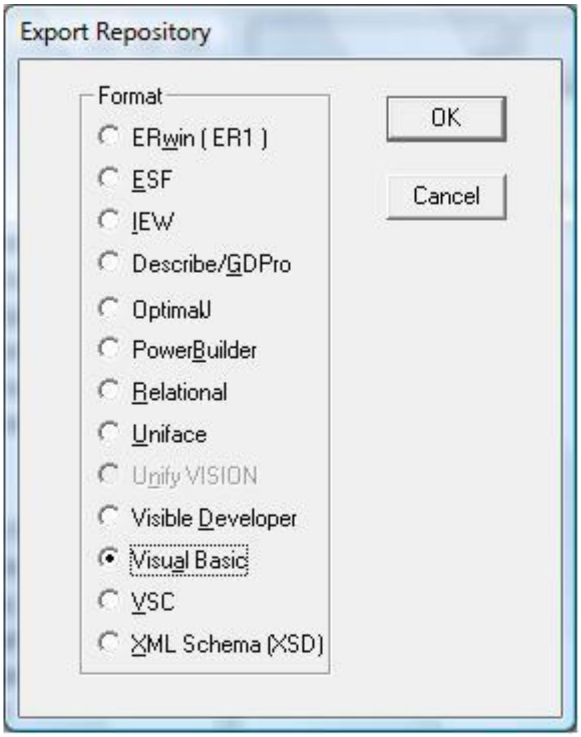
- If a project was created with a version of Visible Analyst prior to version 6, contact Visible Systems for assistance.

- If a project becomes corrupted due to a power outage while Visible Analyst is in use, a bad disk sector, etc., you may be able to salvage all or part of the project files by performing a rebuild. Visible Analyst helps prevent loss of project data during these situations; therefore, this type of rebuild should rarely be necessary.

All of the creation dates and last-altered dates are reset to the current date whenever a project is rebuilt.

EXPORTING REPOSITORY DATA

The Export selection at the Tools menu allows you to convert data repository files for a selected project into ASCII formatted data files for use by other programs. For example, you can export repository information into a file that can be accessed by a database management program.



Export Repository

Format

○ ERwin (ER1)
○ ESF
○ IEW
○ Describe/GDPro
○ OptimalJ
○ PowerBuilder
○ Relational
○ Uniface
○ Unify VISION
○ Visible Developer
● Visual Basic
○ VSC
○ XML Schema (XSD)

Figure 7-2 Repository Export Dialog Box

You can export ASCII data from the repository in one of several formats:

- A relational format that is compatible with most relational databases.
- A proprietary VSC format that is compatible with the ASCII dump used with prior versions of Visible Analyst and other Visible Systems modeling tools.
- KnowledgeWare (IEW/ADW) format, discussed together with the KnowledgeWare import, later in this chapter.
- ESF format, discussed in the section on Shell Code Generation in The Visible Repository.
- SQL format, discussed in the section on SQL Schema Generation in The Visible Repository.
- Powersoft PowerBuilder format.
- IBM AS/400 DDS format, described later in this chapter and also in the section on Data Description Specification Generation in The Visible Repository.

- Visible Prototyper export.
- Visible Developer
- RDBMS Catalog export, discussed in the section on SQL Schema Generation in The Visible Repository.
- GDPro format.
- ERwin format.
- CompuWare OptimalJ format.
- XML Schema (XSD) Export for entities and classes based on W3C standard.

These export formats are selected from the export dialog box, except for SQL and DDS that are generated from selections on the Repository menu. Visible Developer 4.x can be generated from the Export menu or from the Repository menu.

Relational Database Repository Exports

In database terminology, a relation is a common field or fields shared by two different files or tables. The repository in Visible Analyst is a relational database, with files a set of tables containing information about a project linked together by relations. The repository export function converts the internal representation used by the database engine to a delimited text file, as well as eliminating information either proprietary or simply not useful to you. The relation linking the exported files together is an ID number that uniquely identifies each object in the repository.

Export files are named by the project root plus file description method. For example, files in a repository export of a project named LIB would be named as shown below.

LIBROBJ.DTA Primary object file. One record for each distinct entry in the repository.

LIBATTR.DTA Attributes file. Each repository object has properties that are described in this file such as descriptive information, notes, and composition values. Attributes that require more than one record have an associated sequence number. See the table below for a list of valid attribute types.

LIBDIAG.DTA Diagram file. Each diagram exists as a separate record.

LIBDLOC.DTA Location file. Each object location exists as a separate record.

LIBCLOC.DTA Connected location file. Each pair of connected object locations exists as a separate record.

LIBRELA.DTA Related object file. Each object that is related to another object, such as an alias entry, exists as a separate record.

LIBRELL.DTA	Related list file. Each object that is related to a related object, such as columns for a trigger that belongs to an entity, exists as a separate record.

Relational Export File Layout

Files created by the Export program are in delimited ASCII format. This means:

- Fields are separated by commas.
- Character fields have trailing spaces removed and are surrounded by double quotes.
- Numbers are converted to their ASCII representations.
- Null character fields are denoted by “”.
- Fields contain no right or left padding.
- There is no alignment on decimal points.
- Each record ends in a carriage return and a line feed.

Relational Export File Structure

The structure of any database files created from data exported from the repository must be set up to accommodate the maximum field lengths possible for exported data. The field names, field types and maximum field lengths for each file are listed in Table 7-1.

Table 7-1
Relational Export File Structures

Structure for: - ROBJ.DTA				
Field	*Field Name*	*Type*	*Width*	*Dec*
1	ObjectNameLC	Character	128	0
2	ObjectType	Numeric	4	0
3	ObjectID	Numeric	4	0
4	DivisionID	Numeric	4	0
5	ClassID	Numeric	4	0
6	VersonID	Numeric	4	0
7	ObjectName	Character	128	0
8	ObjectSubtype	Numeric	2	0
9	ObjectOwnerID	Numeric	4	0
10	CreationDate	Date	10	0
11	CreationUserID	Numeric	4	0
12	ModificationDate	Date	10	0
13	ModificationUserID	Numeric	4	0
14	AssociatedNumber	Numeric	4	0
TOTAL			314	
Structure for: X ALTTR.DTA				
Field	*Field Name*	*Type*	*Width*	*Dec*
1	ObjectID	Numeric	4	0

Table 7-1
Relational Export File Structures

2	AttributeType	Numeric	4	0
3	Sequence	Numeric	4	0
4	AttributeLine	Character	254	0
TOTAL			266	

Structure for: X DIAG.DTA

Field	*Field Name*	*Type*	*Width*	*Dec*
1	DiagramType	Numeric	2	
2	DiagramName	Character	40	0
3	DiagramID	Numeric	4	0
4	VersionID	Numeric	4	0
5	ObjectOwner	Numeric	4	0
6	AlphaDesignator	Character	40	0
7	LineStyle	Numeric	2	0
8	PageSize	Numeric	2	0
9	DiagramProcNum	Character	40	0
10	CreationDate	Date	10	0
11	CreationUserID	Numeric	4	0
12	ModificationDate	Date	10	0
13	ModificationUserID	Numeric	4	0
TOTAL			166	

Structure for: X DLOC.DTA

Field	*Field Name*	*Type*	*Width*	*Dec*
1	ObjectID	Numeric	4	0
2	LocationNumber	Numeric	2	0
3	DiagramID	Numeric	4	0
4	UpperLeftX	Numeric	4	0
5	UpperLeftY	Numeric	4	0
6	LowerRightX	Numeric	4	0
7	LowerRightY	Numeric	4	0
TOTAL			26	

Structure for: X CLO.DTA

Field	*Field Name*	*Type*	*Width*	*Dec*
1	ObjectID	Numeric	4	0
2	LocationNumer	Numeric	2	0
3	ConnectedID	Numeric	4	0

Table 7-1
Relational Export File Structures

4	ConnectedLocNum	Numeric	2	0
5	ConnectionStyle	Numeric	2	0
TOTAL			14	

Structure for: X RELA.DTA

Field	*Field Name*	*Type*	*Width*	*Dec*
1	ObjectID	Numeric	4	0
2	AttributeType	Numeric	4	0
3	Sequence	Numeric	4	0
4	RelatedID	Numeric	4	0
5	Item1	Numeric	4	0
6	Item2	Numeric	4	0
TOTAL			24	

Structure for: X RELL.DTA

Field	*Field Name*	*Type*	*Width*	*Dec*
1	ObjectID	Numeric	4	0
2	AttributeType	Numeric	4	0
3	Sequence	Numeric	4	0
4	RelatedID	Numeric	4	0
5	AssociatedID	Numeric	4	0
6	Item1	Numeric	4	0
7	Item2	Numeric	4	0
TOTAL			28	

Note

A number of attribute type fields contain multiple pieces of information separated by hex characters. In Table 7-2 below, these separators are indicated by the notation <n> where n can be a number between 1 and 10.

Table 7-2
Attribute Types Used in: ATTR.DTA

Type	*Description*	*Layout*
0	Description	“Text”
3	Values & Meanings	“Text”
4	Composition	“Text”
5	Modules Contained	“Text”
6	Process Description	“Text”
7	Module Description	“Text”
8	Function Description	“Text”

Table 7-2
Attribute Types Used in: ATTR.DTA

Type	*Description*	*Layout*
9	Notes	“Text”
12	Physical Information	“Storagetype<1>StorageLength <2>DecimalPlaces<3>DefaultValue<4> PictureClause<5>Owner<6>AllowNull”
18	Process Number	“Text”
19	Invocation Type	“Text”
20	DM Information	“Type”
21	Unique Index	“IndexName<1>IndexType”
22	Performance Index	“IndexName<1>IndexType”
23	Relationship	“ControllingKey<1>Flags<2> FromToCard<3>ToFromCard<4> OnDelete<5>OnUpdate<6> FromToCardText<7> ToFromCardText<8>ConstraintName”
26	Parameter	“Text”
27	Class Information	“Cardinality<1>CardText<2>Flags”
28	Database File	“FileName, Size, Reuse” Sequence is *Database type**1000+ Sequence where *DatabaseType* equals 1 for Oracle and 2 for SQLServer “Indextype, SortedData, FillFactor, IgnoreDupKey, DupRow” for SQLServer Sequence is Database Type *1000+ SequenceType where *Database Type* equals 1 for Oracle and 2 for SQLServer, and *SequenceType* is 0 to define physical information for the table, 100+ key number for unique keys, and 200+ index number for performance indexes. (Key number for the primary key is zero.
100	User Object Definition *projectcontrolrecord*	“ObjectTypeName<1>LinkSupport<2> LinkToType<3>LinkToCard<4> LinkDisplayName<5>CompSupport<6> CompType<7>CompDisplayName”

Table 7-2
Attribute Types Used in: ATTR.DTA

Type	*Description*	*Layout*
101	User Attribute Definition *projectcontrolrecord*	"AttributeName<1>StorageType<2> StorageLength<3>StorageDecimal"
1000+	User Attribute	"Text"
2000+	User Object Line	"Text"

Table 7-3
Attribute Types Used In: RELA.DTA

Type	*Description*	*Layout*
1	Alias	Sequence – ORder number of Alias entry RelatedID – ObjectID of Alias entry
2	Related To	Sequence – Order number of related entry RelatedID – ObjectID of related entry
10	Associator Element	RelatedID – ObjectID of associator
11	Data Only Module	RelatedID – Object of data only module
15	Trigger	RelatedID – ObjectID of trigger Item1 – Trigger Flags AssociatedID ObjectID of trigger column (RELL.DTA)
16	Check	RelatedID – Object of check AssociateID – ObjectID of check column (RELL.DTA)
21	Unique Key	Sequence – Key number RelatedID – ObjectID of column Item1 – Sequence number that specifies the order of the column in the key Item 2 – Index order (0 = ASC, 1 = DESC)
22	Performance Index	Sequence – Key number RelatedID – ObjectID of column Item1 – Sequence number that specifies the order of the column in the key Item2 – Index order (0 = ASC, 1 = DESC)
23	Relationship	Sequence 0 –From Entity 1 – To Entity 2 – From-To-Relationship 3 – To-From Relationship 4 – Discriminator

Table 7-3
Attribute Types Used In: RELA.DTA

		5 – From Role 6 – To Role 7 – From Qualifier 8 – To Qualifier RelatedID – ObjectID of related item (RELL.DTA) The following is set only if sequence is 1. AssociatedID – Column that is part of the key Item1 – Sequence number that specifies the order of the column in the key Item2 – Index order (0 = ASC, 1 = DESC)
25	Friend	RelatedID – ID of friend class or module
29	Database Phys Inf.	Sequence – See definition in ATTR.DTA RelatedID – ObjectID of Tablespace

Note 1: The User Object Definition and User Attribute Definition attribute types are only used with the system object *projectcontrolrecord.*

Note 2: For Trigger, Check, and Relationship attribute types, there will be records in both the RELA.DTA and RELL.DTA tables.

VSC-Format Repository Exports

When you perform an Export with VSC format selected, Visible Analyst dumps a project data repository to an ASCII format file. The file name of the resulting ASCII file is root + .AD where root is the project root. A layout of the resulting ASCII file is shown in Table 7-4. All fields except those describing the number of description, notes, location, etc., field records are enclosed in double quotes.

Table 7-4
STRUCTUR.DBF File Structure

Field	*Field Name*	*Type*	*Width*	*Dec*
1	FIELD_NAME	Character	10	0
2	FIELD_TYPE	Numeric	2	0
3	FIELD_LEN	Numeric	3	0
4	FIELD_DEC	Numeric	3	0
TOTAL			18	

Note 1: Describes the format of the structure files.

The export file is a variable record length file that is divided into three sections. The first section is the header that is a single record that gives basic information about the file contents.

The next section defines the user attributes and objects that are contained in the project. The third section contains the repository objects, each defined as an Object Block. The Object Block is comprised of an Object Header followed by zero or more Field Blocks and an Object Footer. The order of the Field Blocks within the Object Block is not important except that the KEYS, CHECK, and TRIGGER fields must be present after the COMPOSITION or CLASS COMPOSITION Field Block in the specification.

Note

- A number of fields contain multiple pieces of information separated by hex 01 characters (which appear on your screen as “smiley faces” when using DOS utilities), <crlf> indicates a carriage return line feed sequence.

Table 7-5

Repository Export VSC Format

Record 1 – Header Record	“project root”, “File creation date”, “File creation time”[,User Attribute Count [, User Entry Type Count]]<crlf>
User Attribute Information	“user defined name”, type, length, description<crlf> Note: The number of records is indicated by the User Defined Count in the header record
User Entry Type Information	“user entry type name”, LinkSupport, “LinkToType”, LinkToCardinality, “LinkDisplayName”, CompSupport, “CompType”, “CompDisplayName”<crlf> LinkSupport = digit 0 – no support, 1 – supported LinkToType = VRE standard or aggregate type LinkToCardinality = digit 0 – 1:1, l – l:m, 2 – m:1, 3– m:m LinkDisplayName = Text name for the define dialog box CompSupport = digit 0 – no support, 1 – supported CompType = VRE standard or aggregate type CompDisplayName = Text name for the define dialog box Note: The number of records is indicated by the User Entry Type Count in the header record.
Subsequent Records Object Header	
Item ID record	ENTRY “Entry Key”, “Entry Type | User Defined Name”, “Additional Information” [, “[Time Stamp]”] <crlf>
Entry Key	[Class Name ::] Entry Name [::: [Parameter Key]] This is the ‘ENTRY LABEL’ in the Repository. Class

Table 7-5
Repository Export VSC Format

	Name is the name of the class that owns the object. This is an optional field. Entry Name is the key field that is used in the Define dialog box as the primary name. Both Class Name and Entry Name are ASCII fields up to 128 characters in length. Parameter Key is used for Module types to define the argument types that uniquely identify an instance of a function. Each argument type is an object in the repository of type class, data element, or data structure, and is followed by an ampersand (&) if the parameter is passed by value or reference, or an asterisk (*) if it is passed by address. There is no limit to the number of parameters a function may contain. The colons are required even if there are no parameters. Note: It is possible for there to be duplicate keys (this can happen if there are files and data flows with the same name).
Entry Type	This is a number that indicates the type of entry as follows: 0 = Data element 1 = Process 2 = File, Data Store 3 = Source/Sink, External Entity 4 = Data Flow 5 = Miscellaneous 6 = Alias 7 = Data Structure 8 = Domain 14 = Function 15 = Module 16 = Library Module 17 = Macro 18 = Library Macro 19 = Data Only Module 20 = Info Cluster 21 = Data Couple 22 = Control Couple 23 = Generic Couple 24 = Data Interface Table Row 25 = Control Interface Table Row 26 = Generic Interface Table Row

Table 7-5
Repository Export VSC Format

	28 = Program 34 = Entity 35 = Associative Entity 36 = Attributive Entity 37 = Relationship Name 38 = Cluster 40 = Relationship Line 43 = Local Data Element 45 = Class 46 = State 47 = Event 48 = Tablespace 50 = View Object 52 = Actor 53 = Use Case 54 – System Boundary 55 = Object (Instance of a Class) 56 = Message (Instance of a Class Method)
SubType	If entry type is greater than 65535, then the low order word contains the entry type as described above, and the high order word contains the subtype. This optional field applies only to module types and classes and is used to denote the following special classes: **Module Subtypes** 0 = Standard 1 = Stored Procedure 2 = Check Constraint 3 = Trigger **Class Subtypes** 0 = Standard 5 = Structure 6 = Union 7 = Element 8 = Entity 9 = Associative Entity 10 = Attributive Entity 13 = Domain 14 = View
User Defined Name	This is the name of the user defined entry type and it

Table 7-5
Repository Export VSC Format

	replaces the Entry Type.
Additional Information	For aliases, this field contains the owner entry key and the owner entry type separated by hex 01 characters. For relationship name and relationship line entries, this field contains the From Entity name, To Entity name, From-To Relationship name and To-From Relationship name, separated by hex 01 characters. For structure chart items appearing on a virtual diagram: For Invocation Lines, special compound type of all information necessary to create a virtual diagram location as follows: Type, bounding box, source module, destination module (separated by hex 01 characters). type = Invocation_Normal[0x01], Invocation_Data[0x02], Invocation_Lex[0x03] (bounding box = 1sx, 1sy, 1ex, 1ey) Note: The modules referenced in this record must also be included in the same import file. In all other cases this field is empty.
Time Stamp	This optional field holds the Create and Modify dates for the object. It is used for import only. The format is Create Date followed by Modify Date, separated by hex 01 characters, as follows: yyyy-mm-dd, yyyy-mm-dd (separated by hex 01 characters)
Object Footer	END ENTRY <crlf>
Object Fields	FIELD “Field Type Name”, Number of Records<crlf> The basic format for all object fields is a header that specifies the name of the field and the number of lines in the body of the field. Many types use “plain text” that is defined as a list of lines each with a single quoted string with a maximum line length of 254 characters after processing. Standard Analyst Text means that the field uses the same format per line as the Analyst define screen field bearing the same name. Entry names can be up to 128 characters in length. Names in fields use Entry Key format unless the Entry

Table 7-5
Repository Export VSC Format

	key information is redundant; i.e., Local data elements listed in a class composition must be part of that class.
ADM DATA	Type, min, max, average<crlf> Type is the ASCII number for K (key), M (multiple), or G (group).
ALIAS	“Standard Analyst Text” <crlf> There may be up to 10 aliases per entry.
ASSOCIATOR	“Standard Analyst Text” <crlf> A relationship can have one associator and can be up to 128 characters in length.
ATTR:	“Standard Analyst Text” <crlf> User-defined attribute entries. The actual Field Type is “ATTR:User Attribute Name”. There is one of these for each user attribute used on the object.
CHECK	“Column Name”, “Check Constraint Name” <crlf>
CLASS COMPOSITION	“Local data element”, “Type Name”, Array Limit, Flags <crlf> The Type Name is an object in the repository of type Class, Data Element, or Data Structure. Both Local data element and Type Name can be up to 128 characters in length. Three pieces of information are in the flags fields. One item from each of the following categories is ORed together: Reference Address [0x80] Reference [0x100] Value [0x00] Visibility Implementation [0x18] Private [0x10] Protected [0x08] Public [0x00] Qualification Constant [0x400] None [0x000] Static [0x800] Volatile [0x200]
CLASS DATA	Cardinality, “Cardinality Text”, Flags <crlf> Cardinality can be one of the following values:

Table 7-5
Repository Export VSC Format

	One [0x01] One or many [0x02] Many [0x03] Zero or One [0x04] Zero or Many [0x05] Cardinality text can be up to 20 characters in length. Three pieces of information are stored in the flags field. One item from each of the following categories is ORed together (Persistent and Abstract are optional): Currency Active [0x06] Guarded [0x20] Sequential [0x00] Persistent [0x4000] Abstract [0x8000]
COMPOSITION	“Standard Analyst Text” >crlf> This field can be used with objects that support composition; that is, data structures, entities, data stores.
DATA ONLY MODULE	“Standard Analyst Text” <crlf> This field is used with an information cluster entry, and can be up to 128 characters in length.
DBFILES	Database Type, “File Name”, Size, Reuse <crlf> This field is used with a tablespace entry to specify either the data files that comprise the tablespace or the logical device name. Database Type is 1 for Oracle and 2 for SQLServer.
DBPHYSINFO	“Tablespace Name”, Database Type, Sequence, Index Type, Additional Information <crlf> This field is used with either an entity or a tablespace entry to specify physical storage information. There can be one DBPHYSINFO object for each entity, unique key, and performance index defined. Tablespace Name should be blank {“”} if the current entry is a tablespace. Database Type is 1 for Oracle, 2 for SQLServer, 3 for Informix, 4 for DB2. Sequence is 0 to define physical information for the table, 100+ key number for unique keys, and 200+ index number for performance indexes. (Key number for the primary key is zero.)

Table 7-5
Repository Export VSC Format

	Index type can be one of the following values: Sorted [0x00] Hashed Ordered [0x10] Hashed Scattered [0x20] Additional Information for Oracle: No Sort, PCTFree, PCTUsed, INITrans, MAXTrans, InitialExtent, NextExtend, PCTIncrease, MINExtents, MAXExtents, FreeLists, FreeListGroups, Offline Additional Information for SQLServer indexes: SortedData, FillFactor, IgnoreDupKey, DupRow
DESCRIPTION	“Standard Analyst Text” <crlf> This is a short description for the entry. Up to two lines of information can be specified.
DIAGRAM LINES	lsx, lsy, lex, ley, line)style, terminator_style, color, start_symbol, end_symbol, next_segment, associated_line, loop_height<crlf>
DIAGRAM SYMBOLS	left, top, right, bottom, text_type, associated_object, font_name, format_information, color, text<NL>
FRIEND	“Friend Name” <crlf> This field is used with classes to specify either classes or functions that are friends of the current object. See the description for Entry Key for the format.
FUNCTION DESCRIPTION	“Standard Analyst Text” <crlf>
KEYS	Key Type, Key Number, “Key Name”, Column Count <crlf> “Column Name”, Order <crlf> Key Type is either 21 for unique keys or 22 for performance indexes. Key Number is either zero for the primary key or non-zero for alternate keys and performance indexes. Column Count indicates the number of columns in the key. Order indicates the sort order of the column and is either zero for ascending or one for descending.
LINK	“Standard Analyst Text” <crlf> User-defined object link references. The actual Field Type is “LINK:User Object”. There is one of these for each user object referenced by the object.
LOCATION	Diagram Type, “Diagram Name”, Upper Left X, Upper Left Y, Lower Right X, Lower Right Y,

Table 7-5
Repository Export VSC Format

	Associated Line Count <crlf> “Associated Line Name | Parent Flow Name”, Start X, Start Y, End X, End Y, Direction <crlf> This field is a list of all diagram locations for the object. Each diagram location consists of a header line followed optionally by either a parent flow name record for data flows or list of associated lines for other objects. Diagram Type can be one of the following values: Data Flow 1 Structure Chart 2 Entity Relationship 3 Unstructured 4 Decomposition 5 Class 6 State Transition 7 Entity Life History 8 Use Case 9 Sequence 10 Collaboration 11 Activity 12 Diagram Name can be up to 40 characters long. Each diagram within a diagram type must have a unique name. If the name is prefixed by hex 9B, the diagram is a virtual diagram imported from Application Browser. Associated Line Count is the number of lines attached to the object on the indicated diagram. Associated Line Name is the name of the attached line; it can be up to 128 characters long. Direction can be one of the following values: Input 49 Output 48 or Couple Down 48 Couple Up 49 Couple Bidirectional 50
MODULE DESCRIPTION	“Standard Analyst Text” <crlf>
MODULES CONTAINED	“Standard Analyst Text”<crlf> This field is used with an information cluster entry. There can be up to seven modules each up to 128

Table 7-5
Repository Export VSC Format

	characters long.
NOTES	“Standard Analyst Text” <crlf>
PARAMETER	“Name, “Type Name”, Array Limit, Flags The Type Name is an object in the repository of type Class, Data Element, or Data Structure. Both Name and Type Name can be up to 128 characters long. Two pieces of information are stored in the flags field. One item from each of the following categories is ORed together: Pass By Address [0x80] Reference [0x100] Value [0x00] Qualification Constant [0x400] None [0x000] Volatile [0x200]
PHYSINFO	Data Type, Length, Decimal, “Default Value”, Picture”, “Owner”, Allow Null <crlf> OR Data Type, “Domain Name”, “Owner”, Allow Null <crlf> This field is used with data elements or classes with an elemental subtype. The first version is used if the entry does NOT reference a domain. Default Value can be up to 20 characters long. Picture can be up to 18 character long. Owner can be up to 8 characters long. Allow Null is either ASCII 89 (Y) or 78 (N).
PROCESS DESCRIPTION	“Standard Analyst Text” <crlf>
PROCESS NUMBER	“Standard Analyst Text” <crlf>
RELATED TO	“Standard Analyst Text” <crlf> This field is used with structure chart items to create an association with data flow diagram items. For modules, there can be up to 10 processes listed; for couples, there can be one data element listed; and for data only modules, there can be one data store, data structure, data element, or data flow listed. Each related to item can be up to 128 characters long.
RELATIONSHIP	*Header*

Table 7-5
Repository Export VSC Format

	Controlling Key, Flags, From-To Cardinality, To-From Cardinality, On Delete, On Update, “From-To Cardinality Text”, “To-From Cardinality Text”, “From Role Name”, “To Role Name”, “From Qualifier Name”, “To Qualifier Name” <crlf> *OR* Controlling Key, Flags, From-To Cardinality, To-From Cardinality, On Delete, On Update, “From-To Cardinality Text”, “To-From Cardinality Text”, “Discriminator Name” <crlf> Group/Prefix Group Number, “Prefix 1”, “Prefix 2”,…<crlf> Foreign keys “Foreign Key Column Name”, <crlf> This field is used with a relationship line entry to supply information used during SQL Generation. A relationship field consists of one header, one group/prefix line, and one line for each column of the foreign key. The second form of the header is used with supertype/subtype and inheritance relationships, while the first form is used with all other relationship types. Controlling key is zero to indicate the primary key is used in the owner entity, while any other number indicates an alternate key number. Four pieces of information are stored in the flags field. One item from each of the following categories is ORed together: Relationship Type Normal [0x0] Inheritance [0x01] Aggregation [0x03] Denormalization Option None [0x0] Collapse Child [0x80000] Duplicate Parent [0x10000] Duplicate Parent and Retain [0x180000] Identifying [0x2000] Use Suffix [0x200000] Cardinality can be one of the following values: One [0x01] One or Many [0x02]

Table 7-5
Repository Export VSC Format

	Many [0x03] Zero or One [0x04] Zero or Many [0x05] On Delete and On Update can be one of the following values: Cascade [0x01] Restrict [0x00] Set Default [0x03] Set Null [0x02] Cardinality text can be up to 20 characters long. Discriminator, Roles and Qualifier names can be up to 128 characters long. Group Number is used with supertype/subtype relationships to indicate how relationships are grouped together. Relationships drawn with the same start point on the supertype should have identical group numbers.
TRIGGER	“Trigger Name”, Operations, Fire time, Scope, # of associated columns <crlf> The values to use are: Operation – Note that some dialects allow you to specify more than one operation. Update [0x01] Insert [0x02] Delete [0x04] Fire Time Before [0x00] After [0x08] Scope Table [0x00] Row [0x10] For TRIGGER field records, the above structure is used. For each associated column there is one associated column record, as follows: “Column Name” <crlf>
VALUES AND MEANINGS	“Standard Analyst Text” <crlf>
VIEW SELECT	SQLDialect, SQLStatement<crlf> SQLDialect is one of the supported dialect IDs. SQLStatement is the RAW dialect-specific SQL statement for the view, it is a standard analyst text

Table 7-5
Repository Export VSC Format

	string that may take as many additional lines as necessary. This field is SUBORDINATE to the VIEW SPECIFICATION and should be used only if the view specification is not available.
VIEW SPECIFICATION	This field defines how a view is built from its base entities. It has some basic flags and a list of Sub Query Select (SQS) definitions that make up the view. Each SQS has a header line with a number of entity lines and join lines following it. Leader line. MSCFlags, SQSCount<crlf> SQS header line. SQSName, Union, Distinct, Filter, GroupBy, Having, StartWith, ConnectBy, EntityCount, JoinCount<crlf> Entity line. BaseEntity, JoinAlias, DP_ULX, DP_ULY, DP_LRX, DP_LRY<crlf> Join line. FirstEntity, SecondEntity, JoinRelation, JoinType, JoinExpression<crlf> Filter, GroupBy, Having, StartWith, ConnectBy, and JoinExpression are all pseudo SQL strings that use Visible Analyst standard syntax and have embedded column and SQS references. See the VIEWSPEC repository definition for definition of the named values. The strings are in expanded format.

Note

- Bounding boxes for structure chart types refer to “virtual diagram” positions. The virtual diagram is a temporary object created by import so that invocation lines can exist in the repository without an associated diagram. Virtual locations disappear when the associated objects are added to real diagrams. Virtual locations are not visible when editing repository items.

Table 7-6 lists the record item definitions.

Table 7-6
Record Item Definitions

diagram_type	A number representing the diagram type:

Table 7-6
Record Item Definitions

	1 = Data Flow 2 = Structure Chart 3 = Entity Relationship 4 = Unstructured 5 = Decomposition 6 = Class 7 = State Transition 8 =Entity Life History 9 =Use Case 10 =Sequence 11 =Collaboration 12 = Activity
notation	If the diagram type is data flow, the notation used to draw symbols: 1 = Yourdon 2 = Gane & Sarson 4 = SSADM 5 = Mϑtrica If the diagram type is entity relationship, the notation used to draw relationships: 1 = Crowsfoot 2 = Arrow 3 = Bachman Note: If IDEF1X notation is specified in view_level, this setting indicates the alternate relationship notation.
view_level	The level of detail to be displayed on an ERD: 1 = Entity name only 2 = Entity name and primary key 3 = Entity name and all attributes 8 = Use IDEF1X notation 16 = Expand associator elements (IDEF1X only) 32 = Display discriminators (IDEF1X only) 64 = Display entity name outside box (IDEF1X only) The level of detail to be displayed on a class diagram: 1 = Class name only 2 = Class name and attributes 3 = Class name, attributes, and methods
view_details	Display physical characteristics on an ERD: 1 = Display physical name 2 = Display data type 4 = Display null option

Table 7-6 Record Item Definitions	
	8 = Replace domain references with actual physical type 16 = Use alias name instead of actual name Level of detail to display on a class diagram: 1 = OMT Notation 5 = UML Notation 8 = Display public attributes 16 = Display private attributes 32 = Display protected attributes 48 = Display data type 128 = Display initial value 256 = Display public methods 512 = Display private methods 1024 = Display protected methods 2048 = Display argument list 4096 = Display virtual method tag
lsx	X-coordinate of starting point.
lsy	Y-coordinate of starting point.
lex	X-coordinate of ending point.
ley	Y-coordinate of ending point.
line_style	Style of the line: 1 = single solid (default) 2 = bold solid 3 = extra bold 4 = double 5 = wide double 6 = dotted 7 = dashed 8 = bold dashed 9 = dashed arc 10 = solid arc 13 = generic couple 15 = data couple 16 = control couple 17 = data connection 18 = control connection 19 = loop 20 = dashed elbow 21 = solid elbow 22 = supertype 23 = overlapping supertype

Table 7-6
Record Item Definitions

	24 = aggregation 25 = supertype elbow 26 = overlapping supertype elbow 27 = aggregation elbow 28 = object link 29 = object lifeline 30 = note link 31 = object link elbow 32 = note link elbow Types 13 through 19 should only be used on a structure chart.
terminator_style	Adornment that appears at the startpoint/endpoint of a line. Dataflow, structure chart, decomposition, and state transition 1 = solid arrow at endpoint 2 = open arrow at endpoint 3 = solid arrow at startpoint and endpoint 4 = open arrow at startpoint and endpoint 5 = lexical inclusion at endpoint 6 = decision diamond at endpoint 7 = lexical inclusion and decision diamond at endpoint 11 = single stick arrow 12 = single half-stick arrow For entity relationship and class diagrams, one number from each column must be added together. Parent Cardinality Child Cardinality 0 = undefined 0 = undefined 256 = one 1 = one 512 = one or many 2 = one or many 768 = many 3 = many 1024 = zero or one 4 = zero or one 1280 = zero or many 5 = zero or many
color	Pen color used to draw line, stored as a Red-Green-Blue (RGB) value where each component can have a value from 0 to 255. (((BYTE)(R) | ((WORD)(G)<<8)) | (((DWORD)(BYTE)(B))<<16)))
start_symbol	Number of symbol to which the starting point of the line is connected. If zero, the starting point is not connected to a symbol.
end_symbol	Number of symbol to which the ending point of the line is connected. If zero, the ending point is not connected to a symbol.
next_segment	If this field is non-zero, this indicates the line number of the next segment in a multi-segment line. Segments should be listed

Table 7-6
Record Item Definitions

	sequentially. The line_style and terminator_style for all segments should be the same. If the line is connected to symbols, only the start_symbol field in the first segment and the end_symbol field in the last segment need to be set.
associated_line	On a structure chart, if the line_style is 13, 15, or 16, this field indicates the line number to which the couple is associated. If this value is less than zero, it indicates a return couple, otherwise it is a passed couple.
loop_height	On a structure chart, if the line_style is 19 (loop), this field indicates the height of the loop.
left	X-coordinate of upper left corner of box that completely contains the object.
top	Y-coordinate of upper left corner of box that completely contains the object.
right	X-coordinate of lower right corner of box that completely contains the object.
bottom	Y-coordinate of lower right corner of box that completely contains the object.
symbol_type	Object type as stored in the repository: Data Flow Diagram 1 = Process 2 = File or Data Store 3 = Source/Sink or External Entity Functional Decomposition Diagram 14 = Function Structure Chart 15 = Module 16 = Library Module 17 = Macro 18 = Library Macro 19 = Data-Only Module 20 = Information Cluster 21 = On-page Connector 22 = Off-page Connector Entity Relationship Diagram 34 = Entity 35 = Associative Entity 36 = Attributive Entity 50 = View Object Class Diagram

Table 7-6
Record Item Definitions

	45 = Class State Transition / Activity Diagram 46 = State Entity Life History Diagram 47 = Event Use Case Diagram 52 = Actor 53 = Use Case 54 = System Boundary Sequence/Collaboration Diagram 55 = Object 56 = Message
foreground_color	Pen color used to draw symbol, stored as a Red-Green-Blue (RGB) value where each component can have a value from 0 to 255. (((BYTE)(R) | ((WORD)(G)<<8)) | (((DWORD)(BYTE)(B))<<16)))
background_color	Brush color used to paint symbol background. Stored the same as foreground color.
associated_diagram	Name of the diagram that is associated with this symbol. This entry is used to create linkages between levels on a dataflow diagram, connected pages on a structure chart or class to state transition relationships. If no linkage exists, this field should be NULL ("").
drawing_instruction	Internal drawing instruction for non-methodology symbols. For all others this should be NULL ("").
symbol_name	The name of the symbol. or NULL (“”). If this field is set, there should not be a name for the object in the DIAGRAM TEXT section.
process_number	The process number for the symbol. or NULL (“”). If this field is set, there should not be a process number entry for the object in the DIAGRAM TEXT section. This field is only valid for process numbers.
text_type	A number indicating the type of text: -1 = process/data store number 1 = symbol label 2 = line label (primary) 3 = line label (secondary - used only for relationships) 4 = information cluster – module 5 = information cluster - data module 6 = caption
associated_object	Number of symbol or line to which this text entry is associated.
font_name	Name of font used to draw label.
format_information	Formatting options for text:

Table 7-6
Record Item Definitions

	S# = point size, where # is replaced by a number
	B = bold
	I = italic
	U = underline
	C = text is centered in bounding box

text	This field contains the entry label. Embedded carriage returns are replaced by ASCII 31.
ConstraintType	One of VRCNST_CHECK 0 VRCNST_DEFAULT 1 VRCNST_NULL 2
MemberName	Name of a class member object.
ElementName	Entry key for an elemental type.
ArrayLimit	old. size of an array defined in the text composition field. If DataType is VRT_ARRAY then this is the same as ArrayLength.
Flags	VR_END_GROUP 0x00010000 VR_IN_GROUP 0x00020000 VR_AND_GROUP 0x00040000 VR_EASY_NOT_NULL 0x10000000 VR_EASY_UNIQUE 0x20000000
UseValues	Bit field that defines whether the values in the record are definitions or copied from the base type. VRPHI_USE_NullType 0x0001 VRPHI_USE_Picture 0x0002 VRPHI_USE_Owner 0x0004 VRPHI_USE_Default 0x0008 VRPHI_USE_IdentitySpec 0x0010
DataType	Analyst VRT data type. VRT_UNDEFINED 0 VRT_BIT 1 VRT_VARBIT 2 VRT_CHAR 3 VRT_NATIONAL_CHAR 4 VRT_VARCHAR 5 VRT_NATIONAL_VARCHAR 6 VRT_LONG_VARCHAR 7 VRT_BINARY 8 VRT_VARBINARY 9

Table 7-6
Record Item Definitions

	VRT_LONG_VARBINARY 10 VRT_UNICODE_CHAR 11 VRT_UNICODE_VARCHAR 12 VRT_UNICODE_LONG_VARCHAR 13 VRT_LARGEINT 14 VRT_INTEGER 15 VRT_SMALLINT 16 VRT_TINYINT 17 VRT_MONEY 18 VRT_SMALLMONEY 19 VRT_DECIMAL 20 VRT_ZONEDDECIMAL 21 VRT_FLOAT 22 VRT_SMALLFLOAT 23 VRT_DATE 24 VRT_TIME 25 VRT_DATETIME 26 VRT_SMALLDATETIME 27 VRT_INTERVAL 28 VRT_SERIAL 29 VRT_UPDATESTAMP 30 VRT_ROWID 31 VRT_SYSNAME 32 VRT_USER_TYPE_1 33 VRT_USER_TYPE_2 34 VRT_USER_TYPE_3 35 VRT_USER_TYPE_4 36 VRT_USER_TYPE_5 37 VRT_ARRAY 253 VRT_LIST 254 VRT_DOMAIN 255
Length	Length of the object, type specific in meaning. For bit and floating point types it is a number of bits, for decimal types it is the number of significant digits, for date types it is a special packed value, for all other types it is a number of characters.
Decimal	The number of decimal places to the right of the decimal point. For floating point types it is the scale.
DefaultValue	Constant expression in pseudo SQL that defines the a default for this type.

Table 7-6
Record Item Definitions

Picture	COBOL picture clause.
Owner	User field.
AllowNull	NULL Y NOT_NULL N NOT_NULL_WITH_DEFAULT D IDENTITY I
IdentitySpec	Seed, Increment
DefaultName	Constraint name for the default clause
NullName	Constraint name for the Null clause
BaseTypeName	Entry Key for the base elemental object that is a basis for this type.
ArrayLength	Size of the array for VRT_ARRAY 0 Dynamic, >0 Actual fixed size.

An example of an exported ASCII file is shown in Table 7-7.

Table 7-7
Repository Export VSC Example
Project = BT6, Filename = “BT6.AD”
(‡ is substituted for the hex 01 character)

```
“BT6”,1995 10 27,”10:17:11”,1,3
PowerBuilder Extensions”,4,0,0
annotation”,0,“0”,0,“”,1,“45”,”annotation Description”
requirement”,0,“0",0,“",1,“0",“requirement Description”
“Display Format”,2,“304",1,“",0,“0",“Format:"
ENTRY “264",”40",“Entity1‡Entity2‡may have‡Is an extension of"
FIELD “RELATIONSHIP”, 2
0,8192,2,1,0,0,“",“",“",“",“",“"
0
END ENTRY
ENTRY “A Parent Flow”,“4",“"
FIELD “COMPOSITION”,, 2
“SubFlow1 +”
“SubFlow2"
FIELD “LOCATION”,, 1
1,“TopLevelDiagram",398,594,1016,861,0
END ENTRY
ENTRY “A process for top flow”,"1",""
FIELD “PROCESS NUMBER”, 1
```

Table 7-7

Repository Export VSC Example
Project = BT6, Filename = “BT6.AD”
(‡ is substituted for the hex 01 character)

```
“1.1"
FIELD “LOCATION”, 3
1,"Top Flow",700,250,1084,696,2
SubFlow1",85,273,716,336,49
SubFlow2",85,614,716,550,49
END ENTRY
ENTRY “ColumnID”,"458797",""
FIELD “PHYSINFO”,, 1
22,60,0,"","","",32END ENTRY
ENTRY “ColumnName”,"458797",""
FIELD “PHYSINFO”, 1
1,0,0,"","","",32
END ENTRY
ENTRY “Entity1",”34",""
FIELD “COMPOSITION”, 2“
“[PK] ColumnID”“
“[PI1] ColumnName”
FIELD “KEYS”, 4
21,0,"",0,1
“ColumnID”,0
22,1,"",0,1
“ColumnName”,0
FIELD “LOCATION”, 2
3,"An ERD Diagram",350,250,932,538,1
“264",913,420,1461,813,48
END ENTRY
ENTRY “Entity2",”36",""
FIELD “CLASS COMPOSITION”, 2
“”,"ColumnID",0,0
“SupplementalID”,"ColumnID",0,0
FIELD “KEYS”, 3
21,0,"",0,2
“ColumnID”,0
“SupplementalID”,0
FIELD “LOCATION”, 2
3,"An ERD Diagram",1200,800,1782,1088,1
“264",913,420,1461,813,49
END ENTRY
```

Table 7-7
Repository Export VSC Example
Project = BT6, Filename = “BT6.AD”
(‡ is substituted for the hex 01 character)

```
ENTRY “Is an extension of”,"37","Entity1‡Entity2‡may have‡Is an extension of”
FIELD “LOCATION”, 1
3,"An ERD Diagram",913,420,1461,813,0
END ENTRY
ENTRY “SubFlow1",”4",""
FIELD “LOCATION”, 2
1,"Top Flow",85,273,716,336,1
“A Parent Flow”
END ENTRY
ENTRY “SubFlow2",”4",""
FIELD “LOCATION”, 2
1,"Top Flow",85,614,716,550,1
“A Parent Flow”
END ENTRY
ENTRY “Entity2::SupplementalID”,"43",""
END ENTRY
ENTRY “Top Flow”,"1",""
FIELD “PROCESS NUMBER”, 1
“1"
FIELD “LOCATION”, 2
1,"TopLevelDiagram",1000,650,1384,1096,1
“A Parent Flow”,398,594,1016,861,49
END ENTRY
ENTRY “sample erd”, “41”, 3‡1‡1‡0”, “1997-05-20‡1997-05-20”
Field “DIAGRAM SYMBOLS”, 2
175,212,757,500,34,0,11141119, “”,””
1334,931, 1916,1219,34,0,11141119,””,””
Field “DIAGRAM LINES”, 1
739,350,1639,944,21,261,0,1,2,0,0,0
FIELD “DIAGRAM TEXT”, 5
365,324,567,387,1,1,”Times New Roman”, “CS12”,0,”customer”
1541,1043,1709,1106,1,2,”Times New Roman”,”CS12”,0,”invoice”
1086,275,1291,329,2,1”Arial”,”S10”,0,”generates”
1028,371,1349,425,3,1,”Arial”,”S10”,0,”is generated by”
1283,48,2407,133,6,0,”Times New Romas”,”BIS16”,255,”Sample Entity Relationship
Diagram”
END ENTRY
ENTRY “sample dfd”, “41”, “1‡2”,””
```

Table 7-7
Repository Export VSC Example
Project = BT6, Filename = “BT6.AD”
(‡ is substituted for the hex 01 character)

```
FIELD “DIAGRAM SYMBOLS”, 1
1011,315,1395,761,1,,0,11141119,”process one dfd”,””
FIELD “DIAGRAM LINES”,4
461,495,646,742,1,1,0,0,0,2,,0,0
646,742,1027,632,1,1,0,0,2,0,0,0
1377,586,1869,714,1,1,0,1,0,3,0,0
1903,3157,2895,3735,1,1,0,0,0,0,0,0
FIELD “DIAGRAM TEXT”, 4
1191,336,1214,382,-1,1,”Courier New”,”C8”,0,”1”
1161,551,1290,614,1,1,”Times New Roman”,”CS12”,0,”process”
697,572,927,635,2,1,”Times New Roman”,”S12”,0,”input flow”
1517,534,1773,597,2,2,”Times New Roman”,”S12”,0,”output flow”
END ENTRY
```

IMPORTING DATA INTO THE REPOSITORY

There are several types of imports that can be done to get external data into the Visible Analyst repository.

- Generic format that imports from ASCII files in the proprietary VSC format.
- KnowledgeWare (IEW/ADW) format.
- Excelerator “Excel” format.
- SQL format.
- AS/400 DDS (Data Description Specification) format.
- Application Browser format.
- Powersoft PowerBuilder format.
- RDBMS Catalog format.
- Uniface format.
- Unify VISION format.
- GDPro format.
- ERwin .ER1 or XML format
- XMI format for CompuWare OptimalJ.
- Progress 4GL

All are described below.

Note

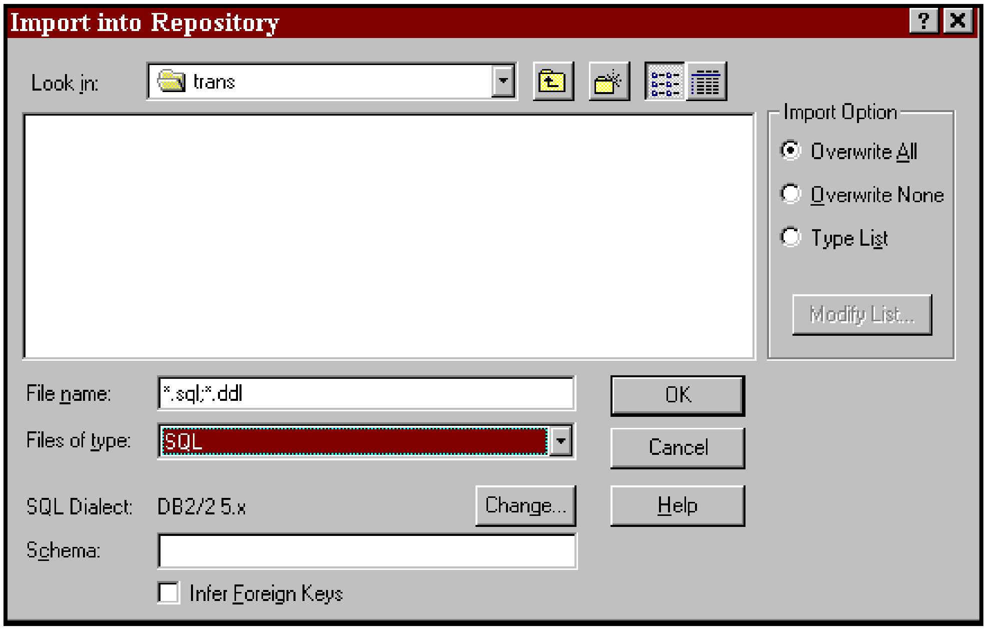
- The Import tool makes significant modifications to your project repository. For this reason it is good practice to backup the project to which you are importing prior to the import. In this way you can easily recover from any unintentional project changes.

General Import Information

After you have prepared the data file contents you want to import into Visible Analyst, there are three stages to the import: preparation, verification, and the actual import.

Preparation

You must first decide on the format of the imported data. You can choose from the formats listed above.   If you select RDBMS Catalog, you must select an SQL dialect. If you select SQL format, you must also select the SQL dialect and type a schema name. (See Figure 7-3.) For both RDBMS Catalog and SQL format, you have the option to infer foreign keys.



Look in: trans

Import Option

● Overwrite All

○ Overwrite None

○ Type List

File name: *.sql;*.ddl

Files of type: SQL

SQL Dialect: DB2/2 5.x

Schema:

□ Infer Foreign Keys

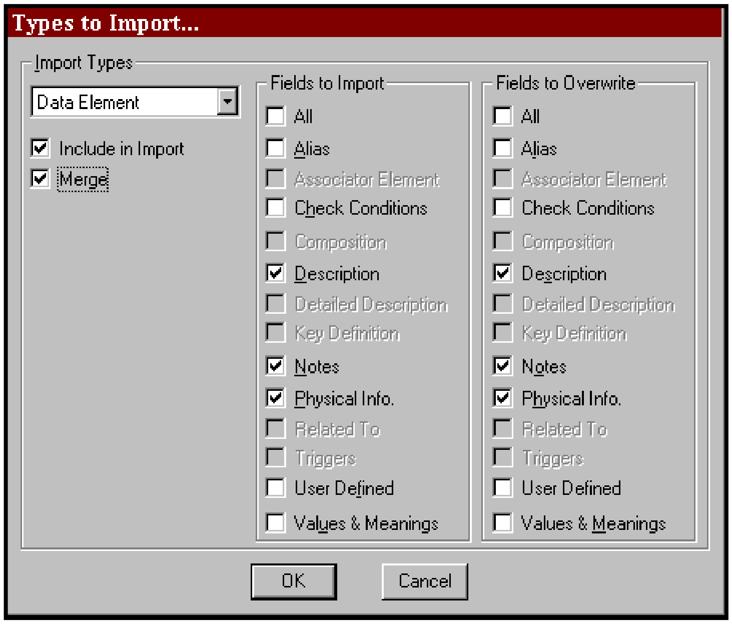
Figure 7-3 Import Dialog Box

Next, you must decide upon the import options, how Visible Analyst should handle imported data vis-à-vis the information already residing in the project repository. You have three choices:

- Overwrite All tells Visible Analyst to add everything in the import file to Visible Analyst regardless of whatever is already there. Visible Analyst overwrites existing data or creates new records and fields as necessary.
- Overwrite None indicates that nothing already in your repository is to be touched. Import file information that would cause such an overwrite is ignored.
- Type List gives you a detailed means to customize the import, as described below.

Import Customization

Customization allows you to make flexible choices on what repository types and what fields within those types are imported. Furthermore, you can specify how import data/existing data collisions are handled. (See Figure 7-4.)



Types to Import...

Import Types

Data Element

☑ Include in Import

☑ Merge

Fields to Import

□ All
□ Alias
□ Associator Element
□ Check Conditions
□ Composition
☑ Description
□ Detailed Description
□ Key Definition
☑ Notes
☑ Physical Info.
□ Related To
□ Triggers
□ User Defined
□ Values & Meanings

Fields to Overwrite

□ All
□ Alias
□ Associator Element
□ Check Conditions
□ Composition
☑ Description
□ Detailed Description
□ Key Definition
☑ Notes
☑ Physical Info.
□ Related To
□ Triggers
□ User Defined
□ Values & Meanings

Figure 7-4 Repository Import Customization Settings

For each repository type (data element, control couple, etc.), you can choose whether items of that type within the import file are to be imported by checking the Include in Import box for that type. Note that you can set import characteristics for subtypes of the compound types individually, regardless of the settings you make for the compound type. For example, you can choose to import All Standard DFD types, but choose not to import Aliases and Miscellaneous items.

For each of the above types you chose to import, you can check both the repository fields you wish to import and whether or not you want existing data in those fields to be overwritten.

You have additional flexibility with the Merge option. The basic idea of merging is that if a field exists in the import file and the corresponding field already in the repository is empty, the file data is imported regardless of overwrite settings for the remaining fields of that repository item. If Merge is on (the default setting), Visible Analyst brings in all data fields for an item except for those fields that contain data and for which overwriting is prohibited. If Merge is turned off, if *any* field for a repository item is non-importable because of an overwrite restriction, *no* fields for that item are imported. You should turn Merge off for a repository type only if this all-or-nothing approach suits your needs.

When you have completed your customization settings, you can review them by selecting the repository types in the dialog box list and noting the options set for each. (The easiest way to do this is with the up and down arrow keys.) If you have set some import characteristics for a compound type, say All Entity Types, and different ones for a subtype of that compound type, say Attributive Entities, when you display the characteristics for the compound type you are informed that those for the subtype are different.

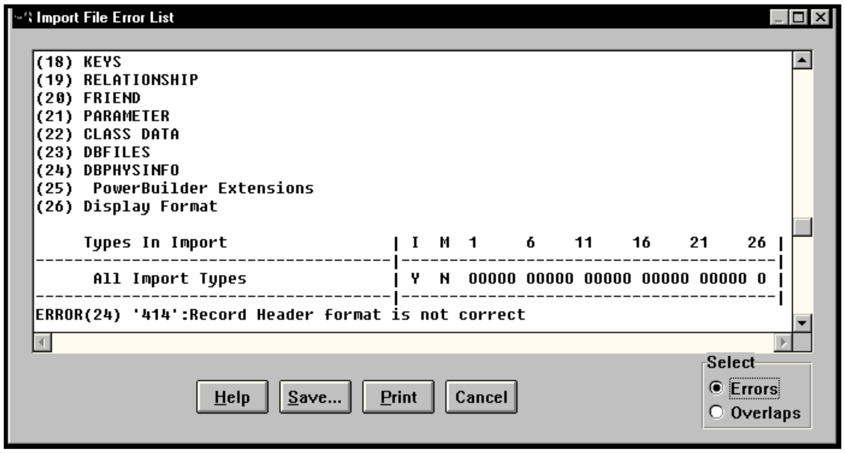
Note

- Customized import characteristics are saved and are applied to future imports for which you pick Type List as the Import Option. You can modify these at the time of future imports and the changed characteristics are saved. You can, however, reset them all (turn all customizations off) if you wish. In the Types to Import dialog box (see Figure 7-4), select All for the import type and toggle the Include in Import box on and then off. All options are then reset, and you can save this status by clicking the OK button.

Verification
Clicking OK at the Import dialog box runs the verification function to allow you to check that the import will be carried out as you expect. There are two reports that result from verification that you can view on the screen, print, or save to a file. You should review these carefully. Both are described below.

Import File Error List

The Import File Error List (see Figure 7-5) shows, for all repository types, what is done to them on import. In other words, it summarizes the import characteristics you specified. At the top are the compound types and below them are the individual types that either are not part of any compound type or that vary from the compound type to which they belong. At the bottom of the report are descriptions of errors in the import file, if any. The existence of errors means that the actual import *cannot proceed* until you correct them and rerun the import verification procedure.



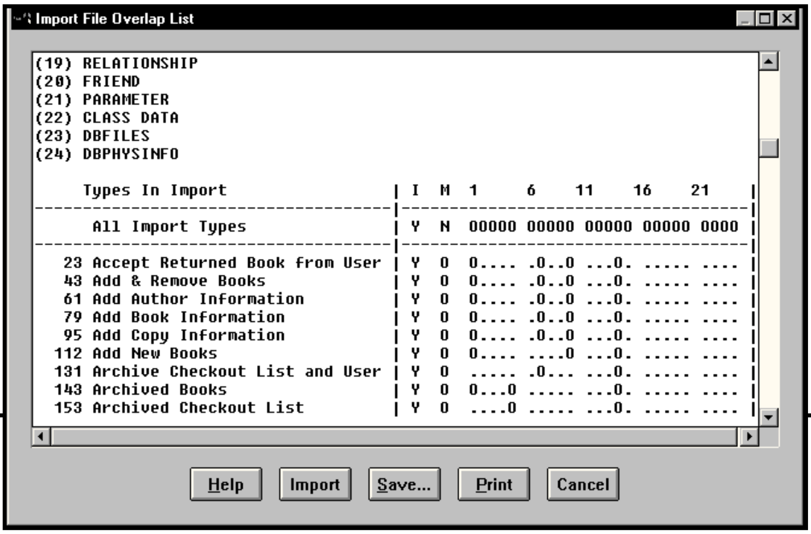
```
(18) KEYS
(19) RELATIONSHIP
(20) FRIEND
(21) PARAMETER
(22) CLASS DATA
(23) DBFILES
(24) DBPHYSINFO
(25)  PowerBuilder Extensions
(26) Display Format

     Types In Import                 | I  M  1     6    11    16    21    26 |
-------------------------------------|------------------------------------------|
      All Import Types               | Y  N  00000 00000 00000 00000 00000 0 |
-------------------------------------|------------------------------------------|
ERROR(24) '414':Record Header format is not correct
```

Figure 7-5 Import File Error List

Import File Overlap List

At the top of the Import File Overlap List (see Figure 7-6) is the same summary of import characteristics as above. Below is a report that shows, for every entry in the import file for which there is an overlap with data already in the repository, how the Import function applies the import characteristics you specified in the customization dialog box.



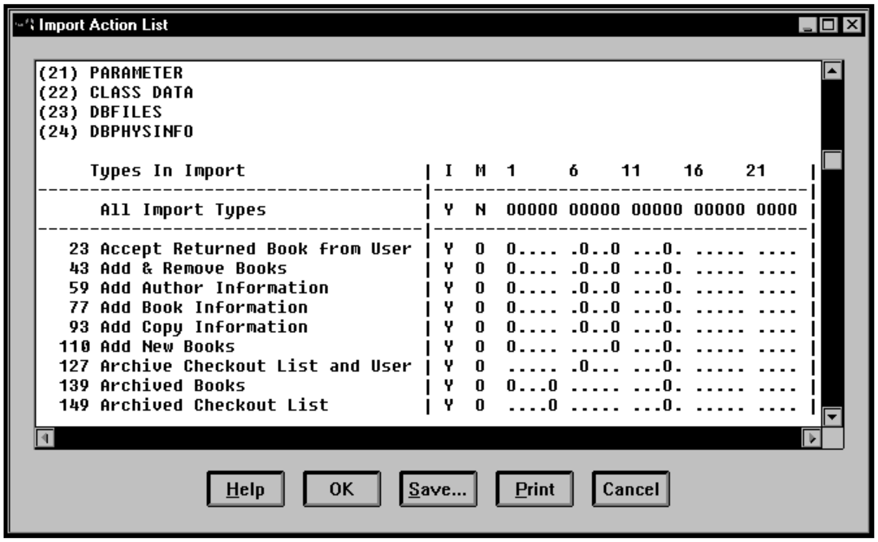
```
(19) RELATIONSHIP
(20) FRIEND
(21) PARAMETER
(22) CLASS DATA
(23) DBFILES
(24) DBPHYSINFO

     Types In Import                 | I  M  1     6    11    16    21    |
-------------------------------------|------------------------------------|
      All Import Types               | Y  N  00000 00000 00000 00000 0000 |
-------------------------------------|------------------------------------|
   23 Accept Returned Book from User | Y  O  O.... .O..O ...O. ..... .... |
   43 Add & Remove Books             | Y  O  O.... .O..O ...O. ..... .... |
   61 Add Author Information         | Y  O  O.... .O..O ...O. ..... .... |
   79 Add Book Information           | Y  O  O.... .O..O ...O. ..... .... |
   95 Add Copy Information           | Y  O  O.... .O..O ...O. ..... .... |
  112 Add New Books                  | Y  O  O.... ....O ...O. ..... .... |
  131 Archive Checkout List and User | Y  O  ..... .O... ...O. ..... .... |
  143 Archived Books                 | Y  O  O...O ..... ...O. ..... .... |
  153 Archived Checkout List         | Y  O  ....O ..... ...O. ..... .... |
```

Figure 7-6 Import File Overlap List

Importing the Data

After the verification step has run error-free, you can run the actual import. When it is complete, you see the Import Action List (see Figure 7-7). This again shows a summary of your import characteristics. Below that, it displays a list of every item in the import file and how it was handled by the Import tool. If you notice that a major error in the way data was imported, you can delete the project and restore the backup you made before beginning the import procedure.



```
(21) PARAMETER
(22) CLASS DATA
(23) DBFILES
(24) DBPHYSINFO

     Types In Import                 | I  M  1     6     11    16    21    |
-------------------------------------|-------------------------------------|
      All Import Types               | Y  N  00000 00000 00000 00000 0000  |
-------------------------------------|-------------------------------------|
   23 Accept Returned Book from User | Y  O  O.... .O..O ...O. ..... ....  |
   43 Add & Remove Books             | Y  O  O.... .O..O ...O. ..... ....  |
   59 Add Author Information         | Y  O  O.... .O..O ...O. ..... ....  |
   77 Add Book Information           | Y  O  O.... .O..O ...O. ..... ....  |
   93 Add Copy Information           | Y  O  O.... .O..O ...O. ..... ....  |
  110 Add New Books                  | Y  O  O.... ....O ...O. ..... ....  |
  127 Archive Checkout List and User | Y  O  ..... .O... ...O. ..... ....  |
  139 Archived Books                 | Y  O  O...O ..... ...O. ..... ....  |
  149 Archived Checkout List         | Y  O  ....O ..... ...O. ..... ....  |
```

Figure 7-7 Import Action List

The Import Procedure

To use the import tool:

Select Import: 1 Select the Import tool from the Tools menu. Visible Analyst displays the import dialog box. See Figure 7-3.

Select the Format of the Import File You Want to Import: 2 Select the format of the import file (what application it comes from (SQL, IEW, etc.)) from the list box. Then choose or enter the name of file you wish to import.

Choose your Import Options:	3	Decide which import options you want—Overwrite All, Overwrite None, or Type List—to customize your import. If you choose the Type List option, the Modify Type List button is enabled.
Customize the Import:	4	If you want to customize your import, click the Modify Type List. Click OK after making your choices to save the current selections. Click Cancel to revert to the previously saved import customizations, if any.
Verify the Import:	5	Follow the import process. See Figures 7-5 and 7-6.
Run the Import:	6	Execute the import. When it is complete, check the results shown on the Import Action List, as shown in Figure 7-7. The import is now complete.

Import Formats

VSC Format Imports

The Import selection on the Tools menu allows you to load information into a project data repository from an ASCII file. The ASCII file format required is the same as that generated by the VSC format of the repository export. This means that you may use the VSC format to export from one project and import into another project. You may also create an ASCII file in this format from another application or database and import it into a Visible Analyst project.

Import from Excelerator™

Project data should be exported from Excelerator using its “E” format, the default. Files in that format consist of a header record and data records. One E-file is created for each project. One data record in an E-file contains one entry from the Excelerator repository. The import from Excelerator to Visible Analyst works one project or E-file at a time.

There are several issues involved in importing Excelerator repository items into Visible Analyst:

- Excelerator allows names no longer than 32 characters, but the uniqueness rules are less strict than those of Visible Analyst. In Excelerator, an object name must be unique only within its entry type (for example, a data element and a process may share the same name). An import naming convention has been developed to resolve possible problems; Visible Analyst appends names with the entry type name when several entries use the same name.

- Excelerator may have more than 70 different entry types, depending on its configuration, and that number may grow in new versions of Excelerator. All Excelerator entry types can be divided into three categories:

  1 Entry types that have direct correspondence with Visible Analyst entry types are converted to the Visible Analyst type. Still, some of the attributes of such entries do not have corresponding attributes in Visible Analyst. These attributes may be lost or be moved to the Notes field during the transfer.

    Almost all of the Visible Analyst objects are among the Excelerator entry types. Missing are couples, interface table rows, information clusters, different module types, different entity types and domains.

  2 Entry types that don't have analogs in Visible Analyst are ignored.

  3 Diagrams are entries in the Excelerator repository. Because of the great differences between the way these two products draw diagrams, there would be so many problems with the conversion of Excelerator diagrams into Visible Analyst diagrams that Excelerator diagrams are ignored during import in the current release.

- Excelerator allows both an input picture and an output picture for a data element. Visible Analyst uses whichever of these is present. If both exist, the output picture is used.

The list of the Excelerator entry types that can be imported into Visible Analyst is below. Only entry types from the basic Excelerator product (XL/IS) are included in the list. Those from others (XL/DB2, XL/CSP) would be ignored in any case.

Table 7-8
Entry Types Imported from Excelerator

Excelerator ID	Excelerator Type Name	Visible Analyst Entry Type
DAE	Data Entity	Entity
DAR	Data Relationship	Relationship
DAF	Data Flow	Data Flow
DAS	Data Store	Data Store
ELE	Data Element	Data Element
EXT	External Entity	Exernal Entity
FUN	Function	Module
PPS	Primitive Process Spec	Process Description
PRC	Process	Process
REC	Record	Data Structure
DNR	Data N-ary Relationship	Associative Entity + Relationships

KnowledgeWare IEW/ADW Import and Export

Project data from IEW (Information Engineering Workbench) can be exported to and imported from a set of four formatted text files. Those files are relational database files converted into an ASCII format. The file names are fixed by IEW. The following is a simplified description of these files:

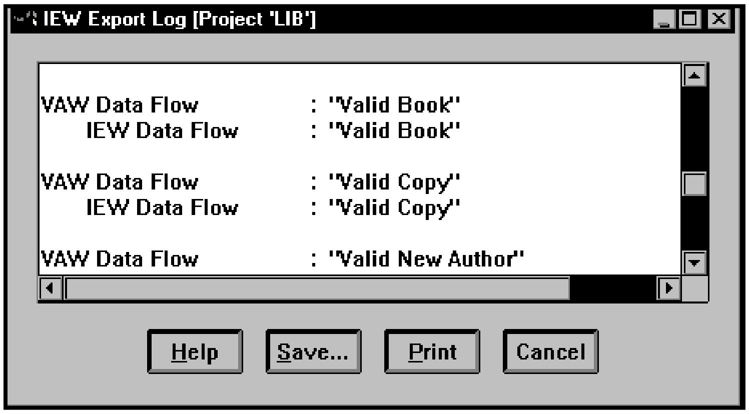
OI.EXP The object file. It contains entry names, types and internal IDs.

AI.EXP The association file. It contains composition/location information and relationships.

PI.EXP The property file. It contains non-text (short) object properties.

TI.EXP The text file. It contains text (long) object properties: description and notes.

The import/export works on a project basis; a Visible Analyst project corresponds to a set of four formatted text files created by/exported to IEW.



```
VAW Data Flow          : 'Valid Book'
    IEW Data Flow      : 'Valid Book'

VAW Data Flow          : 'Valid Copy'
    IEW Data Flow      : 'Valid Copy'

VAW Data Flow          : 'Valid New Author'
```

Figure 7-8 Repository Export Log to IEW Format

Notes on Import

In IEW, a project is designated by the name of the directory in which it is stored. It is probable that the files exported from IEW are stored in the project directory.

- Some IEW attributes get translated to Visible Analyst data elements and some to data structures. If an attribute is a concatenation in IEW, then it is imported as a data structure. Only top-level attributes and concatenations go directly to the composition field of an entity. Attributes that are members of a concatenation are placed in the composition field of the proper data structure. Relationships that are members of concatenations are not placed in the composition field of the data structure; they are imported as relationships.
- Local data types are not imported as Visible Analyst repository entries. They only supply the physical characteristics for the associated attributes (data elements).
- Visible Analyst does not allow entities and relationships in the composition of a data flow. Only data flows and attributes from an IEW flow expression can go to the data flow composition field.
- Visible Analyst does not support exact values for a relationship's cardinality. All values greater than one are interpreted as "many."
- Text fields (IEW definition and comments) get reformatted to fit 60-character lines. For Visible Analyst entries with the composition field (entities, data structures, data flows, etc.), the IEW definition is imported as the Visible Analyst description and is limited to two lines.

Notes on Export

- An export from Visible Analyst in IEW format causes four files to be created in your transient file path.
- In IEW an attribute (data element) cannot exist without being, at some level, in the composition of an entity. Thus, free-standing data elements and data structures do not get exported to IEW. All of the data elements and data structures from the full decomposition of an entity (excluding associator/elements) are exported as attributes of that entity, and data structures generate concatenations.
- In IEW an attribute can only describe one entity. Thus, each Visible Analyst data element or data structure generates as many IEW objects as the number of entities where it is used in the composition field.
- The physical information of data elements generate IEW local data type objects.
- Data flow decomposition is limited to data flows and data elements or data structures that are used in the composition field of an entity.
- For Visible Analyst entries that have three text fields (description, notes and a third one – values & meanings, process description, etc.), the description field and the third field are concatenated into an IEW definition.
- For relationships, text fields for both relationship directions are merged to form one IEW text field.
- IEW requires relationships to be named in both directions. The export procedure generates the relationship name "reverse of …" for the reverse direction of a Visible Analyst uni-directional relationship.
- IEW requires that a creator user name be attached to each object. On export to IEW, this will be set to VAW for each object exported.

General Notes

There are several other issues involved in moving data between the IEW repository and Visible Analyst:

- Creation and modification dates are transported.
- IEW only allows names no longer than 32 characters, but the uniqueness rules are less strict than those of Visible Analyst. Further, IEW has different rules for the characters allowed in an object name. The import and export routines have naming conventions to resolve possible problems.
- IEW objects that have direct correspondence with Visible Analyst entry types are converted to the Visible Analyst type. Still, some of the attributes of such entries do not have corresponding attributes in Visible Analyst. These attributes may be lost or be moved to the Notes field during the transfer. Entry types that don't have analogs in Visible Analyst are ignored.

The list of the IEW objects that can be imported into Visible Analyst is shown in Table 7-9.

Table 7-9
IEW Objects for Import/Export

IEW Object	IEW Type Code	IEW Workstation	Visible Analyst Entry Type
Attribute Type	10003	Planning, Analysis	Data Element or Data Structure
Entity Type	10007	Planning, Analysis	Entity (fundamental, associative, attributive)
Relationship Type	20044	Planning, Analysis	Relationship
Function	10058	Planning	Function
Process	10000	Planning, Analysis	Process
Data Flow	10008	Analysis	Data Flow
Data Store	10012	Analysis	Data Store
External Agent	10002	Analysis	External Entity
Global Data Type	10016	Analysis	Domain
Local Data Type	10048	Analysis	Data Element Physical Inf.

Import from SQL

The SQL import utility allows you to create a Visible Analyst data model from SQL DDL statements. Like other utilities, you can execute it by selecting Import from the Tools menu. The program reads the SQL file, searches for supported DDL statements, and generates Visible Analyst entities, data elements, and corresponding relationships.

The SQL Source File

An SQL file for import is a free-format ASCII file with the following characteristics:

- Tokens (that is, SQL keywords) are separated by spaces, tabs, and new-line characters.
- Single line comments are introduced with two dashes (—) or an exclamation mark (!).
- Multi line comments are introduced with /* and terminated by */ and you cannot have nested comments.

Only DDL statements get Visible Analyst's attention; all others are ignored. Each create table statement generates an entity. Each column in the table generates data elements in the composition field of the entity, with physical information. The create table statements can conform to Ansi-92 standards and are dialect dependent.

There are three types of entities: fundamental, associative and attributive. Visible Analyst determines the entity type based upon the foreign key references. If no foreign key constraints are defined in the create or alter table statements, they can be inferred by enabling the Infer Foreign Keys option.

Alter Table Statements

If a foreign key is referenced outside the scope of the current create table statement, most dialects create an Alter Table Statement at the end of the schema. Visible Analyst reads these alter table statements and generates the appropriate entities if they don't exist and populates the entity with the primary and foreign keys, check constraints, and unique key indexes if this information is provided.

Check Constraints

Visible Analyst supports table and column check constraints as well as primary and foreign key constraints on columns.

Tables/columns are allowed multiple constraints. If no name exists, the constraint is defined using the table/column name. If there are duplicate constraint names, each instance is suffixed with an indexed number.

Primary and Foreign Key Clauses

A primary key clause generates a [PK] prefix before the names of the data elements that are components of the primary key in the entity's composition field. When the Classic User Interface is turned off on the options menu, the primary key is indicated in the repository entry for an entity by a yellow key on a yellow icon displayed before the attribute name in the Attributes field.

A foreign key clause generates a relationship bearing the name of the foreign key. In the case of an unnamed foreign key, the relationship is named Related to. The entity in which the foreign key clause is found is the destination entity for the relationship; the entity referenced by the foreign key is the source entity. The relationship's cardinality on the source entity end is set to 1:1 and that on the destination entity end is set to 0:many. If the entity in which the foreign key is found is an attributive entity (fully dependent on the parent entity), then the relationship's cardinality on the source remains 1:1 but that on the destination entity end is set to 0:1.

When the Classic User Interface is turned off on the options menu, the foreign key is indicated in the repository entry for an entity by a white key on a yellow icon displayed before the attribute name in the Attributes field.

If the Infer Foreign Keys option is selected, Visible Analyst infers foreign key relationships based upon the following criteria:

- Any columns participating in foreign key constraints defined by the schema are ignored during the inferencing process.
- Any column that is part of a primary or unique key is included in the foreign key candidate pool.
- When a column matches an item in the candidate pool in both name and data type (including length and precision), it is marked as a foreign key column.
- A relationship is created between the entity containing the inferred foreign key and an entity that contains the largest number of column matches. If there are multiple entities that have the same number of column matches, a set of supertype relationships is created between a pseudo parent entity named SupertypeNetworkParent_x and each of the matching entities.

Data Element Naming Convention

The Import tool maintains the uniqueness of the generated data element names. If two or more columns under different tables have the same name but have differing data types, the column names are suffixed with an index numbering scheme. If the Import tool generates an element, and the data type is not supported by Visible Analyst, the element is defined as a domain and the type is undefined.

```
create table table_name [ ignored text ]
(
        column_name    storage_type [ ( length [ , dec ] ) ]
                       [ [ not ] null ]
                       [ [ with default | default default_value ] ]
                       [ ignored text ] ,

        [ Primary key  [ primary_key_name]        ( column_name , …) , ]
        [ Foreign key  [ foreign_key_name  ]       ( column_name , …)
                       references                 table_name
                       [ ignored text ],
```

Figure 7-9 Import Format, SQL Create Table Statement

Domains

The import tool supports an object type called domain if importing Rdb or SQLServer dialects.

Unique Indexes

Create unique indexes generates alternate keys in an entity as long as there is a primary key defined or the index is part of a concatenated primary key for the entity. If no primary key exists for the entity, the unique index is referenced as a primary key.

Performance Indexes

Create indexes (that is, non-unique indexes) generate performance indexes in an entity.

Comments on Tables and Columns

Any comments associated with a particular column or table appears in the repository description field of the object. There is an upper limit of 160 characters per comment.

Stored Procedures

If your dialect supports stored procedures, Visible Analyst creates a module with a subtype of stored procedure when a stored procedure is encountered upon import. The text associated with every stored procedure in the DDL is stored in the module description field in Visible Analyst. If using Informix, you must have an “end procedure” statement following the stored procedure. If using SQLServer, you must include a newline Go newline after each stored procedure. For Rdb, you must have an “end module;” and for Oracle Server, a balanced begin/end block is required. Currently Rdb is the only dialect to support a module. When a module is encountered upon import, a corresponding stored procedure is created in the Visible Analyst repository; and the text associated with the module is stored in the module description field.

Triggers

Rdb, Oracle Server, SQLServer, Informix, and DB2 dialects support triggers. For SQLServer, the trigger must be followed by a newline Go newline statement or an EOF. Each trigger is associated with a table. It is always fired after; and Visible Analyst supports insert, delete or update commands. For Oracle Server, each trigger is associated with a table. It can be fired before or after; and Visible Analyst supports insert, delete, update with a column list and a firing scope for either the row or table. For Rdb, the support is the same as Oracle Server except the firing scope is specified as part of the trigger body.

Tablespace

Create Tablespace statements generates a tablespace object in the repository. If an entity or index references a tablespace, the link is maintained. For SQLServer, the sp_addsegment() and sp_extendsegment() stored procedures also generate a tablespace object. You must have DBA privileges to extract physical properties.

Views

Create View statements generate view objects in the repository. The composition field is populated with the list of columns used by the view, and the Select statement is converted to a database-neutral format. This allows views to be regenerated for any target database.

Identity Clause Support

SQLServer version 6.x, 7.x, 2000, 2005 and System 10/11 are the only dialects to support identity clauses. If a numeric data type is encountered and the identity clause follows this data type in the SQL DDL, upon import Visible Analyst creates this column with an Allow Null value of Identity.

RDBMS Catalog Import

Data model information can be transferred between any of the supported RDBMSs and Visible Analyst in a seamless fashion. This function populates a Visible Analyst project database with the data model information as defined in the RDBMS. The supported RDBMSs are Centura SQLBase, SQLServer, Oracle Server, DB2, and Informix, as well as any ODBC compliant database. Please refer to the individual RDBMS for installation guidelines.

The information captured includes all entity and attribute definitions, any key information, as well as views, triggers, constraints and stored procedures, if your RDBMS supports them. Refer to Import from SQL described above for more information.

You may need DBA privileges to extract physical properties.

To import RDBMS database definitions into Visible Analyst:

- Select Import from the Tools menu, choose RDBMS Catalog and click OK.
- The Select Data Source dialog is displayed. This dialog box lists the ODBC drivers installed on the PC which are used to connect to the database. After selecting the appropriate ODBC driver, or creating a new one, the connection is made to the database. You will be prompted to enter the database login and connection information before the connection is made. Refer to Connecting to an RDBMS Engine in Getting Started for more information on the dialog box.
- Visible Analyst reads the RDBMS database definitions, converts them to SQL DDL, and displays an import action list (see Figure 7-7), showing a summary of the objects to be imported. If this import action list is correct, simply click the Import button at the bottom of the screen; and the repository is populated with your data model information. You can

then have Visible Analyst automatically generate an ERD from the data model information that was imported.

DDS Import
An IBM AS/400 DDS (Data Description Specification) import allows the creation of a data model from DDS statements, and is similar to an SQL import. A DDS source file is an ASCII file with 80-character, fixed length, column position formatted lines. The format is shown in Table 7-10.

Table 7-10
DDS Source File Format

Field #	Columns	Meaning
1	1-5	Line number.
2	6	Always “A”.
3	7	‘*’ indicates the line is a comment.
4	17	‘R’ = record, ‘K’ = key, ‘ ‘ = element.
5	19-28	Record/element name (10 characters).
6	30-34	Length (elements only).
7	35	Storage type (elements only): ‘P’ = packed dec.,, ‘S’ = zoned dec.,, ‘B’ = binary,, ‘F’ = float, ‘A’ = character,, ‘H’ = hexadecimal, ‘L’ = date,, ‘T’ = time>, ‘Z’ = timestamp.
8	36-37	Decimal places.
9	45-80	Attributes: ALWNULL = allow null DFT(value) = default TEXT(‘text’) = comment VARLEN = variable length (character). FLTPCN(*DOUBLE) = double precision.

If the attributes for one element do not fit in columns 45-80, they can be continued on the following line(s) in columns 45-80. Key specifications follow the element specifications. Below is a DDS example:

```
00010A          R      PERSON                   TEXT(‘Entity Person’)
00020A*        Elements
```

```
00030A                PERSON_ID   10     A       TEXT(‘Person ID’)
00040A                NAME        40     A       TEXT(‘Name’)
00050A                INFO        400     A      TEXT(‘Info’) ALWNULL
00060A                                           VARLEN
00070A                WEIGHT      6      P2      DFT(200.00)
00080A                HEIGHT      4      P1      DFT(6.0)
00090A*        Key Components
00100A            K   PERSON_ID
```

The DDS Name Translation selection available on the Options menu allows you to tailor how object names are mapped to Visible Analyst object names during DDS import. There are three mapping schemes available:

- Logical to Alias. A Visible Analyst object name is mapped to the ALIAS( ) field.
- Logical to Colhdr. A Visible Analyst object name is mapped to the COLHDR( ) field.
- Logical to Text. A Visible Analyst object name is mapped to the TEXT( ) field.

Import Data from Application Browser

Data from Application Browser can be imported into Visible Analyst. COBOL code can be put into the Application Browser Code Base and viewed in its diagrams. This import procedure enters this COBOL information into the repository of a Visible Analyst project created in Visible Analyst, using the Import tool from the Tools menu.

Conversion Information

COBOL Data Division items are converted into data structures and data elements. The overall structure of the code creates structure chart items in the Visible Analyst project repository. When you print reports on such a project, structure chart items have diagram location references. These references appear as “|virtual”. This is because the import procedure creates a virtual diagram in Visible Analyst while creating structure chart items from Code Base information. You can generate structure charts from this repository information.

In addition to the procedure modules, files, data structures, and data elements created, one data only module is created for each COBOL module (source file). It contains the 01 level Data Division definitions. The name of this module is the COBOL source module name with the suffix _DATA.

If a paragraph contains a PERFORM statement that executes a single paragraph, a standard invocation is created. If multiple paragraphs are performed, for example PERFORM P1 THRU P5, a logical module is created with the name P1..P5 with a lexical inclusion invocation to it. Invocations for the individual paragraphs are created from the P1..P5 module. GOTO statements between these paragraphs generate control connections between the modules in Visible Analyst.

FALLTHRU statements are usually ignored because their action is implied by the standard method of connecting modules. However, in the case of pathological connections that jump out and then back into the list of paragraphs executed by a PERFORM THRU block, FALLTHRU statements are converted into control connections to show the flow through modules outside the list.

A CALL statement in COBOL that calls an external source module generates an invocation from the calling module to the top module of the called program module hierarchy. Any parameters used with the CALL statement in the original COBOL code are lost in the current version of Visible Analyst.

OCCURS clauses in the COBOL source are semantically translated properly into Visible Analyst if the OCCURS value is a simple numeric. Complex OCCURS clauses containing DEPENDING ON, INDEXED BY, etc., are placed into the item Notes field.

Other Conversion Information

Copy libraries used in the original COBOL source are expanded in-line and incorporated into the Visible Analyst repository in their expanded forms.

Currently, the RENAMES statement in COBOL is not supported in Visible Analyst.

COBOL paragraph comments are placed in the Notes field of the module representing the paragraph.

The external name for a COBOL file appears in the Notes field of the file in Visible Analyst.

A Visible Analyst data structure is created for each non-lowest-level item in a COBOL Data Division data definition.

Lowest-level items, including non-subdivided level 01 items and level 77 items, become data elements.

Level 88 statements have their values listed in the Values & Meanings field of the data element with which it is associated in the COBOL source.

Naming Considerations

There are certain name transformations that occur during the import because of different naming rules in the Application Browser and Visible Analyst. First, Application Browser names that begin with a number are prefixed with the letter “A” to conform to the Visible Analyst rule that names must begin with a letter. Second, a suffix is added to Application Browser names that are used in different places with different meanings, so that they can be distinguished in Visible Analyst. During the import, an intermediate list of names is created. Names from the COBOL Data Division hierarchy are appended to the item name, separated by periods. Then the names are shortened, keeping only the information necessary to make the names unique. Because of the 128-character name length limit in Visible Analyst, characters may be removed from the middle of the suffixes and numbers added to maintain uniqueness within the name length limits.

Code Generation Implications

The source code of a COBOL paragraph is saved in the Module Description field of the Visible Analyst module that is generated from the paragraph. If you later want to generate code from the Visible Analyst project, you should first make sure you have made the correct settings in the Shell Code Generation Options dialog box. This is accessed by selecting Code Generation Options from the Options menu. In the box labeled Included Module Information, be sure that the Module Description item is checked so that the Module Description field is included in the generated code. Also, make sure that the Code button is selected so that the Module Definition field is interpreted as code and not just as a comment.

Powersoft PowerBuilder Interface

Data model information can be transferred between Visible Analyst and the PowerBuilder application development system from Powersoft. This information includes entity and attribute definitions and extended attributes, such as comments on tables and columns, and validation rules, information that is stored in the PowerBuilder system tables.

To transfer data to PowerBuilder:

- Select Export from the Tools menu, choose PowerBuilder and click OK.

The process is similar to SQL schema generation in that you must choose the data you want to generate. SQL DDL statements are created and are executed against the PowerBuilder system tables. This, in turn, creates tables, columns and validation rules.

To import validation rules from PowerBuilder:

- Select Import from the Tools menu, then choose PowerBuilder format.

Note

- The directory where PowerBuilder is installed must be included in your PATH, because Visible Analyst must execute PowerBuilder programs to transfer data.

Uniface Interface

Data Model information can be transferred between Visible Analyst and the Uniface application development system from UNIFACE. The Visible Analyst import/export is fully compatible with Uniface v5.2. The information transferred includes entity and attribute definitions, including short descriptions and notes fields, primary and alternate keys, and relationships based upon the foreign key definitions. This information is stored in the conceptual schema derived using IDF.

To transfer data to Uniface:

- Select Export from the Tools menu, choose Uniface, and click OK.

Data Model information is exported to an ASCII file in the Uniface CASE Interface Format (CIF). This file can be imported into Uniface using the CASE load feature.

To import database definitions from Uniface:

- Choose the SQL dialect you would like the conceptual schema data types and interface syntax to map to by choosing SQL Dialect from the Options menu.
- Select Import from the Tools menu, and choose the Uniface format.
- Select the CIF file to import. This file is created using the CASE unload feature of Uniface to export a conceptual schema to the CASE Interface Format. You can use the point-and-click method to highlight the file, or type the filename and PATH.

Describe/GDPro Interface

Object models (class diagrams) can be exported to and imported from GDPro. Information transferred includes class diagrams, class objects, attributes, methods, and friends. It is a two-step process.

To export from Visible Analyst:

- Select Export from the Tools menu, select GDPro and click OK.
- Start GDPro, and select Import From Visible Product from the Tools menu.
- Type the name of the file to be imported and click OK.
- The file created by Visible Analyst is called “project root”.ad and is located in the transient data directory.

A class is created in the target GDPro system for every class, entity, associative entity, and attributive entity defined in the VSC export file. If the class subtype is set to structure, the ClassFormat property is set to struct. If the subtype is union, ClassFormat is set to union. All other subtypes are created as standard classes.

Inheritance/supertype relationships are created as CLD_Generalization_Links, while all other relationship types are created as CLD_ClassAssociations. Class diagrams and entity-relationship diagrams are created as class views. Data elements (attributes) and methods are only imported if they belong to a class/entity. Free standing data elements are ignored. If a data element is used by more than one object, its definition is repeated in each class. Any spaces that appear in the name of an object are replaced with underscores.

To import from GDPro:

- Start GDPro, and select Export to Visible Product from the Tools menu.
- Select Import from the Tools menu, select GDPro, select the file generated in step one and click OK.

For each class defined in GDPro, a class is created in the VSC import file. The subtype is set according to the ClassFormat attribute.

Relationships between classes are created for both CLD_Generalization_Link and CLD_ClassAssociation objects. Class diagrams are created for each class diagram view. Class attributes are created as local data elements.

Table 7-11
Object Property Mappings

	Visible Analyst Property	GDPro Property
Class	Description	ClassDescription
	Notes	ClassDescription (appended)
	Friend	FriendDefinition
	Persistent	PersistentClass
	Abstract	ClassType
	Attributes	ClassAttribute
	Methods	Operations
Data Element	Description	AttributeDescription
	Values and Meanings	AttributeDescription (appended)
	Notes	AttributeDescription (appended)
	Type	Type
	Length	ArraySpecifier
	Default	InitialValue
	Visibility	AttributeVisibility
	Qualification	AttributeScope, Constant, Volotile
Method	Description	OperationDescription
	Notes	OperationDescription (appended)
	Visibility	OperationVisibility
	Qualification	OperationType
	Returns	ReturnType
	Arguments	Parameters
Relationship – Normal	Label	AssociationName
	Role	Role
	Cardinality	RoleMultiplicity

Table 7-11
Object Property Mappings

	Visible Analyst Property	GDPro Property
Relationship – Supertype	Discriminator Visibility Virtual	Discriminator InheritanceMethod Virtual

Note

- Spaces that appear in names are changed to underscores.
- If a global object is referenced by a class, it becomes a local object for each class that references it. All information for that object is duplicated in each class.

ERwin Interface

Data models can be both exported to and imported from ERwin. Information transferred includes entities, relationships, attributes, triggers, and constraints.

To export from Visible Analyst:

- Select Export from the Tools menu, select ERwin (ER1), and click OK.
- Select the name and location of the ER1 files to be created, and click OK.

To import from ERwin:

- Select Import from the Tools menu, select ERwin 3.x choose the desired ER1 file, and click OK.

Select Import from the Tools menu, select ERwin (XML) as the import file type for ERwin v4.x, 7.x, select the XML file and click OK.

XML Schema

An XML Schema based on the W3C standard for the entities and (optionally) classes can be generated via the Tools | Export menu. To include classes in the generated schema, add the classes to an entity diagram.

XMI Import

Static structure models (class diagrams) can be imported from an XMI document created by other modeling tools such as Rational Rose. Before the file is imported, it is parsed for correctness according to the UML.DTD supplied with Visible Analyst. This file must exist in the same directory as the XML file being imported.

To import an XMI document:

- Select Import from the Tools menu, select XMI.

- Choose an XML file that contains an XMI specification, and click OK.

If the XML file is valid, classes and relationships are imported into the Visible Repository.

Chapter 8

Enterprise Modeling

ENTERPRISE MODELING OVERVIEW

Visible Analyst supports enterprise modeling by linking together projects in a parent-child relationship. The enterprise project is a large corporate-wide project that contains data from a number of subordinate or satellite projects. The enterprise project is used to warehouse all information pertaining to an organization, while the satellite project contains only a subset of the enterprise project allowing users to work with components that are important to their development and not be bothered with the details of the entire enterprise.

Information is transferred between the enterprise and satellite projects through the use of divisions. If you are working in the satellite project, you have access not only to the objects defined in that project, but also to objects defined by the division in the enterprise project that you have access to. The type of access depends on how the enterprise link was created. You have a choice of read-only access, meaning enterprise objects *cannot* be modified, or proposed change access, meaning enterprise objects can be modified but the modifications are not reflected in the enterprise project. If a user is given proposed change access, this can be further restricted to individual objects. Users with system manager access see a checkbox on the divisions screen that allows them to see all of the divisions defined for the project.

The Enterprise Copy tool is used to synchronize a satellite project with the enterprise project. There are extensive reconciliation tools to manage the synchronization process, including a difference analysis tool that displays side-by-side the differences between an enterprise object and a satellite object.

When viewing object definitions in the enterprise repository, the satellite projects that use the object are noted; and in the satellite, the enterprise project is displayed. There is no limit to the number of enterprise projects that can be maintained, nor to the number of satellite projects that can be connected to an enterprise project.

Divisions

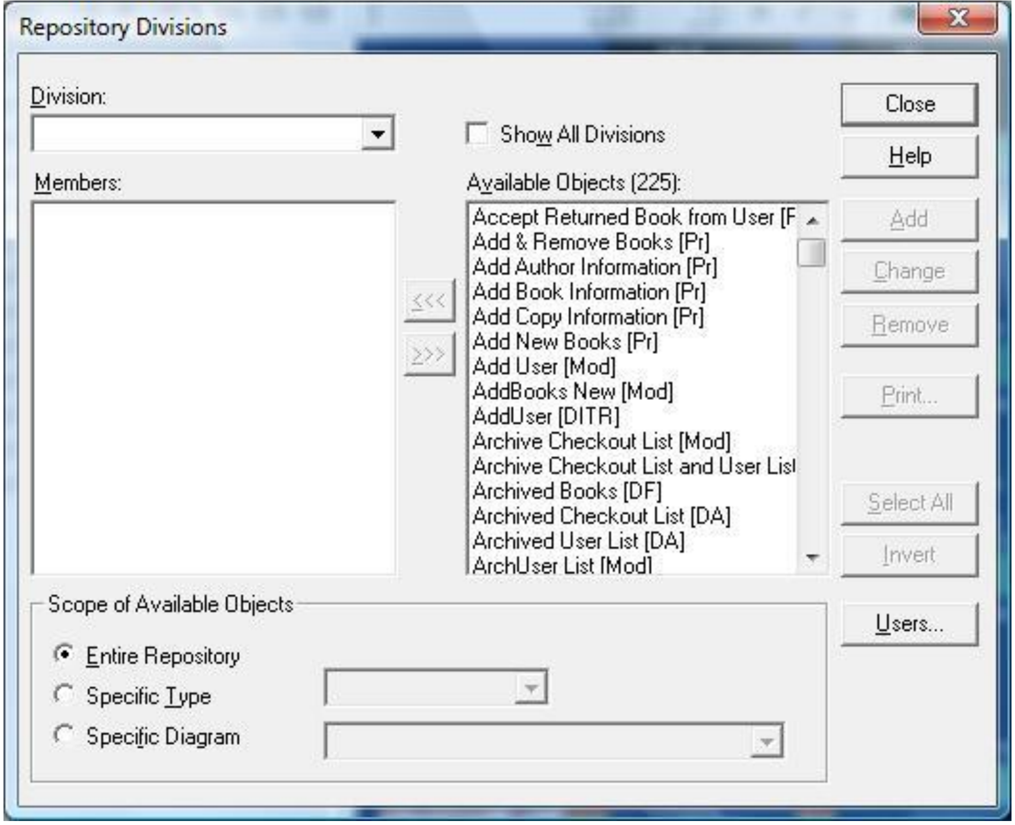
A division is a logical group of repository objects or diagrams that can be used to transfer data between an enterprise project and a satellite project. Divisions can also be used to restrict access to objects within a project. This section explains the techniques for working with divisions within the repository.

Creating Divisions

To create a new division or to modify an existing one:

Open the Dialog Box: 1 Select Divisions from the Repository menu.

Choose a Division Name: 2 If you want to create a new division, type the name in the Division box. If you want to change an existing division, select it from the drop-down list. (See Figure 8-1.)



Available Objects (225):

Accept Returned Book from User [F
Add & Remove Books [Pr]
Add Author Information [Pr]
Add Book Information [Pr]
Add Copy Information [Pr]
Add New Books [Pr]
Add User [Mod]
AddBooks New [Mod]
AddUser [DITR]
Archive Checkout List [Mod]
Archive Checkout List and User List
Archived Books [DF]
Archived Checkout List [DA]
Archived User List [DA]
ArchUser List [Mod]

Scope of Available Objects: ● Entire Repository ○ Specific Type ○ Specific Diagram

Figure 8-1 Repository Divisions Dialog Box

Note

- Only divisions to which you have been assigned *Access Control* rights appear in the division list. See Assigning Users to Divisions later in this chapter for more information.

Narrow the Scope of Available Objects: 3 If you have a large number of objects in your repository, you may want to limit the number of items that appear in the Available Objects list.

To display all objects in the repository, choose Entire Repository.

To display all objects of a specific class or type, choose Specific Type and then select the model type and object type you wish to use.

To display all objects on a specific diagram, choose Specific Diagram and then select the diagram type and diagram you wish to use. The Available Objects list contains all the objects on the diagram and the name of the diagram itself.

Note

- If you choose Specific Diagram and no objects appear in the Available Object*s* list, and the project was created with version 6.0 of Visible Analyst, you may need to Rebuild the project or to edit the diagram for the objects to appear.
- If there are a large number of objects in the repository, the Available Objects list may be updated incrementally. The list is initialized and then updated every few seconds until all objects in the selected scope have been added.

Select Members: 4 A division is made up of repository objects or diagrams.

To add items to a division, highlight the desired objects in the Available Objects list and click ≺≺ to move them into the Members list.

To remove members from a division, highlight the members in the Members list and click ➢➢ to move them into the Available Objects list.

To select all items in the current list (the current list has the focus), click Select All. To invert the selections, click

Invert. Invert is handy if you want to select everything in a list except a few items; select the items you want to exclude and then click Invert. Both Select All and Invert are only applicable if the focus is on either Members or Available Objects.

If the scope is set to a specific diagram and you select the diagram entry, all items in the list are selected. If you then de-select one of the objects, the diagram selection is also de-selected. Any action that causes the diagram entry to be selected causes all items in the list to be selected.

Update the Repository: 5 When you have finished modifying a division, select Add or Change*:*

To add a new division, click the Add button. If the division does not already exist, it is added to the repository.

To change the name of a division or its membership list, click Change. If you alter the name of an existing division, you can add a new division with same membership list or simply change the name.

Note

If you have not been assigned Create Division rights, you cannot add divisions to a project.

Assign Users: 6 To grant access to a division, click the Users button. This allows you to assign specific users to a division, each having a different set of rights. If a repository object or diagram is a member of a division, only users who have been assigned rights to that division can access the object or diagram. See Assigning Users to Divisions later in this chapter for more information.

Deleting Divisions

To remove a division from the repository:

Open the Dialog Box: 1 Select Divisions from the Repository menu.

Choose a Division: 2 Select the division to delete from the drop-down list.

Delete the Division: 3 Click Remove to delete the division from the repository. If the division has enterprise links, you must remove the link before deleting the division. See Removing Enterprise Links later in this chapter for details.

Assigning Users to Divisions

If a repository object or diagram is a member of a division, only users who have been assigned rights to that division can access the object or diagram.

To assign user rights:

Open the Dialog Box: 1 Select Divisions from the Repository menu and then click the User*s* button on the Repository Divisions dialog box.

Choose a Division: 2 Select the division whose user list you wish to modify from the drop-down list.

Note

- Only divisions to which you have been assigned Access Control rights appear in the division list.

Select Users: 3 To add users to a division, highlight the desired items in the Available Users list and click ≺≺ to move them into the Users list.

To remove users from a division, highlight the items in the Users list and click ➢➢ to move them into the Available Users list.

To select all items in the current list (the current list has the focus), click Select All. To invert the selections, click Invert. Invert is handy if you want to select everything in a list except a few items; select the items you want to exclude and then click Invert. Both Select All and Invert are only applicable if the focus is on either Users or Available Users.

Update the User List: 4 When you have finished modifying a user list for a division, click Change to update the repository.

Select User Rights: 5

To change the rights of a user assigned to a division, highlight the user. The user’s current rights are displayed in the Rights box. When a new user is added to a division, the division rights are set to the project rights. Users can have the following rights in a division

Delete Diagrams**.** The user can delete any diagram in the division.

Delete Items. The user can delete any item in the division. This includes deletion from the repository and deletion from a diagram.

Modify Diagrams. The user can add or move items on a diagram.

Modify Items. The user can change the definition of a repository object.

View Diagrams. The user can display diagrams in the division, but if they have no other diagram rights, they cannot alter the diagrams.

View Items. The user can display repository entries in the division, but if they have no other object rights, they cannot alter the entries.

Access Control. The user can assign other users to the division and change their rights, and modify the division membership list.

Note

If more than one user is selected, the rights displayed are a combination of all the rights for the selected users.

Change User Rights: 6

To change the rights of the selected users, click Modify Selected Users. To change the rights of all the users in the division, click Modify All Users.

Enterprise Copy

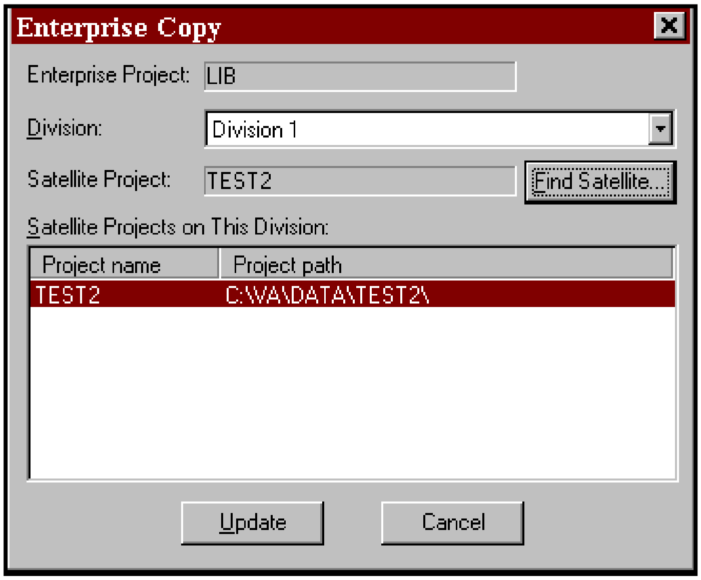
Enterprise Copy is used to create the connection between one project and another. The source project is the enterprise, while the target is the satellite. After Enterprise Copy has been performed, the members of the synchronized division have the same definition in both projects. Enterprise Copy division rights can also be used to "hide" or not display items on the satellite project diagrams copied from the enterprise project. These non-displayed items can be attributes of selected classes, entities, etc. or selected diagram objects and lines. See the section "Hiding Satellite Objects" later in this chapter.

To synchronize an enterprise project with one of its satellites:

Note

Before starting an Enterprise Copy operation, it is a good idea to make a backup of both the enterprise and satellite projects.

Open the Dialog Box: 1 Select Enterprise Copy from the Tools menu. The current project is used as the enterprise project.



Enterprise Project: LIB

Division: Division 1

Satellite Project: TEST2

Project name	Project path
TEST2	C:\VA\DATA\TEST2\

Figure 8-2 Enterprise Copy Dialog Box

Choose a Division: 2 Select the division to be synchronized. If no divisions exist, see Creating Divisions earlier in this chapter for an explanation of the procedure.

Select a Satellite Project: 3 If this is the first time the division has been used in an enterprise copy operation, click Find Satellite to choose a satellite project. Otherwise, select the appropriate project from the list of satellite projects that are already connected to this division.

Start the Synchronization Process: 4 Click Update to start the enterprise copy. If conflicts are detected between objects in the enterprise project and objects in the satellite, you must select the appropriate update action.

Select the Update Actions: 5 For each object that has a different definition in the enterprise and satellite projects, follow the process outlined in Selecting Update Actions to determine the definition to be used in both projects after the synchronization process.

Resolve Name Conflicts: 6 If there are name conflicts, follow the procedure outlined in Resolving Object Name Conflicts.

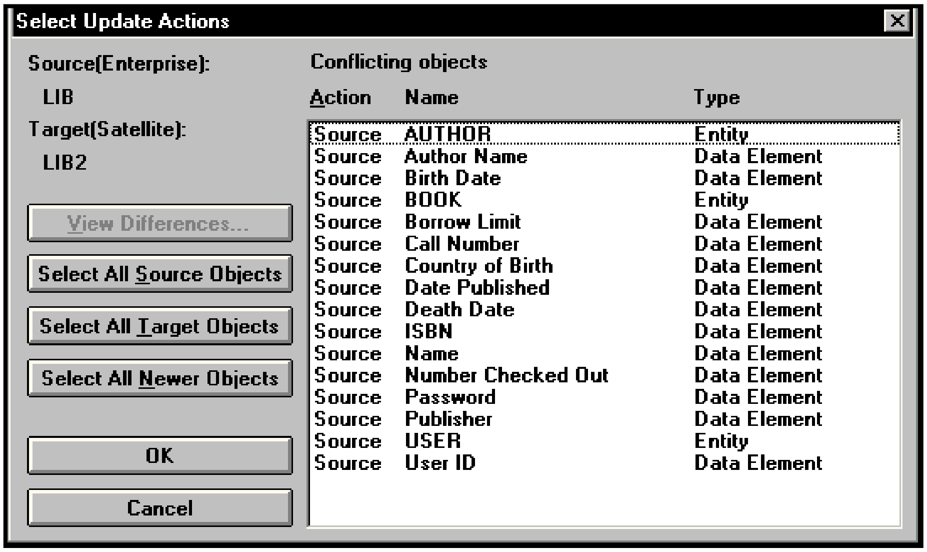
Assign Rights to the Division: 7 The first time a division is used in an enterprise copy operation, only the user performing the copy has rights to the objects in the division in the satellite project. To add other users, follow the procedure outlined in Assigning Users to Divisions earlier in this chapter.

After the Enterprise Copy operation is complete, each object in the synchronized division has a link between the two projects. If you examine the definition of a synchronized object by selecting Define from the Repository menu, the enterprise object has all satellite projects listed in the Locations field, while the satellite object notes the enterprise project in the upper right corner of the Define dialog box. See Displaying Enterprise Links later in this chapter for more information.

Selecting Update Actions

If the enterprise copy procedure detects an object in the enterprise project that has a different definition in the satellite project, you must choose the definition to be used in *both* projects after the synchronization process is complete.

Each object that has a conflicting definition is displayed in the Conflicting Objects list with a default action of either a question mark (?) or Advise. A question mark means neither a source nor a target definition has been selected. Advise indicates either the item exists outside the division being synchronized, or the object has a different type in each project. These items must be reviewed individually before the update process can continue. (Figure 8-3)



Source(Enterprise):
LIB

Target(Satellite):
LIB2

Conflicting objects

Action	Name	Type
Source	AUTHOR	Entity
Source	Author Name	Data Element
Source	Birth Date	Data Element
Source	BOOK	Entity
Source	Borrow Limit	Data Element
Source	Call Number	Data Element
Source	Country of Birth	Data Element
Source	Date Published	Data Element
Source	Death Date	Data Element
Source	ISBN	Data Element
Source	Name	Data Element
Source	Number Checked Out	Data Element
Source	Password	Data Element
Source	Publisher	Data Element
Source	USER	Entity
Source	User ID	Data Element

Figure 8-3 Select Update Actions Dialog Box

In order to complete the copy operation, each object in the list must use either the source definition or the target definition. When making your selections, you have the following options:

- **Select All Source Objects**. All definitions from the source (or enterprise) project are used.

- **Select All Target Objects**. All definitions from the target (or satellite) project are used.
- **Select All Newer Objects.** All objects with the latest modification date are used. If the objects were modified on the same day, you must visually inspect the definitions to determine the appropriate action.
- If you want to inspect the differences between an object pair, select the item from the list and click View Differences. Once all object definitions have been selected, click OK to continue the copy procedure.

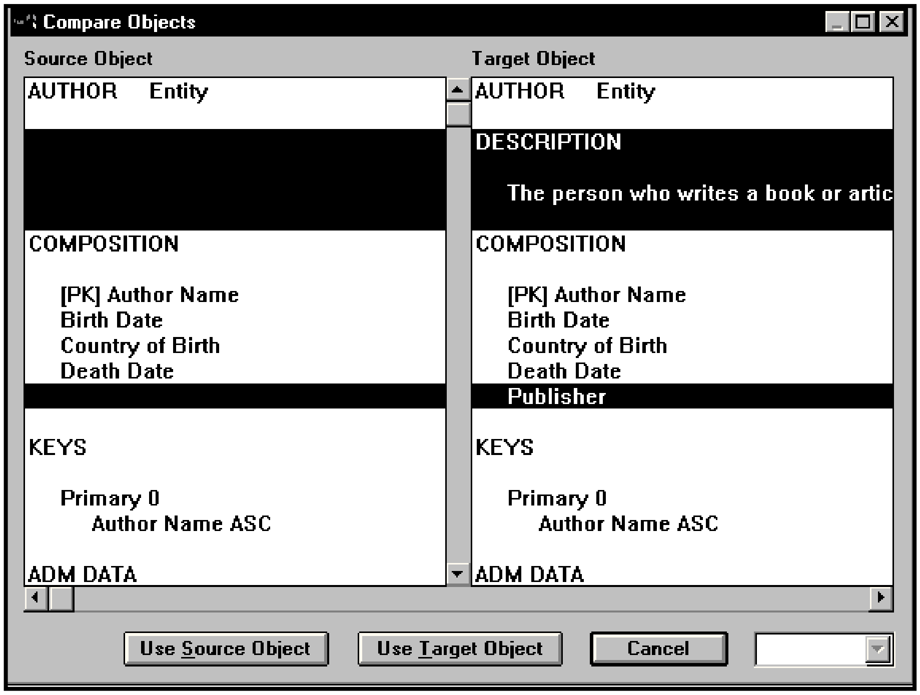
Note

- If an object appears in one project but not the other, it is listed in the conflict list. If you choose the project that does not contain the object, the entry is removed from both projects.
- One selected action may impact another object in the list. For example, if in the enterprise project you add a new data element as a primary key column to an entity, and that column does not exist in the satellite, and you choose to use the entity definition from the enterprise but the column definition from the satellite, not only is the data element deleted but the primary key is also removed from the entity.

Compare Objects

If the definition of an object or diagram differs in the enterprise project and the satellite project, you must choose the definition to be used during the enterprise copy operation.

The Compare Objects dialog box displays the definitions side by side so that you can visually inspect the differences. To use the source (or enterprise) definition, select Use Source Object. To use the target (or satellite) definition, select Use Target Object. To defer making a choice, select Cancel. If the item being inspected is a diagram, you can set the zoom level using the Zoom box in the bottom corner of the dialog box. (See Figure 8-4.)



Source Object

```
AUTHOR     Entity



COMPOSITION

    [PK] Author Name
    Birth Date
    Country of Birth
    Death Date


KEYS

    Primary 0
        Author Name ASC

ADM DATA
```

Target Object

```
AUTHOR     Entity

DESCRIPTION

    The person who writes a book or artic

COMPOSITION

    [PK] Author Name
    Birth Date
    Country of Birth
    Death Date
    Publisher

KEYS

    Primary 0
        Author Name ASC

ADM DATA
```

Figure 8-4 Compare Objects Dialog Box

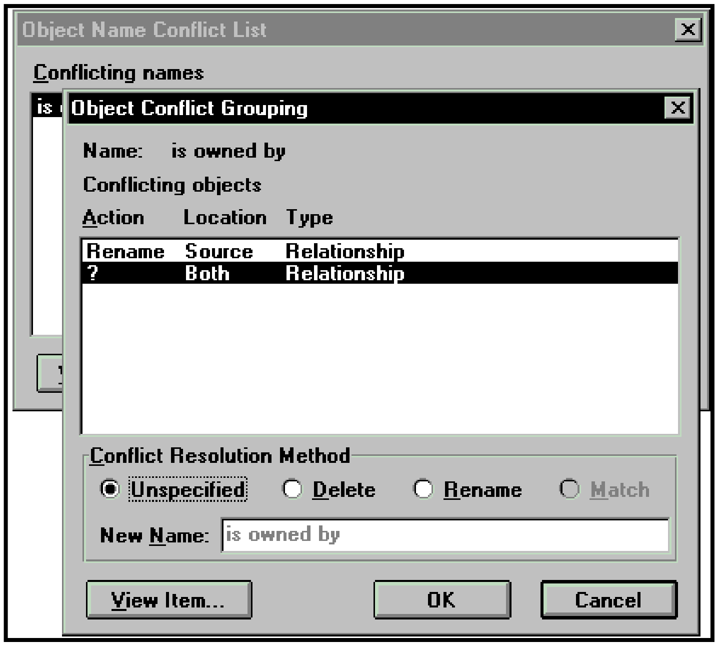
Resolving Object Name Conflicts

To resolve conflicts if the enterprise copy procedure detects a name conflict between the source and target projects:

Select the Conflicting Object:	1	From the Conflicting Names list, select the object and click View Conflict. A list of conflicting objects is displayed indicating whether the object is in the target project, the source project, or both.
Choose a Conflict Resolution Method:	2	Select an object from the Conflicting Objects list. Select Delete to remove the conflicting item or Rename to change its name. These two options are valid if the item exists in the source project, the target project, or both projects. If a conflicting object does not exist in

both, the Match option can be used to connect an object from the source to an object in the target.

You must select two items from the list with the Match option, one from the source and one from the target, and then choose the one to be the new object in both projects. Click OK to confirm your selections. (See Figure 8-5.)



Name: is owned by

Conflicting objects

Action	Location	Type
Rename	Source	Relationship
?	Both	Relationship

Conflict Resolution Method

● Unspecified ○ Delete ○ Rename ○ Match

New Name: is owned by

Figure 8-5 Conflicting Objects Dialog Box

Continue Copy Operation:

3 Click Reanalyze on the Conflicting Names dialog box to continue with the enterprise copy. If there are still problems, repeat steps 1 and 2 until all conflicts have been addressed. When the list contains only conflicts that have been corrected, click OK to continue.

Displaying Enterprise Links

If an object being examined in the repository Define dialog box has enterprise links to other projects, a marker is displayed to the right of the object name. If you double-click on the marker, a list of enterprise links is displayed. Each link displays the following information.

- **Project**. The name of the project to which this object is linked.
- **Type**. The type of link. Enterprise indicates the current project is the parent project, while Satellite indicates the current project is the child.
- **Synchronized.** The last date an Enterprise Copy operation was used to synchronize the projects.
- **Location.** The drive and directory where the linked project is stored.

Removing Enterprise Links

To remove the links between an enterprise project and one of its satellites:

Open the Dialog Box: 1 Select Enterprise Tag Maintenance from the Tools menu. The current project is used as the enterprise project.

Choose a Division: 2 Select the division whose members are to have their link removed.

Note

- Only the link between the two projects is removed. The objects themselves remain in both projects.

Select a Satellite Project 3 Select the appropriate project from the list of satellite projects that are connected to this division.

Start the Removal Process: 4 Click Remove to sever the link. You are prompted to to confirm your action.

Repeat the Process: 5 If you want to select another satellite project or another division to update, start the process again beginning with either step 2 or step 3. When you are finished, click Close.

After the remove operation is complete, each object in the previously synchronized division has its link between the two projects deleted.

Database Synchronization

The enterprise modeling comparison facilities can also be used to synchronize an entity-relationship model with a generated database. To perform the synchronization, follow the steps below.

Create a New Project: 1 Create a new project that contains the database to be synchronized.

Import the Database: 2 Select Import from the Tools menu. Using either the SQL Import or the RDBMS Catalog Import option, create a data model based on the database.

Create a Division: 3 Create a division in the new project encompassing the entire repository. See Creating Divisions earlier in this chapter for details.

Perform Enterprise Copy: 4 Select Enterprise Copy from the Tools menu. Follow the steps outlined earlier in this chapter for synchronizing an enterprise and satellite project. When the process is complete, the original data model is updated to reflect any changes that were made in the database.

Remove Enterprise Links: 5 Select Enterprise Tag Maintenance from the Tools menu to remove the links between the two projects.

Delete the Temporary Project: 6 Select Delete Project from the Tools menu to remove the temporary project.

Note

- If you have denormalized any relationships or used any structures such as domains in your original data model and the target database does not support those structures, you cannot completely synchronize the model with the database.

Printing Division Reports

Selecting the Print button on the Repository Divisions dialog opens the Print Division dialog. This dialog allows you to generate an HTML file listing all items in the selected division. The report can be sorted Alphabetically or by Entry Type. When Entry Type is selected, the Show Composition (Attributes) checkbox includes the attributes of component division items. These items include entities, classes, data elements, etc.

The report can also mark items that have been changed.
None: Specifies no marking of division members will take place
Since: If checked, items that have been changed since the selected date will be marked in red.

Since last synchronization (Enterprise Copy) with: If checked, items that have been changed since the date of the last enterprise copy synchronization with the selected project will be marked in red.

Hiding Satellite Objects

Users may choose not to display or print all of the information displayed on a satellite project diagram. Entity Attributes not germane to the end users, or processes external to the discussion can be hidden through the use of Division Rights assignment. To select the items not to be displayed in the satellite project diagrams, follow the steps below.

Create a Division: 1 Create a division in the enterprise project containing all objects to be transferred to the satellite project.

Perform Enterprise Copy: 2 Select Enterprise Copy from the Tools menu. Follow the steps outlined earlier in this chapter for synchronizing an enterprise and satellite project.

Create a New Division: 3 In the satellite project, create a new division that contain the objects that will be "hidden". See Creating Divisions earlier in this chapter for details.

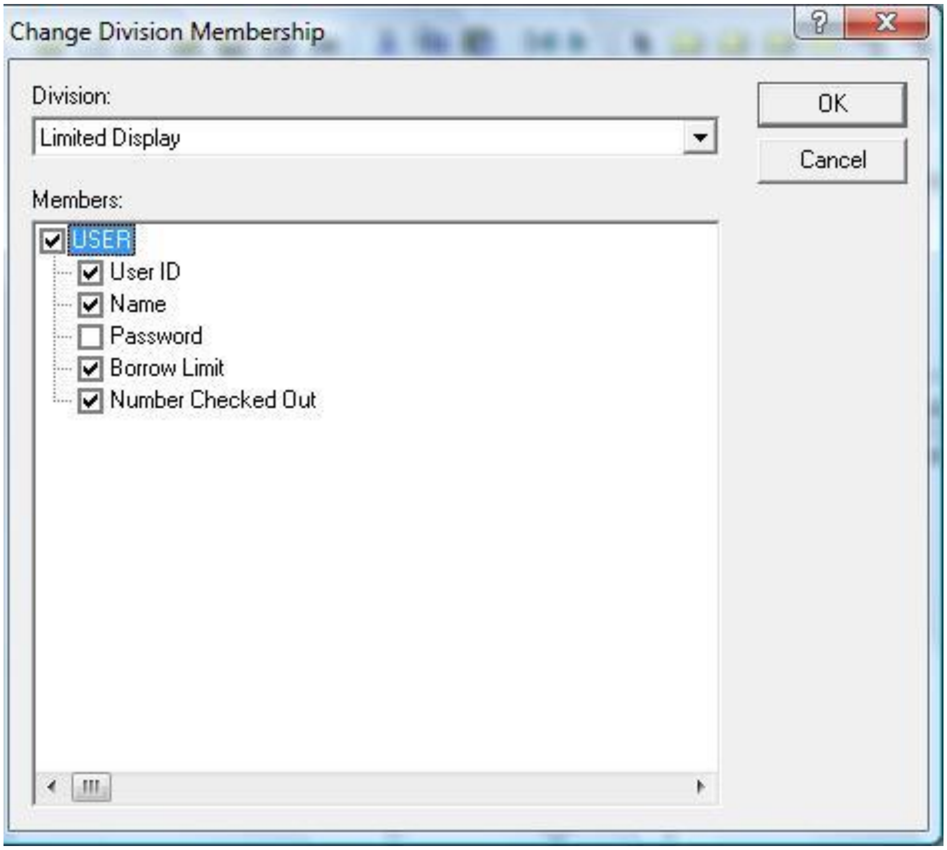
Revoke Division Rights: 4 Revoke all tights to the new division for other users assigned access to the division except “View Diagrams”. Revoke all rights for these users to the original Division. See Assigning Users to Divisions earlier in this chapter for details.

When the user opens a subject area model or prints the model, the hidden objects will not be displayed or printed, even though they are still in the project and have links back to the enterprise project. To add objects to the division and be displayed on the diagram, follow this procedure.

Assign Division Rights: 1 While logged into the project with all rights to the new division, open the subject area, right mouse click on the object and select Divisions to display the Change Division Membership dialog.

Select Display Options: 2 In the Change Division Membership dialog select the new division and check the Members area to add the object to the division and display the object on the diagram. For objects that can display attributes such as entities or classes, check the individual attributes to be displayed.

When users assigned only “View Diagrams” access the project and open the diagram, only those objects added to the new division will be displayed. If an entity is selected but noe attributes of the entity are selected, the entity will be shown at the entity display level. Other entities would be displayed at the Attribute display level if this display level is selected.



Change Division Membership

Division: Limited Display

Members:

- ☑ USER
  - ☑ User ID
  - ☑ Name
  - ☐ Password
  - ☑ Borrow Limit
  - ☑ Number Checked Out

Figure 8-6 Change Division Membership

Chapter 9

Zachman Framework

INTRODUCTION

It has been Visible Systems Corporation's experience that no matter where you start in your application development activities, you will soon find yourself making certain "assumptions" about things not under your control or outside of your scope. To confirm or validate these assumptions, you find yourself addressing the artifacts up and down the Zachman Framework rows and/or across the columns to capture the true drivers for the system: who? what? where? when? why? and how?[1] This means coordinating with the affected or interested business experts, system users, and management.

In 1987 John Zachman wrote, "To keep the business from *dis*integrating, the concept of an information systems architecture is becoming less of an option and more of a necessity.[2]" From that assertion over a decade ago, the Zachman Framework for Enterprise Architecture has evolved and become the model around which major organizations view and communicate their enterprise information infrastructure. The Zachman Framework draws upon the discipline of classical architecture to establish a common vocabulary and set perspectives--a framework--for defining and describing today's complex enterprise systems. Enterprise Architecture provides the blueprint--or architecture--for the organization's information infrastructure and provides a framework for managing information complexity and managing change.

Today the Zachman Framework has become a standard for Enterprise Architecture used by many of the most successful organizations in the world. Evidence of the acceptance of the Framework has been apparent at the annual forums conducted by the Zachman Institute for Framework Advancement (ZIFA, www.zifa.com). At each forum, attendees hear presentations on the many different aspects and practical uses of the Framework. *Visible* fully supports both the concept and philosophy of the Zachman Framework. *Visible* helps clients gain greater control of their information systems and technology requirements through development of an enterprise-wide architecture.

[1] "Visible and the Zachman Framework for Enterprise Architecture" by Alan Perkins p. 2. Copyright © 1997-2001, Visible Systems Corporation.

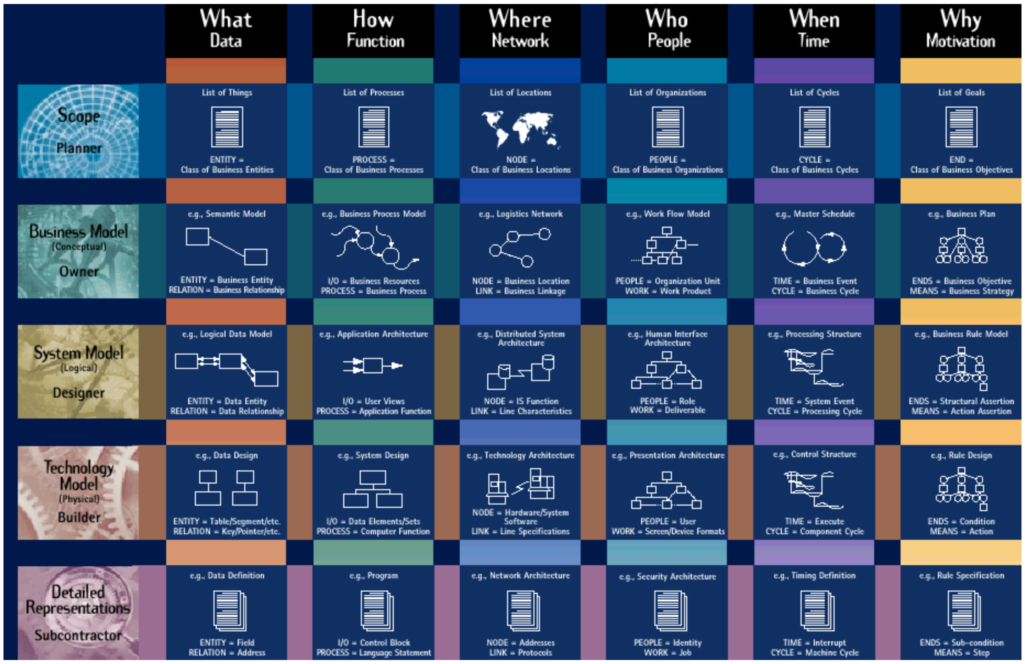
[2] "A framework for information system architecture" by J.A. Zachman p. 454 IBM Systems Journal, Vol. 26, Nos. 3, 1987, ©1987, 1999 IBM.

Visible takes an engineering approach to developing an enterprise architecture. We use a combination of forward and reverse engineering to establish the enterprise architecture. Forward engineering tasks include business planning and data and process modeling. Reverse engineering tasks include analysis and documentation of all existing structures for the organization. The result is a model that represents an integrated view of the enterprise architecture framework, with redundancies and discrepancies resolved and documented. All conceptual and logical architecture components can all be maintained in the *Visible Analyst®.*

The Visible Analyst supports the tasks and techniques involved in the creation and management of an enterprise architecture, with sufficient flexibility to integrate and support other approaches to software engineering. Visible Analyst captures business plans of multiple organization levels and maintains the hierarchy of planning components (mission, goals, strategies, measures, business rules, etc.).

Unlike many other modeling tools, Visible Analyst has the capability of directly linking each business plan component to the entities and attributes of a data model that support/implement the planning elements. This feature is used to control quality and completeness, and to ensure that process and system designs meet business requirements. Visible Analyst can also be used to specify physical information system designs based on the data model or import physical designs of existing data structures into the repository, and then link them back to the business plan component.

It is important to remember that the example Visible Analyst Enterprise Project using the Zachman Framework explained in this chapter is not a static one-time snapshot view of the enterprise. As mentioned in the cell explanations, the artifacts such as the business plan, physical data model, security architecture, strategic goals, etc. will change as the enterprise changes. Using the Visible Analyst and its repository to model the enterprise provides a one-stop location where all information about the enterprise is located. External documents may be changed, but the hyperlinks to the artifacts are maintained within the enterprise project, allowing for both a birds eye and physical implementation perspective of the enterprise.



	What Data	How Function	Where Network	Who People	When Time	Why Motivation
Scope Planner	List of Things ENTITY = Class of Business Entities	List of Processes PROCESS = Class of Business Processes	List of Locations NODE = Class of Business Locations	List of Organizations PEOPLE = Class of Business Organizations	List of Cycles CYCLE = Class of Business Cycles	List of Goals END = Class of Business Objectives
Business Model (Conceptual) Owner	e.g., Semantic Model ENTITY = Business Entity RELATION = Business Relationship	e.g., Business Process Model I/O = Business Resources PROCESS = Business Process	e.g., Logistics Network NODE = Business Location LINK = Business Linkage	e.g., Work Flow Model PEOPLE = Organization Unit WORK = Work Product	e.g., Master Schedule TIME = Business Event CYCLE = Business Cycle	e.g., Business Plan ENDS = Business Objective MEANS = Business Strategy
System Model (Logical) Designer	e.g., Logical Data Model ENTITY = Data Entity RELATION = Data Relationship	e.g., Application Architecture I/O = User Views PROCESS = Application Function	e.g., Distributed System Architecture NODE = IS Function LINK = Line Characteristics	e.g., Human Interface Architecture PEOPLE = Role WORK = Deliverable	e.g., Processing Structure TIME = System Event CYCLE = Processing Cycle	e.g., Business Rule Model ENDS = Structural Assertion MEANS = Action Assertion
Technology Model (Physical) Builder	e.g., Data Design ENTITY = Table/Segment/etc. RELATION = Key/Pointer/etc.	e.g., System Design I/O = Data Elements/Sets PROCESS = Computer Function	e.g., Technology Architecture NODE = Hardware/System Software LINK = Line Specifications	e.g., Presentation Architecture PEOPLE = User WORK = Screen/Device Formats	e.g., Control Structure TIME = Execute CYCLE = Component Cycle	e.g., Rule Design ENDS = Condition MEANS = Action
Detailed Representations Subcontractor	e.g., Data Definition ENTITY = Field RELATION = Address	e.g., Program I/O = Control Block PROCESS = Language Statement	e.g., Network Architecture NODE = Addresses LINK = Protocols	e.g., Security Architecture PEOPLE = Identity WORK = Job	e.g., Timing Definition TIME = Interrupt CYCLE = Machine Cycle	e.g., Rule Specification ENDS = Sub-condition MEANS = Step

Figure 9-1 Zachman Framework

Image provided courtesy of the Intervista Institute, Copyright © Intervista Institute (www.intervist-institute.com)

THE ZACHMAN FRAMEWORK CELL DEFINITIONS AND EXAMPLE ARTIFACTS

When implementing an Enterprise Architecture Framework, it is important where you begin. In our white paper, “Enterprise Architecture Engineering”[3] by Alan Perkins and Clive Finkelstein, available on our web site at www.visible.com, they state, “A well documented Enterprise Architecture is a logical organization of information pertaining to the following multi-level, multi-dimensional, enterprise wide elements.

- Strategic goals, objectives, strategies
- Business rules and measures
- Information requirements

[3] “Enterprise Architecture Engineering” by Alan Perkins and Clive Finkelstein p.3. Copyright © 2000, Visible Systems Corporation.

- Processes, systems and applications
- Relationships between architecture elements
- Technology infrastructure”

They emphasize that *the most important starting point* is that establishing the right sponsorship helps to insure successful development and deployment. Alan also explains, “…all potential users of the applications and systems based upon the architecture must be involved in the process. Without both management sponsorship and near universal involvement, enterprise-wide architecture engineering projects usually fail.”[4] Additional white papers are available on our web site at www.visible.com that help explain the “Critical Success Factors for Enterprise Architecture Engineering”, “Business Rules ARE Meta-Data”, etc.

Each cell of the framework is described using the following format beginning with the “What column Planner perspective” and proceeding down the column in a top-to-bottom left-to-right order.

- Cell location, label, perspective and descriptive type
- An explanation of the cell definition
- The artifact created in the project to implement the cell. The name and location of the cell is included in the artifact label where appropriate, such as "Row 4 Column 1 Physical Data Model".
- A detailed explanation of the project artifact
- Alternative artifacts that could be created in the project to implement the cell

The Zachman project explanation contains many different types of artifacts, consisting of diagrams, strategic planning statements, lists, user defined objects, etc., and each was created to document a specific cell of the framework. Only one example artifact was created for each cell, while additional artifact option types are included in each cell explanation when appropriate. An actual enterprise project will have multiple artifacts representing a particular cell. When using the Visible Analyst in a real-world implementation of the framework, users should consider using the Enterprise Modeling feature to eliminate any naming conflicts, to maintain logical and physical data model separation, program specification, etc. The Enterprise Modeling feature will maintain the linkage between the artifacts in the projects, promoting object re-use.

Framework Rules

[4] “Critical Success Factors for Enterprise Architecture Engineering” by Alan Perkins p. 4. Copyright © 2000 Visible Systems Corporation. http://www.visible.com/AboutUs/whitepapers.html.

Before beginning an enterprise project and creating the cell artifacts, it is important that users know and understand the rules of the framework as described by John Zachman and John Sowa.[5]

1. The columns have no order.
2. Each column has a simple, basic model.
3. The basic model of each column must be unique.
4. Each row represents a distinct, unique perspective.
5. Each cell is unique.
6. The composite or integration of all cell models in one row constitutes a complete model from the perspective of that row.
7. The logic is recursive.

The following cell definitions provide examples of the cell artifacts created in the example project as well as an overview of the Visible Analyst's repository and modeling capabilities. Each cell is detailed in a cell-by-cell review including an explanation of the artifacts created for the cell in the Visible Analyst. A backup copy of the Zachman example project is available; please contact Visible Systems support at ***support@visible.com*** for a copy of the project.

Column 1: The "Data" or "What" column

Provides an understanding of the data important to the business with finer amounts of detail shown at each succeeding perspective.

Row 1, Column 1 - "List of Things Important to the Business"

Objectives/Scope (Contextual)
Data column, Planner role
Entity = Class of Business Thing

Cell explanation[6]
A list of items, objects, assets, etc. important to the business and defined at a high level of aggregation. The list is dependent on the enterprise modeled, and "…defines the scope, or boundaries, of Rows 2 – 5 models of things significant to the Enterprise"[7]. A software or

[5] "Extending and formalizing the framework for information systems architecture" by J. A. Sowa and J. A. Zachman, IBM Systems Journal, Vol. 31, No 3, 1992. Pages 599-603

[6] The cell definitions are based on the cell explanations as described in the following documents:
"The Framework for Enterprise Architecture Cell Definitions" ZIFA 03.doc Copyright © Zachman Institute for Framework Advancement www.zifa.com
and "A different Kind of Life Cycle: The Zachman Framework" by David C. Hay, Essential Strategies Copyright © 2000, Essential Strategies, Inc www.essentialstrategies.com

[7] "The Framework for Enterprise Architecture Cell Definitions" ZIFA 03.doc Copyright © Zachman Institute for Framework Advancement www.zifa.com

manufacturing company would include Vendors, Products, Clients, Product Facilities, etc. A law firm would include specific knowledge areas, trial experience, etc., while educators would include a curriculum, educational levels, specific teaching disciplines, etc.

Project implementation
Implemented as an Strategic Planning Statement

Artifact explanation
The strategic planning statement's Detailed Description repository field contains the list of items important to the business. This list includes Employees, Financial Resources, Accounting Procedures, Equipment and Technology, Profits etc. Using the Links tab of the statements repository entry, this statement was linked to the Row 1 Column 1 "List of things important to the business" cell.

Each item in the list could have been be added as a separate sub-statement, and the repository fields of the individual statements populated with the discrete explanation of the items. Hyperlinks to external documents, which further describe this high level aggregation of the business, can be created if necessary.

Alternative project implementations

- A User Defined Object of type "Business Object" can be created and implemented as an item in the list, with each item maintaining separate repository entries that can be linked to other cell artifacts.
- The list can be maintained in any word processing application and a hyperlink to the file created in any one of the cells descriptive-type fields (notes, detailed description, etc.) in the repository.

Row 2, Column 1 - "Semantic Model"

Enterprise Model (Conceptual)
Data column, Owner role
Entity = Business Entity
Relation = Business Relationship

Cell explanation
Contains a model of the things[8] important to the business, as seen by the participants in the business, and is modeled as a high-level entity relationship diagram. These relationships are later implemented as business rules.[9]

[8] "A different Kind of Life Cycle: The Zachman Framework" by David C. Hay, Essential Strategies Copyright © 2000, Essential Strategies, Inc. www.essentialstrategies.com

Project implementation:
Implemented as an Entity Relationship diagram.

Artifact explanation:
This conceptual data model diagram contains a model of the high-level business objects and the relationships maintained between the objects. Entities include Company Employees, Company Management, Company Business Relationships, Products, etc. The relationships model the business concepts between the entities, such as Employees - *design* - Products; Employees – *produce* – Products; Company Management – *acquires* – Capital Resources, etc.

Alternative project artifact implementations:
A class diagram could be used to model this cell with the classes identifying the business objects and the relationships between these objects defining the business concepts.

Row 3, Column 1 - "Logical Data Model"

System Model (Logical)
Data column, Designer role
Entity = Data Entity
Relation = Data Relationship

Cell explanation
The Technology neutral fully normalized logical data model with attributes and unique identifiers defined to record information important to the business.

Project implementation
Implemented as an Entity Relationship diagram

Artifact explanation
The entities involved in the logical data model can be modeled on one global diagram, and then separate subset entity relationship diagrams created if desirable. Note that key relationships between entities in the Visible Analyst extend across the diagrams for purposes of model analysis to provide additional analysis / and verification of the models.

The subset area diagrams can be copied to satellite projects using the Enterprise Copy feature and implemented as physical data models while maintaining the relationships to the logical data model.

[9] "The Framework for Enterprise Architecture Cell Definitions" ZIFA 03.doc Copyright © Zachman Institute for Framework Advancement www.zifa.com

Note

- A physical data model can be reverse engineered from any ODBC compliant RDBMS, and the physical model used as the basis for creating a new logical data model.

Alternative project artifact implementation

A class diagram can also be used to model this cell.

Row 4, Column 1 - "Physical Data Model"

Technology Model (Physical)
Data column, Builder role
Entity = Segment/Table
Relation = Pointer/Key

Cell explanation

The entities in the subject areas are converted into table definitions of a technology constrained fully attributed entity relationship model. All keys, indexes, table and column check constraints, database storage information, stored procedures, etc., are defined for implementation into a specific RDBMS.

Project implementation

Implemented as an Entity Relationship diagram

Artifact explanation

The fully attributed entities and relationships are added to entity relationship diagram(s) with corresponding Visible Analyst repository entries. All physical information about the entities and elements is defined, including primary, foreign and alternate keys; unique and non-unique indexes; table and column check constraints; database storage information; stored procedures; triggers; etc. Each diagram can be modeled to correspond to specific business subject areas, such as Accounting, Shipping, Sales, etc. and these individual diagrams used as the basis of the generated SQL DDL.

Note

- The RDBMS tables, attributes, keys, index, trigger, stored procedure, tablespace information, etc., can be reverse engineered from the existing RDBMS and used to populate a Visible Analyst project. Diagrams can automatically be generated to view the imported tables and relationships, and data element definitions. Foreign keys can be inferred during the reverse engineering procedure to auto generate relationships if none are defined.

Alternative project artifact implementation

A class diagram can be used to model the physical information, and once the classes are copied to an entity diagram, SQL DDL can be generated.

Row 5, Column 1 - "Data Definition"

Detailed Representations (Out-of-Context)
Data column, Sub-Contractor role
Entity = Field
Relation = Address

Cell explanation
The artifact is the implementation and data definition of the tables and column in the specific RDBMS, as well as the SQL DDL script.

Project implementation
A User Defined Object of type "Database" was created and implemented as the repository object "SQL Server Database".

Artifact explanation
This "SQL Server Database" user-defined object functions in 2 ways:

1. As a container object to list all of the entities associated with a business area implemented in a specific database(s). Entities can be listed in many of these user defined 'database' objects.
2. As the Visible Analyst repository entry linked to the implemented code, which can be stored in a source control application or stored in an external file. When a source code control application is used, the objects Links To field on the Links tab lists the connection to the source code control application. Otherwise, a hyperlink is created to connect the object to the external file containing the SQL DDL script.

The generated SQL DDL code could be pasted into the objects Notes field or a user-defined attribute could be created to store the SQL DDL code as part of this object's repository entry.

Alternative project artifact implementation(s)
A pre-defined Visible Analyst "Cluster" repository object is used to contain a group of entities that can be displayed as one symbol on a diagram. Its purpose is to reduce the amount of displayed detail. The cluster object would be used to define the entities implemented in a specific database based on a specific diagram. The External Link to the source code control application would be entered in the "Links To" field on the cluster's Links tab. Note that entities can only exist within one cluster.

Column 2: The "Function" or "How" column

Describes the process and functions performed by the business. Additional detail is displayed for each succeeding perspective.

Row 1, Column 2 - "List of Processes the Business Performs"

Objectives/Scope (Contextual)
How column, Planner role
Function = Class of Business Processes

Cell explanation
This cell lists the processes /activities the business performs.

Project implementation
Implemented as a Functional Decomposition diagram.

Artifact explanation
The Functional Decomposition Diagram was chosen because the symbols allow the user to display the high-level business functions and processes in a hierarchical relationship. Each methodology symbol maintains a separate repository entry allowing the user to fully describe the function/process, and include hyperlinks to external documents if necessary. Through the use of off-page connectors, each function and its sub-processes can be modeled on separate multi page diagrams and copied to a satellite project if necessary for further decomposition. The Functional Decomposition Diagrams also can be used to spawn a high-level data flow diagram that segues into the next cell in the column, Business Process Modeling.

Alternative project artifact implementation(s)

- The Strategic Planning Statements can be used to define the business functions and high-level child processes in the statement hierarchy.
- A hyperlink from this cell to an external document listing the functions and processes can be used to link the cell to the document.

Row 2, Column 2 - "Business Process Model"

Enterprise Model (Conceptual)
How column, Owner role
Process = Business Process
I/O = Business Resource

Cell explanation

The activities of the business function and processes are described independent of system implementation. The inputs and outputs describe the business resources.

Project implementation
Implemented as a Business Process Model diagram.

Artifact explanation
The Business Process Model diagram, using the BPMN notation developed by the Business Process Management Initiative (www.bpmi.org) and the Object Management Group (OMG www.omg.org) is specifically suited for modeling the business processes. These diagram models communicate the business processes including the business processes, the events (triggers) that begin, end or interrupt the processes, and the information (artifacts) used and developed by the processes. The BPMN notation supports Private (internal), Abstract (public) and Collaboration (global processes).

The repository entries for the diagram symbols capture the processes, events and data details in excruciating detail. The model analysis confirms the integration of the lower level processes with the associated model items.

Alternative project artifact implementation(s)
- The data flow diagram can also be used to model the business processes.
- The functional decomposition diagram can be used to model the business processes
- The activity diagram can be used to model the business processes.

Row 3, Column 2 - “Application Architecture”

System Model (Logical)
How column, Designer role
Process = Application Function
I/O = User Views

Cell explanation
An information perspective of the business processes explaining the controls and mechanisms and conversion of input data to output data.

Project implementation
Implemented as a Business Process Model diagram.

Artifact explanation
The Business Process Model diagram type is used to model the control mechanisms of the processes to include the various events and gateways that define the function of the Application Architecture.

Alternative project artifact implementation(s)

- The data flow diagram can be used as the artifact to define this cell.
- An activity diagram can be used to show the high-level inputs, processes and synchronization of the application architecture.
- A class diagram could also be used to define the business users, the methods of the business, and the relationships between the business users.
- A Use Case diagram can also be used and then "nested" to an Activity diagram, where the inputs and outputs can be shown interacting with the business processes.

Row 4, Column 2 - "System Design"

Technology Constrained Model (Physical)
How column, Builder role
Process = Computer Function
I/O = Data Element Sets

Cell explanation

This system design is converted into to the module definitions or class methods. A high level of abstraction is necessary to model this cell.

Project implementation

Implemented as an Activity diagram.

Artifact explanation

The Business Process Model diagram was chosen because of its capabilities to display not only the events and states of the system design but the concurrency of actions to be completed before processing can continue.

Alternative project artifact implementation(s)

- Structure Chart diagrams could be used to model the programs architecture, i.e. calling structure and information passed from module to module.
- A data flow diagram can be used as an alternative diagram artifact
- An activity diagram can also be used because of its capabilities to include the events and concurrency of the processes.
- A Sequence diagram could also be used to define the calling structure and methods.
- A class diagram can also be used.

Row 5, Column 2 - "Program"

Detailed Representations (Out-of-Context)
How column, Sub-Contractor role

Process = Language Statement
I/O = Control Block

Cell explanation
The programs designed in the above columns are converted / compiled into the actual running programs

Project implementation
The Visible Analyst repository has a predefined repository object labeled as a "Program", which is used to link to the program code stored in a source code control application.

Artifact explanation
This "program" object can be linked to the external code maintained in a source code control application such as RAZOR or Visual Source Safe using the Links To field on the programs Links tab. All methods associated with classes are stored in the Visible Analyst repository with an entry type of "module". These modules can be added to the composition field of the program object, detailing which modules are used in the program. Additionally, structure chart diagrams or sequence diagrams can be used to model the modules and calling structure of the program.

Inclusion of a hyperlink to sections of the code such as header files or code files written in C, C++, C#, VB files, .Net .sln files, etc. can also be created.

Alternative project artifact implementation(s)

- Creation of a user defined object similar to the program repository object mentioned above to maintain a link to the source code control application storing the generated program code.
- Link to Visible Developer, which creates the 3-tier business object program as ASP, VB6 or .Net code.
- A structure chart diagram can also be used to model the program and be the sequence object linked to the code.

Column 3: The “Network” or “Where” column

Describes the geographical distribution of the enterprise’s activities.

Row 1, Column 3 - “List of Locations in Which the Business Operates”

Objectives/Scope (Contextual)
Where column, Planner role

Node = Major Business Location

Cell explanation
A list of locations where the business operates.

Project implementation
Implemented as a Functional Decomposition diagram.

Artifact explanation
The functional decomposition diagram was chosen to create a hierarchy of the business architecture with the corresponding repository entries providing fields to maintain a detailed description of the location. Hyperlinks to external items for each location can also be included as part of the locations repository entry, to record contract information, rules and regulations specific to the location, etc.

Alternative project artifact implementation(s)

- Strategic planning statements could be used to describe each location, with subsidiary locations defined as sub-statements.
- A 'locations' user defined object could be created in the repository, and hyperlinks created to reference the external contracts, rules and regulations, etc, as noted above

Row 2 Column 3 "Logistics Network"

Enterprise Model (Conceptual)
Where column, Owner role
Node = Business Location
Link = Business Linkage

Cell explanation
The detailed communications chart, listing the communications network and the protocols used, such as voice, data, post, rail, shipping, etc. and how the locations interact.

Project implementation
Implemented as a Structure Chart diagram.

Artifact explanation
The structure chart diagram type was selected so that the nodes could be modeled as modules and the links signifying the individual communications between the modules defined as data couples. These couples as well as the modules maintain repository entries allowing for a detailed description of the communications nodes and links. Hyperlinks to external information can also be included in the repository definitions. Additional details of the diagram symbols and the objects they represent can be defined in the repository using user-defined attributes as necessary.

Each location can be modeled independently but connected to the main location diagram via on-page or off-page connections.

Alternative project artifact implementation(s)

- A planning statement or user-defined object can be created to reference this cell, and a hyperlink to an external application supporting a network diagram can be created.
- Hyperlinks to other documents or artifacts associated with this cell but modeled externally can be created.

Row 3, Column 3 - "Distributed System Architecture"

System Model (Logical)
Where column, Designer role
Node = I/S Function (Processor, Storage, etc.)
Link = Line Characteristics

Cell explanation:
Architecture of the data distribution, where it is created, and where used. Technology neutral, it would contain the descriptions of the system facilities, "…controlling software at the nodes and lines (processors/operating systems, storage devices/DBMS', peripherals/drivers, lines/line operation systems, etc)".[10]

Project implementation:
Implemented as a Structure Chart diagram.

Artifact explanation
The structure chart diagram was used so that the data couples signifying the Links show the transfer of the information between the module symbols as Nodes on the diagram. Additional details of the diagram symbols and the objects they represent can be defined in the repository using user-defined attributes.

Alternative project artifact implementation(s)

- A planning statement or user-defined object can be created to reference this cell, and a hyperlink to an external application supporting a system architecture diagram can be created.

[10] The cell definitions are based on the cell explanations as described in the following documents:
"The Framework for Enterprise Architecture Cell Definitions" ZIFA 03.doc Copyright © Zachman Institute for Framework Advancement www.zifa.com
and
"A different Kind of Life Cycle: The Zachman Framework" by David C. Hay, Essential Strategies Copyright © 2000, Essential Strategies, Inc www.essentialstrategies.com

- Hyperlinks to other documents or artifacts associated with this cell but modeled externally can be created.

Row 4, Column 3 - “Technology Architecture”

Technology Constrained Model (Physical)
Where column, Builder role
Node = Hardware/System Software
Link = Line Specification

Cell explanation
Shows the physical design of the computer facilities including the details of the hardware and software used at the business locations.

Project implementation
Implemented as a Structure Chart diagram.

Artifact explanation
The structure chart diagram was used so that the data couples signifying the Links show the transfer of the information between the module symbols as Nodes on the diagram. Additional details of the diagram symbols and the objects they represent can be defined in the repository using user-defined attributes.

Alternative project artifact implementation(s)

- A planning statement or user-defined object can be created to reference this cell, and a hyperlink to an external application supporting a technology architecture diagram can be created.
- Hyperlinks from the cell to other documents or artifacts associated with this cell but modeled externally can be created.

Row 5 Column 3 “Network Architecture”

Detailed Representations (Out-of-Context)
Where column, Sub-Contractor role
Node= Address
Link = Protocol

Cell explanation
The definitions of the node address and line specification, which are translated into specifications of particular protocols, communication facilities, etc., [8]are defined in this cell.

[8] "A different Kind of Life Cycle: The Zachman Framework" by David C. Hay, Essential Strategies Copyright © 2000, Essential Strategies, Inc. www.essentialstrategies.com

Project implementation
Implemented as the User-Defined Object of type" Architecture" and the repository entry "Row 5 Column 3 Network Architecture Implementation".

Artifact explanation
This user defined objects' text fields are used to maintain the network architecture information. It can be hyperlinked to an external application that models network architecture, or hyperlinked to external documents describing the architecture.

Alternative project artifact implementation
A planning statement could be used as a "container" object to maintain information about the network implementation.

Column 4: The "People" or "Who" column

Those involved in the business and their relationship to the technology.

Row 1, Column 4 - "List of Organizations Important to the Business"

Objectives/Scope (Contextual)
Who column, Planner role
People = Class of Agent

Cell explanation
A list of people and organizations important to the business, including organizational units and their scope and boundaries is the artifact created for this cell.

Project implementation
Implemented using a Strategic Planning Hierarchy statement.

Artifact explanation
Only one planning statement was used to identify the people and organizations important to the business. Practically, each person, organization and organizational unit should be entered as sub-statements in the statement hierarchy to maintain individual repository entries. This procedure facilitates the definition of the person / unit especially when hyperlinks are created to external documents describing the relationship. Contact information with vendors; venture capital contracts; rental agreements; technology contracts; shipping agreements are some simple examples of the additional documentation associated with the people and organizations important to the business.

Note

- Not all users should be granted access to the sensitive business documents. In some cases a listing of the documents may be sufficient to define the artifact rather than a hyperlink to the actual documents themselves.

Alternative project artifact implementation(s)

- A functional decomposition diagram could also be used to identify the business units and the individuals, organizations and organizational units in a hierarchical diagram.
- A user-defined object could also be used to identify the people and organizations important to the business.

Row 2, Column 4 - "The Work Flow Model"

Enterprise Model (Conceptual)
Who column, Owner role
People = Organizational Unit
Work = Work Product

Cell explanation

Allocation of responsibilities as described in an organizational chart with secondary documents defining the products. Security requirements are also included within this cell.

Project implementation

Implemented as a Business Process Model diagram.

Artifact explanation

The Business Process Model diagram type was chosen because it models concurrent actions to be completed before the next action begins along with the inputs and outputs. The model also includes the use of Swimlanes (Pools) to categorize the activities performed by the respective roles of the business users

Alternative project artifact implementation(s)

- Data flow diagrams can be used to model the organizations, organizational units and processes performed.
- An Activity diagram can also be used to model the workflow.
- A functional decomposition diagram can be used to model the organizations chart.
- A Use Case could also be used, with links to a nested activity or collaboration diagram modeling the work products.

Row 3, Column 4 - "Human Interface Architecture"

System Model (Logical)
Who column, Designer role

People = Role
Work = Deliverable

Cell explanation
Defines the people, their roles and responsibilities and interacting with the technology to create the deliverables.

Project implementation
Implemented as a Use Case diagram.

Artifact explanation
The Use Case diagram captures the interaction of the people and the work deliverables. Nested links to an activity diagram including the use of user-defined attributes and the use of hyperlinks to the deliverables can be modeled to show additional detail.

Alternative project artifact implementation(s)

- Data flow diagrams can be used to model the processes performed by the organizations interacting with the technology and resulting deliverables.
- A functional decomposition diagram can be used to model the interactions and the deliverables.

Row 4, Column 4 - “Presentation Architecture”

Technology Constrained Model (Physical)
Who column, Builder role
People = User
Work = Screen Format

Cell explanation
The actual interface is modeled with presentation formats including screens, navigation paths, security rules, etc.

Project implementation
This cell was implemented as a Use Case diagram.

Artifact explanation
The Use Case diagram can also be nested to an activity diagram. Each of the repository entries can be tied to the implementation code, such as the screen design as shown in the user interface code generated by Visible Developer. The security considerations can be modeled as user-defined attributes, separate user defined objects, or as planning statements and each of these repository objects linked to the appropriate Use Case symbol artifact. Hyperlinks to some external tools can also be created as necessary.

Alternative project artifact implementation(s)

- A database view object can be used to list the data elements used in the menu screens, and the Extended Attributes tab of the elements repository definition used to store the presentation information.
- A hyperlink from this cell can be used to a human interface architecture diagram or the screen configuration files developed in an external application.

Row 5, Column 4 - "Security Architecture"

Detailed Representations (Out-of-Context)
Who column, Sub-Contractor role
People = Identity
Work = Job

Cell explanation
Individual's program access permissions and work they are authorized to perform.

Project implementation
Implemented as a Class diagram.

Artifact explanation
Implemented as a class diagram with the class representing the users, programs, and the elements defining the class data such as permissions, security mechanisms, etc. Methods can also be defined for classes as an additional level of detail. Hyperlinks to the code stored in a configuration management application can also be created.

Alternative project artifact implementation
An entity diagram can be used with a user-defined attribute or user-defined object substituting for the method's definition.

Column 5: The "Time" or "When" column

Used to describe the effects of time on the business, and interacts with column 2, the How column.

Row 1, Column 5 - "List of Events Significant to the Business"

Objectives/Scope (Contextual)
When column, Planner role
Time = Major Business Event

Cell explanation:

A description of the business cycle and when events significant to the business occur.

Project implementation:
Implemented as a Planning Statement.

Artifact explanation:
The events are modeled as subset planning statements allowing for further definition and linkage to other artifact items listed in subsequent How columns.

Alternative project artifact implementation(s):

- A used-defined object could contain this list.
- Hyperlinks from the cell's definition to external documents describing the event.

Row 2, Column 5 - "Master Schedule"

Enterprise Model (Conceptual)
When column, Owner role
Time = Business Event
Cycle = Business Cycle

Cell explanation
When the business functions occur, including the initiating event and the processing order.

Project implementation
Implemented as a Business Process Model diagram.

Artifact explanation
The Business Process Model diagram models the business events, processes and when functions are to happen and under what circumstances.

Alternative project artifact implementation(s)

- A state transition diagram can be used to model this cell.
- An Entity Life History diagram can be used to model this cell.
- An activity diagram can be used to model this cell.
- A list of events and time lines can be defined as a user defined object or as an external documents hyperlinked to the cell.

Row 3, Column 5 - "Processing Structure"

System Model (Logical)

When column, Designer role
Time = System Event
Cycle = Processing Cycle

Cell explanation
Model of the system events and times to complete the data transformation processes and entity state changes.

Project implementation
Implemented as a Business Process Model diagram.

Artifact explanation
The Business Process Model diagram is used to show the business system processes and the events causing the change in state. Detailed information is documented in the appropriate repository fields with additional user defined attributes added as necessary.

Alternative project artifact implementation(s)

- A data flow diagram can be used to model this cell.
- A state transition diagram can be used to model this cell.
- The activity diagram can be used to model this cell.
- The collaboration diagram can be used to model this cell.
- The sequence diagram can be used to model this cell.

Row 4, Column 5 - "Control Structure"

Technology Constrained Model (Physical)
When Column, Builder role
Time = Execute
Cycle = Component Cycle

Cell explanation
Triggers, messages, responses etc, described as system events with physical properties and processing cycles detailed.

Project implementation:
Implemented as a Sequence Diagram.

Artifact explanation:
The sequence diagram models the calling structure of the programs and the returns, etc. The details are stored in the appropriate repository fields with additional user defined attributes added as necessary.

Alternative project artifact implementation(s)

- The state transition diagram can be used to model this cell.
- A business process model can be used to model this cell.
- The structure chart diagram can be used to model this cell.
- The collaboration diagram can be used to model this cell.

Row 5, Column 5 - “Timing Definition”

Detailed Representations (Out-of-Context)
When Column, Sub-Contractor role
Time = Interrupt
Cycle = Machine Cycle

Cell explanation
Schedule Online and Batch applications (Function Details), showing the interrupts and machine cycles.

Project implementation
Implemented as a Collaboration diagram.

Artifact explanation
The collaboration diagram shows the timing of the application through the use of the messages that implement the business scenario.

Alternative project artifact implementation
A sequence diagram can also be used to model this cell.

Column 6: The “Motivation” or “Why” column

Translation of the business goals and strategies into the ends and means of the business.

Row 1, Column 6 - “List of Business Goals/Strategies”

Objectives/Scope (Contextual)
Why column, Planner role
Ends/Means = Major Business Goal / Critical Success Factor

Cell explanation
The goals and strategies of the business are identified.

Project implementation

Implemented as a Strategic Planning Statement.

Artifact explanation

The strategic planning statement hierarchy is specifically suited to create the artifacts necessary for this cell. The users can extend the statement types, and an editable priority field is available for assignment to each statement in addition to the predefined repository fields. Links to the other artifacts can be defined in the Links To field on the Links tab in the repository. Hyperlinks to external documents can also be created when necessary.

Alternative project artifact implementation(s)

- The functional decomposition diagram can be used to define the hierarchy.
- Hyperlinks to an external document or another statement hierarchy application can be used.

Row 2, Column 6 - "Business Plan"

Enterprise Model (Conceptual)
Why column, Owner role
End = Business Objective
Means = Business Strategy

Cell explanation

The business plan contains the strategies, goals, financial considerations and motivation of the company. These artifacts can include both textual descriptions as well as financial documents.

Project implementation

Implemented as a Planning Statement hierarchy.

Artifact explanation

Individual strategic planning statements should be defined and can include references to external documents and artifacts hyperlinked to the statement. The "Cost Structure", "Capital Funding" statements might reference MS Excel spreadsheets, while the textual description of the "Business Plan" statement contains a hyperlink(s) to MS Word document(s).
Note that these statements are linked to the functional decomposition diagram symbols repository entries as an example of the artifact integration available in the Visible Analyst.

Alternative project artifact implementation(s)

- An alternative implementation is the functional decomposition diagram "Row 2 Column 6 Business Plan Hierarchy". The decomposition diagram allows the artifacts to be listed as a hierarchy, and would include hyperlinks to the external documents the symbols represent. While each artifact can be represented within a symbols Notes field or as a user-defined attribute, maintaining them external to the application allows these artifacts to be updated and maintained in one place while still linking to the symbol in the enterprise project. Note that the decomposition diagram symbols are linked to the individual planning statements to demonstrate the cross artifact reference capability in the Visible Analyst.

- A class diagram could be used to diagram the business plan and the methods used to detail the business constraints.

Row 3, Column 6 - “Business Rule Model”

System Model (Logical)
Why column, Designer role
End = Structural Assertion
Means = Action Assertion

Cell explanation
Business rules can be considered as the meta-data of the enterprise to include the intents and means of the business, and are part of the information implemented as checks on the database and enterprise information. Examples of these business rules as meta-data include Definitions of Business Terms; Data integrity constraints; Mathematical and functional derivations; Logical inferences; Processing sequences; Relationships among facts about the business, etc.

Project implementation
Implemented as a Planning Statement hierarchy.

Artifact explanation
The strategic planning hierarchy provides the structured hierarchy allowing the meta-data to be defined, and most importantly, to be linked to the implementation of these rules to the data. The tables, columns, check constraints, business processes, database access security rules, etc. that implement the business rules are linked to the business rule statement.

Alternative project artifact implementation
Create a hyperlink from this cell to a User Defined Object created to store these business rules.

Row 4, Column 6 - “Rule Design”

Technology Constrained Model (Physical)
Why column, Builder role
End = Condition
Means = Action

Cell explanation
This cell describes the physical specifications of the Business Rules.

Project implementation
Implemented as a User Defined Object of type “Rule Design” labeled “Row 4 Column 6 Rule Design”, linked to the repository entries that implement the rule design, such as the relationship cardinality between entities / classes, column or table check constraints, a business process that enforces these rules, etc.

Artifact explanation
Since this cell describes the physical specifications of the Business Rules, implementation of these rules can also be enforced as part of the relationship cardinality or as table or column check constraints on the Class | Entity in the repository. (Remember, entities can be used on class diagrams and methods, keys, constraints, etc., can be defined and the class/entity used for SQL DDL and code generation). Additional user-defined attributes can be added to the project as necessary to store specific textual descriptions of the specifications.

Note

- There is some agreement in the enterprise community that the UML OCL Language be used to represent the artifacts of this cell. Enter the following links into a web browser for an explanation of the OCL language. The artifacts referenced by the OCL can natively be created in the Visible Analyst using the supported diagram types or identified as a user defined attribute(s). See http://www-3.ibm.com/software/awdtools/library/standards/ocl.html and http://www.klasse.nl/ocl for details.

Alternative project artifact implementation(s)

- Program rule design can also be detailed in the methods associated with a class, either defined on the Class or Sequence diagram.
- The cell’s repository entry may also contain hotlinks to appropriate external applications.

Row 5, Column 6 - “Rule Specification”

Detailed Representations (Out-of-Context)
Why column, Sub-Contractor role
End = Sub-condition
Means = Step

Cell explanation
Enforcement of the business rules in the programs are the artifacts.

Project implementation
Implemented as Application modules and Database tables, Data and Function Details. Hyperlinks from these repository entries to the implementation artifacts (programs) from the cell can be created.

Artifact explanation
The Rule Design artifacts as defined in the previous cell in this column are implemented in the code and applications as part of the Application modules, Database tables, Data and Function Details, etc. These programs or code should be hyperlinked to the Rule Design artifacts.

Alternative project artifact implementation
Application modules could be detailed as sequence or structure chart diagrams, but it is more appropriate to link to the code implementing the rules.

Appendix A

Error Messages

INTRODUCTION

This appendix lists all Visible Analyst error messages and their explanations. The error messages are listed in alphabetical order.

Note

- These messages are listed in absolute alphabetical order. This means that if a message begins with the word “The,” it is alphabetized under “The” not under the first important word.
- Some error messages contain the names of individual objects. In this generic listing, these individual names have been replaced with “X”. Further, such a message is alphabetized using the “X”.

A change is pending for this diagram - Cannot edit.

Another user has performed a global change that could affect this diagram. You cannot access the diagram until the other user saves his or her work or aborts the edit.

A class must be labeled before a Nest operation can be performed.

An unnamed class cannot be decomposed.

A couple may only be changed into another couple type.

Once a couple line type is created, the Change Type option limits its transformation to other couple line types. For instance, a generic couple can be changed to a data couple; however, the generic couple cannot be changed to a loop line.

A diagram is being inserted into the tree - This branch cannot be altered.

Another user is inserting a diagram into the tree that could affect the diagram you have selected. The diagram cannot be edited. To determine the user ID of the person inserting the diagram, select the Current Activity option from the File menu and find the users with an activity of "is creating new diagram." The person in question is one of these users. You may contact him/her by using the message facility (press the F5 key).

A gateway cannot be the source or target for a message flow.

A message flow cannot be attached to a gateway.

A message flow must connect objects in different pools.

A Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them. In BPMN, two separate pools on a diagram will represent the two entities. A Message Flow MUSTconnect two pools, either to the pools themselves or to flow objects within the pools. They cannot connect two objects within the same pool.

A join already exists between this pair of entities.

Only one join relationship can exist between a pair of table instances in a view. For non-relationship based joins, you can add restrictions to the join expression either by editing the join expression field or by dragging columns from the parent table to the child table in the view window.

A non-couple cannot be changed to a couple.

The Change Type option in the Change Item dialog box cannot be used to transform an invocation line or loop line type into a couple.

A non-loop may not be changed into a loop.

The Change Type option on the Change Item dialog box cannot be used to transform an invocation line or a couple into a loop line.

A process must be added to this context diagram.

A process must be added and named before the diagram is saved when a context diagram is requested.

A process must be labeled before a nested decomposition can be performed.

An unnamed process cannot be decomposed.

A program can only have one item as the root module.

The composition field of a program indicates the top level module of that program. Only one module may be specified.

A relationship can be labeled only when two NAMED entities are attached.

A relationship only has meaning when attached to two entities. It must join two entities before a relationship can be labeled and added to the repository.

A relationship cannot be connected to a view.

When a view appears on an entity relationship diagram, relationship lines cannot be connected to the view because a view does not contain foreign keys.

A supertype/inheritance relationship already exists between this pair of objects.

Only one supertype or inheritance relationship can be drawn between a pair of entities or classes.

A supertype/inheritance relationship cannot be recursive.

A supertype or inheritance relationship cannot be drawn such that the start and endpoints are the same object.

A supertype relationship cannot be used with a non-fundamental entity.

A supertype relationship cannot be drawn if the child entity is either an associative or attributive entity. These entity types imply special relationships to their parents that are in conflict with supertype relationships. Either draw a different type of relationship or change the entity type to fundamental.

A tree file for project “x” is damaged or missing.

When copying a portion of a project, the tree file could not be found or was damaged. Restore the project.

A user-defined object cannot have the same name as a VAW entry type.

When creating a user-defined object, the name of the object cannot be the same as any of the built-in Visible Analyst types. For example, you cannot create an object named Process.

Access denied - another user is editing this diagram.

A diagram can be edited by only one user at a time. The selected diagram is already in use.

Activity diagram ‘x’ is not associated with a class, use case, or activity.

This warning is the result of not exploding a class, use case or activity to the indicated activity diagram. To resolve the warning, open a class, use case or activity diagram, select the appropriate object, and use the explode feature to attach to the indicated activity diagram.

Actor labeled ‘x’ is dangling.

The above error message is the result of not connecting the indicated actor to another actor or use case on the use case diagram. To resolve the error, draw a line from the actor to any use case or actor using the Connect function.

Aggregation line not added.

A construct that was originally created from a class diagram can be loaded into an entity relationship diagram. However, aggregation relationships are not supported on ERDs, so that relationship type is ignored.

All pure virtual methods will be reset to virtual. Continue?

If you want to change an abstract class to a concrete class, the class definition cannot contain pure virtual methods because a concrete class can be instantiated. Visible Analyst resets pure virtual methods to virtual in this case.

All users have been added to this project’s edit list.

All users currently defined to NetWare have been assigned to this project. There are none left to add.

All virtual methods will be reset to pure virtual. Continue ?

If you want to change a concrete class to an abstract class, Visible Analyst resets virtual methods to pure virtual.

An intermediate event attached to the boundary of an activity cannot be the target for a sequence flow.

When an intermediate event is attached to the boundary of an activity, it represents exception or compensation handling. The event must be the source of a sequence flow and cannot be a destination. Any input sequence flows must be connected to the activity itself.

To be attached to the boundary of an Activity, an Intermediate Event MUST be one of the following Triggers: Message, Timer, Error, Cancel, Compensation, Rule, and Multiple.

An intermediate event attached to the boundary of an activity can only have one outgoing sequence flow.

When an intermediate event is attached to the boundary of an activity, it represents exception or compensation handling. The event must be the source of a sequence flow and can only have one outgoing sequence flow.

To be attached to the boundary of an Activity, an Intermediate Event MUST be one of the following Triggers: Message, Timer, Error, Cancel, Compensation, Rule, and Multiple.

An item could not be added due to space constraints.

When loading a construct, the maximum number of objects was reached before the entire construct could be loaded.

An off-page connector with output connections cannot be used.

An off-page connector with input connections (invocation lines drawn from a module symbol to the off-page connector) is acting as a reference to a module on another diagram. The Page function can operate on this type of page connector. An off-page connector with output connections (an invocation line drawn from the off-page connector to another symbol) cannot be the object of a Page function.

Another user is currently adding this item - Create cannot be performed.

Someone has created the specified item, but it has not been saved. Click the Details button for more information.

Another user is currently adding this item - Label cannot be used.

Another user is adding this item to the repository, but it has not been saved. Try again later or click the Details button for more information.

Another user is currently modifying this item - Cannot be used as a subflow.

If another user is modifying a data flow repository entry, it cannot be used as a subflow.

Another user is currently modifying this item - Edit cannot be performed.

Only one user can edit a repository entry at one time.

Another user is currently modifying this item - Split cannot be performed.

Another user is modifying the repository entry for this data flow. Split operation cannot be performed.

Another user is editing an item that would be affected - Operation denied.

If a name change is attempted on a repository item and another user is editing a component item, the operation is denied.

Another user is modifying the user list of this project.

Only one user at a time can change the list of users that can access a project.

Another user is performing an operation requiring exclusive project access.

This project cannot be accessed until the user currently performing an operation requiring exclusive rights is complete. These operations include all the operations on the Tools menu such as Backup, Restore, Copy, etc.

A sequence flow cannot cross a pool boundary.

Sequence flows must connect objects within a single pool. If no pool is shown on a diagram, it is assumed that all objects are in the same pool.

A sequence or message flow cannot be attached to a data object.

Sequence flows can only be used between flow objects, while message flows can only be used between flow objects or pools.

A start event cannot be the target for a sequence flow.

A start event signals the beginning of a process so it cannot have an incoming sequence flow.

At least one user must be assigned to a division.

If all users are removed from a division, there is no way to maintain division information or use the division in an Enterprise Copy operation. In addition, the user must have Access Control rights to the division in order to modify the division contents and user list.

At least one user must have Access Control rights to a division.

If no user has Access Control rights to a division, there is no way to maintain user information for the division.

Attached relationship(s) exist, cannot delete this entity.

An entity cannot be deleted inside the repository if relationships are attached to that entity on a diagram. All relationships attached to this entity on all views where it appears must be deleted before it can be deleted.

Bad entry label.

Label does not begin with an alphabetic character.

Boilerplate not found.

The boilerplate you requested for this diagram cannot be found.

Boilerplate too big to fit on diagram.

The current boilerplate is larger than the selected diagram size. Either make the diagram drawing area larger or change the boilerplate.

Both the Related To and Values & Meanings fields contain information.

Only one of these fields may contain a value. If you declare that this item is related to another existing item, the values and meanings information should be specified in the repository entry for that item.

C struct “x” has no composition defined.

A data structure exists in the repository that has nothing in its composition field.

Can’t add Check condition “x” to Repository - Name Conflict.

The name of the check condition you are trying to add to the repository is the same as another object already in the repository. You must select another name to add it.

Can’t add Trigger “x” to Repository - Name Conflict.

The name of the trigger you are trying to add to the repository is the same as another object already in the repository. You must select another name to add it.

Can’t change process # - Diagrams that would be affected are being edited.

A process number change affects all child diagrams in the same branch of the tree. Another user is editing one of these diagrams so the operation is denied.

Can’t Get Device Context for Printer.

Windows was unable to access the selected printer. Verify the printer settings.

Can’t Get Repository Handle.

Visible Analyst could not find the repository for this project. Make sure that your files have not been moved or deleted.

Can’t perform global change - Diagrams that would be affected are being edited.

Another user is currently editing a diagram that would be affected by a global change so the operation is denied.

Can't perform conversion, project is not in version 6.0 format.

In order to be converted to the format of the current version, a project must first be in version 6.0 (or later) format. If you have a version of Visible Analyst older than version 6.0, please call Visible Systems Corporation so that we can assist you in this conversion.

Can't perform Nest - Diagrams that would be affected are being edited

Another person is editing the diagram or a child of the diagram whose name was selected as the object of a Nest operation. Use the Current Activity facility on the File menu to determine the user IDs of these people. Send an appropriate message to them using the messaging facility (press F5).

Cannot add because record already exists.

A user cannot be added to the list of valid users because the name is already used.

Cannot add this alias - It already exists as another entry.

The specified alias already exists as another entry type. All items in the repository must be defined with unique labels.

Cannot copy portion of project that contains context diagram.

A context diagram cannot be copied into an existing project. However, the image of the context diagram can be copied into another project by using the Construct function on the Diagram menu. The construct should be created from the existing context diagram and loaded into the copy-to project.

The loading of the construct into the copy-to project does not migrate the data repository information maintained for the objects appearing on the diagram, but the information for these objects is migrated if they are referenced by other diagrams included in the Copy operation.

To determine whether the information was migrated, use the Define option on the Repository menu and select each object to view its data repository definition.

Cannot delete record because it doesn't exist.

This user has not been added to the security list, so his/her ID cannot be deleted.

Cannot get record because it doesn't exist.

There is no security record for the user you specified.

Cannot open reports file.

The custom reports definition file REPORTS.TBL could not be opened. It has been deleted or is otherwise unavailable. Since it is necessary to the operation you just requested, you should copy it from your master media.

Cannot open SQLDIAL.TBL file.

The file SQLDIAL.TBL contains information on all SQL dialects supported by Visible Analyst. It has been deleted or is otherwise unavailable. Since it is necessary to the operation you just requested, you should copy it from your master media.

Cannot perform Spawn - Diagrams that would be affected are being edited.

The Spawn function affects all diagrams in same branch of the DFD tree, and one or more users are editing some of these diagrams. Use the Current Activity function on the File menu to determine the user IDs of these people and send an appropriate message to them using the messaging facility (press F5).

Class type not specified for ‘x’.

When defining attributes for a class, the type is required.

COBOL configuration file ‘x’ cannot be opened. Can’t generate COBOL code.

The file COBOLTYP.TBL contains information on how COBOL code should be generated. It has been deleted or is otherwise unavailable. Since it is necessary to the operation you just requested, you should copy it from your master media.

C**OBOL configuration file ‘x’ contains invalid data. Can’t generate COBOL code.**

The file COBOLTYP.TBL contains information on how COBOL code should be generated. It has been improperly edited or otherwise corrupted. Since it is necessary to the operation you just requested, you should fix the editing improprieties (if you are *certain* how to do so) or copy it from your master media.

Component must have a name.

When defining components using the Add Components dialog, a name must be given for each component. The name is used to reference the component in the repository and is thus required.

Conflict between default VALUE and USAGE INDEX or REDEFINES for item ‘x’. The default VALUE has been ignored.

The file COBOLTYP.TBL contains information on how COBOL code should be generated. There is a conflict between specifications in this file and the information you specified in the repository, or an internal inconsistency in specifications in this repository entry. The code generator continues to generate code, but it ignores what you specified for the default value for this item.

Conflict between picture and USAGE INDEX for item ‘x’. The picture has been ignored.

The file COBOLTYP.TBL contains information on how COBOL code should be generated. There is a conflict between specifications in this file and the information you specified in the repository, or an internal inconsistency in specifications in this repository entry. The code generator continues to generate code, but it ignores what you specified for the picture for this item.

Conflict between picture class and the number of decimal places for item ‘x’. The number of decimal places has been ignored.

The file COBOLTYP.TBL contains information on how COBOL code should be generated. There is a conflict between specifications in this file and the information you specified in the repository, or an internal inconsistency in specifications in this repository entry. The code generator continues to generate code, but it ignores what you specified for the number of decimal places for this item.

Connected page does not exist.

This diagram has been connected to another diagram with the Page function. The connected diagram has been deleted.

Construct ‘x’ not found. Enter new location:.

The construct file for the construct you selected has been deleted or is otherwise unavailable. If the file has been moved to another disk location, enter its path. If the file no longer exists, you should eliminate the dangling construct name and prevent recurrences of this error message. To do this, click Cancel. Then click Delete in the Construct dialog box to eliminate the construct name that generated this error.

Construct exceeds page size.

The current construct is of easel size and cannot be loaded into the current diagram, which is standard size. Either make the diagram drawing area larger or use a different construct.

Could not add a user-defined type for the current field.

The import program could not add a user-defined type for this field.

Couple ‘x’ used in ‘y’ has the same C name as a module.
Couples ‘x’ and ‘y’ used in ‘z’ have the same C name.

The naming rules and name length requirements for Visible Analyst are different for those in C. When generating shell code for C, repository item names must be transformed into C names. It is possible for two differently named items in Visible Analyst to end up with the same C name. This would cause problems if you try to compile this generated code. You should change one of the names so that the generated code name is different from that for the other item.

Database is corrupt. Run Rebuild and try again.

Something happened to the project database. Run Rebuild from the Tools menu to restore it to normal functioning.

Date constant expected.

A date constant is required in a ‘where’ clause restriction.

Decomposition hierarchy incompatible with data flow diagrams.

Changes have been made to a set of data flow diagrams spawned from an FDD function that are inconsistent with the set of processes descending from that function on the FDD. The Spawn Verify function cannot reconcile them. You must manually change one set of diagrams to match the other. To avoid this error in the future, changes to the hierarchy of functions and processes should be made to the FDD. Then you can let the Spawn function carry the changes to the DFDs rather than edit them both independently.

Diagram exceeds maximum page size.

Visible Analyst could not automatically draw an editable view because all of the items you specified to include in the view could not be included without having to overlay symbols. You have several options:

- You can print the view to see what it looks like and decide what to do later.
- You can return to the selection process and select fewer items to include in the view.
- You can decrease the size of the symbols in the symbol template, which might sufficiently ease the crowding of symbols to make it possible to draw an editable diagram.
- You can create clusters, including some of the entities on this view, and create an unstructured cluster diagram.
- You can reduce the point size of the fonts used for the text used to label lines and symbols before generating the view. This causes Visible Analyst to scale the size of the symbols and lines down to match the text and squeeze more information onto the view. The more important of the two is the text used to label the relationships because it decreases the space between the entities.

Your best option is probably to reduce the number of entities in the view, for a view of the current complexity is probably difficult to understand even if you create an editable diagram.

Diagram label already used in this project - Enter another label.

Each diagram in a project must have a unique name.

DLL ‘x’ not found. Cannot import.

The DLL program file specified is missing or not in the VA directory.

DOS file name for program ‘x’ is not unique, file ‘y’ has been overwritten.

During shell code generation, the output file created overwrote an existing file.

Duplicate ‘x’ clause.

The indicated clause is duplicated in the custom report specification.

Duplicate couple ‘x’ in call between ‘y’ and ‘z’.

A couple appears twice on an invocation line between modules “Y” and “Z”. This would cause a duplicated parameter in generated C code. Note that it is possible that this couple appears along the invocation and is also listed in the composition of an ITR that is attached to the invocation. In this case, its duplication would not be obvious from looking at the diagram.

Duplicate definition for ‘x’.

When defining methods for a class, a method name can be used more than once, but each instance must have a different set of arguments to make it unique.

Duplicate label encountered, object label not added.

During a construct load or tree copy operation, any label that could cause two different entry types to share the same name or create a second instance of an object limited to a single appearance on a diagram is ignored. The text of the symbols defined with duplicate labels is removed, resulting in diagrams containing unnamed symbols. An analysis of the project can be used to identify unlabeled symbols.

Duplicate parameter ‘x’.

There is a duplicate parameter in the ‘params’ clause of a custom report specification.

Enterprise connections exist for this project. Should the copied project be a satellite of the same project(s) as its source?

When copying a project that has enterprise connections, you can choose to duplicate the connections in the new project or remove them. Answering yes maintains the connections between the copied project and all projects that had links to the original project.

Entity labeled ‘x’ is dangling.

This error message is the result of not connecting the indicated entity to any events on an entity life history diagram. To resolve the error, draw a line from the entity to any event type or structure box using the Connect function.

Entity labeled ‘x’ should have no input connections.

The entity on an entity life history diagram must be at the top of the hierarchy. To resolve the error, remove all input connections to the entity.

Entry could not be found that matches current entry characteristics.

When searching for an item in the repository, an item with the selected entry characteristics could not be found.

Entry type expected.

An entry type is required in the selected report query.

Error reading file.

There was a system error reading a requested file. The current task cannot proceed.

Error reading reports file.

There was a system error reading the saved reports file. The requested report cannot proceed.

Error reading user-defined object links. Delete entry and run rebuild.

The structure containing links to user-defined objects has been damaged. You must delete this entry and re-create the link. This may happen as a result of a power loss during a repository update.

Error writing reports file.

There was a system error writing the saved reports file. The requested report cannot be saved.

Event labeled ‘x’ must be of the same type as the other events at this level.

All symbols hanging from a single node must be of the same type, or be a structure box that is used as a place holder. To resolve the error, change the symbol type of the indicated event using the Change Item function.

Event labeled ‘x’ has more than one input connection.

The above error message is the result of connecting the indicated event to more than one superior event/structure boxes. To resolve the error, move or delete all but one of the lines to connect the event to one and only one superior event/structure box.

Export failed - Could not create export files.

The files necessary to store exported data could not be created. It could be that there is not enough disk space, that you do not have rights to the destination directory, or some system failure. You must rectify the situation before the export can be done.

Fatal error splitting view(s).

A global view could not be split into pages to print. There is probably insufficient disk space.

File not found.

The file specified for import could not be found in the specified directory.

Files damaged, must rebuild.

The repository for the current project has been corrupted. To correct this condition, execute the project Rebuild tool. If the problem persists, contact Visible Systems.

First character of repository name not a letter.

All repository entries must start with a letter.

Flow selected is not attached to the parent process of this diagram.

Only a net input or output flow can be Split. A flow attached to a process specified as the target of a Nest (decompose) operation is considered a net input/output flow.

Function must be decomposed directly to processes before a Spawn operation.

Only a function, all of whose descendants on the FDD are processes, can be spawned. If some descendants are subsidiary functions, the Spawn function cannot proceed.

Functional decomposition error(s) encountered - Try analyze.

A Spawn Verify operation could not be performed because the structure of the functional decomposition diagram does not match the data flow diagram set. Run the Analyze function to determine the cause.

Illegal character - Used as a delimiter in database.

The label of an entry type such as a data flow, data store or couple, which can appear in the composition field of another data repository entry, may contain up to 128 characters. Each can consist of any upper/lower case letter, numbers, spaces, periods, underlines and hyphens. The first character must always be a letter. All other characters are considered delimiter characters.

Import can not open log file, ‘x’.

The listed file cannot be opened for reading and writing.

Import cannot create an invocation

The Import function cannot create an invocation described in the import file. This could be because the source and/or destination module names were not present or because the module names were present in the import file but they couldn't be found in the repository and Visible Analyst couldn't create them (perhaps they were of incompatible types or violated some naming uniqueness rule).

Import DLL error in ‘x’: #y Import file format incorrect.

The import file specified does not conform to the format required. See specifications for import files from other systems and make sure that the file was created correctly.

Import DLL error in ‘x’: #y Internal - exit and restart Visible Analyst.

The Import DLL (Dynamic Link Library) failed and its state is invalid. You may have to close and restart Microsoft Windows in order to run the import again. Please contact Visible Systems.

Import DLL error in ‘x’: #y Unable to open import file(s).

One or more of the import files was not found or cannot be opened.

Import DLL error in ‘x’: #y Undefined - Contact Visible Systems.

An internal error has occurred. Please contact Visible Systems.

Import file not found.

The VSC format import file was not found.

Import object type conflicts with existing object, item not imported.

The object in the import file has a type that cannot coexist with an object already in the repository of a different type.

Import procedure aborted, x records have been imported.

If Import is canceled after the actual import has begun, the number of records imported at the time of the cancellation is displayed.

Initialization failure - Contact Visible Systems.

The network version of Visible Analyst was not able to access the network. If the network shell is loaded and you are logged on, contact Visible Systems.

Integer constant expected.

An integer constant is required in a ‘where’ clause restriction.

Invalid Alias Owner.

The alias owner information for an alias record is not valid or compatible with the project in use.

Invalid condition.

There is an invalid restriction condition in a custom report ‘where’ clause.

Invalid construct name.

The name you specified for a construct contains illegal characters. Please correct and try again.

Invalid construct name, reserved DOS device.

The entered construct name is a reserved DOS device name.

Invalid entry types list.

There is an invalid entry types list in a custom report ‘where’ clause.

Invalid format value ‘x’.

The format value indicated in a custom report ‘format’ clause is invalid.

Invalid label to use in split. - Label used elsewhere in project.

Data flow labels can be the same as data store labels, but they must be different from process and external entity labels.

Invalid Location entry.

The entry you made in a Location field does not correspond to a repository item. Please enter a correct value, or use the Search function to locate it and enter it that way.

Invalid ‘order by’ value ‘x’.

The value you entered in a custom report ‘order by’ field is not a valid repository entry label or entry type. Please enter a correct value.

Invalid path specified.

The path specified could not be created. Correct and try again.

Invalid placement of ‘|’ in Composition of ‘x’.

The logical OR operator ‘|’ is not placed in a syntactically correct position in the field. Correct and try again.

Invalid project name.

The project name you entered contains invalid characters. Please enter a correct value.

Invalid report name.

The report name you entered contains invalid characters. Please enter a correct value.

Invalid ‘select’ value ‘x’.

The value you entered in a custom report ‘select’ field is not “*”, “detail,” or “summary.” Please enter a correct value.

Invalid SQL dialect format in SQLDIAL.TBL file - Reload from master disk.

The format of information in the file SQLDIAL.TBL is incorrect. Perhaps an attempt to customize the file was done incorrectly. Inspect the file customizations for correctness or reload the original file from the master media.

Invalid storage types list.

There is an invalid storage types list in a custom report 'where' clause.

Invalid VA construct file format.

The format of the file containing the construct you requested is invalid. Please restore a correct file from a backup or delete the construct and re-create it.

Invalid 'where' expression.

The restriction entered in the 'where' clause of a custom report does not evaluate to a valid restriction expression. Please consult the manual and correct.

Invocation(s) from Macro 'x' cannot be generated — not a recommended type of connection.

A Macro should not invoke another module. Change it to a normal module.

Item named 'x' altered to name 'x' to meet COBOL rules.
Item named 'x' altered to program name 'x' to meet COBOL rules.

The naming rules for Visible Analyst and COBOL are different. When generating COBOL shell code, the names of some repository objects may be transformed to conform to COBOL specifications.

Item not decomposed beyond the 01 level: 'x'.

A non-element repository item has been named but has not been decomposed at all. A correct COBOL data declaration cannot be generated.

Item not fully decomposed to data elements: 'x'.

A non-element repository item has not been fully decomposed to data elements. A correct COBOL data declaration cannot be generated.

Iteration event labeled 'x' is dangling.

The above error message is the result of not connecting the indicated selection event box to an entity or a structure box on an entity life history diagram. To resolve the error, draw a line connecting the event to an entity or structure box using the Connect function.

Key already exists.

You attempted to save a new repository item with the same name as one that is already in the repository of your project.

Key does not exist.

The label you entered in the Define dialog box does not exist in the repository. In the education version of Visible Analyst, items cannot be created directly. They must be created on a diagram or in the composition field of another object, such as an entity or data store.

Labels must be unique - Label used elsewhere in project.

If you are labeling a process, all names must be unique. If labeling a data flow, data store, or external entity, the name of one type cannot be the same as another type. However, files and data flows can have the same name.

Last record in file is not complete.

The VSC import file ended in the middle of a record.

Loading this construct would exceed diagram limits.

A construct could not be added to a diagram due to size constraints.

Location(s) exist, cannot delete this entry.

An item cannot be deleted from the data repository if it exists on a diagram or as an entry in the composition field of another data repository entry. To delete this item, it must be deleted from any diagram it appears on, and its label must be removed from the composition field of entries that reference it.

Location(s) exist to which you do not have Modify rights!

The object you are trying to rename or delete, appears in the composition or attributes field of another object to which you do not have the right to make changes. User rights are restricted by including objects in divisions. If you have not been assigned Modify rights to the division, you cannot alter objects that belong to the division.

Local data element is already used with a different class type.

When defining role names for a relationship, the name selected for the role already appears as a member of the opposite class but with a different class type than that of the current class. Because local data elements must be unique within a class, you must choose a different name for the role. For example, if you have a class named Window with a relationship to Item, and you defined the role for Window as pWindow, there must be a member called pWindow of type Window in the member list of Item.

Low memory warning: Approaching limit of diagram complexity. Save work

The amount of space allocated for this diagram is almost depleted. To avoid loss of work, execute the Save function. We suggest you either partition the diagram into multiple diagrams or not add more objects to it.

Matrix report is empty.

There were no repository objects that met the requirements you specified for a matrix report.

Maximum number of projects reached, can't create.

The completion of a New Project operation would have exceeded the maximum project count restriction. The maximum project count for each configuration is:

Education – 2
Corporate - limited to the available disk space.

To determine your product's configuration, select About Visible Analyst from the Help menu.

Maximum number of levels in decomposition hierarchy has been reached.

You have already nested to the maximum level allowed by Visible Analyst (20 levels). The current Nest operation would exceed that maximum and is not allowed.

Maximum number of symbols for this diagram.

You have already reached the maximum number of symbols on a diagram allowed by Visible Analyst (500). The current operation would exceed that maximum and is not allowed.

Maximum number of users already logged on.

The maximum number of users (or nodes) allowed for the tool you purchased are already logged on. You cannot log on until someone logs off.

Member 'x' is used more than once.

Local data elements must be unique within a class.

Message 'x' refers to a method that does not exist in the derivation tree of class 'y'.

The indicated message is based on a method that does not exist in the indicated class, or one of its base classes. To correct the problem, select the item and use the Change Item function.

Message 'x' uses a pure virtual method.

A message used on a sequence or collaboration diagram is based on a pure virtual method. Since pure virtual methods cannot be executed, a derived class should be used instead.

Message 'x' uses a virtual method.

A message used on a sequence or collaboration diagram is based on a virtual method. Since virtual methods are meant to be overridden, a method in a derived class should probably be used instead.

Missing 'format' value.

There is no value entered in a custom report 'format' field. It should be "multiple" or "single." Please enter a correct value.

Missing 'order by' value.

There is no value entered in a custom report 'order by' field. It should be a valid repository entry label or entry type. Please enter a correct value.

Missing 'restriction' condition.

There is no restriction entered in the 'where' clause of a custom report. Please consult the manual and correct.

Missing select value.

There is no value entered in a custom report 'select' field. It should be "*", "detail" or "summary". Please enter a correct value.

Modify rights to the project directory are required to perform this operation.

The creator of this project or a supervisor has given you view-only rights. One of those people must upgrade your rights before you can make any changes to this project.

More than one returned couple in call between 'x' and 'y'.

More than one couple appears on an invocation line between modules 'x' and 'y' in the return direction. This causes a function with more than one return value in generated C code and is not allowed.

Multiple associator elements on a relationship.

A relationship may have only one associator element.

Multiple Data Only Modules specified in an Information Cluster.

One of the standard modules you specified for the information cluster is in fact a data only module. An information cluster can have only one data only module.

Multiple invocations of 'x' have different number of parameters.

You invoked module “X” in different places on your structure chart with different numbers of couples along the invocation line. The code generator cannot produce proper C code in this situation.

Multiple invocations of ‘x’ have parameter of different type.

You invoked module “X” in different places on your structure chart, with couples corresponding to repository items of different types. The code generator cannot produce proper C code in this situation. Note that it is possible that these couples could exist as components of ITRs attached to the invocations. In this case, the variations are not obvious from looking at the diagrams.

Multiple invocations of ‘x’ have return values of different type.

You invoked module “X” in different places on your structure chart with one couple appearing on invocation lines to module “X” in the return direction corresponding to repository items of different types in different instances. The code generator cannot produce proper C code in this situation.

Multiple Related To items specified.

More than one Related To item was specified in a field in an import file.

Name exceeds 128 characters.

The name you entered exceeds the Visible Analyst maximum of 128 characters.

Name not specified where required.

You have not entered the name of a repository item where one is required.

Name of connected item is not a valid VSC name.

A name you entered contains invalid characters. Please reenter.

Nested braces in composition of ‘x’.

Curly braces cannot be nested in a composition field to show array structure.

NetWare shell is not loaded.

You have attempted to run the LAN version of Visible Analyst when both of the NetWare shell programs IPX.COM and NET*x*.COM are not memory resident. Load them into memory and then restart Visible Analyst.

No complete clusters exist on this view.

A cluster view replaces a group of entities with a cluster on a cluster diagram only when all of the entities composing the cluster appear on the current view. Partial clusters are ignored. Either none of the entities appearing in the current view is a member of a cluster or there is no cluster for which all of the composing entities exist on this view. In either case, the cluster view is the same as the existing view.

No entities are available for inclusion in cluster.

Either there are no entities in the project or all of the entities are currently members of clusters and the free entity pool is empty. An entity can be a member of only one cluster.

No entries in repository.

A view can be created by the selection process only when there are already entities and relationships in the repository from which to select. You must either create your view manually, by adding entities and relationships to blank diagrams, or you must enter the repository using the Define function from the Repository menu and create entity entries. Then you can add them to a diagram and draw relationships between them.

No length specified for item 'x'. Using 'y'.

Item 'x' has no length specified in its repository entry. Shell code is generated using the default value y.

No more entries.

While searching for the next/prior item in the repository, the end/beginning of list has been reached.

No reports found in reports file.

The report saver file is empty.

No space left on drive.

The target drive is full or there is not enough space to copy all of the files.

No valid programs in the project 'x'. Using the entire project.

There is no program item in your project repository referring to a module for which to generate shell code. The code generator assumes that you want a code for a program corresponding to your entire project structure chart.

Not enough memory to execute this function.

While printing a report, Windows could not allocate the memory space required to process the report.

Object type is not supported by this version of Visible Analyst.

The item you are attempting to open is of a type that is not supported by this version of Visible Analyst. For example, if your tool set does not support object modeling, you cannot access class and member objects that were created by other Visible Analyst tool sets. Contact Visible Systems to upgrade your product configuration.

Object ‘x’ is based on an abstract class.

An object used on a sequence or collaboration diagram is based on an abstract class. Since abstract classes cannot be directly instantiated, a derived class should probably be used instead.

Object ‘x’ is not used. There are no messages attached.

This error is the result of not having messages attached to either an object lifeline on a sequence diagram, or an object link on a collaboration diagram. To correct the error on a sequence diagram, connect two object lifelines with a message line. To correct the error on a collaboration diagram, add a message to an object link line.

One or more of the selected tables have columns that are either members of the view or are used in an expression. Remove?

When removing a base table from a view, if columns from the table are specified in the column list or are used in an expression (Filter, Having, Group By, etc.), the user is prompted before the table is removed. After confirming the delete, any expression that references the table will have the table name replaced by the work DELETED.

Only one processes allowed on a context diagram.

Because the context diagram defines the scope of the project, only one process is allowed. It is numbered as process 0.

Only two projects allowed in the Education Version.

The education configurations of the product limit you to editing two projects at a time. However, you may maintain other projects on backup disks created by the Visible Analyst Backup tool. When these projects are needed, you should Backup and Delete the current project then Restore the required project.

Only someone logged in as supervisor can turn off security.

To disable security for the single user version of Visible Analyst, the user must be named supervisor or be assigned as system manager.

Operation labeled ‘x’ is dangling.

This error message is the result of not connecting the indicated operation to an event on an entity life history diagram. To resolve the error, draw a line connecting the operation to an event using the Connect function.

Operation labeled ‘x’ cannot be connected to an entity.

This error message is the result of connecting the indicated operation to an entity. Operations can only be connected to events. To resolve the error, connect the operation to an event using the Connect function, or remove the connection to the entity.

Operation labeled ‘x’ has more than one input connection.

This error message is the result of connecting the indicated operation to more than one event. To resolve the error, move or delete all but one of the lines to connect the operation to one and only one event.

Parameters missing.

There is one (or more) missing parameter in the ‘params’ clause of a custom report specification.

Parent already has 40 children.

Each diagram in a project can have a maximum of 40 diagrams on the level immediately below it. The New Diagram, Nest or Page operation cannot continue.

Parent missing. Must restore before editing children.

The parent diagram of the selected child cannot be located in the directory containing the project. A DOS delete or rename command could have caused this condition. To correct this condition, Restore the project data files from the backup disk(s) generated by a Visible Analyst Backup operation.

Problem accessing database files. - Operation canceled.

An unexpected error status was returned from the record manager. To correct this condition, execute the project Rebuild function. If the problem persists, contact Visible Systems.

Problem accessing security file.

The VASECUR.VAW file is missing or damaged. Please contact Visible Systems.

Problem creating directory - Cannot proceed with operation.

When creating a new project, there was a problem creating the needed subdirectory. Either you have insufficient rights to perform the operation, or there was a problem with the drive.

Problem creating network transaction file - Cannot proceed with create.

Critical error; contact Visible Systems.

Problem creating project repository - Cannot proceed with operation.

When creating a new project, there was a problem creating the needed files. Either you have insufficient rights to perform the operation, there is insufficient disk space, or there was a problem with the drive.

Problem opening tree file - Cannot proceed with operation.
Problem reading project tree file - Cannot perform save.
Problem reading project tree file - Cannot print.

Critical error; contact Visible Systems.

Problem sending message.

A message could not be sent to the selected user. It could be that the user is not logged in on the network. Otherwise, try reloading the network shell.

Problem with current device.

Windows had a problem accessing the output device you specified.

Problem with transaction log - Cannot proceed.

The transaction file for this project has been corrupted. To correct this condition, contact Visible Systems for instructions.

Process number already in use.

The specified process number is already used. A process number must be of the form 'Parent-number + "." + Child-number' where child-number is a unique number between one and one hundred.

Process view is empty.

You have attempted to create a process view and indicated a process that does not use data elements listed in the composition field of any existing entity. There must be data flows either entering or leaving the process that are composed directly or indirectly of data elements which one or more entities list in their composition fields.

Program 'x' has invalid root module.

The specified program repository item does not have a module listed in its composition field.

Project already exists.

The project specified for creation already exists.

Project already has maximum number of diagrams.

The diagram count for the current project involved in a New Diagram operation or the Copy or Copy Diagram operation exceeds the maximum allowable diagram count if the operation were to complete. The following table describes the maximum diagram count *per diagram type* for the each configuration:

Education – 10
Corporate –limited to the available disk space

To determine your product configuration, select About Visible Analyst from the Help menu.

Project does not exist.

The project root that was entered could not be found in the project master list. Try restoring the project from a set of backup disks.

Project files damaged or not convertible.

A project created with an earlier version of Visible Analyst is damaged or not complete and cannot be converted to the current release.

Project is currently being created by another user - Cannot proceed.

The project specified is being created by another user. Either select another project root name or wait until the other user saves his or her work.

Project master file cannot be opened - May be damaged.

There was a problem accessing the project master file. Contact Visible Systems.

Project not found in master file.

The selected project could not be found in the list of projects.

Project version (X.Y) not supported.

This project was created with a prior version of Visible Analyst. It must be converted to version 6.0 before being used. To convert a 3.1 or *later* project to version 6.0, execute the Rebuild tool. If the project was created by a version *prior to* 3.1, then contact Visible Systems. We can convert it for you.

Project 'x' not found. Enter new location:

This project could not be found in the directory path where other projects reside. Perhaps it is on another drive or in another directory, or perhaps it is on a server or a local drive.

Project's database files not found.

The database files for this project could not be found in the directory where the project resides. Either the files have been deleted or moved with DOS functions. They need to be restored to the proper directory before Visible Analyst can proceed. You also have the option of restoring the most recently backed up version of the project.

Query definition error: ‘x’.

The specified error occurred when you defined a custom report query. Please correct and try again.

Rebuild failed. Cannot create data store list.

Rebuild could not access or update the temporary project.

Rebuild failed. Cannot create work files for project ‘x’.

Your system may be low on memory or disk space.

Record header format is not correct.

The header for an import record is not properly formed.

Recursive definition for ‘x’ is not permitted. Set Reference type to Address.

A class cannot be used in its own definition unless the reference type of the member indicates it is a pointer to the class.

Recursive views are not allowed.

When defining the base tables for a view, the list cannot contain the view itself. Any other view or table is allowed.

Relationship already exists with different cardinality.

A relationship actually consists of the relationship itself and the two entities to which it is attached. An entity/relationship combination can appear on multiple views, but only once per view. However, each occurrence must specify the same cardinality. You are attempting to specify a cardinality for a relationship different from how it appears on another view.

Relationship already exists with different set of names.

A relationship actually consists of the relationship itself and the two entities to which it is attached. An entity/relationship combination can appear on multiple views, but only once per view. However, on each occurrence, the relationship must have the same names in both directions. You are attempting to specify a relationship with mismatched names in the opposing directions of the relationship.

Report name missing.

The report name is missing in the ‘report’ clause of a custom report specification.

Report saver file is empty.

No report format files have been created. To create a saved report, define a report in the Reports dialog box and click the Save Report button. In the subsequent dialog box, give the report a name and click OK.

Report “x” not found.

The report you requested from the report saver file could not be found. The report names contained in the file are listed in the dialog box. You can choose one from the list.

Repository string exceeds the maximum length.

One of the strings in a VSC import file exceeds 254 characters in overall length.

Restriction too long.

The restriction entered in the custom report definition ‘where’ clause is too long.

Return value from “x” is sometimes ignored.

Some invocations of module “x” do not have a return couple and some do. Invocations should be consistent.

Root module for C program “x” is not named ‘main’.

The module named in the repository Program item for which shell code is being generated needs to be named “main” for C programs.

Security cannot be disabled for network version of Visible Analyst.

In order for Visible Analyst to work properly on a network, security must always be turned on.

Selected SQL dialect is not defined in SQLDIAL.TBL - Reload from master media.

The information for the SQL dialect you have selected is not in the file SQLDIAL.TBL. You should reload the file from the master media.

Sequence event labeled ‘x’ is dangling.

This error message is the result of not connecting the indicated sequence event to an entity or any events on an entity life history diagram. To resolve the error, draw a line connecting the event to an entity or structure box using the Connect function.

Selection event labeled ‘x’ is dangling.

This error is the result of not connecting the indicated selection event box to an entity or a structure box on an entity life history diagram. To resolve the error, draw a line connecting the event to an entity or structure box using the Connect function.

Some components already exist - Conflicting information was not changed.

When defining components using the Add Components dialog, an item was defined that already existed in the repository but with different physical characteristics. The original characteristics were not modified.

Statement type 'x' is being used in the current project and cannot be removed.

You cannot delete a statement type from the current project if there are statements that us it.

Stereotype not defined for relationship between the use cases 'x' and 'y'.

For relationships between a pair of use case symbols, a stereotype must be specified, indicating the type of relationship. A name is optional. To resolve the error, find the relationship and set the stereotype using the Change Item function.

Storage type expected.

A storage type is required in a custom report specification.

String constant expected.

A string constant is required in a 'where' clause restriction.

Structure box labeled 'x' is dangling.

This error message is the result of not connecting the indicated structure box to an entity or any events on an entity life history diagram. To resolve the error, draw a line connecting the structure box to an entity or event using the Connect function.

Subdirectory could not be created.

A default subdirectory could not be created for your new project because your disk is full, there are no more available directory entries (a very rare occurrence), or because you do not have rights to create subdirectories from the current directory. Check your available disk space or check with your network administrator.

System boundary labeled 'x' has no components.

The above error message indicates the named system boundary does not have any use case object defined in its composition. A system boundary is used to group objects. To resolve the error, either delete the system boundary, or add components using Define.

Template files missing - Reload from master media.

One of the symbol template files (TEMPLATE, TEMPLATE.SC, TEMPLATE.ER, TEMPLATE.US, TEMPLATE.FD, TEMPLATE.CL, or TEMPLATE.ST) is missing. Reload from master media.

Text string is not in the proper format.

All strings in the import file must be enclosed in quotes.

The first character of a label must be a letter when rules are enabled.

All entries in the project repository must begin with a letter when rules are enabled.

The item "x" is not valid when combined with the other line parameters you have previously chosen.

Not all combinations of line elements are valid. If you select an invalid combination, you cannot continue. Either reset to a valid setting or cancel the operation.

The Main Transaction Log could not be accessed

Visible Analyst could not open the Main Transaction Log for some reason. Please call Technical Support so that we can assist you.

The maximum number of projects have already been created.

In the education version of Visible Analyst, only two projects can be created. If you want to create another project, you must delete a project before continuing.

The requested scaling factor is too small for the selected page size.

The scaling factor you selected cannot be used with the selected page size because the workspace required to represent the diagram would be too large. The minimum valid scaling factor is set. As the page size increases, so does the minimum scaling factor.

The requested diagram settings are too small to include all diagram objects. The scaling factor has been changed.

The page settings you specified (workspace, orientation, page size, and scaling) are not large enough to encompass the entire diagram. The scaling factor has been changed to accommodate all the diagram objects.

The selected object cannot be used because of the way the current object is used on the diagram

You cannot change the name of the current object to the name that was selected using the Search function because some characteristic of the selected object conflicts with the definition of the current object. For example, you may have chosen to change the name of a gateway that was defined as data-based to one that has a definition of complex.

The selection that you made requires the existence of objects that you have chosen to delete. The selection will not be changed.

When resolving conflicts during enterprise copy, you cannot make your selections such that a dependent object is retained while the parent object is deleted. For example, if you choose to delete a class, all member functions (or modules) associated with that class must also be deleted.

The settings are too small to include all diagram objects, but changing the scaling factor conflicts with the page size.

The page settings you specified (workspace, orientation, page size, and scaling) are not large enough to encompass the entire diagram. However, the scaling factor cannot be changed because the workspace required to represent the diagram would be too large. Change the workspace, orientation, or page size to accommodate all the diagram objects.

There are more than two description lines.

An import file has more than two lines of description information.

There are no virtual methods defined for this class.

Only classes with virtual or pure virtual methods can be marked as abstract. Visible Analyst automatically marks classes with pure virtual methods as abstract.

There are 'x' unnamed actor(s).

This error is the result of not labeling actors on your use case diagram. To resolve the error, find and label them using the Change Item function.

There are 'x' unnamed iteration event(s).

This error is the result of not labeling iteration events on your entity life history diagram. To resolve the error, find and label them using the Change Item function.

There are 'x' unnamed message(s).

This error is the result of not labeling messages on your sequence or collaboration diagram. To resolve the error, find and label them using the Change Item function.

There are 'x' unnamed note(s).

This error is the result of not labeling note symbols on one of several types of UML diagrams. To resolve the error, find and label them using the Change Item function.

There are 'x' unnamed object(s).

This error is the result of not labeling objects on your sequence or collaboration diagram. To resolve the error, find and label them using the Change Item function.

There are 'x' unnamed operation(s).

This error is the result of not labeling operations on your entity life history diagram. To resolve the error, find and label them using the Change Item function. Note that even though the name does not appear in the operation symbol, the name must exist and will be displayed in the operation table.

There are 'x' unnamed relationship(s).

This warning is the result of not labeling relationships on your use case diagram. To resolve the warning, find and label them using the Change Item function.

There are ‘x’ unnamed selection events.

This error is the result of not labeling selection events on your entity life history diagram. To resolve the error, find and label them using the Change Item function.

There are ‘x’ unnamed sequence events.

This error is the result of not labeling sequence events on your entity life history diagram. To resolve the error, find and label them using the Change Item function.

There are ‘x’ unnamed swimlane(s).

This error is the result of not labeling swimlanes on your activity diagram. To resolve the error, find and label them using the Change Item function.

There are ‘x’ unnamed system boundary(s).

This error is the result of not labeling system boundaries on your use case diagram. To resolve the error, find and label them using the Change Item function.

There are ‘x’ unnamed use case(s).

This error is the result of not labeling use cases on your use case diagram. To resolve the error, find and label them using the Change Item function.

There must be at least two event blocks defined on this diagram.

Every entity must have a creation event and a destruction event. In addition, there may be other events that take place in between. This error indicates there are not at least two connections from the defining entity to event symbols. To resolve the error, connect the entity to at least two events using the Connect function.

There should be only one entity on this diagram.

An entity life history diagram describes the events and operations of a single entity. Remove all but the desired entity for this diagram.

These columns are components of a relationship. Do you want to include all members of the relationship?

When adding a join relationship by dragging and dropping columns, if the pair of columns selected are part of a relationship based join, the user can choose to add all the columns from the relationship. Note: the columns will not appear in the Additional Join Restrictions control because they are part of a relationship definition.

These PERFORM statements are generated from invocation lines.

You specified that module description fields are to be interpreted as COBOL code and placed into your generated shell code. Therefore, the PERFORM statements that would otherwise be generated from module invocations appearing on structure chart diagrams are not used as generated code. Rather, they are placed in the code as comments for your reference, to compare against the code stored in the module description field.

This class appears on an entity-relationship diagram.

If a class appears on an entity-relationship diagram, the persistent indicator cannot be turned off. This is because an entity or table outlasts the execution of a program.

This couple exists on other diagrams, so it may not be changed.

Couples of different types cannot have the same name. You have tried to locally change a couple to a different type when the original couple has locations on other diagrams. Either choose a new name for the couple before attempting to change the type, or make a global type change, throughout your project.

This division has been used in an Enterprise Copy. Delete Enterprise Tags before removing.

If a division has been used in an Enterprise Copy operation, the link between the enterprise project and the satellite project must be removed before a division can be deleted. See Removing Enterprise Links for details.

This entry can only be associated with one 'x' planning statement type.

When this statement type was defined, the Cardinality was set to 1:1, so it cannot be linked to more than one entry.

This entry has been deleted.

The item selected has been deleted by another user since the window was displayed. Select another entry.

This file is currently locked by another user.

The file specified for import or export is being accessed by another user. Try again.

This is disk "x" of the backup. Please insert backup disk "y".

The wrong disk was inserted during a Restore operation. The disks must be accessed in the same sequence as when they were produced during Backup operation.

This is not a conditional invocation line. Reposition start point.

This message is issued when an attempt to specify an additional conditional invocation line fails because the conditional terminator (the diamond shaped one) was not chosen. To associate another conditional invocation line to a pre-existing conditional terminator, select the conditional terminator with the Line Settings function on the Options menu. Now point to the middle of a preexisting conditional terminator and select it as the start point of the line. Next choose the appropriate module as the endpoint of the line. A second conditional invocation line is drawn as if it were emanating from the conditional terminator you selected as the start point. In fact, there are two conditional-terminated invocation lines with the conditional terminators exactly superimposed on each other. Either can be moved or deleted separately from the other.

This item cannot be deleted or modified.

The text caption that appears on diagrams generated by the education version of Visible Analyst cannot be altered or removed.

This page already contains the maximum of 2000 text items per page.

You are allowed up to 2000 blocks of text on a diagram.

This page already contains the maximum of 1000 lines per page.

You are allowed to have any combination of 1000 lines and line segments on a diagram.

This page already contains the maximum of 500 symbols per page.

A diagram may contain up to 500 symbols. Once this maximum is reached, you can't add more. Try subdividing the diagram.

This PERFORM statement is generated from an invocation line.

You have specified that module description fields are to be interpreted as COBOL code and placed into your generated shell code. Therefore, the PERFORM statements that would otherwise be generated from module invocations appearing on structure chart diagrams are not used as generated code. Rather, they are placed in the code as comments for your reference, to compare against the code stored in the module description field.

This planning statement is already associated with another entry.

When this statement type was defined, the Cardinality was set to 1:1, so it cannot be linked to more than one entry.

This project is stored on a local drive to which you do not have access.

The selected project is stored on a local drive on another workstation. You must be using that station in order to access the project, even if you have been given rights.

Too many flows to fit on diagram. - Remaining flows ignored.

This message is displayed by the Nest and Split operations. When splitting a data flow, there was not enough space in the margin to draw all the subflows. Therefore, only the subflows specified before this message appeared are drawn in the diagram's margin. If you wish to continue splitting the parent data flow, move the flows on the margin (either individually or in a block) about two inches toward the middle of the diagram and once again select Split. Now select one of the subflows as the object of the Split. The split operation further decomposes the parent data flow, not the subflow. Then enter or select the component subflows. Repeat this procedure until you have fully decomposed the parent data flow.

The instructions for the Nest operation are similar. Follow the above instructions to move the flows from the margin(s) of the child diagram. Then, Save the diagram and return to the parent diagram using the Parent option of the Nest function. The parent diagram containing the nested process should be the current diagram. Now locate the nested process and re-nest this parent process using Nest Explode operation. Another set of data flows is dragged down and drawn on the child diagram. Repeat this procedure until the message is no longer displayed.

Too many parameters.

There are too many parameters in the ‘params’ clause of a custom report specification.

Too many reports.

You have already saved the maximum number of custom reports. You cannot save another.

Tree file already exists in this subdirectory - Select another path.

The subdirectory to which you are moving the project already contains the current project tree file.

Trigger “x” already in repository. Add to Table?

You tried to add a trigger to the repository that already exists. If you want to attach it to the current table (entity), click OK. Otherwise, click Cancel.

Type for couple “x” cannot be determined.

The element related to by couple “x” has no physical characteristics defined.

Unable to access path\file.

Problem accessing a file in the specified directory. You might not have rights to the directory.

Unable to delete directory.

The directory containing the project you are deleting cannot be deleted. You might not have rights to the directory or the directory might not be empty.

Unbalanced braces in composition of “x”.

The curly braces in the composition field of repository item “x” are not balanced.

Unbalanced parentheses.

The parentheses in an expression are not balanced.

Unexpected EOF.

An end-of-file was encountered in the REPORTS.TBL file where other report definition statements were expected. Please correct the file and try again.

Unexpected error during VA operation.

Critical error, contact Visible Systems.

Unknown column "x".

The column name you entered in a 'where' clause restriction is invalid. Please consult the manual and enter a correct value.

Unknown parameter "x".

The parameter you entered in a 'where' clause restriction is does not match any parameter expression you entered in the 'params' clause. Please enter a correct value.

Use case labeled 'x' is dangling.

The above warning message is the result of not connecting the indicated use case to another use case on the use case diagram. To resolve the error, draw a line from the use case to any use case using the Connect function.

User file cannot be accessed if security is disabled.

The Users option on the Tools menu cannot be selected if security is not enabled. To enable security in the single user version of Visible Analyst, select the Security option from the Options menu and enter a user ID of "SUPERVISOR." This enables security and prevents unauthorized users from entering Visible Analyst. After enabling security, the supervisor can assign a password for the Supervisor User ID. To disable security, sign on as SUPERVISOR and deselect Security.

User not logged on.

When security is enabled, a list of valid user IDs is checked. The one you used is not among them. Check with the supervisor.

VR Repository Access Error (Error Displays as a Dialog Box)

This error is issued by the BTRIEVE data base manager employed by the repository. The message usually occurs because the data repository and diagram parameter files are unsynchronized due to a latent problem or power loss during a Visible Analyst operation. It could also happen because one or more repository files have become unavailable due to deletion. There are other possible causes as well.

Because it indicates a serious problem, we recommend that you call Visible Systems so that we can help you correct the situation. It is important that you record the other information that appears in the dialog box. A simple way to do this is to press ALT+PrintScreen and capture the dialog box to the Windows Clipboard. You can then open Windows Write program (or another software package that can deal with bitmaps). Once in Write (or whatever other software you choose), select Paste from the program Edit menu and the image from the Clipboard appears. Then you can select Print from the File menu and print it, so that you can refer to it when you talk to Visible Systems technical support personnel.

Warning: A pure virtual method should not have a definition. Continue?

If a class method is defined as pure virtual, or abstract, the Module Description field should be empty because pure virtual methods are never executed. They are meant to be defined by derived classes.

Warning: Both Related To and Values & Meanings contain information.

It may be redundant to specify information in the Values & Meanings field when a couple is related to an element, data structure, data flow, data store or file. We suggest you specify information in the Related To field when a couple represents a data flow diagram element, data structure, etc., and remove any information from the Values & Meanings field.

In other instances, when the couple only exists within the context of the structure chart diagram and bears no meaning in terms of project data flow diagrams, then the Values & Meanings field should describe this couple. For example, a control couple that signals an end of file condition would likely contain information in its Values & Meanings field because a condition or signal should not be defined in a data flow diagram.

Warning: Entity life history diagram ‘x’ is not associated with an entity.

This warning is the result of not exploding an entity on an entity relationship diagram to the indicated entity life history diagram. To resolve the warning, open an entity relationship diagram on which the entity appears, select the entity, and use the Explode feature to attach to the indicated ELH diagram.

Warning: Related to entry “x” does not exist in the repository.

A label was entered into the Related To field which does not reference a dictionary entry. Correct the spelling of the label. To determine the correct spelling of the label, execute a repository search.

Warning: Relationship “XrY” omitted from schema because it doesn’t appear on the included view(s).

The named relationship either joins two entities (somewhere in this project) or is attached to an entity included in the current schema but does not appear on any of the views you chose to include in this schema. Therefore, it is omitted from the schema you are generating. If this is not what you want, add one of the views containing it to the list for the schema generation.

Warning: Relationship “XY” omitted from schema because Entity “Z” doesn’t appear on the included view(s).

The named relationship is attached to Entity “Z”, that is included in the current schema but does not appear on any of the views you chose to include in this schema. Therefore, it is omitted from the schema you are generating. If this is not what you want, add one of the views containing it to the list for the schema generation.

Warning: There are ‘x’ unnamed structure box(es).

This warning is the result of not labeling structure boxes on your entity life history diagram. To resolve the warning, find and label them using the Change Item function. However, since structure boxes are simply place holders, it is not necessary to label them.

Warning: This entry type is normally related to a data element.

A label was entered into the Related To field of a control couple that does not refer to a data element. A control couple represents an item used to coordinate the processing between procedures. In most instances, it is a single conditional signal. The Related To field should only contain an element reference due to the data flow diagramming restriction that prevents the definition of conditionals and the elementary nature of a control couple.

Warning: This entry type is normally related to a process.

A label was entered into the Related To field of a structure chart module entry that does not refer to a data flow process. A structure chart module represents the encoding of one or more data flow processes. Therefore, the Related To field should contain a label that references a process.

Warning: This entry type is normally related to an element, data structure, data flow, or file.

A label was entered into the Related To field of a data or generic couple that does not refer to a data element, data structure, data flow or file. A data or generic couple represents the data items passed between structure chart modules. These data items are inputs furnished to and the outputs produced by the modules. A high correlation exists between the data flow diagram entry types such as an element, data structure, etc., and a data or generic couple because the structure chart is derived from the data flow diagrams of the project.

Warning: “x” in composition field has an invalid type for this object.

You entered a name in the composition field of “x” whose type cannot be used for object “X.”

Warning: “x” in composition field is referenced by another cluster.

You entered an entity into the composition field of a cluster that is already in the composition of another entity. No entity may be referenced by more than one cluster. To select from the free entity pool (entities not referenced by any cluster), position the cursor in the composition field of the cluster and click the repository Search button. Multiple entities can be selected from the repository search dialog box.

You do not have rights to change this object.

The selected object belongs to a Division to which you do not have Modify rights. Division rights are inherited from a user’s project rights, but can be further restricted within the division.

You do not have rights to this directory.

Either you have not been assigned as a trustee to the directory selected or you have insufficient rights in the directory. There are two other ways this error can be caused. First, the network drive mappings changed since directory rights were assigned. Second, some users are attempting to access this network volume with different drive mappings than other users.

Trustee assignment is accomplished through NetWare functions, normally performed by the supervisor. The following trustee rights must be granted for the user to gain access to the project: Read, Write, Open, Create and Search.

You do not have rights to this diagram.

The selected diagram belongs to a division to which you do not have access rights. Division rights are inherited from user project rights, but can be further restricted within the division.

You do not have rights to this object.

The selected object belongs to a division to which you do not have access rights. Division rights are inherited from user project rights, but can be further restricted within the division.

You do not have rights to update the VAW.INI file.

In order to change user-defined attributes or objects, you must have write access to the system VAW.INI file that is stored in the directory where VA32.EXE was loaded.

You do not have the required rights to perform the enterprise update requested.

In order to complete an Enterprise Copy operation, you may need modify rights on the selected division in either the enterprise project, the satellite project, or both depending on the selections you made in the Update Actions dialog box. Division rights are inherited from user project rights, but can be further restricted within the division.

You do not have view rights for all objects in the specified division.

In order to perform an Enterprise Copy operation, you must have view rights for the division being copied. Division rights are inherited from user project rights, but can be further restricted within the division.

You have insufficient rights to the directory where the Visible Analyst is installed.

In order to use Visible Analyst, you must have read, write, create, and erase rights to the directory where the Visible Analyst system data files are stored. If the System Data Path was set to be different from the program directory during the install procedure, you only need to assign read rights to the program directory. Establishing a separate directory for programs and data gives you the most flexibility in assigning rights to control access.

You have not been assigned access rights to this project.

In order to access this project, the user who created the project or a user with supervisor equivalent privileges must include your user ID in the access list maintained by the project. Furthermore, you must be defined as a trustee of the directory containing the project either implicitly, through the cascading effect of parental privileges, or by an explicit directory entry.

To include the user ID in the access list maintained by the project, select Users from the Tools menu. Trustee assignment is accomplished through NetWare functions, normally performed by the supervisor. The following trustee rights must be granted for the user to gain access to the project: Read, Write, Open, Create and Search.

You have not been assigned modify rights to this project.

The creator of this project or a supervisor has given you view-only rights. One of those people must upgrade your rights before you can make any changes to this project.

You may not use arcs and elbows for invocation lines that have couples.

Because of the way couples are drawn, you cannot add couples to invocation lines of these types and you cannot change an invocation line with existing couples into one of these line types.

You must have exclusive rights to branch - Another user is editing a child diagram.

When a diagram is inserted into the tree, all diagrams below it in the same branch could be affected. Someone is editing a diagram within the branch. Therefore, the create cannot be performed.

You must have exclusive rights to project to perform this operation.

The selected function can only be performed if you are the only user accessing the current project.

You must have rules & repository support to access this project.

The project you are attempting to open was created with a repository, and the version of Visible Analyst that you are using does not have repository support. Contact Visible Systems to upgrade your product configuration.

You must install printers from Windows Control Panel before this function can be used.

You cannot change printers until at least one is installed. You must do this from the Windows Control Panel.

You must select an invocation line before pasting unassociated couples.

If a couple is copied to the Windows Clipboard without an associated invocation line, you must select an invocation line before performing a Paste operation so that Visible Analyst knows where to place the couple.

You must Undo changes to the current line-in-progress to change modes.

If a line is being drawn, but has not yet been completed, you must either complete the line or select Undo from the edit menu to cancel the line before you can switch to another drawing mode.

'X' statement types cannot have children.

When this statement type was defined, the Composite Type option was not enabled.

'X' statement types cannot be linked to other objects.

When this statement type was defined, the Allow Linkage option was not enabled.

'X' statement types cannot be linked to this type of object.

When this statement type was defined, the Link To type was different than the object type to which you are attempting to link.

Appendix B

Customizing an SQL Dialect

After selecting the Options function SQL Dialect, you see a dialect named User-Defined. Selecting this and then choosing Generate SQL from the Repository menu generates SQL DDL that includes all possible options. There is no existing SQL dialect that includes all of them; however, you can customize this dialect if you need an SQL dialect that Visible Analyst does not yet support.

To customize the dialect:

Keep a Backup of the Original File:

1 Make a copy of the file \VA\SQLDIAL.TBL so that you can restore it and start over in case you make a serious error.

Edit the Dialect Description Table:

2 With an ASCII text editor, edit the file SQLDIAL.TBL. Each dialect is described in a section beginning with the name of the dialect in brackets. Search for [sql user] at the beginning of the section describing User Defined SQL.

3 You see the various options listed followed by a colon, and then Yes or No (indicating whether or not the option applies) or a numeric value describing a length constraint. Change these values to reflect the requirements of your SQL dialect. If ‘Semicolons:’ is set to yes, each DDL statement is appended with a ‘;’.

Select Applicable Data Types:

4 This is followed by a section defining the available data types for the customized dialect. For any data type that is not supported by your dialect, the text following the colon should be deleted. Alternatively, the entire line can be deleted. For data types available in your dialect, you must also indicate how many parameters are to be passed to SQL when indicating a certain data type. Examine how it is done in other dialects if you have questions.

For example, examine the definition for the data type “decimal”. You see three possible syntaxes:

DECIMAL(,), DECIMAL(), and DECIMAL. The first indicates that SQL would expect a definition of a decimal data type to include both the field length and the number of decimal places (DECIMAL(6,2)). Similarly, DECIMAL() would describe a declaration with just one length value, and DECIMAL would contain no length value. Consult the documentation for your SQL dialect to find out how it expects data types to be defined and what the numbers mean to that specific dialect.

If the list of data types begins with a single colon (:), this indicates the logical-to-physical mapping wastes storage space. If the list starts with a pair of colons (::), the SQL dialect does not have a native type that corresponds to the logical type.

Note

- In the LAN version, this file is Read Only. A user with network supervisor privileges must change the file to Read/Write before editing can be done.

Appendix C

Customizing the Trigger Wizard

When performing SQL schema generation, you can enforce referential integrity either declaratively or through the use of the Trigger Wizard. The Trigger Wizard automatically generates the triggers that are necessary to enforce the referential integrity constraints specified by the relationships defined in your data model. If you want to customize the trigger code that is produced by the Trigger Wizard, you must follow the guidelines outlined in this section.

Create a Supplemental Trigger Definition File:

1 When generating triggers, the Trigger Wizard uses the TRIGSYS.TBL file to create the necessary SQL code. This file contains macro definitions for all triggers that can be generated for each SQL dialect. This file should ***never*** be modified. Instead, create a file named TRIGUSER.TBL to use to redefine macros that you wish to change. (You can simply copy the TRIGSYS.TBL.)

Determine the Scope of the Changes:

2 The trigger definition file is divided into sections. In the first section, [Template global], define macros that are used by all other sections in the file. For each SQL dialect that you use, there should be a section named [Template dialect], where dialect is the name of the target RDBMS. Valid dialect names include Server System 10, Oracle, and Watcom.

Create New Macros:

3 There are two types of modifications that you can perform: modify the code of an existing macro or create a new macro that is to be called by one or more of the standard macros. To create a new macro, a C-style convention is used:

#define *identifier* {*substitution-text*}
O*R*
#define *identfier* (*parameter-list*) {*substitution-text*}

where all occurrences of *identifier* are replaced with *substitution-text*. The *substitution-text* can contain zero, one, or more statements. Note that the opening and closing braces ({ }) are required.

There are two operators that can be used for token passing:

token##*parameter, parameter*##*token*
AND
token#^*parameter, parameter*#^*token*

If ## precedes or follows a formal parameter in the definition of a macro, the actual argument is concatenated with the token on the other side of the ## before the macro is expanded. If #^ is used, the expansion is performed before the passing.

To remove a macro definition, use the directive #undef *identifier.*

To define a single-line comment, use the **#—** directive. Anything after the directive is ignored by the Trigger Wizard. To define a multi-line comment use **#/*** to start the comment and ***/** to end it. Macro comments do not appear in the generated trigger.

Modify the Trigger Preamble Macro: 4 The Trigger preamble is a special macro that gets expanded at the beginning of every trigger generated by the Trigger Wizard. It can be used to store code that is common to all triggers.

Modify the Referential Integrity Macros: 5 The Trigger Wizard uses a set of pre-defined macros to enforce the referential integrity constraint defined by a relationship. The names of the macros cannot be altered. However, you are free to make any changes you wish to the body of the macro.

OnDeleteCascadeChild
Delete dependent rows when the parent is deleted.

OnDeleteRestrictParent
Don’t allow parent to be deleted if dependent rows exist in the child.

OnDeleteSetDfltChild
Set dependent foreign key columns to a default value when the parent is deleted.

OnDeleteSetNullChild
Set dependent foreign key columns to NULL when the parent is deleted.

OnUpdateCascadeChild
Change dependent foreign key columns when the primary key (or any unique constraint) columns in the parent are modified.

OnUpdateRestrictParent
Don't allow primary key columns in the parent to be modified if dependent rows exist in the child.

OnUpdateSetDfltChild
Set dependent foreign key columns to a default value when the primary key columns in the parent are modified.

OnUpdateSetNullChild
Set dependent foreign key columns to NULL when the primary key columns in the parent are modified.

OnInsertChildRestrictParent
Don't allow an insert to be performed on a dependent table if the foreign key does not exist as a primary key in the parent.

OnUpdateChildRestrictParent
Don't allow an update to be performed on a dependent table is the foreign key does not exist as a primary key in the parent.

Use Built-in Macros: 6 The Trigger Wizard has several built-in macros to ease the construction of a trigger. Except where noted, these macros can only be used in the body of one of the referential integrity macros described above.

ChildTableName
The name of the dependent table in a relationship.

ColumnUpdateTest(*Conjunction*)
Creates a test condition comparing column values before and after the table is modified. This should only be used for SQLServer dialects.

Parameter	**Description**
Conjunction	If a multi-column key is used, this parameter determines how the column comparisons are grouped. Acceptable values are *and* or *or*.

ColumnValueTest (*CorrelationNameOld, RelationalOperator, CorrelationNameNew, Conjunction*)
Creates a test condition comparing a pair of columns in either the parent or dependent table.

Parameter	**Description**
CorrelationNameOld	The name of the table. Depending on which table is being examined, either the ParentTableName or ChildTableName macro or a correlation name should be used.
RelationalOperator	The operator used to compare the columns. Any relational operators are acceptable.
CorrelationNameNew	The name of the table. Depending on which table is being examined, either the ParentTableName or ChildTableName macro or a correlation name should be used.
Conjunction	If a multi-column key is used, this parameter determines how the column comparisons are grouped. Acceptable values are *and* or *or*.

ForeignKeyColumnNameList(*CorrelationName*)
Comma delimited list of foreign key column names in key order.

Parameter	**Description**
CorrelationName	The name of the dependent table. Either the ChildTableName macro or a correlation name should be used.

JoinParentChild (*ParentTableName, RelationalOperator, ChildTableName, Conjunction*)
Creates a join condition that is used to compare the unique columns in the parent table against the foreign key columns in the child.

Parameter	**Description**
ParentTableName	The name of the owner table. Either the ParentTableName macro or a correlation name should be used.
RelationalOperator	The operator used to compare the columns. Any relational operators are acceptable.
ChildTableName	The name of the dependent table. Either the ChildTableName macro or a correlation name should be used.
Conjunction	If a multi-column key is used, this parameter determines how the column comparisons are grouped. Acceptable values are *and* or *or.*

JoinParentParent (*CorrelationName1, RelationalOperator*,*CorrelationName2, Conjunction*)

Creates a join condition that is used to compare the parent table against itself.

Parameter	**Description**
CorrelationName1	The name of the parent table. Either the ParentTableName macro or a correlation name should be used.
RelationalOperator	The operator used to compare the columns. Any relational operators are acceptable.
CorrelationName2	The name of the parent table. Either the ParentTableName macro or a correlation name should be used.
Conjunction	If a multi-column key is used, this parameter determines how the column comparisons are grouped. Acceptable values are *and* or *or*.

ParentRelationshipName
Foreign key relationship name from the parent perspective.

ParentTableName
The name of the owner table in a relationship. This table has a primary key that is used as a foreign key in the dependent table.

SetForeignKeyDefaultValueList (*ChildTableName*)
Comma delimited list of assignment statements required to set foreign key columns in the dependent table to default values.

Parameter	**Description**
ChildTableName	The name of the dependent table. Either the ChildTableName macro or a

	correlation name should be used.

SetForeignKeyNullList (*ChildTableName*)
Comma delimited list of assignment statements required to set foreign key columns in the dependent table to NULL.

Parameter	**Description**
ChildTableName	The name of the dependent table. Either the ChildTableName macro or a correlation name should be used.

SetForeignKeyCascadePkey(*ChildTableName, ParentTableName*)
Cascades primary key column values from the parent into foreign key columns in the child.

Parameter	**Description**
ChildTableName	The name of the dependent table. Either the ChildTableName macro or a correlation name should be used.
ParentTableName	The name of the owner table. Either the ParentTableName macro or a correlation name should be used.

TriggerName
The name of the trigger being created. This macro is available throughout the body of a trigger.

TriggerTableName
The name of the table for which the current trigger is being created. This macro is available throughout the body of a trigger.

UniqueKeyColumnNameList(*CorrelationName*)
Comma delimited list of the unique constraint column names in key order.

Parameter	**Description**
CorrelationName	The name of the owner table. Either the ParentTableName macro or a correlation name should be used.

Combine Generated Trigger Code with User Triggers:

7 If you have defined triggers in the repository for any entities that are used with the Trigger Wizard, you must insert the following directive into the body of the trigger:

```
—VA_Extension InsertTrigger
```

This indicates to the Trigger Wizard where the generated referential integrity code should be placed. If this directive is not used, an error message is issued during SQL generation and no referential integrity code is produced.

When a trigger is produced by the Trigger Wizard, it has the following form:

```
CREATE TRIGGER trigger-name
Begin
Trigger Preamble code
—VA_Extension InsertTrigger
—VA_Extension begin
referential integrity constraint code
—VA_Extension end
End
```

Note

The macro facility of the Trigger Wizard is available to all trigger bodies, even those that are stored in the repository. A macro defined within the body of an existing trigger is visible only within the given macro body. To extend the visibility of a macro to all trigger bodies, place the definition of the macro in the [Template global] section of the TRIGUSER.TBL file, or if you want to limit it to particular an SQL dialect, place it in the [Template dialect] section.

Index