Visible Developer - Value Proposition

Develop Applications Rapidly, Not Recklessly. Why is a prototype frequently thrown away? Probably because minimal effort was expended on design and architecture (the most enduring qualities) and the most time was spent on the user interface (the most appealing quality).

Visible Developer's business objects provide a rapid and well-constructed start to a business application. A large amount of time and effort are invested in the design of generated business objects. They demonstrate good programming practices and sound architectural structure. Generated business objects are prototypes only in the sense that they are not the final product. They represent the starting point of the evolution toward the final business object.

With Visible Developer you quickly turn prototypes into working business applications.

Visible Developer Generates Components That Can be Packaged to Create a Single, 2-tier, or Even a 3-tier Architecture Without Changing a Single Line of Code. The building blocks are created for you; you decide how to assemble them. Grow your business application over time without rewriting a single line of code.

Visible Developer can generate a single executable, a 2-tier or a 3-tier application. The first choice places all the objects in a single, monolithic project – a good first step for newcomers. The other choices divide objects into layers according to the following scheme:

- **User Interface Layer** – contains all the forms that participate in the project. Objects in this layer know nothing about business rules or data access or database schemas. What they do is communicate to the user; forward requests to the second layer, and return the results of those requests to the UI.

- **Logical Business Object Layer** – contains the business objects which know all about business rules and data validation and know nothing at all about either the UI or the data access mechanisms beneath them in the layer three.

- **Physical Business Object Layer** – contains the persistence objects that read from, and write to the database. The objects in this layer know all about the physical storage underlying the application including: column and table names, relationships among tables and referential integrity constraints.

Focus Your Effort on Adding Business Value. How much time does your programming staff spend solving technical versus business issues? How much of the code in the final application is actual business logic as opposed to code required to move data between application layers? Precise figures aren't available, but intuition and experience suggests that we spend far too much time sweating the technical details to the detriment of the business solution we are striving to develop.

And the bad news is distributed architectures have many more details. But let Visible Developer provide those details for you. The code generated by Visible Developer is the predictable, but absolutely necessary, code required to implement distributed business objects. It is code you would normally have to write but now you don't. Your starting point is a Visual Basic 6 or Visual Studio .NET project with forms, standard modules, and classes that combine to create a 3-tier business object. You are free to concentrate on adding business value, which is what software development is really about in the first place.

Scale Business Applications without Rewriting Code. One size seldom fits everyone and the same is true when it comes to business applications. And, to continue the analogy, even if it does fit today there is no guarantee it will in the future. Building a business application once is an expensive proposition, so designs that require us to rewrite even a small portion of an application are not acceptable.

Reduce Time Needed to Learn Distributed Application Design. Courses and books are useful training methods but they do have one shortcoming: examples from books and classrooms seldom apply to your business application. Theory and guidelines are great, but nothing beats putting them both to work on your business application. Visible Developer does that for you. Visible Developer offers a degree of tailoring no classroom or book can possibly match because the examples aren't really examples; they are working code based on your database schema.

Visible Developer training begins with clearly documented and executable code implementing your business data in a distributed architecture. If a programmer wants to know how a user interface interacts with the logical business object, just step through the generated code line by line. It's like having a training course developed around your application!

Make Maintenance Easier and Predictable. When programmers are artists, software maintenance consumes large amounts of resources for "restoration and preservation" of their masterpieces. Business objects generated by Visible Developer are closer to mass produced manufactured goods rather than individual artistic creations. But they are beautiful to behold if you are responsible for software maintenance.

Easily Reuse Business Logic with New Interfaces

The user interface is the most tangible and visible aspect of your business application. It is also the one most likely to require changes. One business application may have interfaces to accommodate technology ranging from web browsers to character-based terminals. Even if your technology base is uniform, multiple styles of interfaces might be required to meet the needs of different users ranging

from sophisticated analytical tools for experts to simple point-and-click designs for novices.

The one constant in the equation is the underlying business information, rules, and processes; in other words, the business objects. Visible Developer's business objects work with all styles of user interfaces.

Visible Developer can generate different UIs on the same objects (WinForms and ASP.NET). Users can write custom code patterns to generate other types of UIs or extend the UIs that are supplied.

Learn and See How Distributed Business Applications are Designed. The quickest way to learn distributed application development is by examining working code. But why settle for sample code about a fictitious business when the code you learn form can be written for your application? The classes generated by Visible Developer are extensively documented with explanations.

Use a Variety of Interfaces. Visible Developer business objects work with any user interface that can create ActiveX components. Develop your user interface in a variety of languages, (C++, Java, Visual Basic, Delphi, etc.) or desktop applications (Word, Excel, any product with VBA), or even web browsers.

Import Existing Database Schemas. The foundation of any business application is the database design. You work hard to create it, so Visible Developer makes certain all of the information contained in it is utilized. Using the latest version of Microsoft's ActiveX Data Objects (ADO), database design information (tables, fields, data types, keys, etc.) is extracted and stored in a local repository. Generated code matches your database design.

Add Design Information. Visible Developer gives you control over how code is generated. Information from the database schema is augmented with application design information to make the generated code fit your requirements as close as possible. Pick the type of control used to display a database field and provide a label displayed next to it. Define business operations that the business object will provide. Identify relationships among business objects and specify container and contained objects.

Create a User Interface. Forms complete with controls and code are generated to provide a working user interface that creates, updates, displays, and deletes business objects. The generated forms demonstrate techniques for working with business objects and provide a starting point for more elaborate user interfaces.

Synchronize After Changing Database Design. Added a new field? Changed the key? Discovered new tables? No problem, just synchronize your application model with the new database schema and regenerate the code.

Preserve Changes to Generated Code - A Major Benefit. As long as new statements are added between provided edit points, all of your custom code is saved and inserted in the correct place when you regenerate. Edit points are clearly marked points in the code, where you might want to place customizations or embellishments. The comments even suggest what sort of code should go there. This becomes significant on regeneration of your objects. Any custom code added at the edit points can be preserved and incorporated in subsequent generations. This feature allows developers to work iteratively, refining their objects little by little and regenerating the code several times in the course of a project.

Use Guidelines to Extend Generated Code. You will need to add business logic to extend the code generated by Visible Developer. To make this task easier, numerous clearly marked edit points are included in the generated code. Notes prior to the edit point explain the types of changes typically made. A companion handbook explains how to extend the generated code for common situations.

Recognition of Contained Objects Another significant benefit of Visible Developer is its recognition of contained objects. A business object is not just a single collection, but also one that can have child collections of directly associated records. When adding a join to a business object, where there are many records associated by that join, the result is a new child collection in the business object. For example, you can have a Company object with a child collection of Purchase Orders, even though both entities are fundamental and stand on their own.

Code Patterns Generate the Application Framework. Eases the Transition to .NET with Developer's template-like code patterns which enable you to reuse existing logic which can save you a rewrite. You can develop once and use the business logic in different configurations without changing a line of code.

Deploy a Single, Two, or Three Tier Application. Package the generated business objects to fit your requirements without changing a single line of code. Create one executable containing the user interface and all business object classes for a single tier architecture. As your requirements expand, repackage the classes to create a distributed architecture.

Interface to Front-end Development Tool. Visible Developer can export UML class definitions of your business objects which can be imported into Visible Analyst. You can also take class diagrams from Visible Analyst and export them into you Visible Developer Model Schema.

Visible Systems
201 Spring Street
Lexington, MA 02421

1-800-6VISIBLE

www.visible.com

Download a Free Evaluation Copy Today.

Visible

Integrated Enterprise Application/Code Generation