University of Massachusetts

Medical Center - Success Story

Whenever software is completed on time there is cause for celebration. When the project is the first 3-tier application built with business components ever attempted by the development team, then a larger celebration is required! Consider what a team of six developers at the UMass State Laboratory Institute accomplished in three months:

- Learned how to build 3-tier business components (none of the team had put a 3-tier application into production prior to this project)
- Developed 23 business components each separated into a business logic layer and a database access layer (46 DLLs in all) based on a database design of 88 tables
- Created 51 forms using a MDI style interface. Forms use the business components for all business rule validation and database access
- Started coding three weeks behind schedule and finished on time

The business components totaled over 180,000 Visual Basic statements (notes and white space not included). Considering that coding did not start until the beginning of the second month, if the six members of the team developed all of the code by themselves they would average 15,000 lines of code per developer per month. However, there was another team member Visible Systems' Visible Developer – and according to the project leader, it generated 95% of the code in the business logic and database services layers. That means the development team could focus on the 9,000 lines of code that addressed unique business requirements and provided the business value needed by the State Laboratory. Visible Developer generated the remaining 171,000 lines of code freeing them to concentrate on the business issues instead of the intricacies of distributed business components.

The key ingredients of this success story:

- Management commitment
- Strong technical leadership

201 Spring Street • Lexington, MA 02421
(800) 6VISIBLE • Fax (781) 778-0208
Email: sales@visible.com • www.visible.com

Visible SYSTEMS CORPORATION

- And, of course Visible Developer

Background

John Munies is the Deputy Assistant Vice Chancellor of Information Systems at the State Laboratory. His top priority upon assuming his position was to formalize the process of software development throughout all departments. Software development was "project centric" with each new project approaching analysis and technical design in a different fashion. There was no campus-wide approach to software development and no formal way of measuring progress. John's first task was to institute a life-cycle with explicit stages of development so teams would share a common way of planning and managing software projects. John describes this first step as "defining the view from 50,000 feet."

The next step created a standardized approach for developers to accomplish the tasks in the life-cycle. John said, "We needed a forum for communication; a way to improve communication between projects and to share expertise without imposing too much structure. Until now, commonality between projects was achieved with a similar style of user interface. But, if you drilled down into the code, the similarity quickly ended and it became very difficult for other developers to maintain projects they didn't work on. Turnover is increasingly an issue, particularly with public sector organizations, so the ability to leverage staff across projects in crucial." John saw Visible Developer as a way to achieve the next level of detail in the definition of the State Laboratory's software life-cycle. The viewpoint shifted from 50,000 to 1,000 feet in a hurry!

Bob Tumposky, Director of Computer Sciences, oversees a staff of eight full time developers that is currently augmented by six contractors. Bob reports to John Munies and you could describe Bob's job as taking the viewpoint from 1,000 feet to ground level.
Bob agrees with John's assessment that the biggest long-term value provided by Visible Developer is simplified maintenance. In the past, each project developed systems using different designs and techniques. Two client/server applications shared general architecture but the underlying structure and design varied from programmer to programmer. Software maintenance, never a popular task, became even less popular.

A closely related benefit of the common structure created by Visible Developer is that it makes it easier for projects to share ideas and help each other solve technical problems. Bob Tomposky said, "Visible Developer provides a common grammar and shared methodology that improves communication among developers. Visible Developer accelerates development, but as with any

software project, success requires strong team leadership. It is important to have an advocate with technical credentials to direct the team."

Rich Carr, the team leader for the Newborn Screening project, provides the advocacy and technical credentials required by Bob Tomposky. Rich is a MS Certified Professional and MS Certified Trainer with over ten years of professional experience. He discovered Visible Developer when searching for a 3-tier business object design to use for the project. "I was struggling to come up with a consistent, repeatable, and best practice 3-tier solution for Newborn Screening. Visible Developer cam along at a perfect time.

Newborn Screening

Newborn Screening is a legacy application developed over a ten-year period with FoxPro. The Newborn Screening system stores laboratory tests performed on babies born in Massachusetts and four other New England states. Approximately 150,000 new babies are added to the system each year and 25 or more clinical tests are recorded for each one. Integrity of information is extremely important because doctors use test results to identify serious defects that are treatable if detected shortly after birth.

As it grew over time, the lack of a unifying design became more evident and the system was increasingly more difficult to enhance and maintain. The accumulated affect of years of different programming styles and a non-relational database design created an inflexible system. Additional contractor resources were brought on-board to help.

A client/server architecture was planned for the initial release of Newborn Screening, but management wanted the option to shift to a web-based design in the future without having to start over again. A 3-tier architecture would provide the flexibility to meet the current need and would easily adapt to a web-based solution in the future. Without experience with developing distributed applications, management was hesitant to make 3-tier design a firm requirement.

Getting Started: Requirements Analysis

The first four months were spent in business analysis and prototyping. Some members of the team attended a training course in requirements analysis using the Unified Modeling Language (UML). The development team used some UML analysis techniques, primarily Use Case Analysis, Scenario Traces, and Scenarios, but did not develop formal models of the application. A UML

201 Spring Street • Lexington, MA 02421
(800) 6VISIBLE • Fax (781) 778-0208
Email: sales@visible.com • www.visible.com

Visible SYSTEMS CORPORATION

modeling tool was evaluated but the team opted to record analysis using a simpler tool they built with Visual Basic, MS Access and Crystal Reports.

Rich Carr said, “The UML modeling tool would allow us to express a 3-tier design with its diagramming tools, but we would have to define the classes, properties, and methods in each layer. We didn’t have an experienced-base with 3-tier designs, so we were looking for more than assistance with diagramming. We would need to solve all of the difficult technical issues ourselves.”

The team developed a Visual Basic prototype concurrently with the business analysis task. User Interface design received the emphasis and developers were instructed to “get data on the forms the quickest way possible.” The plan was to preserve the Visual Basic forms at the end of the prototyping phase, but to throw away the database access code behind them and replace them with a standard design.

What would the standard design be? Rich Carr researched the most popular VB web sites for information on 3-tier architectures. Rich said, “I included one week into the project schedule to define the architecture for Newborn Screening. Fortunately, Visible Developer was featured on DevX and we downloaded a sample 3-tier application it generated. Rich spent a week dissecting the code looking for holes. After close examination, Rich decided he had found their standard 3-tier architecture, and as an added bonus the generated code would give the project a big jump-start.

The next step was critical, although it seems obvious after the fact. Instead of immediately starting code development, the team took an entire month to study the code generated by Visible Developer and to plan how they would use the tool. A month might seem like a long time, especially when faced with a development deadline just three months away, but it proved to be a wise investment of time. Furthermore, during that month, most team members were introduced to object-oriented, business components and 3-tier architectures for the first time.

The days were spent in internal classes, team discussions, working through Visible Developer tutorial, and many hours pouring over generated code. Most developers on the team had never written a VB class before so the code contained many new concepts and programming techniques. During interviews for this case study, developers unanimously agreed that the month spent studying the code was necessary and critical to their success. When asked to estimate how long it would take before a developer with their background would

201 Spring Street • Lexington, MA 02421
(800) 6VISIBLE • Fax (781) 778-0208
Email: sales@visible.com • www.visible.com

Visible SYSTEMS CORPORATION

be proficient in building 3-tier applications with Visible Developer, the common reply was one month.

The following month the team started developing code, they were three weeks behind schedule. However, they had a standard 3-tier design to follow and had carefully mapped how they planned to use the code generated by Visible Developer. The up-front training and careful planning would enable them to make up the schedule deficit and still finish on time. The plan revolved around creating the core objects that other objects would need to reference first. Without this approach, the developers would constantly be waiting for others to finish or even start critical components. Rich Carr said, "We developed what I call a pyramid approach, carefully figuring out what components belonged at the bottom. Out last business component referenced seven components that had already been created before work on that one even started."

Rich Car created a few rules at the start of development that proved to be important as development progressed. "We started with the assumption that we would only change Visible Developer's business and database access layer code if absolutely necessary. Our security model, for example, required changes in a few places. Each change was clearly documented with explicit instructions. We made sure that when developers touched the business logic and database services layers they did it in the same way." Rich's insistence on solving problems within Visible Developer's design decreased the amount of code that developers had to write. He calculates that over 95% of the final delivered code in the business logic and database services layer was generated by Visible Developer.

The user interface for the Newborn Screening application is very different from the standard forms generated by Visible Developer. An MDI style of interface is used for important business objects and the Visible Developer's List and Detail forms are used for maintenance of reference data. The team preserved the forms generated for the prototype but completely replaced the code behind each form.

A frequently used technique involved generating forms for a business object with Visible Developer and then replacing controls on the generated forms with the controls from a similar prototype form. The types of changes required for the new UI were planned in advance and provided to each developer as a checklist to follow when creating the UI logic. Rich estimates that over 50% of the form logic generated by Visible Developer was reused.

201 Spring Street • Lexington, MA 02421
(800) 6VISIBLE • Fax (781) 778-0208
Email: sales@visible.com • www.visible.com

Visible SYSTEMS CORPORATION

The Newborn Screening application was divided into 23 subsystems or business components that were assigned to developers. Each subsystem was packaged as three separate VB projects: UI, business logic, and database services. Between one and six database tables were included in each subsystem. As subsystems were completed, the forms were collected into a single project and packaged as a standard executable.

Developers followed these steps when developing a subsystem:

- Import the tables required for the subsystem into Visible Developer.
- Generate the business objects and standard forms.
- Add stored procedures and test once again to validate that the basic create, read, update, and delete transactions work.
- Use the standard forms to thoroughly test the business logic and data services layer – before making changes.
- Implement the changes outlined in the checklist of layers two and three.
- Connect prototype UI and make it work with the generated business objects.
- Deliver UI project and 2 DLLs (business logic and database services) for integration test.

The area that required the most relearning by developers was the separation of all business rules, including basic data validation, from the User Interface. "It was difficult to break the habit of adding logic to the UI. Often the most direct and quickest solution was to add code in a control's lost focus event. But that is contrary to the design goal and would create maintenance problems later, especially if the system evolves to a web-based design throughout the project."

Keys to Success

Success in software development requires doing many things right. There is no "silver bullet." Visible Developer played an important role in the Newborn Screening project, but a tool doesn't guarantee success; at best it makes the difficult job of software development easier. The UMass State Laboratory Institute deserves the majority of the credit because they were able to introduce a new technology and make it work for them. Their

experience offers important guidance to future projects attempting the same undertaking:

- Obtain Management Commitment **- John, Bob, and Rich saw the potential benefits offered by Visible Developer and took the required steps to make it happen. Even when an entire month was spent training the team and the project started three weeks behind schedule.**
- **Find a Strong Advocate – A new initiative is more likely to succeed if it has a strong advocate. For Newborn Screening, Rich Carr was the chief advocate and principal enthusiast. It always helps to have an advocate (even if he or she has to be appointed) who has the desire and incentive to make it work.**
- Understand the Business Problem **– Visible Developer is a code generation tool so it addresses just one portion of the software life-cycle. The Newborn Screening system is successful because it satisfies business user requirements. The up-front analysis created a solid UI prototype and database design.**
- Plan Before Coding **– The Newborn Screening team took an entire month to study Visible Developer's 3-tier design and to understand how they could use the generated objects with the prototypes UI. When they started coding, they had detailed checklists for the developers to follow.**
- Stay Within the Design **– "Code within the box" sums it up nicely, according the Rich Carr. Adhere to the 3-tier design and avoid the temptation of quick fixes like combining business logic with UI controls.**

UMass State Labs is already planning to expand the success of the Newborn Screening system to five other projects.

201 Spring Street • Lexington, MA 02421
(800) 6VISIBLE • Fax (781) 778-0208
Email: sales@visible.com • www.visible.com

Visible SYSTEMS CORPORATION

Next Steps

With the success of the Newborn Screening project, John, Bob have made Visible Developer the standard development tool for Visual Basic. Five more projects using Visible Developer are slated for completion within a six-month period. The State Laboratory is not resting on their laurels; they introduced the new life-cycle and Visible Developer to other development teams and have gleaned valuable lessons from their experience with Newborn Screening.

New technology, especially a code generator like Visible Developer may receive a lukewarm reception by the development staff and could even be considered as a threat. To address these concerns, Bob Tomposky gives his developers adequate time to appreciate the 3-tier design and business object concepts generated by Visible Developer. Bob said, "I try to communicate the positive benefits to developers, emphasizing the the code generator automates the boring stuff and allows human beings to do what we do best – solve unstructured problems. Standardization is important for the larger good. You may feel it impacts your creative style, but consider the benefits when you have to maintain someone else's code!"

Conclusion

Corporate information technology organizations and commercial application vendors' view distributed business components as strategically significant. A Gartner Group study indicates that over 60% of new software development projects plan to implement distributed business components. It is widely reported that projects struggle to implement these new technologies because they place a high premium on software design and architecture skills and not commonly found in the majority of "code-centric" organizations.

Visible Developer simplifies the hard task of distributed business component development. A committed management team with a sound implementation plan receives the best return on their software tools investments. The combination of good tools, developers and management results in success.

201 Spring Street • Lexington, MA 02421
(800) 6VISIBLE • Fax (781) 778-0208
Email: sales@visible.com • www.visible.com

Visible SYSTEMS CORPORATION