Selecting a Best-In-Class Technology: When Enterprise Solutions need Collaboration, Adaptability, Robustness and Supportability

By: Visibility Corporation

Synopsis

Visibility Corporation is conceivably the first ERP application developer to release an enterprise class application designed from the ground up on the Microsoft©.NET architecture. The objective from the onset for the VISIBILITY.net ERP solution was to develop a robust integrated business enterprise system that would take full advantage of what the latest technology could deliver such as: increased efficiencies, optimized business procedures, improved access to high volumes of enterprise data, easy integration of multiple business entities and languages and a platform easier for in-house technical support. While effectively replicating and significantly extending the functionality of its predecessor product, VISIBILITY.net contains no legacy code. This "no previous baggage approach", enabled Visibility to take full advantage of the .NET framework, providing a system that can be accessed using a web-browser with a user interface equal in functionality to a client-server application.

All of this is achieved with a so called 'Zero-Client' interface. In other words, VISIBILITY.net does not download any software onto the client PCs (workstations) in order to operate. Any client PC with Internet Explorer 5.5 or later is capable of operating VISIBILITY.net without compromising on the quality of the user experience. VISIBILITY.net is deployed as a true internet application – allowing system administration without concern regarding the client side (workstation) hardware and no installation on the client device. In doing so, ubiquitous access is provided for users inside the four walls of your organization as well as for users at remote offices and mobile locations.

Most importantly it provides huge benefits in terms of total cost of ownership. No IT support is required to install VISIBILITY.net on a user PC, no client or workstation installation is necessary and client-side application upgrades are never a concern. In fact the more PCs an organization has, the greater the financial benefit which their business will realize when implementing VISIBILITY.net.

This white paper reviews the major factors considered by the Visibility Technology Team along the road to deciding the proper technology for an entire new product generation. Additionally, this long term decision path provides insight to the team's ultimate choice of technology in the context of today's market and to provide an informed comparison with the technology of other ERP products.

Background/Need for Change

The VISIBILITY ERP application was originally developed in 1989. It was one of the first ERP packages designed specifically to satisfy the needs of To-Order manufacturers. At that time, Fourth Generation Languages (4GL) were the technology of the day. VISIBILITY was originally developed on the Cognos PowerHouse 4GL framework, on Hewlett Packard ("HP") hardware using an HP proprietary file system. Over the next 15 years the Cognos framework continued to support the development of the VISIBILITY product, providing a platform that allowed the product to evolve into a business system capable of supporting small, mid-size and large multi-site manufacturing organizations. Visibility Corporation

(Visibility) subsequently created versions of the VISIBILITY application that would operate on several different hardware platforms, support UNIX operating systems, utilize the Oracle relational database, support n-tier implementation and provide a client-server interface. In doing so created a scaleable, robust platform that could cost-effectively support from 10 to more than 1000 concurrent users 24 hours a day, 7 days per week. Today, VISIBILITY ERP continues to operate successfully in hundreds of To-order businesses worldwide in Release version 6.4 or earlier based on the PowerHouse framework.

Technology Objectives

Visibility's long term goals for its product were defined to reflect the desired business and technical needs of its mid-size To-Order manufacturing target market. They are:

- Low Total Cost of Ownership
- User-friendliness and easy access to business data
- High reliability and scalability
- Easily installed and maintained
- The capability to integrate easily with other business critical systems
- The ability to be modified to customers' existing processes without the need for custom coding
- Development based on leading technologies which can best assure application stability and long term evolutionary potential
- Integration with the office desktop and mail systems for document processing, and workflow notification to support internal business processes
- Incorporation of all existing functionality found in VISIBILITY 6 and provision of a number of new functions designed to add additional power and efficiencies.

In 2001, with these objectives in mind Visibility Corporation made the business decision to upgrade or re-write the entire VISIBILITY system in a new technology framework. This would be accomplished either by a complete re-write or by upgrading the existing code. The resulting application must be able to deliver the requirements listed, and provide clients with a strong ROI through enhanced, value-added functionality and collaborative capabilities which would enable an efficient, adaptable and supportable business solution. Rather than evolve a solution, by adding application functionality to VISIBILITY 6 using new technologies, Visibility made the decision to completely re-write the technology. With the objectives that Visibility wanted to achieve, a hybrid solution of new technology and old was very limiting and carried additional mixed bag support implications associated with new and legacy technology.

Business functionality of the existing (at the time) ERP product was highly successful, well liked by its users and provided excellent To-Order capabilities. So VISIBILITY 6 (release at that time) provided a good functional baseline for the future product blueprint. However, there were a number of major decisions to be made regarding the technological framework for the next generation application. What technology platform should the company use for its new product? Should the new application be a client-server solution, a web application, or even a hybrid? Which technologies should be selected for use in application development: Java, Visual Basic, C#, Oracle Developer or some combination of Active Server Page / HTML web form coding?

Technology Review

To answer these questions Visibility formed a team (Team) consisting of technology architects and senior developers. Over the next year this team reviewed all of the major contemporary technologies and sought the opinion of industry watchers and technology research firms in order to be sure that they would reach the right decision in support of the long term.

Selecting a Platform for the Future

Pre Dot Net (".net") – ActiveX

The first concept that the Team considered was hybrid technology which involved using client-server computing concepts within a web-browser user interface.

Up until 2001, one of the most common hybrid technologies was to incorporate the use of Microsoft ActiveX components into application web forms. ActiveX provided a Windows style of user interface which could be run from within a web browser. This enabled the creation of highly functional and interactive user interfaces, but each ActiveX control had to be downloaded to the client browser. The ActiveX approach was one option that Visibility reviewed in some depth. At that time Microsoft was promoting the use of ActiveX as an element of its DNA architecture which also included Microsoft Transaction Server (MTS), the Common Object Model (COM), for client to server communication, and Active Server Pages (ASP) for web-based applications development.

Given that Visibility's objective was to become the platform of choice for Mid-Size To-Order companies and that it already had a substantial user base, ease of supportability was an important consideration. Common to businesses in the mid-size market Visibility serves, is having a limited technical staff and a small budget for IT. This places an emphasis on system supportability and lower total cost of ownership (TCO). The team was concerned that the ActiveX architecture would cause continuing implementation and support headaches.

When an ActiveX application is used in a browser, the application is first downloaded to the PC. In essence ActiveX controls are an applet with the benefit, that they can be written using a variety of Windows programming tools. Visibility's technical architects were concerned that there were some practical issues that needed to be addressed before contemplating an ActiveX architecture.

The first of these was the communication between the browser application and the server. At that time, Microsoft was using Remote Data Services which had been provided to execute both read/write database record queries and remote procedure calls which were used to run application code deployed to an application server. The Team's benchmarking experiments determined that the performance achievable using this approach was less then acceptable for an enterprise class ERP application such as VISIBILITY ERP.

The Team's biggest concern was with respect to maintainability. They concluded that running an ActiveX application was effectively the same as installing Dynamic Link Libraries (DLL) on each client device. It required the installation of at least a part of the application on every client PC. As with Windows applications which relied on the use of application DLLs, this approach frequently caused problems. There were frequent incompatibility problems with the different client operating systems. Each installation of a new DLL or ActiveX download added the challenge of affecting version management. Constant checking was required to assure that the installation or download was not overwriting a newer DLL or ActiveX control. From a maintenance stand point requiring the download of ActiveX components was the equivalent of installing a client-server application on a desktop. It had all the same support headaches that were present with client-server, the only advantage being that to install the application would not require the users to insert a disc; the browser would download the software for them.

The Emergence of ".NET"

In 2001, before the team had completed their evaluation of ActiveX and its related technologies, Microsoft announced the new "Dot Net Framework", which became known simply as ".NET". This new technology platform had the potential to have a huge impact on software companies that were already part way through major development projects using the Microsoft DNA architecture.

At that time the Visibility Team had not yet selected the next generation technology platform and was therefore well positioned and educated to consider the adoption of the new .NET Framework. The major alternative was the Java EJB framework. There were a number of similarities between the .NET and Java EJB frameworks as they were both run-time, server-based environments.

Technology Selection - User Interface

The emergence of Microsoft© .NET Framework caused the Visibility team to re-start its technology evaluation to a major degree. There were several decisions to be made, including the choice of application server technology, which databases to support and the reporting tool technology. The first decision, one that would have a significant impact on maintainability and the user overall experience, was which user interface technology to use.

Was it going to be web-based, or client-server? If it was going to be web-based, then which web technology should be utilized? Was it going to be HTML or was it going to be a Java applet running in the browser, a technology direction which at that time had already been used by Oracle Applications? With enterprise applications built around Java each form which is required by the client requires the download of a Java applet which runs on the client PC and communicates with the server. This is in effect a variant of a client-server implementation. The Java applet is downloaded through the browser and is installed on the client PC or workstation. For Windows client PCs the installation of a Java Virtual Machine ("VM") or a Java plug-in is also required because Microsoft no longer implicitly supports

Java within its browser. For Visibility's clients this would have meant that a system manager would need to ensure that their entire user base of client PCs was up to date with the required Java VM or plug-in installed. Java offered several potentially attractive features worthy of consideration. It offered a rich set of components which could be used to define a highly functional user interface. It could be defined to run on any platform for which a VM had been deployed. Oracle and others were designing and were generally successful in the delivery of business applications which utilized the Java platform.

Rather than rushing into a decision that would affect the future of its customers and its business for many years to come, Visibility sought the advice of industry analysts and technology partners.

One company Visibility sought the opinion of was The Hurwitz Group a highly respected ERP and technology research group which had significant experience working with all the major ERP applications including Oracle, SAP, Baan, and Microsoft Great Plains. Additionally, Hurwitz had consultative experience with many of these firms as it related to current technology and application directions shared by those throughout the ERP industry. The Visibility team posed one particularly important question. For new enterprise class applications is the most appropriate current and longer technology choice: client-server, web-based, Java or Java applet, HTML or hybrid?

Their unequivocal answer was that it had to be Internet technology and it had to be HTML based. In their opinion there was no other choice. They were very firm in their opinion that if Visibility was serious about being competitive in the future and wanted to play in the enterprise market, then its next generation applications had to be HTML based.

That consultative direction left Visibility with several concerns and potential problems to be sorted out. HTML can typically only run one form (screen) at a time, but an ERP application is a complex software product that cannot be easily done on a single page or a single browser frame. To provide a high level of functionality and usability an ERP application needs to be a delivered as an application with multi-document interface ("MDI"). To present the system through a user interface that could open only one form at time would be extremely limiting for an ERP application or any large-scale enterprise application. ERP users typically need to be able to interact with multiple forms and have those forms interact with each other and run reports or execute inquiries at the same time. The Team knew they needed to be able to provide an application with an MDI in order to provide this level of flexibility.

In Visibility's peer group, a number of companies had attempted the implementation of a web-based MDI interface. To the Visibility team these choices provided a high quality user interface, but had the potential to provide major support problems.

Microsoft©.NET vs. Java EJB

Given that the .NET Framework was designed as Microsoft's platform for web-based computing, Visibility's technical team turned their attention to reviewing the Microsoft platform.

Conceptually the Microsoft©.NET Framework had many similarities to the Java EJB framework, it was implemented within a runtime environment and came with a number of supporting classes and frameworks within which applications could be written. Both frameworks provided a server-side application and a client-side application. Oracle Applications demonstrated this with Java EJB applications on the server and a Java applet running in the client browser that communicated with the server.

An alternative approach was to have an application that was all Java on the server and HTML on the client browser. .NET offered a similar type of a framework. .NET applications could also run on the client in the form of WinForms that communicated with an application which was deployed on a server, and could also run in the form of HTML, or standard browser forms. In that respect the ability to implement an application using "n-tier" architecture was virtually identical to the client-server concept and could help to assure scalability.

Java Applets and WinForms

Beginning before 2001, there had been much media hype about Java and Sun and several other companies that had become very Java centric. Prior to 2001, Java was frequently considered to be the easiest choice of platform because it provided a fully scalable framework with offerings from Web Logic, IBM Web Sphere and HP with each development environment provider supplying an EJB engine. Each of these engines provided a platform that could be used to build enterprise level applications.

One of the most important objectives of Visibility's technology strategy had been to provide a system which was supportable by the market Visibility served (mid-market) which meant an objective to driving the Total Cost of Ownership (TCO) as low as possible and to insure the system was easily supported by the limited IT resources the mid-market Companies typically have. To that end, Visibility wanted to avoid, as much as possible, any maintenance tasks on client PCs which required substantial internal IT support of end user devices, as well as avoiding the need to require a powerful client PC to support the application. Assuring that this objective could be achieved meant that Visibility clients would be able to avoid significant desktop capital and maintenance expenditures for each user. Utilizing Java applets required both a VM on the client PC and the download of applets onto every client PC that accesses the Java application. This made the Visibility team cautious of adopting Java applets, mostly because running a Java applet on the front end is memory intensive on the client and requires a download each time a new PC is used or a new release is provide. Visibility's team was also cognizant that Microsoft was proposing to stop supporting Java applets in their browser in the foreseeable future.

WinForms are essentially client-server applications that have to be installed on the client PC, or can be run as a .NET control that gets downloaded and runs within the client browser in much the same way that ActiveX controls required. In both cases, the software is ultimately loaded onto the client PC itself and runs in client-server mode. The concept of WinForms was nothing new; Microsoft had already done the same with ActiveX, but now with WinForms the form and the associated application ran in the new .NET Framework.

To Visibility's team of system architects, WinForms had a development-cost benefit over Java applets. In the team's view the cost to develop a Java application, with the resources, associated overhead and projected limitations, was prohibitive. Additionally, it did not make good long term business sense, to ramp up the whole company to the Java skill level needed when most of Visibility's developers already knew VB, C++ or one of the other languages which were supported for use in association with the .NET Framework. This assessment was made on the basis of projects that had involved Visibility directly with many companies that had developed similar applications. The assessment kept coming back to a factor of 4 as it related to Java development vs. .NET. Visibility assessed that to develop a Java-based application when compared to the effort and costs projected to undertake the same project as either a VB application or a .NET application selecting the use of Java would:

- Be 4 times more expensive;
- Take 4 times longer; and
- Need 4 times as many resources.

For a company like Visibility that has a history of providing an agile response to market demands, a Java decision did not make sense from a business, economic or customer supportability perspective for the Company's target market. Ultimately, with these factors considered the use of Java was ruled out.

This left the Team with two options remaining. Select either WinForms or use HTML forms.

Supporting Complex Transactions using HTML Forms

Although HTML provides a big advantage by not requiring anything to be downloaded, from a programming point of view it is more limited in terms of the functionality of the user interface. Many simple screen handling capabilities that a programmer would use to write a GUI application are not by default available for use when creating HTML forms.

This did not mean that an enterprise class application could not be done using HTML forms and programming. Within the HTML client there are a number of implementation options which do not require additional downloads or component installs on each client PC. These options include DHTML (Dynamic Hypertext Markup Language) and client-side Java Script (a light-weight scripting language that can be encapsulated in HTML). Both of these technologies provide the capability to develop a more dynamic user interface.

However, to provide the type of user interface ("UI") functionality required within a business system means developing some very complex coding, which would have presented a problem in terms of the skill set required to develop the application code. Although the .NET Framework supports a wide range of programming languages, to develop an enterprise business application requires the coordination of a large team of programmers. This typically requires that a standard development language must be adopted. To the technology team, requiring programmers to develop UI functionality using DHTML or Java script and server-based business logic in a .NET language would have required a unique hybrid skill set. They would have needed to understand HTML, Java Script and also be capable of understanding the object module in the browser that the HTML framework uses as well as understanding how each implementation element integrates and interacts with the application business logic. They would need to be experienced in three different languages just to write one application.

With the new .NET Framework, Microsoft had bridged this gap somewhat between writing a client-server application, a regular Windows application and a HTML application. The .NET Framework offered a set of run time classes and GUI classes in a common architecture within which the whole system ran, and a development environment in the form of Visual Studio .NET which could be used to develop applications for .NET. Visual Studio .NET allowed the development of HTML based applications within an environment which was already familiar to developers. The Visibility research team came to the conclusion that the .NET Framework could be capable of providing a platform that would facilitate the development of a product with an HTML interface, but without the need for convoluted coding.

There remained one problem. HTML provided an environment where virtually nothing needed to be pushed to the clients, thereby eliminating the need for client PC maintenance. HTML forms and applications could be developed using the new .NET development tools. The problem was that in its native form HTML applications had a limitation of operating within one form at a time. The Team still needed to determine how to affect the use of an MDI within the HTML browser interface.

Technological Innovation

The exacting requirements set by the company's executive team were proving hard for the technical team to deliver. From The Hurwitz Group and other industry analysts they knew that the application needed to run in a browser, the team had concluded, for supportability and to eliminate client PC maintenance costs that they would need to develop a product that did not require the download of application controls. They also had to bear in mind the primary corporate objectives; Visibility wanted VISIBILITY.net to provide a highly functional user interface and excellent user friendliness in the form of an MDI. The Team was presented with a technical hurdle which had to be overcome. The team's decision hinged upon whether it is possible to overcome the limitations of using a pure HTML browser interface in a highly scalable multi-user application.

From talking to industry watchers, they knew that none of their competitors had been able to solve the conundrum that Visibility had set for itself. Other ERP companies had apparently opted for Java or WinForms solutions, with their associated high development costs and client maintainability issues, or had been forced to develop using dual paths which required a combination of client-server applications and some basic HTML forms. The Visibility team did not want to do that, they wanted one code base and one supportable framework which would be both familiar and accessible by all members of the development team. The team also determined that the implementation must be supportable using industry standards for the long term benefit and protection of Visibility clients.

Major Decision & Major Solution

The research team accepted the challenge to find a solution. Investing in research and development they looked at great depth into the extended functionality that the combined use of DHTML, HTML and client-side Java Script could achieve. They determined that they could achieve an MDI environment running in a web browser using only HTML, DHTML and client-side Java Script. This provided a user interface that can run in virtually any standards compliant commercial web browser such as the latest generations of Internet Explorer, Netscape Communicator and Opera, all without requiring the download of application controls, applets or browser plug-ins.

With this determination the team had been able to prove that it was possible to use an HTML user interface to develop an application that was technologically more advanced than any competitive offering yet would be standards based and simpler to implement, use and administer. There was more work to be done to finalize the technical environment before work could be started on developing application code.

Server-Side Choice

The decisions still to be resolved were the choice of development language for the server-based business logic and the framework that the server code would run in. Having already eliminated Java EJB, they chose .NET as the server-side architecture using ASP.NET as the runtime environment and Visual Studio .NET as the integrated design environment and VB.NET as the principle application development language.

The team decided that Visual Studio .NET without enhancement remained too complex to develop the application, as developers still needed to understand Java script and DHTML in order to achieve the browser-based MDI and provide a functionally rich user interface. In order to simplify the task of developing potentially complex ERP application code, the technical architecture team wrote a comprehensive set of forms and controls which were implemented on top of the standard .NET Framework. Ultimately, the Visibility team added more than 70 controls and architectural framework components to the base Microsoft Visual Studio .NET development environment.

A Development Environment for Enterprise Apps

By adding original Visibility controls to the environment, the team created a development environment that removed the need for the programmers to know HTML, Java Script and DHTML. As a result, for the programmers experienced in developing these applications it was just like writing a WinForms application. Developers would be able to work as if they were writing code using the VB.NET highly interactive application development form. The Visibility technical team had bridged the gap between WinForms development and HTML development by successfully implementing and adding back each of the standard 'Windows like' events that were missing from standard HTML encoded forms.

Dynamic Form Updates

In addition to enhancing the development environment to provide Windows functionality, the technical team also worked out a way to make the screen behavior more dynamic than had previously been possible using a 'standard' browser application. In a standard browser application, if the user changes some of the data whereby the application needs to communicate with the server, the whole page must be submitted and pushed down to the server. The web server must then respond back and the client browser must redraw the whole page to display the resulting changes. This frequently results in a visible page refresh.

Visibility's architects achieved a means of providing the same functionality without requiring the submission of the whole web page to the server after each event. The forms running in the browser session are enabled such that they only communicate changes and update events to the server, so the network traffic is a fraction of what it would be using via a standard web form. The VISIBILITY.net application forms are maintained as objects in memory on the server so that there is no requirement to marshal and un-marshal the state engines on the application web server. The effect of this is that the events and the interaction with the form are effectively instantaneous, and the user does not notice the inherent delays associated with standard web forms. Leveraging this original technology the new VISIBILITY forms behave like a powerful client-server application, but run in a web browser without requiring any downloads to the client PC.

For maintainability, all of the custom web controls that the Visibility team developed adhere to DHTML and Java-script 2.0 standards (3.0 HTML or 4.0 HTML), which all the major browser companies support today.

Maintaining the Visibility originated custom controls has proven to be straight-forward. With each new release of the Microsoft©.NET Framework, the controls are simply re-compiled and are immediately ready for use by the enterprise development team.

Selecting ASP.NET and the .NET Framework has allowed the Visibility technical architects to satisfy another key objective, that of providing a scaleable application. The .NET Framework supports the concept of load balancing and this facilitates the use of server "farms" where an ever increasing number of application users can be supported by simply adding additional application servers.

XML Web Services

The Visibility team found significant additional advantages in its use of the Microsoft©.NET platform. .NET made it very easy to create XML web services – component code objects that can run anywhere and be called from anywhere using standardized XML to communicate. That allowed Visibility to make XML the language for communication of data throughout the system, and because firewalls can be configured to allow for the controlled passage of XML, the VISIBILITY application supports virtually any network hardware configuration imaginable. Visibility's widespread adoption of web services has also had big benefits in terms of re-use and supportability. Many of the core ERP business transactions in VISIBILITY have been developed as Web Services that are called by many different forms. For example, inventory processing, which is at the heart of many business transactions such as sales order shipping, receiving goods and material issues, has been encapsulated in a single web service that can be called whenever an inventory transaction is performed. This means that if Visibility defines an enhancement for implementation within inventory transactions, or if a "bug" is found in transacting inventory, the enhancement or the bug "fix" only needs to be implemented in one place. It also provides the opportunity to deploy appropriate forms and transactions from within the VISIBILITY application onto any of the client devices that are supported under the .NET Framework, including devices such as tablet PCs and hand held devices.

The Advantage of Adhering to Standards

Part of Visibility's objective with the new architecture for VISIBILITY.net has been to adopt a standards-based approach that supports the concept of choice when a user chooses to implement the VISIBILITY.net application. The adoption of HTML is a standard; browser choice is not limited to a single browser client. The VISIBILITY.net application supports virtually any standards compliant web browser. For VISIBILITY users this means that clients can use the VISIBILITY.net application on Linux and Macintosh PCs when equipped with a standards compliant browser.

On the server side, the VISIBILITY.net application can be installed on any application web server which includes support for the Microsoft©.NET Framework. Currently, the recommended application server platform is any version of Microsoft Windows 2003 because those operating systems incorporate the .NET Framework by default.

Database Independent Support

The technical team made the decision to adopt a database independent approach, in part because the VISIBILITY 6 application is designed to run on the Oracle RDBMS as a standard, and Visibility did not want to force its existing customers to abandon their existing investment in Oracle. Adopting a database independent approach also served to provide a choice for companies that have implemented alternative corporate database policies.

The Visibility design team elected to use a layer which would separate the application from the database. This was not a new concept, but Visibility's technical architects created an object based approach that is unique. As implemented, the multi-layered architecture efficiently stores all of the SQL that is used to communicate with the database in XML-based resource files outside of the application programs. This means that there is no SQL embedded in any of the VISIBILITY.net application code, and because of this there is no need for recompilation if the user switches databases. The VISIBILITY.net data layer has built-in intelligence whereby it knows what database language the SQL was developed for and knows how to translate it to the language of the destination database thereby giving it the ability to translate that SQL before it is executed against the destination database. The benefit of this approach is a VISIBILITY system data store can be exported from one database and imported into a completely different type of database without having to change any part of the application software. On completion of the import to the new database the VISIBILITY.net application will be able to connect to it and run. The enterprise application developers only need to write a single set of SQL, and it will run against any commercial database. VISIBILITY.net provides support for both Oracle and Microsoft SQL Server from a single code base.

Enterprise Application Components

Having developed a user interface layer, a development framework and a data layer, Visibility's technical architects set about developing a layer of components designed specifically to address common ERP development issues. The objective was to provide a series of reusable components that would be capable of performing many of the common functions of an enterprise business system thereby allowing the application developers to concentrate the majority of their efforts on developing high performance, efficient business transaction logic.

This collection of components is referred to as the VISIBILITY.net Component Tool Kit. A good example of this kind of component is the VISIBILITY.net Tree component. Many business data structures consist of parent-child relationships. Obvious examples include: bills-of-material, corporate organization structures, or a chart of accounts. For each of these structures a program may need to read down the structure adding up values from each level. Many developers will be familiar with the problem of reporting a parent-child structure; typically they would have had to write a complex stored procedure or looping code. For many reporting tools, producing an indented bill-of-materials is virtually impossible. Visibility's team built a data layer component which allows developers to create and execute record sets against parent/child relationship tables. This allows programmers to write very simple SQL and the tree-reader will do the rest very quickly and efficiently against any brand of database which can accessed using standard SQL.

Other examples of VISIBILITY.net components affected at the enterprise layer include objects to provide data filters, complex screen grids, objects for attaching files to data records and multi-language capabilities. In addition, the base development environment that the technical team has created is metadata driven so many of the screen handling components have built in intelligence that recognizes the data structures being used. This means the application programmers can write forms and applications almost without writing code. With proper metadata definitions recorded to the database, it is possible in certain cases to write large parts of an application module group and affect the associated user experience with security and transaction level validation in relation to form interaction without actually writing any code.

All of these capabilities are designed to allow Visibility to standardize the look and feel of the application and provide an environment for development of robust business transactions. The benefit is huge; to create a new form, the developer has the ability to work within the drag and drop development environment of Visual Studio .NET extended by the VISIBILITY.net components. The developer creates the form the way they want it to look and how they want the behavior to be and the environment does the rest.

The programmers do not have to think about all of the validations on form fields in terms of whether text, dates or number values are to be allowed. The VISIBILITY.net component enterprise architecture takes care of that for them.

Having metadata driven functionality also provides additional benefits for VISIBILITY.net clients because data definitions and validations are not hard coded in the software. This provides Visibility clients with the ability to add additional validations tailored to their business use of the VISIBILITY application forms. This is achieved by simply altering or adding new metadata without requiring a new version of the form or application code.

ERP functionality, Desktop productivity

To enhance all of these elements, the Visibility technical team also created a portal interface designed for maximum ease of use and user productivity. The VISIBILITY.net Portal consists of any number of panels or portlets. The content of each portlet can be defined and controlled by the individual user. For maximum productivity, Portal components can provide instant access to a client's most frequently used transactions, workflow tasks requiring attention, key performance indicators giving online information and favorite web pages.

As a business tool VISIBILITY.net integrates comprehensive To-Order ERP functionality with office productivity tools. For example, users are able to attach an unlimited number of documents to any data record in the system. Whether a client wants to attach a CAD drawing to a part-code, a scanned image to an invoice, or a photograph to an employee record, VISIBILITY provides a simple means of attaching these documents and storing them subject to secured access in the application database.

VISIBILITY.net also makes use of the latest concepts for storing internally-used text information. Users can enter notes against any data record as either labeled comments or in the form of threaded comments. Using threaded comments, notes can be stored in a topical or conversational thread for multi-user collaboration. This is particularly useful for maintaining notes against items that need approvals or for expediters' notes on projects or orders.

Inquiry and Reporting

The final element of the application environment that the Visibility technical architects designed is the reporting and inquiry aspect. Like the entire VISIBILITY.net environment, the reporting (and inquiry) aspect of the VISIBILITY application is 100% web-based.

Once again the team faced a choice: go with a 3rd party reporting product or develop their own? Client-server based reporting engines from companies such as Crystal or Cognos were ruled out immediately. The team reasoned that the whole benefit of zero client footprint implementation and lowest cost on-going maintenance would be lost if a client-server report writer had to be installed on each client PC. Few true web-based reporting solutions were available. The Visibility team also decided that the functionality of the entire application would be enhanced if the reporting and inquiry tool was capable of integrating and interacting with the application forms. That basically left only one choice. Visibility would develop the report writer themselves.

The result is a product that integrates seamlessly with the VISIBILITY.net application environment and one that can operate as a stand-alone product capable of reporting against 3rd party application databases which can be accessed using SQL. Visibility IQPlus is one of the first web-based reporting tools developed on the Microsoft©.NET Framework and was certified as a Microsoft Dot Net Connected product in March 2004. IQPlus allows reports to be utilized in a browser where they can be designed, prototyped and tested as well.

The integration of the reporting tool with the VISIBILITY framework has provided some very pleasing results. The reports are capable of interacting with the enterprise application directly. For example, VISIBILITY has financial reports that can be displayed on the screen, printed, or e-mailed in PDF format; the user can click on an item of information on the screen and drill-through into a more detailed inquiry or even jump to the appropriate maintenance form to edit the information. As each IQPlus report represents an HTML form, the reports are able to contain dynamically generated links to constructed URLs used to access other web forms, or to other reports which may be written to extract data from either the VISIBILITY database or any other database which is available for access from the IQPlus application web server.

Summary

The Visibility technical team finished its research project at the end of 2001. The Visibility application developers immediately began working hard to roll out the business logic for the VISIBILITY.net application. The VISIBILITY.net application was formally released in January of 2006 after having had three βeta test sites complete their testing of the application.

Many ERP vendors have written extensively about industry standards such as HTTP, XML, .NET and web services. Much of this talk is misleading, referring in fact to ‘bolt-on’ technology modules and integration modules. Visibility has truly embraced these standards. Embracing native use of the .NET architecture and object-oriented technology means that VISIBILITY.net can easily use and integrate with other applications by communicating via XML web services. The core of the VISIBILITY application makes use of this technology today without the need for additional middleware or integration applications (no wrappers). VISIBILITY.net ERP provides a common user interface, a single integrated platform and a single software technology based on established industry standards. Truly a technology platform for the long term that supports the objectives established in support of the mid market served.

Contact Us

Corporate Headquarters
Visibility Corporation
10 New England Business Center Dr.
Suite 203
Andover, MA 01810
Phone: (978) 269-6500
Fax: (978) 269-6501
sales@visibility.com
www.visibility.com

European Headquarters
Visibility Europe Limited
11th Floor, Regent House
Stockport
Cheshire SK4 1BS
Phone: +44(0)161 475-0633 or +44(0)161 475-0632
Esales@visibility-europe.com