nVISIA™

Enterprise Architecture and Construction

Changing the Economics of Enterprise Software Development

A Point of View

By

Mark Panthofer and Christopher Rollyson

nVISIA presents this Point of View and solutions in public and private seminars:

Fall 2003 Public Seminar Series

Twin Cities	Milwaukee	Chicago
November 11	November 13	November 18

http://www.nvisia.com/events

Changing the Economics of Software Development

The Slow Evolution of Mission-critical Systems

As 2004 dawns, IT finds itself at a crossroads: the 90s wave of e-business investment mandated pervasive, real-time access to enterprise information—and burdened technology executives with steadily increasing complexity. Most of the corporate "crown jewels," mission-critical information systems, still reside in trusted legacy applications developed with proprietary technology that predated today's network-oriented platforms. While these legacy applications are rock-solid in themselves, they typically were not built to share information very easily, which has frustrated efforts to open them up to the rest of the enterprise—and outside. This situation has led to a wave of proprietary Enterprise Application Integration (EAI) tools in an attempt to retrofit legacy systems into today's networked enterprise environment. While some successes have emerged from proprietary EAI initiatives, their staggering development and maintenance costs have often exceeded initial expectations, and many have siphoned IT funding from new development. Systems integration is the profligate stepchild of new development in today's enterprise environment.

Meanwhile, the rush to the Web has enterprise architecture playing catch up, which has resulted in the creation of enterprise architecture groups.

The Struggle to Adopt New Technology

Over the last decade, corporations have worked diligently to take advantage of emerging software development trends such as object-oriented development[1]; however, their efforts have yielded little improvement in the timely delivery of stable systems that meet functional expectations. Corporate development organizations have been hampered by the deeply ingrained waterfall-style[2] development habits that are more suitable for mature technology. They have often failed to adopt the best practices of iterative development[3] that are critical to managing the risks associated with emerging technology. Meanwhile, the rush to the Web has enterprise architecture playing catch up, and this has resulted in the formation of enterprise architecture groups, which attempt to impose standards and create the architecture to drive new flexible technology infrastructures.

In addition, the Internet has accelerated the evolution of distributed[4] object-based technology because distributed applications' value increases in the presence of pervasive networks. This acceleration has led to intense technology churn—and disillusionment in

corporate IT-because the complexity of distributed applications and architectures typically demands experience and skills beyond those of most developers'.

The good news is that Java 2 Enterprise Edition (J2EE)[5] has emerged as stable, standards-based distributed object platform that is capable of supporting the enterprise. However, corporate IT's struggle to attain the skills needed to succeed with the technology has created frustration and disillusionment to the extent that many IT investments are being questioned-because many applications have failed to deliver on the promise of the technology. Enterprises too often lack any pervasive enterprise architecture, and they lean on IT resources that have insufficient skills and experience to lead their organizations. Symptoms of this are manifested in budget busting IT maintenance costs, which already average 80% of IT spend at many corporations, but often end up consuming nearly 100% of the budget. In an era of little tolerance for IT cost increases, many technology executives have little choice but to pull funds from new development, which stifles innovation and weakens competitiveness.

An Inflection Point for Enterprise Software

Several developments are now converging to enable astute executives to overcome the difficulties of advanced enterprise software, permanently enhancing their competitiveness:

- Software development organizations are starting to grasp the core best practices of iterative software development—after substantial investments in training and on the job experience. They are beginning to develop internal competency with distributed object architectures. At a minimum, developers are recognizing what they don't know.
- Service-oriented architecture (SOA)[6] has driven the industry to put robust distributed application environments within the reach of most IT shops. Distributed applications, when properly architected, have proven to decrease maintenance costs significantly and drive a new model of reuse that markedly reduces costs and time to market.
- SOA, with J2EE and Web services[7], has created opportunities to provide standards-based open alternatives for EAI; it has delivered native integration of applications throughout the enterprise-and has quickly developed into a universal means for deep B2B connectivity. As applications become more distributed, their functionality can be shared much more easily and accessed via open XML[8] protocols and Web services without long-term vendor lock-in.
- Modern integrated development tools and innovative specialized software have matured to the point where they address the corporate developer's needs by facilitating the development of distributed applications with sophisticated architectures.

Gartner and the META Group mandated service-oriented architecture because the maturity of Web services served as a trigger to expose the functionality and information of service-based applications.

The Rise of Architecture-and the Two-edged Sword

In 2002, Gartner and the META Group mandated service-oriented architecture at conferences worldwide because the maturity of Web services served as a trigger to expose the functionality and information of service-based applications:

> *"Web services have potential as a low-risk, high-utility data integration catalyst... Companies should begin to experiment with Web services now, developing pilots for deployment no later than 2003."*

However, many of the details required to achieve these visionary initiatives have lagged behind. Web services' value is largely derived from connectivity, and if applications are not architected for connectivity, Web services are trivial. Hence, service-oriented architecture.

On the corporate front, the IT investment downturn gave corporations some breathing room to take stock of their IT, and they recognized that architecture and standards were woefully lacking. They created enterprise architecture groups to determine and enforce standards on myriad application development groups enterprise-wide.

Yet the majority of consultants and in-house enterprise architects have won their spurs at large technology consultancies and, consequently, they have product-oriented backgrounds. Taking a product-oriented approach within a complex standards-based enterprise environment often doesn't produce optimal results because key (legacy) systems are the result of many generations of custom development, and their idiosyncrasies are not fully addressed by packaged solutions and product-focused approaches.

Asset-based Software Engineering is an emerging, critical facet of software development that is currently practiced by a handful of the Fortune 1000.

In fact, the function of architecture is arguably to abstract away[9] from products and even technologies: it is the strategy and foresight for how the system or environment is to function, taking into account business goals, desired solution components and resources. The demand for enterprise architecture experience woefully exceeds the supply, forcing a long learning curve on enterprises.

Distributed applications, enabled by service-oriented architecture and Web services, represent an increase in complexity for enterprise developers, but the upside is significant and achievable.

Asset-based Software Engineering

Asset-based Software Engineering (ASE) is a crucial mindset that guides the development of distributed applications with pliable, robust, service-oriented architecture. It is an emerging, critical facet of software development that is currently practiced by a handful of the Fortune 1000.

Central to ASE is its organizational focus and emphasis on aligning investments with strategic value by distinguishing between two types of software: "strategic" software is classed as an asset and is closely tied to an enterprise's competitive advantage while "commodity" software is required to run the business (Geoff Moore would say "core" and "context") and is classed as an expense. The business strategy underlying ASE is to create and manage distributed systems' services as assets, attaining ROIs that are exponentially higher than average.

Breakaway results occur with ASE when assets are published as services and managed as assets. Attaining breakaway results requires a balanced approach between the development of a robust distributed computing environment to support run-time services and a process/mechanism that institutionalizes the ASE mindset, which is required to encourage cooperation and sharing among enterprise developer and manager communities.

Changing Economics

Technology and approaches for creating distributed systems have existed for over a decade, but they have been mired in the challenges of immature technology, the classic early adopter phase. The Internet's mass adoption during the 90s has created a pervasive network, which dramatically facilitates the spread of distributed applications[10]: as the demand for interactivity increases, technologies that facilitate interactivity have matured, and standards-based protocols such as J2EE and Web services are a practical means to achieve deep B2B integration without multimillion dollar, vendor-licensed integration projects.

J2EE and Web services are a practical means to achieve deep B2B integration without multimillion dollar, vendor-licensed integration projects.

The maturation of distributed systems will create a new economics of enterprise software. Distributed systems have begun to deliver on their promise of efficiency to enterprise software developers. To accelerate the benefits of distributed applications, executives must procure rare architecture expertise, overcome resistance to cultural changes (waterfall to iterative development) and have a realistic roadmap to negotiate the adoption of ASE.

About the Authors

Mark Panthofer is Vice President Professional Services at nVISIA. He is a 17-year software engineering veteran with ten years of enterprise object-oriented technology experience. He has extensive experience with delivering distributed object solutions to global enterprises while guiding their adoption of new technology and has been instrumental in the firm's astounding success ratio.

Christopher Rollyson is Vice President Marketing at nVISIA. He has 15 years of technology-focused marketing and consulting experience in several markets. A former management consultant at PwC specializing in market strategy, e-business strategy and technology adoption, he leads the strategy and execution of nVISIA's marketing activities.

End Notes

[1] Object-oriented technology is the next generation of software analysis, design and programming. It enables data structures to become objects that include data and functions, and objects have relationships among one another, which enable software engineers to create modules that do not need to be changed when a new object is added. Objects are said to "encapsulate" complexity because they are self-contained and only share the information needed to execute certain defined requests. Object-oriented programs are often easier to modify.

[2] Waterfall development is marked by long development cycles. Business requirements are gathered by the software development team in a long, up-front effort that precedes development. Once requirements are complete, the software is developed and tested. Waterfall development worked well when the business needs driving requirements were less dynamic than they are in 2003.

[3] Iterative development arose in tandem with object-oriented technology as a response to the limitations of waterfall development. Iterative development anticipates shifting requirements and makes the software development process more flexible to accommodate changes "on the fly." Iterative development modularizes large software development projects, with each "module" having its own requirement phases and deliverables. The end result is that the development process is more interactive between developers and customers, and changes are not as disruptive as in waterfall development.

[4] In distributed computing, different components and objects that comprise an application can be located on different computers that are connected by a network. It enables applications to be much more flexible because various components reside where they run the best (hardware) or where they are easiest to maintain; components are not limited to being on the same computer.

[5] Java 2 Enterprise Edition is a platform-independent, Java-centric environment for developing, building and deploying Web-based enterprise applications. It consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.

[6] Service-oriented systems make their services (resources, software components) available in a structured format that describes their capabilities and how to access them. Other parts of the system (or other systems) can request those services on demand but have no power to modify their makeup, which ensures that their capabilities always remain available to any and all other "consumers." This loosely coupled, on-demand assembly of resources has the advantage of being highly adaptable to change.

[7] Web services are a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone.

[8] eXtensible Markup Language (XML) is pared-down version of SGML that is designed especially for Web documents. XML allows designers to create their own customized tags , enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

[9] Abstraction refers to the process of picking out (abstracting) common features of objects and procedures. It is one of the primary techniques in software engineering and is used to reduce complexity.

[10] The more poweful the network, the more devices or software components it can connect. For example, basic telephone service was a new kind of network that enabled the spread of telephones at first, followed bynumerous other devices like fax machines and modems which, without the network being in place, would probably not have succeeded in the market place.