WHITE PAPER

Combining models and patterns: delivering on the promise of increased IT productivity

Ton Blankers, OptimalJ Product Manager, Compuware

Maintaining IT productivity in constantly changing technological and business landscapes is very much like trying to hit several moving targets at once. Many IT managers now have the daunting task of completing more application development, testing and maintenance while working with reduced staffs and smaller budgets, leading to longer working hours and a more stressful working environment. Meanwhile, the technologies that IT staffs must work with and integrate across their enterprises—platforms, networks, servers, hardware and software—are becoming more complex on an almost daily basis.

Increasing productivity in such an environment is a strain, at the very least. Results from *InformationWeek*'s annual salary survey for 2002 show more than 50 percent of IT professionals and more than 60 percent of IT managers said their jobs are more stressful than a year ago. Ultimately, increasing productivity means finding ways to speed development in a wide variety of technologies, achieving software reusability and increasing overall code quality—without burning out reduced workforces.

That's where advances over the past decade in pattern-based development, especially as it relates to object-oriented programming, are beginning to deliver on the promise of greater productivity. Pattern-based development helps IT departments design software more quickly from a greater level of abstraction—while at the same time improving the quality and reusability of the code.

Simply put, patterns are created by architects to describe a general solution to recurring software design problems; developers then apply the pattern solution to their specific projects. Patterns capture specific knowledge about the architectures, platforms and technologies to help create reusable code.

As architects create and link together more patterns to create a pattern framework, they can significantly reduce the time it takes to develop an application, compared to creating one from scratch.

Speaking in June 2002, Rikki Kirzner, Research Director at IDC, a world leader in IT market research and analysis, explained the benefits of pattern-based development for Java applications and J2EE in particular.

“The ability to develop specific patterns, containing specialized knowledge that can be reused throughout the company, goes a long way toward helping architects create and maintain an individual architectural style. Developers can then focus on what needs to be built rather than on how it should be built.”

Although pattern-based development does deliver on the promise of increased developer productivity, does it go far enough? It can be argued that the answer is no. The future to attaining even greater productivity heights is combining models with patterns, as well as continuing to enhance the power of patterns. Combining models and patterns can achieve even higher levels of abstraction because models simplify the business process while patterns simplify technology.

Giga presented the need for models plus patterns in January 2002 in a GigaTel report entitled, “Application Development Tool Trends for .NET, J2EE and Web Services.” Combining models and patterns results in design and technology abstraction (requiring less coding and design skills), as well as high productivity and consistency in application development.

The goal: reuse

Comparing the advances in hardware productivity to software development is, frankly, comparing apples to oranges. Moore’s Law—originally a prediction made in 1965 by Gordon Moore, the co-founder of Intel—states that the number of transistors on a computer chip will double every 18 months. Thirty-seven years later, the “law” still stands. No such law, or even prediction, exists for programmer productivity. As engineers continue to find new ways to print circuits smaller and smaller, no such level of productivity gains is visible on the software development side.

Software development languages have gone through several generations: 3GL languages such as Pascal, COBOL and C; and 4GL/Rapid Application Development (RAD) object-oriented languages such as Forte, Powerbuilder and UNIFACE. Each has made some improvements in speeding application development. But current development, especially in Java environments, has seen an erosion in developer productivity. As Java environments incorporate and communicate with an increasing number of platforms, application complexity grows—and developer productivity slows.

Reuse has long been a "philosopher's stone" for IT departments seeking a way to improve productivity. There have been a number of attempts to facilitate greater levels of reuse, the main one being component-based development. Just as the goal is for software components to move from a huge mass of software toward a repository of clearly defined reusable software components, the goal is for patterns to componentize knowledge into smaller, articulated chunks for reuse.

What patterns are, and how they are used

To understand how patterns work, and how they benefit developers and application quality, it's helpful to understand the concept of patterns.

Patterns have long been used in a variety of scientific disciplines to create new products or gather new information. Chemists use patterns of existing chemical compounds to create new ones; anthropologists use cultural patterns to extrapolate social events from ancient artifacts; medical scientists use research methods such as double-blind tests to collect data on new treatments and verify their efficacy.

The concept of software patterns appeared in the early 1990s. In 1995, four developers who had been working on the concept published their landmark work, "Design Patterns: Elements of Reusable Object-Oriented Software." The developers—Ralph Johnson, Erich Gamma, Richard Helm and John Vlissides—came to be known as the Gang of Four, or GoF. Their book helped create a burgeoning movement within the developer community to further the use and development of patterns in software design.

Pattern-based application development is designed to deliver repeatable, reusable code that addresses common—and even uncommon—business tasks by first defining all the elements that affect a solution, and providing tested examples of code that can then be implemented.

There's a frequent question about patterns that those new to the concept often ask: How does a pattern differ from a development method, a protocol, a design or a coding standard? The answer is that a pattern can focus on any one of these structures for developing code, but is not comprised solely of any one of them. Patterns describe how and why a protocol or method or design or coding standard may apply in a specific development context, and includes detailed guidance to apply them.

The future of patterns

There are many types of patterns in use today from many sources, including those from the GoF. Sun's J2EE patterns constitute another popular use of pattern-based development. Strictly speaking, however, it can be argued that these so-called patterns are more code templates or code patterns than patterns in the true meaning of the word. Whereas developers do gain some

economies of time by using these code patterns, as they speed up initial development, their limitations as "true" patterns are beginning to be felt. The reason: Code patterns do not help to accelerate ongoing maintenance and change. Most of the patterns in use today are code templates and perform a very specific task, which is to provide code to developers that can be cut and pasted where required.

The power of patterns has to be brought to the next level. The key requirement, however, is to combine patterns and models to achieve even greater productivity improvements. The combination of patterns and models reduces technology and business complexity.

Object Management Group's (OMG's) Model Driven Architecture (MDA) is a key emerging standard. PricewaterhouseCoopers highlighted MDA in its 2002-2004 Technology Forecast as a "significant technology trend" to watch. MDA promotes the concept of models and patterns. MDA aims to automatically transform models into a proper architecture and eventually into a working application. The key element MDA does not offer yet is the set of rules to transform a Platform Independent Model (PIM) into a Platform Specific Model (PSM) and eventually into code. The solution here is the use of transformation patterns. These transform higher-level business models and PIMs automatically into lower-level models and code, to reduce complexity.

OMG is working currently with Compuware and Sun, among others, to define this standard and make it part of MDA. When this standard is accepted, it will be possible to solve the real problem: accelerating application delivery by simplifying development. In this way, business-focused developers with varying experience levels can rapidly produce reliable business applications on complex platforms such as J2EE and .NET.

To gain further benefits from patterns, patterns themselves also need to be taken to the next level. A key requirement is pattern editing. Pattern editing functionality enables software architects to build and maintain their own patterns, tailored to their companies' requirements, to automatically implement working applications. Patterns result in fast, efficient, standard development throughout the development process—focusing developers on what to build rather than how to build it. Editing gives architects a mechanism to define patterns that developers automatically use when building applications. Developers who use patterns no longer need to exert time and effort on complex, technical infrastructures.

Improvement is also needed to allow patterns to be used by other patterns—so-called pattern "frameworks." Pattern frameworks are the combination of patterns at different levels, the platform-independent level, the platform-specific level and the code level. Together, these patterns form a pattern framework, which reuses best practices or implements software development standards, to accelerate development.

Online exchanges and open sharing of patterns between developers are still in an early stage, but are necessary and will grow quickly as more developers and IT managers realize the benefits from pattern-based development. In its most basic sense, patterns don't define a new way to code, but rather a new way to collect and share all relevant information about how to code for a specific problem, and open sharing of patterns will aid this.

Using "true" patterns, organizations can:

- increase reuse and productivity, leveraging pre-defined designs, structure and code, and capture the specific knowledge of the few skilled architects within an organization
- achieve fast, efficient, standard development throughout the development process—focusing developers on what to build rather than how to build it
- retain business and technology knowledge from IT staff that would otherwise disappear once a skilled staff member leaves the organization
- smooth implementations of new software versions, as patterns can incorporate the latest versions and applications are updated automatically
- increase the quality of code, as each pattern contains code that has already been tested and used in production.

Conclusion

Patterns hold great promise for software development, and it can be said that they're already delivering some degree of that promise to application development, easing developer workloads and making development faster. It is the combination of models and patterns, however, that will deliver the real promise of increased productivity, together with initiatives to take patterns to the next level. As the number of information technologies increase, some form of business and technology abstraction must come into place if developers are to keep pace. Pattern-based, model-driven development reduces the pressure of time to market, delivers more agile and less costly applications, and answers the need to "do more with less" in tough economic conditions.

Compuware OptimalJ

Compuware OptimalJ is an enterprise application development environment designed for organizations that are adopting J2EE standards and Model Driven Architecture (MDA), as standardized by the Object Management Group (OMG), in order to accelerate the development, integration and maintenance of applications for competitive advantage.

OptimalJ uses patterns to automatically translate business models into working applications by implementing OMG's MDA in its entirety. OptimalJ enables organizations to rapidly respond to change, increase development efficiency and dramatically decrease maintenance costs.

Compuware products and professional services—delivering quality applications

Compuware is a leading global provider of software products and professional services which IT organizations use to develop, integrate, test and manage the performance of the applications that drive their businesses. Our software products help optimize every step in the application life cycle—from defining requirements to supporting production service levels—for web, distributed and mainframe platforms. Our services professionals work at customer sites around the world, sharing their real-world perspective and experience to deliver an integrated, reliable solution.

Please contact us to learn more about how our comprehensive products and services can help your organization improve productivity, create higher quality applications and ensure performance in production.

COMPUWARE®

www.compuware.com

4/03