

Architected RAD & zeroCode



www.zeroCode.com

www.pk4.in

Bangalore

London

Los Angeles

Rome

San Jose

Abstract

The rapid rise of Web services and the concurrent need to deploy existing mainframe systems and enterprise services to the Web and beyond highlight a need for scalability, agility and an on-demand delivery capability that is more often than not beyond traditional in-house application development teams. Gartner has identified trends in methodology which have shown organizations moving from traditional waterfall structured analysis and design approaches to iterative, incremental processes and business process analysis (BPA) methodologies. These trends, according to the research house, are continuing as organizations evolve toward service-oriented architectures such as Java 2 Platform, Enterprise Edition (J2EE) and .NET, which require a better understanding of business process flow and object orientation for effective implementation in OO languages, such as Java and C#. Worryingly, Gartner recently reported that 75% of all non-trivial Java-based projects fail - mainly due to developer skills - and that the time to productivity for RPG, COBOL JCL, and other legacy programmers to skill-up for J2EE can be as long as 12 months, at a significant cost to companies. Of these trainees, the research house says, one third will not make the transition. It is also estimated that the average cost of developing an enterprise Web application is between \$1 million and \$12 million, with 80% of the cost being labor.

The above gives an indication of the limitations of conventional techniques in the on-demand environment. There are, however, new and powerful alternative tools - architected rapid application development (ARAD).

The Gartner Group recognizes tools of this nature as Architected RAD or "ARAD" tools. Some characteristics of these tools include:

- ? *Support for industry standards such as J2EE, UML, XML, SOAP, and MDA.*
- ? *Use of industry standard design and construction patterns, such as MVC2.*
- ? *Engineered quality in constructed code.*
- ? *Architectural consistency across teams of developers.*

According to Gartner,

There's no better way to improve productivity, reduce cost, and ensure scalability and performance of applications, while still putting methods in place that are "minimally invasive."

For most enterprises and applications, ARAD should prove to be a near-ideal approach to balancing speed and cost with "just enough" application quality and performance.

The New Buzzword- ARAD

Simply described, ARAD is Rapid Application Development (RAD) that has grown up to meet the needs of development teams charged with creating enterprise class applications for on demand business. Picture a team of eleven people working on an enterprise application -- a project leader, one analyst, eight developers, and one technology architect. Whereas RAD focused only on the developers, ARAD provides a unified environment for all these team members that comprises all the elements and capabilities essential to software development success: a unified process; business modeling via the Unified Modeling Language (UML); rapid application development; integration of legacy and enterprise systems; and scalable n-tier development.

Salient Features:

- ? It is an agile, rapid, iterative, and time-boxed development process that brings traditional RAD concepts such as prototyping and joint application development to the world of enterprise applications. It also provides a team-unified platform that supports parallel development, distributed teams, version management, release management, and change management.
- ? With Architected RAD, however, analysts on the project team can use UML to analyze business requirements, define business use cases, define the information model for the application -- consisting of business objects and their storage in databases -- and define the business processes and business rules that will drive the application.
- ? In today's corporations, most project teams consist of application developers with experience in a variety of coding environments, such as Visual Basic and PowerBuilder, Java and C++, and COBOL. Their experience also covers a diverse array of deployment scenarios, from n-tiered architectures, to client/server, to legacy/mainframe environments. Although they have diverse technology skills, these developers all understand the need to write code at an increasingly rapid pace, and to support requirements that change ever more rapidly as well. With ARAD, companies can leverage the valuable technical and domain skills these developers have acquired through the years to maintain vital legacy applications, but also enable them to quickly develop n-tier applications -- without dealing with the complexities of the underlying n-tier platform.
- ? Enterprise applications for on demand enterprises are never standalone; they connect to other legacy and mainframe systems across the enterprise and serve as the vital link between the enterprise and its customers and partners. On demand organizations need the ability to quickly construct Web applications that integrate with databases, messaging protocols, and information from a variety of application servers. Then, after the Web application is deployed, the organization needs an efficient approach for maintaining the underlying technologies, which are constantly subject to change. The ARAD process addresses these needs. It supports rapid application integration with J2EE and other new techniques that help teams quickly leverage information from disparate sources. Then, users can select target technologies for deployment without extensive knowledge of J2EE.
- ? The ARAD process allows the architect to design and fine-tune n-tier patterns, using n-tier components such as JSP, servlets, stateless and stateful session beans, and entity beans. Then, developers who keep their focus on the business can apply these patterns as they create applications to support business processes.

The zeroCode ARAD Approach

The Architected RAD approach adds to the speed of development advantages of RAD a level of architectural integrity consistent with true enterprise development. The skills of developers, system architects and business analysts are integrated into a single development vision, with support for key technology standards.

PK4's product strategy has always been focused on making developers more productive and efficient, and in recent years has added capabilities focused on business process integration and management, legacy integration, support of Web services standards and J2EE development and deployment, and automation of the process of building Web services. The Architected RAD approach adds to the speed of development advantages of RAD a level of architectural integrity consistent with true enterprise development. The skills of developers, system architects and business analysts are integrated into a single development vision, with support for key technology standards. The point-and-click, model-driven zeroCode approach takes is exemplary of the need

for tools that permit developers to build enterprise applications quickly and with minimal training in Web services, J2EE, and other technologies.

PK4 correctly recognizes that the evolution of e-business will eventually require the full integration of assets and resources that not only span departments and groups within organizations, but also must cross company boundaries in order to optimize operations and meet the needs of partners, suppliers, customers, and other communities in a timely and dynamic manner. The resulting need to integrate information, IT assets, and business processes is placing demands on development organizations and the tools they use that are likely unprecedented in IT history.

Unlike RAD approaches of the past, zeroCode provides architects and senior developers with a great deal of control over application development. Architects can ensure architectural consistency by controlling the code generation process, ensure layout and branding consistency through the style repository and templates, and ensure deployment success through advanced partitioning controls. They can also provide less experienced developers with templates and tools that make their work faster and more efficient. The end result is a quality application that is built using the best practices for the target platform, and a team that works well together and has a head start on the next project.

zeroCode and the ARAD Model

Having understood the basic ARAD concept, let us now take a look at how zeroCode fits the ARAD framework. Unlike most other ARAD frameworks, zeroCode is a full-lifecycle solution that takes a "full picture" approach to software construction.

Agile, rapid, iterative, and time-boxed development process

Most frameworks generate simple pages on single tables, the rest to be built using laborious "wizards" (non-updating) or hand-coding. zeroCode, on the other hand, begins where the others end: starting with an ER model, it builds a complete, basic app in a few minutes and then gives you a high-speed browser environment to build complex transactions and reports. Constructing even complex visual renditions is merely a matter of running the data model through a suitable "meta-template" in a few browser clicks. Most other products need significant time for each screen/page to be created. Constructing input and editing forms has an even bigger payoff, since they are automatically built up front.

Analyze business requirements and define business use cases

Business analysts and domain experts can leverage the data - model-centric perspective used in the zeroCode framework well. This is typically an easier model than UML-based products that require significant technical sophistication. Senior analysts can develop the business logic by studying and interpreting the domain, while the juniors need not have an extensive programming skills to use zeroCode for analysis and development. They can build the application from the existing database schema with little programming skills involved.

Filters and action sequences provide reusable abstractions at a much higher level than the Java object - much closer to a domain expert's level of thinking. Automatic computation of dependencies between transaction sections saves a great deal of time and effort on the part of the designer. The zeroCode Design Environment remembers schema intricacies and presents them in context during the construction of complicated transactions.

Optimal utilization of development teams

With zeroCode, the existing development teams need not get into the complexities of learning and adapting to new technologies and languages. zeroCode uses industry-standard technologies like Java and XML. Designers don't need to learn new languages to be able to use the framework. While the completed applications are Java-based, their creation and maintenance do not require

Java programming skills. The basic skills of a web-master or business analyst are normally sufficient. The resulting application is robust, scalable and brings the average zeroCode user from design to functional testing directly, with little intermediary coding. Complex needs that zeroCode does not deal well with (math-intensive algorithms, for example) can be plugged in as standard Java beans or objects.

Many commonly used business functions – security, sorting, searching, foreign-key dereferencing and relators, to name a few - are built into the environment, so that the effort needed to design or construct them reduces.

Ease of Integration

Applications built with zeroCode integrate into any kind of information systems environment, as we have seen above. They are particularly well-suited for large, multi-platform organizations that have developed systems over the years and would like to leverage them with today's technology, for tomorrow's user demands.

The code generated by zeroCode is tight, effective and well-organized. It includes techniques like connection-pooling, for example, where the generated application scales to a large number of users in a linear manner. A large number of factors can be configured in the application, increasing its scalability to a growing number of users. A multi-tiered architecture further reduces the sensitivity of the applications to the number of transactions handled. Business logic functions can be shared across many machines using commercially available load-balancing hardware and software.

Design and fine-tune n-tier patterns

zeroCode uses industry-standard technologies like Java and XML. Designers don't need to learn new languages to be able to use the framework. While the completed applications are Java-based, their creation and maintenance do not require Java programming skills. The basic skills of a web-master or business analyst are normally sufficient.

In addition, the zeroCode-built application has no "proprietary" components – the code delivered is all Java, JSP/FreeMarker, XML and HTML/WML/whatever. And the code is very efficient in high-use applications, using advanced features like connection pooling. Given the fact that zeroCode can generate J2EE-compliant EJBs for all functions, its ability to build an entire application from an ER model can be combined with the skills of UML-based developers who build and integrate other objects for compute-intensive tasks. That increases productivity across the board.

The look and feel of a generated application can be modified or completely replaced using a standard HTML editor such as FrontPage. Designers can use other editors that they are comfortable with. Also, zeroCode can generate applications in other user-interface languages like WML, further enhancing their usability.

Reliability

The mean time between failures of the generated code is very high and is controllable. The generated code is based on mathematical, patent-pending algorithms that are very robust in deployment environments. These objects can be integrated into app server environments from vendors like IBM and BEA, for further execution-time scalability.

In a computing environment that is reasonably standardized and includes a minimum number of platforms to work with, the ability of zeroCode-built applications to address a large number of platforms equally well may be irrelevant. But in reality, most organizations have a need to access data from a variety of platforms. This is especially true of larger organizations - and older organizations. Fortune 1000 companies continue to rely on proven systems running on older hardware, since they provide the best bang-for-buck even today. In that kind of an environment, zeroCode-built applications are ideal mechanisms to truly leverage the information assets of a company.

Bottom-Line

zeroCode is best described as a RAD that has matured to meet the needs of development teams tasked with creating enterprise-class applications for on-demand business - without extensive J2EE re-training.

It provides a unified environment that encompasses the needs of all development team members - project leaders, analysts, programmers and architects - while comprising the elements essential to software development success.

zeroCode surpasses traditional RAD offerings in its support for the Unified Modelling Language (UML) and model-driven development, its ability to facilitate integration with legacy enterprise and business-to-business systems, its use of pre-built patterns for deploying scalable n-tier J2EE applications. In short, ARAD combines the benefits of RAD with the benefits of good architectural design. Using zeroCode, development teams can deliver on-demand applications in less time and at a lower cost. Team members with valuable business domain and legacy skills, but limited knowledge of J2EE, can be trained quickly to become skilled n-tier developers.

They can deploy applications with high scalability and performance, and with the flexibility to mix and match technologies as well as adapt to new ones. The repeatable ARAD process helps developers deliver applications with predictability and a high rate of success.

Gartner has confirmed that there is no better way to improve productivity, reduce cost and ensure scalability and performance of applications, while at the same time instilling methods that are "minimally invasive." ARAD should prove to be a near-ideal approach to balancing speed and cost with 'just enough' application quality and performance," the research house says, adding that it expects the ARAD approach to eventually represent between 45% to 55% of the application development portfolio. The take-up of ARAD is expected to be particularly rapid due to the skill shortage, the current move in both the corporate and public sectors towards Web services and, importantly, the urgent need to train developers from previously disadvantaged backgrounds.