FAQ's:

What is JAG?

Related Links:

Blue Ink Overview

Time Savings

ARTICLES

What is Rapid Application Development?

QUESTIONS?

Call us toll free: (877) 756-3595

Or Visit Forums

Rapid Application Development

Republished on Developers.net October 14 2005.

Rapid Application Development (RAD) is a marketing buzzword that almost every software development tool uses, yet one that rarely applies. At a high level it is an Application Development technique that uses Prototypes, Iterative Customization, and CASE Tools. This article will focus on RAD: its history, advantages and disadvantages, appropriateness for various problems, core elements, process, and additionally will focus on CASE tools that support RAD.

Overview

Rapid Application Development (RAD) is a software development methodology that focuses on building applications in a very short amount of time; traditionally with compromises in usability, features and/or execution speed. The term has recently become a marketing buzzword that generically describes applications that can be designed and developed within 60-90 days, but it was originally intended to describe a process of development that involves application prototyping and iterative development.

History

Rapid Application Development has been in existence for nearly 20 years, but is as valid today as it was when it was initially conceptualized.

The Problem

Processes developed in the 1970's, such as the Waterfall development methodology, often resulted in the development of applications that did not meet client needs because applications took so long to build that requirements had changed before the system was complete. Thus, for larger projects, these methodologies frequently resulted in complete, but unusable, systems.

The cause of the problem was identified in the strict adherence to completion of one lifecycle stage before moving on to the next lifecycle stage. Specifically, building an application based on requirements that have been frozen at a point in time means that the longer development takes, the more likely that business needs will change and invalidate the requirements that the system being developed is based upon.

The RAD Solution

There have been many responses to this problem from the 1980's through today. In 1986 Barry Boehm wrote *A Spiral Model of Software Development and Enhancement*, which initially defined the concepts of prototyping and iterative development and had a focus on risk reduction. During the late 1980's Scott Shultz and James Martin refined the ideas of prototyping and iterative development into a methodology called Rapid Iterative Production Prototyping (RIPP) that focused on developing systems in a short timeframe with small teams of highly qualified, motivated, and experienced staff. James Martin (who was nominated for a Pulitzer prize and is frequently called "the Guru of the Information Age") further expanded and formalized Rapid Iterative Production Prototyping and in 1991 published the book *Rapid Application Development*, the details of which are explained in the remainder of this article.

Other Solutions

Other solutions have arisen out of the need for adaptable development techniques, most notably the agile methodologies developed during early 2000's, the most popular of which is Extreme Programming. Kent Beck wrote the first book on the topic in 1999 and expanded it with a second edition in 2005. Extreme programming is a somewhat controvercial methodology that deemphasizes formalized requirements gathering and instead puts a heavy focus on developing programming code in short development cycles with constant feedback from the user.

Advantages and Disadvantages

Speed and quality are the primary advantages of Rapid Application Development, while potentially reduced scalability and feature sets are the disadvantages.

Increased Speed

As the name suggests, Rapid Application Development's primary advantage lies in an application's increased development speed and decreased time to delivery. The goal of delivering applications quickly is addressed through the use of Computer Aided Software Engineering or CASE tools, which focus on converting requirements to code as quickly as possible, as well as Time Boxing, in which

features are pushed out to future releases in order to complete a feature light version quickly.

Increased Quality

Increased quality is a primary focus of the Rapid Application Development methodology, but the term has a different meaning than is traditionally associated with Custom Application Development. Prior to RAD, and perhaps more intuitively, quality in development was both the degree to which an application conforms to specifications and a lack of defects once the application is delivered. According to RAD, quality is defined as both the degree to which a delivered application meets the needs of users as well as the degree to which a delivered system has low maintenance costs. Rapid Application Development attempts to deliver on quality through the heavy involving of users in the analysis and particularly the design stages.

Reduced Scalability

Because RAD focuses on development of a prototype that is iteratively developed into a full system, the delivered solution may lack the scalability of a solution that was designed as a full application from the start.

At Automated Architecture our Just-In-Time Application Generation methodology provides the benefits of Rapid Application Development while minimizing many of the disadvantage, such as reduced scalability, through the generation of an enterprise level "prototype" that provides as a starting point a scalable, efficient, and well designed application.

Reduced Features

Due to time boxing, where features are pushed off to later versions in favor of delivering an application in a short time frame, RAD may produce applications that are less full featured than traditionally developed applications. This concern should be addressed as soon as possible through clear communication with the client as to what will be delivered and when.

Appropriate RAD Projects

Rapid Application Development is not appropriate for all projects. The methodology works best for projects where the scope is small or work can be broken down into manageable chunks. Along these lines project teams must also be small, preferably two to six people, and the team must have experience with all technologies that are to be used.

Business objectives will need to be well defined before the project can begin, so projects that use RAD should not have a broad or poorly defined scope. Furthermore, in order to keep the project within a short time frame, decisions must be able to be made quickly, so it imperative that there be very few client decision makers, preferably only one, and they must be clearly identified up front. Client decision makers need to understand and agree to a RAD approach and ideally should be willing to accept a product that is less full featured and/or be willing to accept higher development cost (due to the emphasis on purchasing reusable components over building them) in exchange for increases in speed.

Core Elements of Rapid Application Development

RAD has many core elements that make it a unique methodology including prototyping, iterative development, time boxing, team members, management approach, and RAD tools.

Prototyping

A key aspect of RAD is the construction of a prototype for the purpose of jumpstarting design and flushing out user requirements. The objective is to build a feature light version of the finished product in as short an amount of time as possible, preferably days. The initial prototype serves as a proof of concept for the client, but more importantly serves as a talking point and tool for refining requirements.

Developing prototypes quickly is accomplished with Computer Aided Software Engineering CASE tools that focus on capturing requirements, converting them to a data model, converting the data model to a database, and generating code all in one tool. CASE tools were popular in the late 80's and early 90's, but as technology has changed (and COBOL has become obsolete) few tools take full advantage of the full potential of CASE tool technology. The Rational company is the most well known although its prototype generating potential is limited.

At Automated Architecture our Blue Ink product focuses on generating enterprise level web applications that serve as a prototype due to the speed with which they can be created (in minutes).

Iterative Development

Iterative development means creating increasingly functional versions of a system in short development cycles. Each version is reviewed with the client to produce requirements that feed the next version. The process is repeated until all functionality has been developed. The ideal length of iterations is between one day (which is closer to Agile Methodologies) and three weeks.

Each development cycle provides the user an opportunity to provide feedback, refine requirements, and view progress (in focus group session meetings). It is ultimately the iterative development that solves the problems inherent in the inflexible methodologies created in the 1970's.

Time Boxing

Time boxing is the process of putting off features to future application versions in order to complete the current version in as short amount of time as possible. Strict time boxing is an important aspect of RAD, because without it scope creep can threaten to lengthen development iterations, thus limiting client feedback, minimizing the benefits of iterative development, and potentially reverting the process back to a waterfall methodology approach.

Note from the author: From personal experience I can say that client communication, specifically limiting client expectations, is extremely important with time boxing and iterative development, because without sufficient warning, and even with a theoretical understanding, clients may be presented with a version of an application that is well below their idea of an ideal, complete system. I can not stress enough the need to identifying the approach and set a versioning timeline from the very beginning of the project.

Team Members

The RAD methodology recommends the use of small teams that consist of experienced, versatile, and motivated members that are able to perform multiple roles. As the client plays a vital role in the development process, dedicated client resources must be available during

to perform multiple roles. As the client plays a vital role in the development process, dedicated client resources must be available during the initial Joint Application Development (JAD) sessions as well as Focus Group Sessions conducted at the end of development cycles. Development teams (also known as SWAT or Skilled Workers with Advanced Tools) should ideally have experience in Rapid Application Development and should have experience with the Computer Aided Software Engineering tools.

Management Approach

Active and involved management is vital to mitigate the risks of lengthened development cycles, client misunderstandings, and missed deadlines. Above all management must be stalwart and consistent in their desire to use the Rapid Application Development methodology. In addition to enforcing a strict timeline, management must focus on team member selection, team motivation, and on clearing bureaucratic or political obstacles.

RAD Tools

One of the primary objectives of the Rapid Application Development methodology developed by James Martin in the late 1980's was to take advantage of the latest technology available to speed development. Clearly the technology of the 1980's is obsolete today, but RAD's focus of the latest tools is as important today as it was when the methodology was initially created. This article has a section dedicated to tools following the process section.

Process

Below is a brief overview of the RAD process, which consists of four lifecycle stages: Requirements Planning, User Design, Construction, and Implementation. Also described are typical pre and post project activities.

The Gantthead project management portal is an excellent resource that contains more far more details on the process and additionally contains sample project plans and details on techniques such as time boxing, project estimation, recommended tools, and work breakdown structures. *Do not begin your project without at least looking at this vital resource*. Additional information is also available on the slightly dated but otherwise excellent website put together by Walter Maner of Bowling Green State University.

Pre-Project Activities

As with any project it is vital to identify the details of the project up front in some form of document such as a Project Management Plan (PMP). All parties should agree up front on details such as potential risks and mitigation strategies, a development schedule including resources, milestones and deliverables such as a completed data model or types of documentation to deliver, an approach including standards, tools, and technologies to be used, a desired end result, terms and constraints and financial considerations including budget and cost of tools.

Requirements Planning

The Requirements Planning stage (also known as the Concept Definition Stage) consists of meetings between a requirements planning team and key client users. Meetings focus on both developing a high level list of initial requirements as well as setting the project scope. The requirements planning team identifies primary business functions (such as "sell widgets to the Acme corporation") and initially breaks them down into business entities (such as Product, Sale, Company, Sales Person). The Requirements Planning stage should result in a list of entities as well as action diagrams that define the interactions between processes and data elements and should take between one and four weeks. Ideally requirements should be captured in a structured tool such as IBM's Rational Rose or Rational RequisitePro or Microsoft's Visio (the enterprise edition, since it can generate databases from a data model) rather than an unstructured document (see the tools section below for more details). At the end of the Requirements Planning stage project estimation should be considered. The International Function Point Users Group as well as the International Software Benchmarking Standards Group offer project estimation methods based on "function points" along with a vast database of actual projects with the effort involved to complete them.

User Design

During the User Design stage (also known as the Functional Design Stage) the analysis team meets with end users in Joint Application Development (JAD) Workshops. During the workshops the analysis team flushes out the requirements in more detail, develops the entities developed in the Requirements Planning into a data model (Entity Relationship Diagram), formalizes business rules, develops test plans, and creates screen flows and layouts for essential parts of the system. During the later half of the User Design stage the development team (also known as the SWAT or Skilled Workers with Advanced Tools team) aids the analysis team in activities such as creating a working data model that can be converted to a functional database, and in identifying reusable components (such as Microsoft's Application Blocks, which, incidentally, are an excellent time saver on Microsoft .Net projects). Again, as mentioned in the Requirements Planning stage, all requirements should be captured in a tool.

Before moving to the Construction Stage the analysis team should focus on next steps by flushing out the project plan and focusing on effort estimates. Focusing on next steps is an important element of the User Design phase, because the initial iteration of the

Construction Phase should focus on a feature light prototype. In order to keep development iterations as short as possible, and to gain the maximum benefit of RAD's agile nature, core requirements should be identified and targeted for the initial prototype, and secondary requirements should be identified and targeted for future development iterations. Beyond a vertical limiting of scope, such as removing entities or use cases from the initial scope, a horizontal type limiting of scope should be considered as well, such as not developing field validation, not developing file upload/download capabilities, or focusing on just the strengths of the particular CASE tool being used without manually adding much developer customization.

The User Design stage should last between three and five weeks.

Construction

During the Construction Phase the Design Team develops the application in iterative cycles of development, testing, requirements refining, and development again, until the application is complete. Development iterations should last between one day and three weeks. The development team should convert the Data Model that was developed during the User Design Stage into a functional database (all data modeling tools have this ability). The CASE tool used (which may be the same as the data modeler or a separate tool) should now generate large sections of the application, at a minimum data access code, but preferably business functions and user interface as well.

At Automated Architecture, our Blue Ink product will read information from database that has been generated, pre-generate answers to meta-data about the project (in other words make a best guess as to the details of your application that you may then customize in more detail later), and generate an entire application that can serve as a working prototype without a line of development code. A prototype developed in this way may reduce the first iteration of development to days instead of weeks.

It is vital to keep each development iteration on track, and functionality may need to be dropped to keep development within the time box. Management plays a vital part in ensuring everything is progressing according to schedule, keeping the customer in the loop regarding changes in the functionality, and keeping the team motivated.

Once the prototype has been developed (within its time box), the construction team tests the initial prototype using test scripts developed during the User Design stage, the design team reviews the application, the customer reviews the application and finally the construction team, design team, and customer meet in Focus Group meetings in order to determine the requirements for the next iteration. Focus group meetings consist of facilitated sessions that last about two hours. The facilitator should know ahead of time the areas that require discussion and should ensure that each issue receives enough attention, keeping a list of issues that require additional attention in a separate meeting as appropriate.

After the meeting (additional meetings may be necessary), the development team and design team should update the requirements, data model, test scripts, and project plan as during the User Design stage. Again the teams should identify core and secondary requirements, plan out the next development iteration, keep the user in the loop regarding what will be done, and then start the next iteration of development over again. As the system approaches a sufficient state the development team should focus on the system as a finished application rather than a prototype.

During the final iterations of development the design team should update user documentation, perform User Acceptance Testing and define the steps necessary for deployment/implementation.

Implementation

The Implementation Stage (also known as the Deployment Stage) consists of integrating the new system into the business. The Development Team prepares data (such as lookup values like States and Countries) and implements interfaces to other systems. The Design Team trains the system users while the users perform acceptance testing. and are trained by the Design Team. The Design Team helps the users transfer from their old procedures to new ones that involve the new system, trouble shoots after the deployment, and identifies and tracks potential enhancements (read wish list). The amount of time required to complete the Implementation Stage varies with the project.

Post-Project Activities

As with any project final deliverables should be handed over to the client and such activities should be performed that will benefit future projects. Specifically it is a best practice for a Project Manager to review and document project metrics, organize and store project assets such as reusable code components, Project Plan, Project Management Plan (PMP), and Test Plan. It is also a good practice to prepare a short lessons learned document.

Rapid Application Development Tools

As mentioned in the Core Elements section above, the RAD methodology was designed to take advantage of the latest technologies. Unfortunately RAD is such a nice buzzword that many tools have taken to using it to describe what they offer for purely marketing purposes. I believe that James Martin had a particular type of tool in mind when he wrote about RAD tools and below I provide my personal view of what type of tools support RAD development as opposed to the tools that just speed up development on any project.

Data Integration

Carnegie Mellon's Software Engineering Institute has a 107 page report on Rapid Integration Tools for Rapid Application Development written in December 2004 that includes a detailed analysis of Pervasive Data Junction, RoughWave's LEIF, IBM Rational Rapid Developer, Microsoft SQL Server, Host Integration Server, Microsoft BizTalk Server, IBM WebSphere Business Integration, Artix Relay, Encompass and Mainframe, PiiE Smart Client and Fusion Server, InterSystem Ensemble, and Jboss. While these tools can all be used for integrating with legacy systems, some more "rapidly" than others, their primary function makes them no more suitable for RAD development than SCRUM development or Waterfall development. One could certainly envision a RAD project that does not use a data integration product.

Development Environments

ZD Net wrote an article in October 2004 called Five IDEs tested that compares the following tools that claim to support Rapid Application Development: Microsoft Visual Studio.NET 2003, Sun Java Studio Creator, BEA Web Logic Workshop 8.1, Borland C# Builder, and IBM WebSphere Studio Application Developer 5.1.2. The article is well written and useful, and compares the relative RAD merits of the tools (e.g. how easy it would be to create a prototype in them). However, all these tools are just development environments. These tools are necessary for development and some may be more "rapid" than others, but ultimately none of them supports RAD more than Microsoft Project supports RAD (in fact, I would argue that Microsoft Project supports RAD *more* than any development environment).

Requirements Gathering Tools

All stages of the RAD Methodology, particularly the requirements planning stage, specify that requirements should be captured in a tool rather than an unstructured document. For this reason, and because the Unified Modeling Language (UML) is the only language I am aware of for this task, tools that support writing in the Unified Modeling Language (UML) support RAD. There are numerous tools that support UML notation: Microsoft Visio and IBM's Rational Rose as mentioned in the Requirements Planning are two of the most popular, while Enterprise Architect 5.0 by Sparx Systems is another leading UML tool. For more information about UML, which was created by the Object Management Group (OMG), see the Introduction To OMG's Unified Modeling Language™ (UML®) on their website. For a list of tools that support UML see the UML Resource Page (check the bottom of the page).

Data Modeling Tools

If requirements gathering tools are import for RAD, Data Modeling Tools are vital. Modifying a database manually with SQL statements or through an IDE goes against RAD because data modelers capture requirements *and* speed development by generating the database. Not using data modeling on even the smallest project is just a terrible idea. Often tools that support requirements gathering and UML also contain a data modeling tool and everything links together. The Introduction to Data Modeling site is a detailed practical guide maintained by the University of Texas at Austin's Information Technology Services. The next most popular after the Microsoft and IBM tools is probably ERwin by Computer Associates.

As far as Microsoft Visio, I have put together an in depth article on how to use Visio for data modeling. The Microsoft documentation is fairly scarce, but this guide helps flush out the details for an otherwise very good tool.

Code Generation Tools

RAD was designed in large part to take advantage of CASE Tools. Computer Aided Software Engineering technology involves aspects of requirements gathering and data modeling, but most especially of code generation. Code generation is involves taking some input (often UML models, but also Databases themselves) and transforming it to the source code that a developer might otherwise have to write according to some rules (often called templates).

There are a great many tools that provide code generation technology almost all of which are listed at the Code Generation Network. Good criteria for selecting a code generator includes:

- Supports existing technologies, architectures, and best practices
- Produces enterprise level code (i.e. n-tier code)
- Generates prototype applications without having to write a line of code (do not accept a code generator that does not at least attempt to generate a presentation layer code)
- Uses templates or a technology that allows complete control over outputted code
- Provides a flexible meta-data mechanism
- Can be used throughout the entire development process, and specifically will not overwrite your code

At Automated Architecture our Blue Ink code generator supports all of these requirements including basing the generated application on Microsoft's best practices described in Application Architecture for .NET: Designing Applications and Services and Designing Data Tier Components and Passing Data Through Tiers.

Conclusion

Clearly Rapid Application Development is about more than just delivering applications as quickly as possible. James Martin intended it as a well defined approach to application development involving short, iterative development cycles; timeboxing; prototyping; and the use of modern technology for requirements capture with an eye toward turning those captured requirements into a working application using code generation or similar technologies. This approach is just as valid today as it was in the late 1980's, only its meaning has been somewhat obscured by modern marketing.

Additional Resources

Aside from the resources already mentioned in the text of this document Case Maker has a slightly dated document with good diagrams that may be useful and WikiPedia has information on everything imaginable, but also a concise but useful description of RAD.