Rational Unified Process

Group Members:

Najam Shahid (1110)
Ovais Ahmad Khan (1105)
Syed Kashif Anwar (1115)
Umar Tariq Pirzada (1085)

TABLE OF CONTENTS

TABLE OF CONTENTS....................2

INTRODUCTION....................3

THE TRADITIONAL STRUCTURED ANALYSIS....................3

WATERFALL MODEL OF ANALYSIS AND DESIGN....................3
ASSUMPTIONS OF THE WATERFALL MODEL....................3
WATERFALL PROCESS LIMITATIONS....................3
REUSE – ANOTHER MAJOR PROBLEM WITH STRUCTURED ANALYSIS....................4
THE SOLUTIONS....................4

THE RATIONAL UNIFIED PROCESS....................5

A BRIEF HISTORY....................5
DETAILS....................5

ADVANTAGES OF USING RUP....................6

DISADVANTAGES OF USING RUP....................7

CASE STUDY: BUILDING A NEW RATIONAL WEB SITE....................8

THE TOOLS....................8
OTHER DETAILS....................8
RUP UNIFIES EFFORTS OF DISTRIBUTED TEAM....................8
INCEPTION PHASE....................9
ELABORATION PHASE....................9
CONSTRUCTION PHASE....................9
CONFIGURATION & CODE MANAGEMENT....................9
SOURCE CODE REPOSITORY....................9
TESTING PHASE....................10
SITE PROFILE....................10
BETTER QUALITY DESPITE THE TIGHT TIMEFRAME....................10

CONCLUSION....................11

INTRODUCTION

This report gives a high level description of the philosophy and structure of the Rational Unified Process® (RUP), a process framework, refined over the years by Rational® Software, that's being widely used on a variety of software projects, from small to large.

THE TRADITIONAL STRUCTURED ANALYSIS

W. W. Royce described the traditional structured analysis in 1970. It involves decomposition in terms of Function and Data. The analysis model followed in this approach is the Waterfall Method of Analysis and Design. The major problems with it were that the mobility was available only at the file level and the data was not encapsulated. There were only 3 types of scopes available namely, Global Scope, File Scope, and Function Scope (automatic, local).

Waterfall Model of Analysis and Design

The waterfall or the linear sequential model suggests a systematic sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support. It encompass the following activities:

- System / Information Engineering and Modelling
- Software Requirement Analysis
- Design
- Coding from Design Specifications
- Testing
  - Unit Testing
  - System Testing
  - UAT Testing
- Support

Assumptions of the Waterfall Model

The following are the assumptions of the Waterfall model:

- Requirements are known up front before design
- Requirements rarely change
- Users know what they want, and rarely need visualization
- Design can be conducted in a purely abstract space, or trial rarely leads to error
- The technology will all fit nicely into place when the time comes (the apocalypse)
- The system is not so complex. (Drawings are for wimps)

Waterfall Process Limitations

Having a look at these assumptions tells us a lot about the problems of the waterfall model. Among the problems that are sometimes encountered when the waterfall model is applied are:

- Real projects rarely follow the sequential flow that the model proposes
- It is often difficult for the customer to state all requirements explicitly
- User doesn't get to *see* anything real until the very end, and they always *hate* it.
- System Testing doesn't get involved until later in the process
- The model often leads to *blocking states* in which some project team members must wait for others to complete dependent tasks

Reuse – Another Major Problem with Structured Analysis

In structured analysis, reuse is usually defined as code reuse and is implemented through cutting and pasting of the same code in multiple places. It results in:

- coding changes need to be made in several different places
- changing the function often changes the API which breaks other functions dependent upon that API
- data type changes need to be made each time they are used throughout the application

The Solutions

In order to mitigate problems described previously in proprietary software development processes, base your process framework on open, published and supported standards. Two software development processes stand out as strong candidates for consideration:

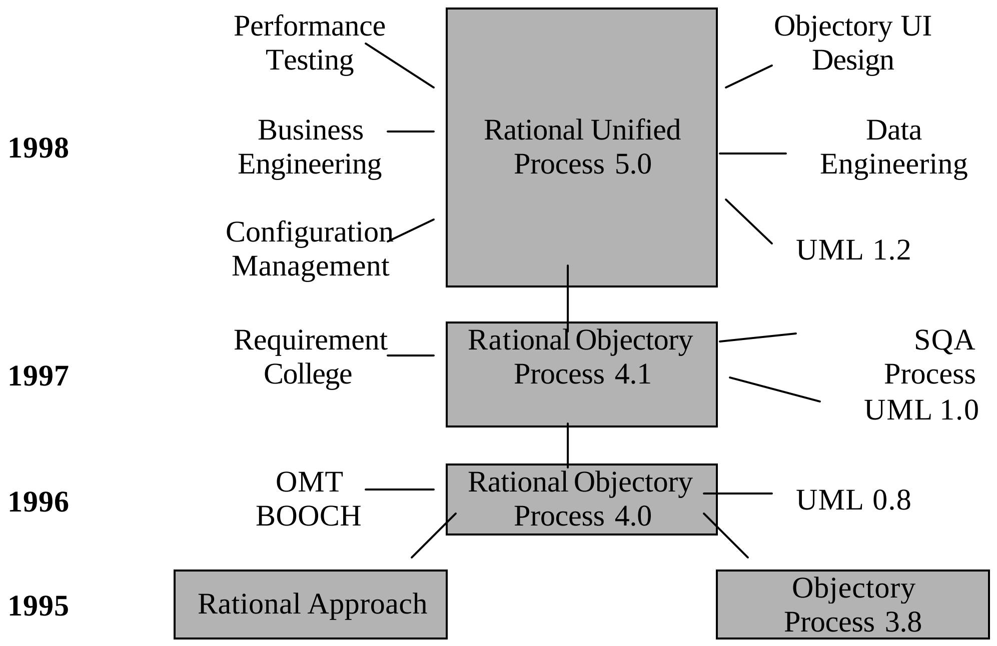
- Extreme Programming (XP)
- The Rational Unified Process® (Unified Process)

These processes are well documented and publicly available. In this paper, we will briefly describe The Rational Unified Process® and highlight why it is so important and how it addresses specific shortcomings of proprietary development processes.

THE RATIONAL UNIFIED PROCESS

<Intro goes here>

A Brief History



Details

< details goes here >

ADVANTAGES OF USING RUP

- **Well-documented and comple te methodology** – RUP is a complete methodology with all of its documentation easily available.
- **Open and public** – The Rational Unified Process is openly published, distributed and supported.
- **Training readily available** – The on-line version of the Rational Unified Process walks users through the process in a step-by-step tutorial manner. A lot of institutes also offer training courses.
- **Changing Requirements -** proactive resolve of client's changing requirements and related risks
- **Reduced integration time and e ffort** – As the development model followed is iterative in nature, so we integrate the code in phases resulting in lesser time and effort spent on integration.
- **Higher level of reuse -** The reuse of code is easy and faster.

DISADVANTAGES OF USING RUP

- **The process is too complex** – Unless you have a real expert, it is likely that you will succeed in adapting to this process. The process is too complex, too difficult to learn, and too difficult to apply correctly.
- **Sociological Aspects** - The Unified Process does not capture the sociological aspects of software development and the details of how to truly develop incrementally.
- **Disorganized Development** - may lead to a totally undisciplined form of software development

CASE STUDY: BUILDING A NEW RATIONAL WEB SITE

In April of last year, Rational Software determined that its Web site (http://www.rational.com) no longer measured up to the high standards the company had set for itself. Obviously, what was cutting edge technology in 1997 had become insufficient for a company whose identity centered on industry-leading tools and services for e-development! Based on a proprietary infrastructure that the original vendor was no longer supporting, the site could not scale to keep up with Rational's rapid growth and long-term business needs. We soon concluded that it was time to rebuild from the ground up.

The Tools

When it came to securing the right tools for the team to do the job, we had to look no farther than our own Rational Suite. In fact, we viewed the complete site overhaul as an opportunity to showcase Rational Suite as an end-to-end e-development solution in a demanding implementation environment.

Using the Suite enabled the team to launch the new and improved company Web site right on schedule, after two and-a-half months of intensive development. The site was also more tightly integrated with existing back office systems within the same timeframe, dramatically improving both customer self-service and other e-business capabilities.

Other Details

Why did the team have such a tight time frame for this project? Rational has two major software releases each year that occur in our first and third financial quarters, respectively, and the aim was to have the new site up and running well before the second release cycle. In addition, the team wanted to have the site up in time to support Rational's annual User Conference, scheduled for the end of August 2000. It was evident that only tight organizational practices would enable us to get the project completed on time. It was decided to supplement the in-house team of Web developers and engineers, who specialize in Java development, by engaging an outside vendor to assist in the systems integration for the Web site. The choice was Context Integration, a Burlington, Massachusetts based Internet professional services firm and a member of Rational's Unified Partner Program who uses the Rational Unified Process (RUP).

RUP Unifies Efforts of Distributed Team

RUP is a software engineering process that enhances team productivity and puts the experiences of thousands of projects into the hands of each project member through its embedded best practices. Because it's totally Web-enabled, everyone could access what it had to offer through a browser.

Together, the Context and Rational teams used RUP to develop guidelines, templates, and examples as we moved through each of the RUP's critical phases: Inception, Elaboration, Construction, and Transition (see Figure 1). Because RUP is tightly integrated with other Rational tools, the entire combined development team could reap the benefits of using Unified Modeling Language (UML), software automation, and other industry best practices.

As Matthew Burnett, Context's project manager for the Rational Web site redesign put it, "When developing software this fast, it is essential that you follow the Rational

Unified Process principles. Because we were working from a shared vision and using the Rational Unified Process, the Inception and Elaboration phases went smoothly."

In fact, throughout every phase, RUP enhanced communication among all the developers and engineers by providing the distributed team with one knowledge base, one modeling language, and one view of how to develop the content and infrastructure software to support the Web site.

Inception Phase

During the Inception phase of the new Web site development, Context performed a comprehensive evaluation of the application and a thorough technology assessment of personalization servers currently on the market.

Context used the Unified Modeling Language (UML) with Rational Rose throughout the Inception phase and up through the end of Construction. Using the UML enabled the entire Context team to comprehend and provide feedback on the overall system architecture.

The team's Technical Architect analyzed and documented the user requirements and use-case requirements using both Rose and Rational RequisitePro.

Elaboration Phase

After the selection of the server, the projected headed into a four week Elaboration phase. This included developing a prototype of the site and using Rational Rose to visually model the software architecture. Unified Modeling Language (UML) is used with Rational Rose for the analysis and detailed design of the system.

By modeling, artifacts were created that the entire team understood and that were not subject to individual interpretation. This, in turn, enabled the team to generate whatever other documents and deliverables we needed via Rational SoDA, in less time than it might otherwise have taken.

Construction Phase

During the Construction phase, Rational Rose saved the team a significant amount of time by generating code automatically, based on the architecture. In addition, throughout these phases, the rapid prototyping methodology embodied in the RUP and the Suite tools was invaluable when it came to tackling high-risk design challenges, such as the back-end integration of the Web site's e-commerce capabilities with our Rational legacy infrastructure.

Configuration & Code Management

Both Context and Rational's in-house team used Rational ClearCase as the project's CM (Configuration Management) solution and Rational ClearQuest for defect tracking and workflow management. Because there were so many publishers, Rational ClearCase's versioning functionality was particularly useful, and it gave the ability to store every file connected with each version, not just source code. Together, the two provided the team with a single, comprehensive platform for efficient Web content and code management.

Source Code Repository

Rational ClearCase stores all project work in a shared Versioned Object Database (VOB). With access to this centralized repository, team members could work on related development activities simultaneously. This was a great advantage because the team was cross-functional, and widely distributed, with people on both the East and West Coasts. We had content developers, designers, web developers, and software engineers all working in concert, and every one of them had access to the assets stored in the database.

Testing Phase

As soon as Context generated the system requirement, the testing team began creating its own test requirements and plans based on those specifications. Then, as we moved into the load-testing phase, we used Rational Suite TestStudio tools to record basic functional testing for our GUI (Graphical User Interface) and to create virtual users for load and performance testing. We were able to record a wide variety of test cases that could be rapidly executed at any time during the development cycle. The tests had to be repeated many times to verify integration with the back-end systems. But each time we corrected an error, Rational Suite TestStudio's regression-testing capabilities allowed us to quickly regenerate and run all the scenarios to ensure that the bug fix didn't break the rest of the site.

Because of the schedule constraints, the team started load and performance testing in our staging environment, even before the production environment was completed. With user profiles and patterns clearly defined, the team was able to generate a set of virtual user groups that performed various virtual activities. One group, for example, simply browsed the site, while another used its e-commerce capabilities. This testing enabled us to discover and address performance bottlenecks in a controlled and iterative fashion. That, in, turn, helped us to move into production far ahead of where we would normally have been, given how complex the development process was.

Site Profile

Availability (IBM Hosting) Java Server Pages	99.999 percent 5,895
Static Content Pages	8,000
Enterprise JavaBeans	14
Application Subsystems	14
Reusable Design Patterns and Java Components	100
Page Views/Month	4,708,508

Better Quality Despite the Tight Timeframe

Our new Web site offers Rational customers a far better user experience than our old one: greater ease-of-use, faster response time, and a high level of personalization. Visitors can now obtain individualized content on support issues, information on appropriate courses and schedules at Rational University, and more.

The above site statistics clearly shows the stability and success of the web site. This project also demonstrates that Rational tools really do help you get the job done faster without sacrificing quality.

CONCLUSION

The Rational process puts an emphasis on addressing very early high risks areas, by developing rapidly an initial version of the system, which defines its architecture. It does not assume a fixed set of firm requirements at the inception of the project, but allows refining the requirements as the project evolves. It expects and accommodates changes. The process does not put either a strong focus on documents or 'ceremonies', and it lends itself to the automation of many of the tedious tasks associated with software development. The main focus remains the software product itself, and its quality, as measured by the degree to which it satisfies its end-users, and meets its return on investment objective altogether.

We cons ider the Rational Unified Process® to be a well documented and complete but complex methodology. However, unless you have a *real* expert on-staff it is likely that you will not significantly increase your likelihood of success trying to adopt this process. The process is too complex, too difficult to learn, and too difficult to apply correctly. If you don't have an expert, an expert who has actually delivered similar projects using this process, then either hire or rent one and plan to engage the expert for at least one year.

The Unified Process does not capture the sociological aspects of software development and the details of how to truly develop incrementally. In order to complement your Unified Process initiative consider studying the core development pr actices of Extreme Programming (XP).