



White paper on the Agile process : RUP and Scrum

Version : V1.0
Company: CoMakeIT
Date : 10/10/2007

Table of Contents

Introduction.....	3
What is Rational Unified Process or RUP ?.....	3
Characteristics.....	3
What is Scrum?	4
Best practices of the RUP and how Scrum is integrated.....	5
RUP disciplines.....	7
Life cycle of the RUP.....	8
Two Dimensions process.....	9
Tools for tailoring RUP in context of Scrum.....	10
Conclusion.....	10
References.....	11

Introduction

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

- The Rational processes Unified Process(Unified Process)
- Scrum Methodology

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 and how scrum is integrated with in it.

What is Rational Unified Process or RUP ?

*The Rational Unified Process is a **Software Engineering Process**. It provides a disciplined approach to assigning tasks and responsibilities within a development organization.* Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget. The Rational Unified Process is a **process product**, developed and maintained by Rational Software.

*It enhances **team productivity**, by providing every team member with easy access to a knowledge base with guidelines, templates and tool mentors for all critical development activities.* By having all team members accessing the same knowledge base, no matter if you work with requirements, design, test, project management, or configuration management, we ensure that all team members share a common language, process and view of how to develop software.

Its activities create and maintain **models**. Rather than focusing on the production of large amount of paper documents, the Unified Process emphasizes the development and maintenance of **models**—semantically rich representations of the software system under development. The Rational Unified Process is a guide for how to effectively use the **Unified Modeling Language** (UML).

The Rational Unified Process is supported by **tools**, which automate large parts of the process. They are used to create and maintain the various artifacts—models in particular—of the software engineering process: visual modeling, programming, testing, etc. They are invaluable in supporting all the bookkeeping associated with the change management as well as the configuration management that accompanies each iteration.

The Rational Unified Process is a **configurable process**. No single process is suitable for all software development. The Unified Process fits small development teams as well as large development organizations. The Unified Process is founded on a simple and clear process architecture that provides commonality across a family of processes.

Characteristics

The Unified Process has three distinguishing characteristics. These characteristics are

- **Use-Case Driven** - The process employs Use Cases to drive the development process from inception to deployment.

- **Architecture-Centric** - The process seeks to understand the most significant static and dynamic aspects in terms of software architecture. The architecture is a function of the needs of the users and is captured in the core Use Cases.
- **Iterative and Incremental** -The process recognizes that it is practical to divide large projects into smaller projects or mini-projects. Each mini-project comprises an iteration that results in an increment. An iteration may encompass all of the workflows in the process. The iterations are planned using Use Cases.

What is Scrum?

Scrum is an agile software-management process that can help you navigate your project through iterative, incremental software development. Scrum achieved more popularity after the formation of the Agile Alliance. This lightweight process can function as a wrapper, which allows you to combine Scrum with other agile process frameworks, for example, the Rational Unified Process, or RUP.

Scrum offers an empirical approach, which allows team members to work independently and cohesively in a creative environment. It recognizes the importance of the social aspect in software engineering: the name derives from the game of rugby. The process is quick, adaptive, and self-organizing, and it represents a significant change from sequential development processes. Scrum believes that software should not be developed according to the processes used in typical manufacturing -- that is, in a repeating fashion. This repetition makes the input and output parameters more predictive and descriptive, but this is not a helpful goal in today's software engineering. Time to market, return on investment, and the need to build a vision alongside the customer are among the major challenges in modern software engineering. The RUP and other agile software engineering processes are capable of addressing these challenges.

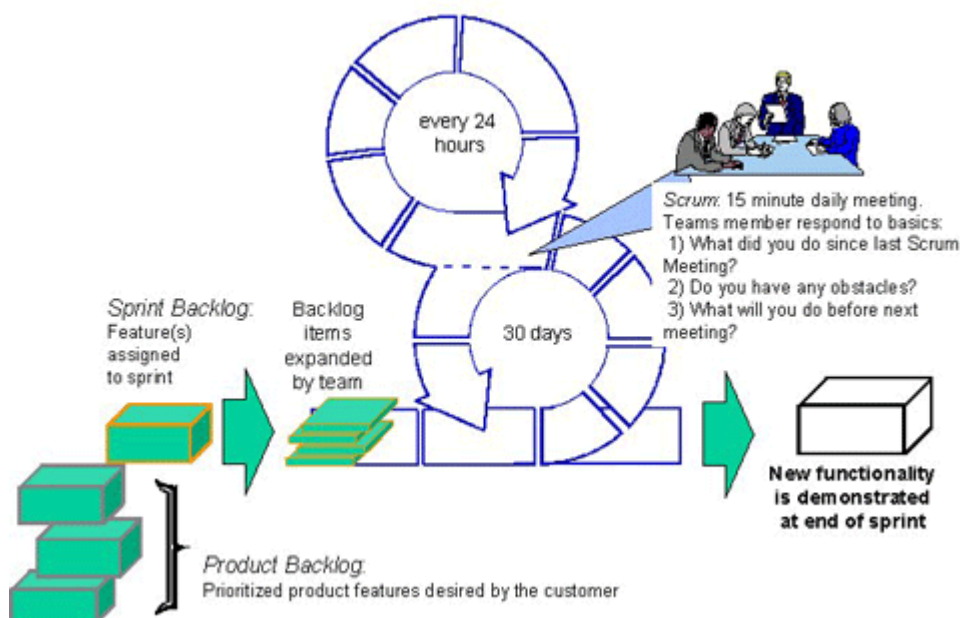


Figure 1. The SCRUM work flow diagram

Best practices of the RUP and how Scrum is integrated

The Rational Unified Process describes how to effectively deploy commercially proven approaches to software development for software development teams. These are called “best practices” not so much because you can precisely quantify their value, but rather, because they are observed to be commonly used in industry by successful organizations. The Rational Unified Process provides each team member with the guidelines, templates and tool mentors necessary for the entire team to take full advantage of among others the following best practices:

1. Develop software iteratively
2. Manage requirements
3. Use component-based architectures
4. Visually model software
5. Verify software quality
6. Control changes to software

Develop software iteratively - Given today’s sophisticated software systems, it is not possible to sequentially first define the entire problem, design the entire solution, build the software and then test the product at the end. An iterative approach is required that allows an increasing understanding of the problem through successive refinements, and to incrementally grow an effective solution over multiple iterations. The Rational Unified Process supports an iterative approach to development that addresses the highest risk items at every stage in the lifecycle, significantly reducing a project’s risk profile. This iterative approach helps you attack risk through demonstrable progress frequent, executable releases that enable continuous end user involvement and feedback. Because each iteration ends with an executable release, the development team stays focused on producing results, and frequent status checks help ensure that the project stays on schedule. An iterative approach also makes it easier to accommodate tactical changes in requirements, features or schedule.

A Scrum project enforces iterative development; the rhythm is defined as 30 days fixed in length. Even though RUP is more flexible in the definition of an iteration, the 30-day window is often used in RUP and determined to be a useful size in length. Instead of defining the length of an iteration in a RUP project, the workload is always aligned with the 30-day length sprint.

Manage requirements - The Rational Unified Process describes how to elicit, organize, and document required functionality and constraints; track and document tradeoffs and decisions; and easily capture and communicate business requirements. The notions of use case and scenarios prescribed in the process has proven to be an excellent way to capture functional requirements and to ensure that these drive the design, implementation and testing of software, making it more likely that the final system fulfills the end user needs. They provide coherent and traceable threads through both the development and the delivered system.

Requirements are managed by the entire team but are under the control of the product owner. In RUP, requirements engineers own the requirements, but this responsibility in Scrum belongs to the product owner, which is often represented by business stakeholders. Requirements management, as in RUP, requires a collaborative effort.

Use component-based architectures - The process focuses on early development and base lining of a robust executable architecture, prior to committing resources for full-scale development. It describes how to design a resilient architecture that is flexible, accommodates change, is intuitively understandable, and promotes more effective software reuse. The Rational Unified Process supports component-based software development. Components are non-trivial modules, subsystems that fulfill a clear function. The Rational Unified Process provides a systematic approach to defining an architecture using new and existing components. These are assembled in a well-defined architecture, either ad hoc, or in a component infrastructure such as the Internet, CORBA, and COM, for which an industry of reusable components is emerging.

The architecture is outlined in the company's strategy and guidelines, or it is proposed by the team itself. The architecture-centric approach promoted by RUP leads to careful consideration and selection of various applicable architectures. In Scrum, this definition is looser and depends on guidelines and standards given to the team. Many agile software projects use modern software tools and programming languages - i.e., technologies in which component architectures are dominant.

Visually model software - The process shows you how to visually model software to capture the structure and behavior of architectures and components. This allows you to hide the details and write code using “graphical building blocks.” Visual abstractions help you communicate different aspects of your software; see how the elements of the system fit together; make sure that the building blocks are consistent with your code; maintain consistency between a design and its implementation; and promote unambiguous communication. The industry-standard Unified Modeling Language (UML), created by Rational Software, is the foundation for successful visual modeling

Scrum does not force teams to build specific visual artifacts, but rather delegates this to the Scrum team. The decision regarding which artifact is created, and to what degree and quality, depends on the set sprint goals. With the industry-wide acceptance of UML, many industry experts apply visual techniques, and the use of visual modeling is most likely a common practice, in both RUP and Scrum.

Verify software quality - Poor application performance and poor reliability are common factors which dramatically inhibit the acceptability of today's software applications. Hence, quality should be reviewed with respect to the requirements based on reliability, functionality, application performance and system performance. The Rational Unified Process assists you in the planning, design, implementation, execution, and evaluation of these test types. Quality assessment is built into the process, in all activities, involving all participants, using objective measurements and criteria, and not treated as an afterthought or a separate activity performed by a separate group.

Iterative, incremental development already offers a great measure of quality and progress for every software project. Focused exclusively on their sprint goals, Scrum teams must demonstrate results at the end of each 30-day cycle (each sprint). In this way, functionality is tested, measured, and demonstrated in an iterative fashion. However, in larger organizations, where quality control is observed from various angles and higher quality control programs are in place, Scrum would benefit from RUP's Quality Assurance Plan, which is controlled by the project manager and is an important component of iterative success.

Control changes to software - The ability to manage change-making certain that each change is acceptable, and being able to track changes is essential in an environment in which change is inevitable. The process describes how to control, track and monitor changes to enable successful iterative development. It also guides you in how to establish secure workspaces for each developer by providing isolation from changes made in other workspaces and by controlling changes of all software artifacts (e.g., models, code, documents, etc.). And it brings a team together to work as a single unit by describing how to automate integration and build management.

As in any modern software development project, changes occur (and are welcome!) during a Scrum project. But Scrum Masters don't interrupt a sprint cycle by introducing changes to the agreed goals. Instead, the Scrum Master captures the changes required, and waits until the end of the sprint to present them as part of the next set of goals. This is essential to the "spirit" of Scrum. Teams work apart from the outside world, and changes are not allowed to affect the current sprint. Otherwise, constant adjustment to the scope of an iteration (the goals of a sprint) will cause the team to lose focus and under deliver the functionality agreed at the beginning of the sprint.

RUP disciplines

The core process workflows are divided into six core “engineering” workflows:

1. Business modeling workflow
2. Requirements workflow
3. Analysis & Design workflow
4. Implementation workflow
5. Test workflow
6. Deployment workflow

And three core “supporting” workflows:

1. Project Management workflow
2. Configuration and Change Management workflow
3. Environment workflow

In the context of the scrum these disciplines can be grouped into two different categories. The first category embraces umbrella activities, which indirectly benefit the system development or organization, such as "Project Management," "Configuration and Change Management," and "Environment," The second category are material activities, including "Business Modeling," "Requirements," "Analysis and Design," "Implementation," "Test," and "Deployment."

Umbrella activities

The umbrella activities are administrative overhead for a Scrum team and should be handled by external stakeholders -- e.g., project management. Except for the Scrum Master, who takes on the role of project manager, the team should only deal with those activities necessary to achieve the Scrum goal and not get bogged down with administrative work.

The RUP discipline "Project Management," and in particular the project plan, offers the software development team a unique opportunity to experience the nature of Scrum. Once the Scrum Master executes the project, a RUP project will feel like a Scrum project to the team. Internal organizational, administrative, verbal, and non-verbal communication should be eliminated for the Scrum team and handled by the Scrum Master (in RUP terms, the project manager). A RUP project plan would be the ideal artifact to:

- Outline the Scrum rules applied in the project.
- Define the milestones.
- Describe ongoing sprint goals.
- Present risks.
- Present estimates and estimation techniques.

In Scrum, the Scrum Master serves as a mediator between management and the Scrum team. The master manages the scope and functionality of the project, and communicates internally and externally, but does not manage at the micro-activity level. This difference of responsibilities and duties needs to be defined in the project plan. In addition, someone in the Scrum team usually plays the role of an architect.

In a Scrum-managed project, the "Development Case" in the RUP Environment discipline is affected by constant revision. The Development Case lists both mandatory and optional artifacts as a roadmap to success. But since the Scrum team works as an independent unit, all the artifacts that are possibly produced by the team should be declared "optional" to reflect the control that the team has in a Scrum environment. Alternatively, you could remove all the artifacts from the development case that are related to the Scrum team. The generic development case template in RUP would serve this need. Over time though, as the Scrum team gains a rhythm of useful documentation during the sprints, the actual development case will take shape.

Material activities

The remaining six RUP disciplines with imminent influence on software engineering offer the Scrum team a set of optional artifacts and activities. The RUP process framework can serve as a guide, providing useful steps and artifacts to consider. On an ad hoc basis, the team can decide to create variations or modifications in each of those disciplines, as long as they serve the objectives of their sprint goal. Newer or less experienced software engineers can use RUP as a map for communication and structured context.

Life cycle of the RUP

The Unified Process consists of cycles that may repeat over the long-term life of a system. A cycle consists of four phases: **Inception, Elaboration, Construction and Transition**. Each cycle is concluded with a release, there are also releases within a cycle. We will briefly review the four phases in a cycle

Inception phase - During the inception phase the core idea is developed into a product vision. In this phase, we review and confirm our understanding of the core business drivers. We want to understand the business case for why the project should be attempted. The inception phase establishes the product feasibility and delimits the project scope.

Elaboration phase - During the elaboration phase the majority of the Use Cases are specified in detail and the system architecture is designed. This phase focuses on the "Do-Ability" of the project. We identify significant risks and prepare a schedule, staff and cost profile for the entire project.

Construction phase - During the construction phase the product is moved from the architectural baseline to a system complete enough to transition to the user community. The architectural baseline grows to become the completed system as the design is refined into code.

Transition phase - In the transition phase the goal is to ensure that the requirements have been met to the satisfaction of the stakeholders. This phase is often initiated with a beta release of the application. Other activities include site preparation, manual completion, and defect identification and correction. The transition phase ends with a postmortem devoted to learning and recording lessons for future cycles.

Scrum principles can be applied during the entire lifecycle of the project, the Elaboration and Construction phases, which consume most project resources, are best suited to the Scrum approach. Compared to the Inception phase, the Elaboration and Construction phases allow the team to work independently toward tangible goals. Scrum sprint planning, review, and daily meetings could be very powerful instruments during these two phases. If the project team consists of various business analysts in Inception or even prior to the start of the software project, Scrum can also be beneficial during Inception. To better ensure successful deployment of the project, the entire Transition phase could be executed as its own project, again with Scrum deployment team members. Scrum project management techniques may not seem ideal for your entire RUP project, but they can definitely be applied in various forms throughout the project. The project plan can reflect these decisions and can present the reasons why some phases exclude Scrum techniques.

Two dimensions process

The process can be described in two dimensions, or along two axis:

- the horizontal axis represents time and shows the dynamic aspect of the process as it is enacted, and it is expressed in terms of cycles, phases, iterations, and milestones.
- the vertical axis represents the static aspect of the process: how it is described in terms of activities, artifacts, workers and workflows.

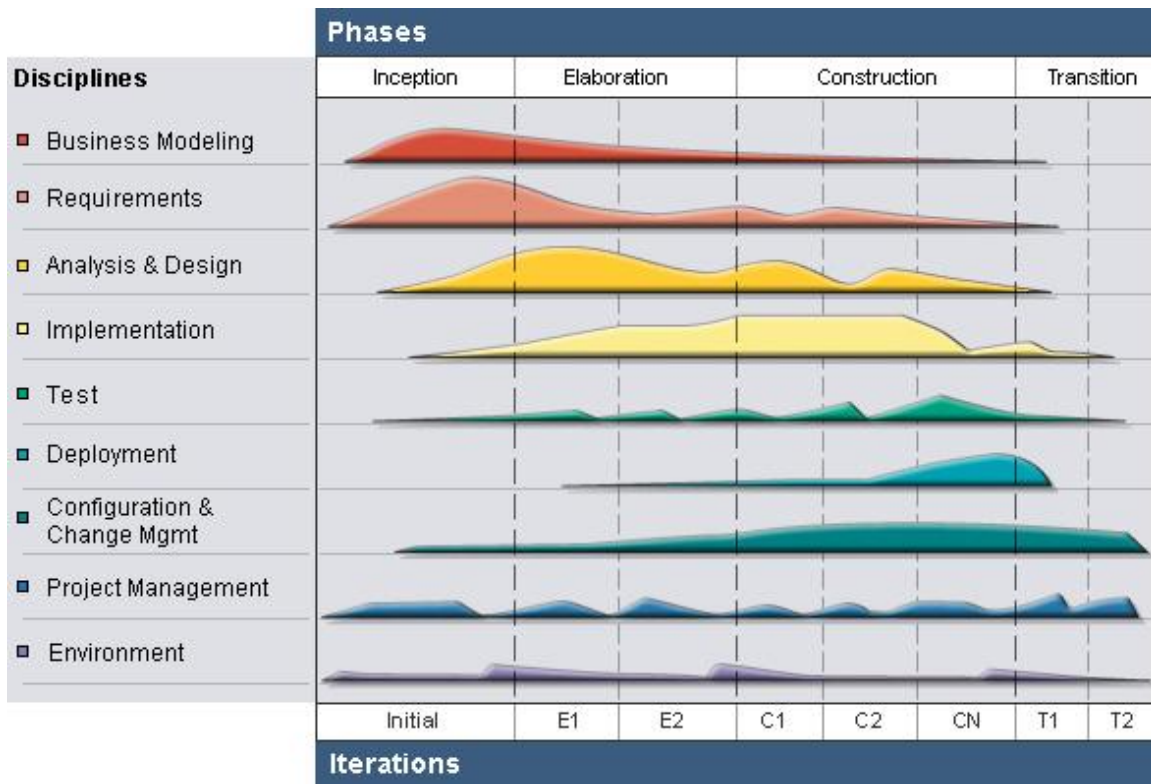


Figure 2. The Iterative Model graph shows how the process is structured along two dimensions.

Tools for tailoring RUP in context of Scrum

Used in conjunction with the IBM Rational Process Workbench, RUP offers enormous advantages as a process framework that can be fully customized. The Workbench product consists of RUP Modeler, RUP Organizer, and RUP Builder.

RUP Modeler is a visual tool used to modify and manage the core model. The process engineer can make modifications to the existing RUP framework to include, in our case, Scrum concepts. Dependencies between Scrum roles and artifacts are easily managed with the RUP Modeler. With the RUP Organizer, the process engineer describes the modified Scrum workflows, which helps project team members better understand the Scrum process. To share process changes within the organization, the process engineer releases the process via RUP Builder. This tool ties together the process, artifacts, and descriptions and creates a navigable Website.

Conclusion

Scrum and RUP can enhance each other in various ways. The RUP process framework can be tailored and customized to your project needs; the RUP development case and the software development plan, for example, can reflect your decision to use Scrum techniques. In my experience, the Scrum team will fall into a rhythm of activities and artifacts, which yield a most effective development case. This output of artifacts can serve as an input document for other projects within your organization.

In general, RUP satisfies organizational demands by bringing a structured and proven process framework to the table, and Scrum patterns can add additional dynamics to the project. For example, Scrum management patterns can be easily added to your current or future RUP projects. Addressing the social side of software engineering, Scrum's proven process patterns can offer immediate benefit in requirements management, change control, and project management. With RUP artifacts guiding the development process and project documentation, Scrum team members -- especially new or inexperienced ones -- are better equipped to avoid software development pitfalls.

References

RUP in the dialogue with Scrum by Joe Krebs, IBM Rational.

Rational Unified Process - Best Practices for Software Development Teams by A Rational Software Corporation.

The Menlo Institute LLC - The Rational Unified Process.

<http://www-128.ibm.com/developerworks/rational/library/feb05/krebs/>

<http://www.menloinnovations.com/>