RUP in the dialogue with Scrum

Level: Introductory

Joe Krebs
Senior IT Specialist , IBM Rational
15 Feb 2005

from The Rational Edge: This article introduces the agile software development process known as Scrum. The author presents techniques on how software development teams can add Scrum ideas to an existing RUP environment.

As you may know, the Rational Unified Process®, or RUP®, is a widely used software process framework that can be tailored to your process needs and can accommodate other techniques. Scrum is a collection of interesting project management patterns used to wrap agile software projects. This article introduces some important characteristics of Scrum and presents techniques on how you can add Scrum ideas to your existing RUP environment. I provide a glossary for Scrum and agile terms in the sidebar, and I use an asterisk (*) after the first occurence of these terms in the text below.

What is Scrum?

Scrum is an agile* software-management process that can help you navigate your project through iterative, incremental software development. Introduced in 1995 by Advanced Development Methodologies, Inc.[1], Scrum achieved more popularity after the formation of the Agile Alliance in 2001. 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, and refers to "*a rugby play in which the forwards of each side come together in a tight formation and struggle to gain possession of the ball when it is tossed in among them.*"[2]

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.

How it works

So what makes Scrum different from other processes? Assuming you are a first-time observer of a Scrum project, the most noticeable difference is the daily stand-up meeting, usually conducted at the same time and in the same team room. During the meeting, which is often itself referred to as "the Scrum," every team member comments only on the following three points:

- What have you done since the last Scrum meeting?
- What will you do between now and the next Scrum meeting?
- What got in your way of doing work?

Scrum team composition

Because a Scrum team is up to seven members large, the meeting should not consume more than 15 minutes. The Scrum Master* runs the meeting and is in charge of the overall project success; this person monitors the answers and removes impediments identified during the meeting. The Scrum Master provides instant answers or guidelines in the meeting to overcome an impediment, which will keep the team moving toward its goals. Scrum meetings should not become project meetings, but instead serve as quick briefings for the team. If issues do not disappear, the team members should challenge the Scrum Master or the group members.

Team members (pigs*) are the only participants allowed to speak during the Scrum meeting, but silent visitors (chickens*) are welcome. For project teams with more than seven members, Scrum advises breaking into multiple teams rather than increasing the team size. Teams can be formed based on functionality, architectural subjects, or applications, including sub-applications. Teams can then work in parallel with each other, synchronized by the Scrum Master through Scrum meetings. Scrum even offers solutions for team members working remotely.

A Scrum team should not only be seen as a group of programmers, but should be assembled using roles in various backgrounds, including business analysts, designers, programmers, and testers. Often times, members can play multiple roles; the energy and efficiency is all about the right mix.

Project planning

Scrum iterations are called sprints* and are 30 days in length. As in RUP, where the iteration is time-boxed between 2 to 6 weeks, each sprint ends with executable, testable code.

Glossary

Agile development
Iterative software engineering with open scope, focusing on working software over comprehensive documentation.

Chicken
Observer of the Scrum project.

Defined process
Predefined process flows generating the predicted results.

Discipline
Ongoing workflow in a RUP project with specific emphasis depending on the state of the project.

Empirical process
Frequent inspection and adaptation of activities to meet desired outcomes.

Iterative development
Software engineering activities are performed frequently in short and repetitive cycles, with feedback from stakeholders after each iteration.

Pig
Committed member of the project.

Product backlog
Feature repository

Scrum Master
Project manager

Sequential (waterfall) development
Software engineering activities are executed in one activity, one after another, usually separated by fixed, pre-defined milestones.

Sprint
An iteration in RUP projects, 30 days with scoped features. Sprints and iterations result in workable, demonstrable software.

Sprint backlog
Feature and requirements scope for the sprint.

Umbrella activity
Administrative task, which is not directly linked to the actually software product.

The product owner owns the product backlog*, and controls and prioritizes the functionality, but the Scrum team estimates the effort. All stakeholders together, including the Scrum team and the product owner, inspect the product backlog and decide what functionality will be tackled first based on priority. By comparison, iteration planning in RUP is also based on risk in addition to priority.

The sprint objectives defined by the team now serve as guidelines for progress control. Once the sprint is started, all external communication to the team must be initiated through the Scrum Master. The Scrum Master must give the team the ability to work without distractions on the given objectives.

The Scrum team has its own sprint backlog*, which records more detailed tasks for accomplishing the goal of the sprint. After the team has gained more knowledge about the functionality of the sprint, the members may adjust the original estimate captured in the product backlog and expand the product backlog to include details learned during the sprint. These metrics are useful for Scrum progress tracking.

Because the Scrum team is tied through the organization by a daily Scrum meeting and a monthly sprint planning and sprint review meeting, a vertical visibility through the entire organization exists by default. This makes organizational problems and challenges visible. Because all team members observe the project first-hand, communications are short, quick, and efficient. The team is self-organized and focused on the sprint goals, which brings out the best of each individual on the team. The Scrum Master acts as a clearinghouse for issues and communication rather than a "boss" controlling the team.

During the sprint review meeting, which is a half-day long, the team presents the completed functionality to stakeholders. The team focuses on the given sprint goals to demonstrate the accomplishments.

The backlogs, sprint planning and review, management engagement, daily Scrum meetings, metrics tracking, and other Scrum techniques are based on process patterns (Schwaber and Beedle, 2004) primarily used for managing software projects. These patterns have been successful in the past for small and large projects and different business domains. If you are ready to apply the Scrum process patterns combined with the power of RUP, the next section will give you a better feeling for the touchpoints.

Scrum meets RUP

Current software projects using RUP can easily benefit from Scrum principles, even for projects that are already started. The first area of investigation is the RUP process model and how Scrum impacts the best practices in RUP, the four phases (Inception, Elaboration, Construction and Transition), and the nine disciplines within RUP. The relationship of the RUP phases and disciplines is shown in Figure 1.

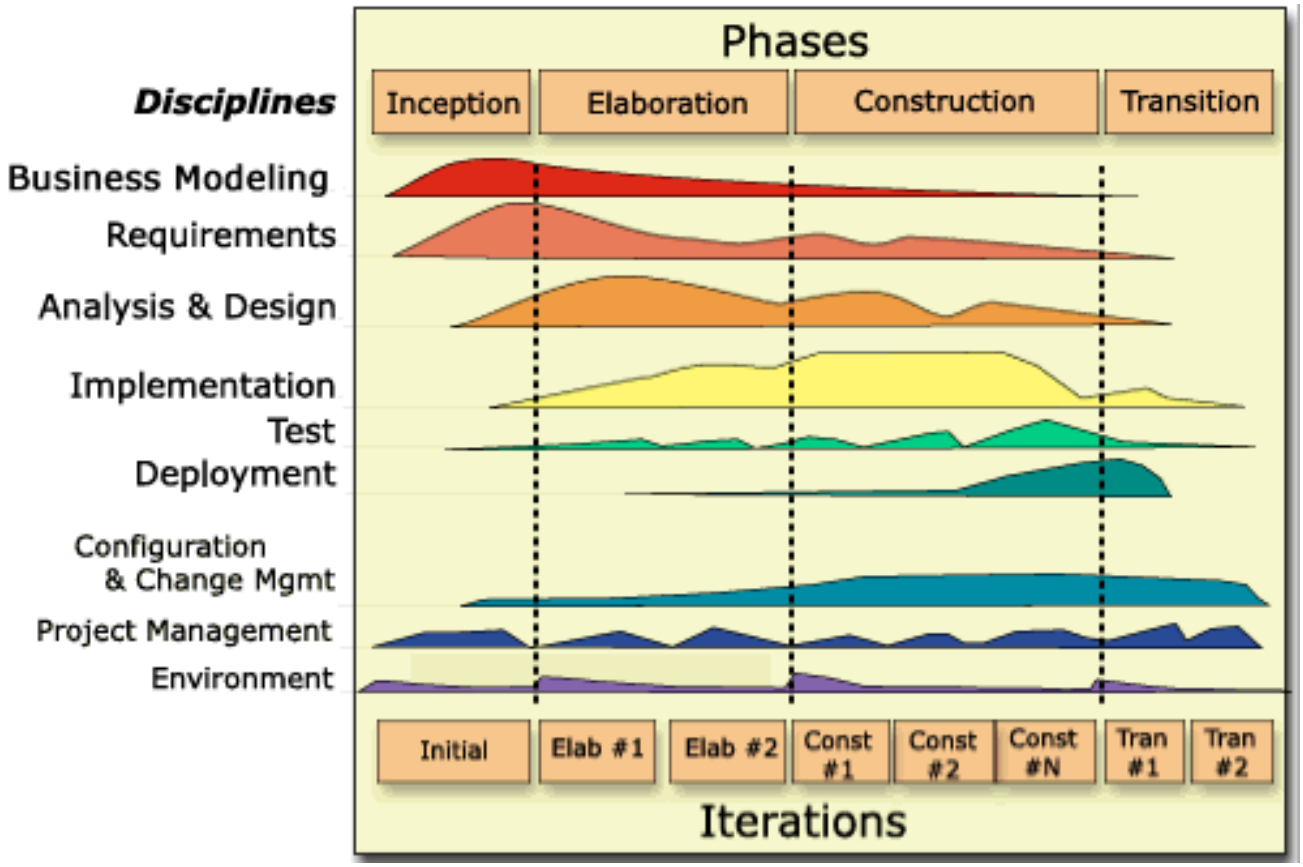

Figure 1: Each of the disciplines in RUP, listed at left, is correlated to the four project phases.

While most disciplines are involved in each phase, the chart shows varying degrees of involvement by phase.

RUP: Embracing the best practices in software development

Let's begin first with the six best practices outlined in the RUP process framework.

Develop iteratively

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

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 architecture

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.

Model visually

RUP promotes visual modeling. Even the RUP framework itself is guided by visuals, many of them in UML. Like RUP, 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.

Continuously ensuring quality

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.

Manage change
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 underdeliver the functionality agreed at the beginning of the sprint.

Scrum and the RUP disciplines

To discuss the nine RUP disciplines* in the context of Scrum, I have grouped the disciplines in two different categories. The first category embraces umbrella activities*, which indirectly benefit the system development or organization, such as "Environment," "Project Management," and "Configuration and Change Management." 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.

The four phases in a RUP project represent different project styles in the software engineering process. While 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.

Tools for aligning RUP with 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

Agile Alliance (2004) http://www.agilealliance.org

Ken Schwaber and Mike Beedle, *Agile Software Development with SCRUM* Prentice Hall, 2002.

Scrum (2004). See http://www.controlchaos.com/ The diagram at http://www.controlchaos.com/about/ is particularly helpful for visualizing the flow of a Scrum project iteration.

Notes

[1] See the Scrum Website at http://www.controlchaos.com/

[2] According to Merriam-Webster's *Collegiate Dictionary*, 11th Edition, Merriam-Webster, 2003.

About the author

Jochen (Joe) Krebs (jkrebs@us.ibm.com) is a senior IT specialist for the Rational Software Brand within the IBM Software Group. He is responsible for successful enablement of Rational products and services for clients in the financial sector. Prior to joining IBM Rational, he worked as an instructor and senior consultant with a focus on project management, requirements management, software engineering processes, and object-oriented technologies using Smalltalk and Java. He holds his MSc in computing for commerce and industry at the Open University.