Agile Record

The Magazine for Agile Developers and Agile Testers

www.agilerecord.com free digital version made in Germany

January 2010

issue 1

Scrum and RUP - A Comparison Doesn't Go on All Fours

by Remi-Armand Collaris and Eef Dekker

Introduction

Many developers who have embraced Scrum or any other 'agile way of working' perceive the Rational Unified Process (RUP) as the opposite of what they see as useful and fruitful. So, we have heard many professionals in the field saying things like: 'If you do Scrum, you have no room nor need for RUP', 'You should always apply Scrum exactly, otherwise it doesn't work. So it is dangerous to try to apply more.' On the other hand, developers who are used to RUP or another process-based development method perceive Agile software development as being unstructured, undisciplined and lacking any form of documentation.

In our experience, Scrum and RUP do not collide but rather complement each other. In the following, we show

- what (mis)interpretations of both Scrum and RUP are behind the collision view, and
- what evidence there is to sustain the complementary view.

Popular but flawed views of RUP

The Rational Unified Process is a massive development process, which you need to swallow completely. Indeed many organizations tried to apply full-blown RUP without much tailoring. Not surprisingly, they came to a complete standstill pretty quickly. Nowadays, as intended all along, the first key principle of RUP is to 'adapt the process'. That is, to apply only those parts which you need in your organization, and to adapt those parts so they fit in your organization.[1]

Some views that can be traced to the same misunderstanding include:

- RUP lets you produce massive heaps of paper (or electronic counterparts), for as many as 128 work products are prescribed. It was never intended for any organization, no matter how large, to apply each and every work product. Choosing the right work products to support development and production is an important part of adapting the (development) process.

- If you apply RUP, you must have a large team, for RUP prescribes 33 different roles. Roles are meant to be like 'caps' a person wears. Various roles can be played by the same person, just as one role can be played by more than one person. The ultimate consequence of this is that you can have a team of one person, playing all roles. Although this is seldom the case, it is not precluded by anything in RUP. Furthermore there is nothing that keeps you from describing fewer roles in your organisation's RUP implementation.

- The RUP is a complex process, you are bound to get lost in it. Unfortunately, this one seems to hold for IBM's distribution of RUP for large projects. However, you are supposed to tailor RUP for your organization or project. The result can be clarified in a Development Case (a work product describing your development process). Such a Development Case can include a Responsibility Matrix and Workflows / Work product flows[2] to illustrate the process. There are also tools like Rational Method Composer or its open-source counterpart Eclipse Process Framework (EPF) that can help you tailor RUP to your needs.

- RUP is just a disguised waterfall process. Admittedly, the RUP disciplines are modelled after waterfall phases and RUP doesn't forbid you to divide your project in iterations of 3 months each. However, the opening picture of IBM's distribution of RUP shows multiple iterations per phase and shows all disciplines working together throughout the lifecycle. Furthermore, the fourth key principle of RUP to 'deliver value iteratively', makes it impossible to see a waterfall project as a valid RUP implementation.

Popular but flawed views of Scrum

Scrum is a complete software development methodology that should be used "as is"# without any adaptations or additions.

Scrum is an Agile process framework for managing Agile teams. With only 3 roles, 3 important meetings and just a handful of work products it sets up an easy-to-learn process for managing incremental software delivery, guided by business needs. It helps the team to deliver value to the customer early and gives complete openness to all stakeholders concerning tasks that are being done, progress of the team and impediments that keep the team from performing at its best. It does, however, focus only on the management aspects of Agile software development and does not comprise development practices.

Some views that can be traced to the same misunderstanding include:

Scrum is so wonderful and so widely applicable that you don't need any of those thick-headed processes anymore. We agree fully with the first part. Scrum is wonderful and most lessons it teaches can be applied generally. However – as we hope to show – you can and must complement Scrum with something else.

Once you think Scrum in itself does it all, you forget that most of the time Scrum comes with a whole bunch of implicit or explicit assumptions. For example, Scrum assumes there is a vision for the project and defines a product backlog, but Scrum in itself does not say how these work products are filled initially. This is no rocket science, but our point is that Scrum leaves many open ends to be filled in – be it by common sense or by some other methodology as long as you stay Agile (focused on the situation at hand).

Another example: the product owner represents all stakeholders. Which stakeholders and what their stake may be, is outside Scrum's scope. Of course, it is good to have a product owner as the single point of customer responsibility. Nevertheless, if it somehow belongs to the project scope to get your stakeholders aligned, some guidance on how to do this might be welcome.

Scrum advocates cross-functional teams; the team as a whole is responsible. Therefore we don't need role descriptions for team roles. We have seen that a team, in which individuals really take responsibility for the team result, performs a lot better than one in which individuals take responsibility only for stuff within the scope of their own role. The point is that individuals should not stick to only one role. They should be able and willing to switch roles if needed. For example, the individual who has the role of Analyst can switch to the role of Tester. This is just another way of saying that John (normally being the Analyst) takes up some testing tasks. It is still valid and useful to have good role descriptions. People can see what is expected if they take up that role. A role is a cap, and you should be able to judge if that cap fits you.

Furthermore, roles and associated skills and competencies are indispensable when assembling a development team. Sharing responsibilities within the team doesn't make it less important to put together all skills and competencies that will be needed to get the job done.

Compatibility

Here follow a few quotations or observations on points in which Scrum and RUP resemble each other:

Scrum (from the Scrum Guide[3])	**RUP** (from 7.5 large projects)
"Scrum employs an iterative, incremental approach to optimize predictability and control risk."	"Deliver Value Iteratively" and "Attack major … risks early" (from Key Principle 4)
"The ScrumMaster helps the Team understand and use self-management and cross-functionality."	"Collaborate Across Teams" and "Create self-managed teams" (from Key Principle 2)
"Each Scrum Team member applies his or her expertise … the resultant synergy … improves code quality and raises productivity."	"Focus continuously on quality" and "Ensure team ownership of quality for the product." (from Key Principle 6)
"By the end of the Sprint retrospective, the Scrum team should have identified actionable improvement measures that they implement in the next Sprint."	"Adapt the process" and "Improve the process continuously" (from Key Principle 1); "The Iteration Assessment captures…lessons learned and process changes to be implemented" (from Process Essentials 7)

Scrum and the complementary view

Although the core of Scrum is very simple, it is not completely self-explanatory and has a limited scope. The first point is illustrated by the many books and articles dedicated to the application of Scrum. Scrum gives you the simple basics that all can easily understand. Yet every situation is different so you need to adapt the process.

The second point becomes clear by looking at its scope. Scrum is about managing the development process:

- The Team is self-managing and self-organizing
- The Product Owner manages the Product Backlog
- The Scrum Master manages the Scrum process.

Any rules not stated in Scrum are supposed to be figured out by the Team and all others involved in the project. They may consult best practices, such as those described in other (Agile) methodologies.

Scrum is often used in conjunction with eXtreme Programming (XP), an Agile methodology providing day-to-day practices for developers. RUP can complement them with guidance on organizing iterations, working towards a release to production. XP is mainly concerned with requirements, architecture, development and testing, whereas RUP also provides practices for other software development disciplines. Furthermore RUP supports higher levels of ceremony (if needed) than Scrum and XP do.

Scrum and RUP both provide guidance on team management,

but are not focused on project management. For this Scrum could be pulled to a higher level (Scrum of Scrums) or the current management practices – like those described in Prince 2 – can be used.

We have done several projects with a combination of Scrum and RUP, and are writing a book on it which we hope to publish in the near future. You can take a preview look at: www.ScrumUP.eu/preview.

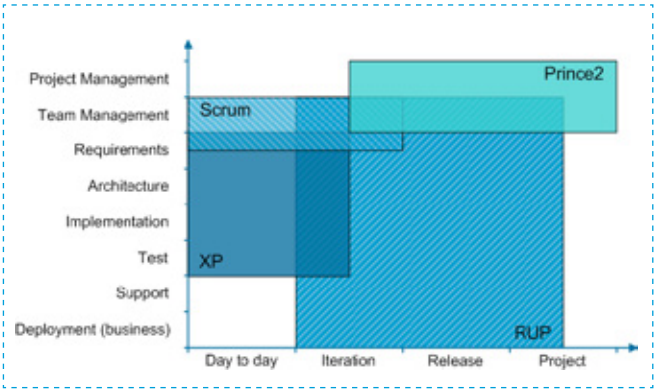

Figure 1: The Focus of Scrum, XP, RUP and Prince 2

In figure 1 we have visualized the scope of Scrum together with that of RUP, XP and Prince 2. On the x-axis we see time ranges: Does a method focus on day-to-day tasks, or is it focused around iterations, or perhaps releases or the complete project? We see that Scrum is focused on day-to-day tasks, and RUP not at all. We find iterations to be an overlapping area of concern, whereas Scrum does not say much about releases or a project as a whole. On the y-axis we see several 'disciplines'. Scrum is mostly concerned with team management and a little bit with requirements. RUP has a lot to say about other disciplines as well, although of course there are areas which RUP does not cover.

One could well place XP as a set of day-to-day practices in the field of architecture, implementation, requirements and test. In this way, we can show that Scrum and XP complement each other in the day-to-day area, and that even Scrum and XP together are nicely complemented by RUP. ■

1 For guidance on this point, see our article Tailoring RUP made easy: Introducing the Responsibility Matrix and the Artifact Flow , published in the September 2006 issue of The Rational Edge, http://www-128.ibm.com/developerworks/rational/library/sep06/collaris_dekker_warmer/index.html

2 See our implementation of RUP: www.rupopmaat.nl (Dutch), and the publication we're working on: www.ScrumUP.eu. See especially the workflows and how they bring everything together.

3 The Scrum Guide can be downloaded at http://www.scrumalliance.org/resources/598.

> About the author

Remi-Armand Collaris
is a consultant at Ordina, based in The Netherlands. He has worked for a number of financial, insurance and semi-government institutions. In recent years, his focus shifted from project management to coaching organisations in adopting Agile using RUP, Scrum. An important part of his work at Ordina is contributing to the company's Agile RUP development case and giving presentations and workshops on RUP, Agile and project management. With co-author Eef Dekker, he wrote the Dutch book RUP op Maat: Een praktische handleiding voor IT-projecten, translated as RUP Tailored: A Practical Guide to IT Projects, second revised edition published in 2008 (see www.rupopmaat.nl). They are now working on a new book: ScrumUP, Agile Software Development with Scrum and RUP (see www.scrumup.eu).

Eef Dekker
is a consultant at Ordina, based in The Netherlands. He mainly coaches organizations in implementing RUP in an Agile way. Furthermore, he gives presentations and workshops on RUP, Use Case Modeling and software estimation with Use Case Points. With co-author Remi-Armand Collaris, he wrote the Dutch book RUP op Maat, Een praktische handleiding voor IT-projecten, translated as RUP Tailored, A Practical Guide to IT Projects, second revised edition published in 2008 (see www.rupopmaat.nl). They are now working on a new book: ScrumUP, Agile Software Development with Scrum and RUP (see www.scrumup.eu).