Agile Record

The Magazine for Agile Developers and Agile Testers

April 2010
issue 2

www.agilerecord.com free digital version made in Germany

Software Process Improvement Using Scrum, RUP and CMMi: Three Reasons why this is a Bad Title

by Remi-Armand Collaris & Eef Dekker

Under current economic circumstances, "software process improvement" is hot again. It's all about changing the way we work, and doing things right for a change. To decide what's right, people turn to popular software development methodologies like Scrum and RUP or to a maturity model like CMMi.

Software process improvement is often seen as a tool to shorten time-to-market, increase quality and increase customer satisfaction. In order to reach these objectives, however, it is not enough to look only at software development. In order to reach the three objectives mentioned, we have to look beyond the software development process and get the business involved in the creation of software that will deliver value for them. So, the first problem with the title lies in the word "software".

Another problem is that the process is not the first thing that should be changed, but the mindset that led to the current process. For any improvement initiative to be effective, we need to involve the people that execute the process. So, the second problem with the title lies in the word "process".

Initially, the implementation of a methodology or the attainment of a certain level of maturity may work fine as a means to reach our objectives. Once this is formulated, we often see a loss of focus: the means become the end, and everybody strives to implement the methodology or reach the maturity level, losing sight of the objectives. So the third problem lies in the words "using Scrum, RUP and CMMi".

In this article we will show how the mindset needs to be changed, in order to successfully improve on realizing value for the business. Furthermore we will show how this change in mindset can be initiated and consolidated using Agile and iterative practices from Scrum and RUP. Once such a mindset takes root, the organization's maturity will increase. Finally, we propose a way to keep ourselves focused on the objectives rather than on the means to reach them.

Creating a change of mindset

You need people to really feel responsible for the product and for the process used to make that product – responsible not only on a personal basis but as a team. This is true for the software development team, but should also include stakeholders. This means you need to find stakeholders who really hold a stake in the process improvement and involve them. You need to lower the fences between customer and supplier, so that they can at least look at each other instead of looking at the fence between them. This is even more true in outsourcing and offshoring situations, in which the contract should stimulate cooperation. You also need commitment from stakeholders so that the time the stakeholders need to invest will indeed be available. These are delicate points in most organizations, but they are crucial for success. Below, we present a number of Agile practices that can help to reach this mindset.

Start iterating so that people can build trust on the basis of results.

This point is so important that we will discuss several aspects of the iterative way of working:

- Get subject matter experts from the business involved within the development team. Our experience is that once they have been actively involved in a software development effort, they become enthusiastic and will demand to always be involved in this way.

- Put a relentless bias on timeboxing, but do it in combination with a 'neutral', non-punishing culture. Once you start slipping in time, the relevance of timeboxing will be diminished. It is counterproductive to 'punish' individuals for not having x, y or z ready at the end of the timebox. It is far better to assume each individual has done everything in their power to contribute to the success of that iteration. At the iteration assessment, measures to avoid downsizing or even failure of delivery can be discussed and implemented in the next iteration.

- Shorten the time between detailing requirements and having software delivered. This will greatly improve business involvement. The subject matter experts and other stakeholders will quickly see their direct influence on the result.

Create effective communication channels and facilities.

An important ingredient for successful cooperation is communication. The project setting should foster mutual understanding of objectives, needs and backgrounds. This leads to a feeling of team responsibility, where every member knows what is needed and how he can contribute.

It's all about changing the way we work, and doing things right for a change.

Practices involved:

- User stories or use cases as a means for communicating requirements
- One project team room
- A planning meeting at the start of each iteration, in which the goals and priorities are clarified and in which the team decides on its commitment for the upcoming iteration
- Daily stand-up meetings
- Working in pairs (not only the developers)
- An evaluation session (retrospective or assessment) at the end of each iteration

Create transparency in work products and their status.

This ensures that everyone interested has access to the same status information. You may use the following tools to create this transparency:

- Daily visual updates on progress (task board and burn-down chart)
- A visible list of issues that block progress or slow the team down (impediments list)
- A demo of completed working software at the end of each iteration

It is no use to hide a problem. You foster trust by showing your problems as well as your ability to solve them. Even if you can't solve it, it is better to clearly say so, and why. This stimulates people from outside the team to get involved in helping the team out. Doing all this will establish a transparent reporting structure with a minimal amount of work.

Look for 'in between' responsibilities and solve them.

These are responsibilities which nobody takes up because they are not perceived to be 'our' responsibility. If two parties think of a particular responsibility as 'not ours' and there is no check whether it is covered, we have orphan responsibilities. For example, imagine a service call, in a complex service-oriented environment which involves various departments, does not return an answer. For the past two weeks, several persons have looked into the problem and found 'their' part to be okay, whereas the company spirit should inspire them to form a small taskforce committed to cooperate until the problem is solved.

Create a contract that fosters cooperation.

Contracts between supply and demand organizations often focus on the scope of the system to be built and do not explicitly address cooperation. As a result, conflicting interests arise, and it is hard to accommodate change. To create value for the business, however, accommodating change is crucial. Therefore focus your contracts on cooperation and include a reward system, which gives clear signals that loyalty to the project is the same as loyalty to the individual organizations.[1]

The same point is true at the level of individuals: let the interests of individuals coincide with the interests of the project. If a software developer will be evaluated not only on the quality of the code, but also on the usability of the product, this will enhance his awareness of responsibility for that product. If a tester will be evaluated not on the amount of bugs found but on the quality of the product, his behavior will change. Make sure that responsibility for the product as a whole is in line with personal concerns for every role involved.

Keep focus on the objectives

While implementing a methodology or trying to reach a higher maturity level, people involved in the improvement initiative often lose sight of the objectives they were trying to reach. An improvement plan will help keeping focus on these objectives. Elements of an improvement plan format, that we have had good results with, are:[2]

- description of the current process
- problem statement
- stakeholders
- stakeholder needs
- a prioritized list of improvement actions
- planning

First, you need to know what the problem is. Why does an organization want to improve their software development process? They want to improve, because they have encountered problems in their current process. Investigate these problems. Start with a high-level inventory in something called a problem statement. The problem statement includes the persons or groups affected by that problem, its impact and the solution as perceived by the stakeholders.

Stating the problem in this way automatically leads to a clear view on who the stakeholders are: they are the ones that have these problems. Then, stakeholders hold a stake. What is that stake, or in other words, what is their interest in the software process improvement?
Stakeholders have needs. What are they, and how should they be dealt with? What improvement actions do you propose to meet these needs? For every suggested improvement action, the plan should describe:

- why you want to change something;
- when the action should be executed;
- what the desired result is;
- who is waiting for that result and willing to pay for achieving it;
- how you can measure whether that result has been reached.

Once you have all this, time has come to think about how the improvement actions can be planned in time.

Having such a plan helps to avoid an over-enthusiast introduction of 'the new golden hammer method'. It is good to have a clear idea of what the problem is, and why you think this problem can be diminished or solved by the proposed method. What does your proposal contribute to obtaining the right mindset? We want to make your software improvement effort as successful as possible. This requires a little thought at the outset. Make it clear that you're about to make a good investment.

Conclusion

Software improvement efforts should be guided by a very down-to-earth practical plan, and should have a very clear focus on stimulating a mindset rather than on introducing a new process, however promising that new process may be. We have given a quick glance at elements that help create the right mindset of responsibility and cooperation:

- Start iterating so that people can build trust on the basis of results.
- Create transparency in work products and their status.
- Create effective communication channels and facilities.
- Look for 'in-between' responsibilities and solve them.
- Create a contract that fosters cooperation.

If this mindset takes root, increased customer satisfaction, increased quality and a shorter time-to-market will come along. ■

[1] See the excellent book of Mary and Tom Poppendieck, Implementing Lean Software Development: From Concept to Cash, Addison Wesley 2008, p. 216-217.

[2] You can download the template at www.scrumup.eu/downloads.html.

> About the authors

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 organizations 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).*