Journal of Universal Knowledge Management, vol. 1, no. 2 (2006), 103-111
submitted: 15/5/06, accepted: 15/6/06, appeared: 28/9/06 © J.UKM

Learning to Tailor Documentation of Software Requirements

Eric Knauss
(FG Software Engineering, University Hannover, Germany
eric.knauss@inf.uni-hannover.de)

Daniel Lübke
(FG Software Engineering, University Hannover, Germany
daniel.luebke@inf.uni-hannover.de)

Thomas Flohr
(FG Software Engineering, University Hannover, Germany
thomas.flohr@inf.uni-hannover.de)

Abstract: In software projects, it is important to determine the right amount of requirements documentation. If the documentation is not detailed enough, it is an insufficient base for contracts. If it is too long, it is expensive to maintain. Therefore the amount of documentation should be adjusted for each project. Often this step is omitted, partly to avoid the effort of tailoring, but mainly because project members do not know how to tailor or even are afraid of the consequences and associated risks. In this paper we share our experience in addressing both aspects with a mixture of organizational and individual learning. We successfully used our approach in university teaching and in parts with industrial partners.

Keywords: documentation tailoring, experience base, agile hour, FLOW, university teaching
Categories: D.2.1, D.2.9

1 Introduction

Process-driven software projects requirements (e.g. the V-Model, [IABG, 1997]) aim for a very high documentation standard: Requirements should be fully and clearly documented and possibly traceable. This leads to very large documents which cannot be easily managed, used and maintained. In practice we encountered that this high standard is often the goal of management and quality assurance teams, but many attempts to reach that standard normally fail. For example, in one company requirements are written down but only *after* they were communicated to other stakeholders. This way documents become only a persistent memory and loose their role as a communication channel: It seems that the demanded theoretical standard cannot be reached in practice.

Therefore, "agile" methods, like eXtreme Programming (XP) [Beck, 2000], try to capture requirements differently: Requirements are orally exchanged between the customer and the developers. For example, in XP only small story cards containing some notes are used to support and document the communication.

However, neither approach seems to be appropriate under all circumstances. Participants in software projects need to learn to mix both approaches to requirements management in order to achieve a good solution which is well-suited to their project.

In university teaching this leads to the problem of presenting contrasting techniques and teach students the ability to combine these techniques' strengths appropriately. Moreover, the experience to tailor and improve the development process can be valuable for experienced developers, too. They are normally used to and involved in highly process-driven projects and do not see problems and possible improvements associated with their approach. They can benefit from being confronted with contrasting techniques in order to think about their daily work.

For presenting the tailoring ideas this paper is organized as follows: In section 2 we show our approach for organizational learning applied to software projects in university teaching. Section 3 contains our experiences with individual learning based on so called Agile Hours. The benefices of the combination of these two approaches are shown in section 4. Based on our conclusions in section 5 we suggest a new way to support the tailoring process of requirements in our outlook.

This paper is aimed at people teaching requirements engineering. It investigates necessary skills and explains how the mechanisms *Agile Hour* and *Experience Base* can support the learning process. Nevertheless we expect our results to be useful in the related areas as well: tailoring, requirements engineering and quality management.

2 Organizational Learning in process-driven approaches in University

As part of our curriculum, students have to participate in a one term software project. These software projects are organized as a simulation of a process-oriented company [Lübke, 2005a] in which different projects are being worked on by different teams. These teams are coordinated using Quality Gates [Lübke, 2004] which impose a certain development process by defining several phases, like requirements gathering.

For the requirements process, students get the following assistance:

1. Requirements Template: Students receive a template for their documents which they have to use to document the project's requirements.
2. Checklists for Quality Gates: By this the students learn about the formal requirements for their specification.
3. Experience Base: Students get access to an internal experience base web-based tool [Buchloh, 2005] in which example documents, comments and experiences by older projects are provided and can be viewed and downloaded. Our experience base resembles the ones introduced at large commercial software organizations.

At the end of each project, all teams elicit experience. With a light-weight Post-Mortem (e.g. [Birk, 2002]) technique, the LIDs [Schneider, 2000] method, experiences are collected and written down in LIDs documents containing approximately 12 pages. The LIDs session, in which all projects members are able to share their experiences and insights, is guided by a template. Everything which is

talked about is instantly written into the LIDs template by a moderator. The document is visible all the time to the project members. This facilitates feedback and improves the discussion. The LIDs template contains following sections:

- Motivation: What was the motivation to participate? Is it a typical situation that will reoccur?
- Expectations and fears in advance of the project.
- The course of events from the project members' point of view.
- What worked out, what did not?
- Description of the best and the worst moment during the project.

These experiences are used to refine the projects' templates and checklists as well as to feed back examples, experiences and best practices to the experience base as is shown in figure 1.

Figure 1: The Experience Base arranges experiences around the process description.

In this way, the entire simulated software company learns. Consequently, we could observe that student teams improved from term to term: interview techniques (e.g. using dictating machines) which proved to be useful in one term were used more frequently in following terms. Several changes to the templates have improved the overall documentation level. Furthermore the experiences led to changes of the software development process used.

The second way of learning takes place within our department. The feedback and experiences provided by the students are used to improve teaching of critical or not well-understood techniques and aspects of software engineering. Therefore, overall teaching quality has improved as well.

All techniques described above aim to improve the overall organization's performance. The organization as a whole learns as it collects LIDs documents and additional entries in its experience base which can be used by teams and the process designers to better conduct and manage the next projects.

3 Reason for Reflection: Agile Requirements in Agile Hours

Within software organizations individual learning is as important as organizational learning beyond individuals: First-hand experiences by developers are more valuable to them and facilitate their own improvement ideas and implementations.

Especially, within organizations which employ the same processes for a long time, new aspects introduced by externals can break up old behaviour and lead to new improvements.

In this context, Agile Methods are often new to large software organizations. Especially for demonstrating problems within requirements management, e.g. contact to the real stakeholders, they provide an efficient way to bring up deficiencies because they are provocative and completely different than established methods.

As part of our curriculum and in cooperation with commercial software development organizations we conducted 18 so-called Agile Hours [Lübke, 2005b]. Agile Hours are simulations of small XP projects done within 70 minutes and a following discussion. They focus on customer interaction and XP-style requirements management and documentation techniques. Within a prototype phase and two iterations a small project is "developed" by drawing a product on sheets of paper or building it using Lego bricks.

The requirements are documented *story card*-like by the customers as one-line requirements as is demonstrated in figure 2. These story cards are used for discussion in the planning game. The planning game is a meeting of all developers and the customers in which the story cards for the next iteration are selected.

Because XP is very customer-oriented and the interaction with the customer is very direct, deficiencies in requirements gathering are normally uncovered. Especially, experienced software developers get new points of view: Playing a customer in a simulated project is often very helpful for recognizing and understanding the problems of conveying requirements between the different parties. In any case, the relationship between the development and organization and the stakeholders can be discussed. The questions if the software organization knows the real requirements and who can be asked in case of problems with requirements

documents are very important. Astonishingly, these questions normally cannot be answered by the participating developers.

Figure 2: "Customers" writing down requirements during an Agile Hour.

All in all, Agile Hours are a very good way on the individual level to raise the interest in the problems of customer interaction and requirements management. Students and professional developers can benefit from attending an Agile Hour in order to improve their own behaviour related to requirements and customers' demands concerning their real-life projects. The individual experience of the participants is the basis for the tailoring of established processes to more effectiveness.

4 Learning to Tailor Documentation

We imagine a spectrum that runs from strict processes to absolute agility as in figure 3 (taken from [Boehm, 2002]). Tailoring the documentation of software requirements translates into finding the optimal point in this spectrum. For this reason learning to tailor is learning about the spectrum. Only if project members know about the alternatives they are able to choose the appropriate ones for the specific project.

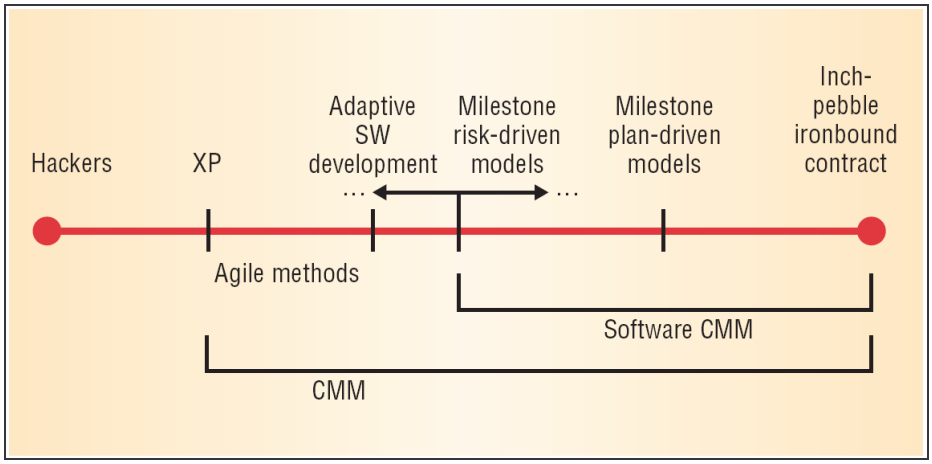

Figure 3: The planning spectrum (taken from [Boehm, 2002]) runs from unplanned ad hoc development on the left to micromanaged milestone planning on the right.

In our experience people often decide not to tailor their process, because they are unsure about the effect of the tailoring measures. A good knowledge about the spectrum of possible techniques is the only way to reduce this fear. In the domain of software requirements documentation one part of this learning process is gaining experiences with different templates for requirement documents. In our opinion this experience should be gathered by organizational learning, because the individual normally cannot try out all the different flavours of requirements documentation.

One aspect of lightweight requirements documentation like story cards is that it relies heavily on customer interaction. Project members need experience in this area for which individual learning, for example in an Agile Hour, is much better suited.

The importance of a good mixture of individual and organizational learning for tailoring requirements documentation becomes also evident from a more process driven point of view. On the one hand, organizational learning can optimize existing processes but is not able to break up process driven thinking if required or beneficial. For example, the tailoring of templates alone will never lead to an agile approach.

On the other hand agile techniques can be beneficial to know even for process-driven projects: culture of stakeholder interaction can be a crucial success factor.

Modern processes try to include rules into the tailoring process e.g. if a project has two or less members or will last less than 3 month a milestone trend analysis can be omitted. This leads to a questionnaire: project members answer a few questions

and get a tailored process that reflects the given project. This is an interesting approach but it seems that a lot of work needs to be done in this area. As projects are defined by their unique setting such more or less general advices have to prove their usefulness.

Especially small but important details like personal behaviour towards a customer cannot be tailored this way, because they are unique to a given project and are too small to be used in generally applicable rule-sets. For example, from our experience inexperienced developers require more guidance through processes than more advanced developers. However, more advanced developers can utilize agile methods due to their project experience and better technical knowledge. But they need to have more self-discipline.

Therefore, detailed knowledge about the environment of a project remains the most important prerequisite for tailoring. Only project members have enough knowledge about their surroundings to give a good guess about how a given tailoring will evolve in future. This means that these people need to be personally responsible and consequently be involved in the tailoring and adaptation themselves. Agile Hours and Software Project set-ups as described in this paper are from our experience good ways for teaching students abilities to cope with such responsibility.

5 Conclusions

We often observe organizations that are very process-oriented. Even our students learn "heavy" processes before more agile techniques like XP are introduced. We experienced that process-driven approach is better to start with, because it provides more guidance for "novice programmers" like students. The general *direction* of tailoring seems to be introducing more agile concepts into existing processes: The organization has to learn to use as few processes as necessary in order to be as agile as possible because no unnecessary work is done. This especially holds true for requirements documentation.

The problem with too extensive documentation of requirements is well known (e.g. [Cockburn, 2001]): If there is too much documentation, it will not be read. At the same time it prevents developers from asking their customers, because they are unsure, if the answer to the question has not been written down already and the documentation can be a barrier between the customer and the developer over which only indirect, error-prone communication happens. Furthermore large documents are hard to maintain. Organizations have to learn the right amount of documentation in their specific context.

In this paper we pointed out two ways to help with this tailoring process. Within our software projects we have successfully established a learning environment using traditional approaches like experience bases combined with modern teaching like Agile Hours. On the one hand a slow but continuous learning takes place. Checklists are modified, templates are adjusted to the right degree of freedom and more generally the important and useful parts of the process are identified.

On the other hand we make use of Agile Hours as a foundation for a discussion that often leads to new insights. Even if this does not cause a shift to a more agile approach, it does support the tailoring process. In our experience software

organizations benefit from this kind of organizational combined with individual learning. The resulting experiences have already been used with commercial partners.

6 Future Work

As part of our research we are now looking into analyzing the way requirements are being passed within different project settings using our FLOW notation [Schneider, 2005]. Figure 4 shows the Planning Game (a XP practice introduced by [Beck, 2000]) in this notation.

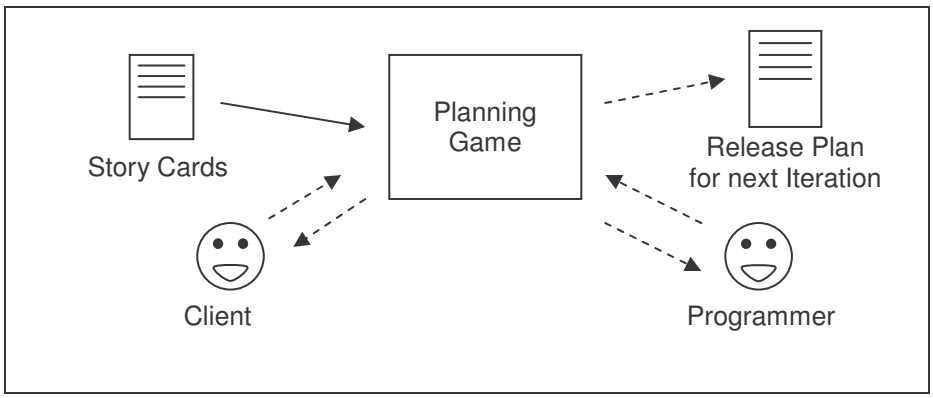

Figure 4: Planning Game in FLOW-Notation.

Note the focus on information flows that allows us to model how the resulting documentation (a sorted stack of story cards) is enhanced by communication between customer and programmer. In this example the solid arrows represent document based information. The input of the planning game is the requirements which were written down to story cards. If read again a story card will produce the same information.

Currently we try to establish the analysis of processes with means of this notation as a third source for tailoring. We already observed certain patterns that become visible when processes are displayed in FLOW and point to problems in a process (like documents that are never read). But FLOW might also be useful in planning where to introduce more “dashed lines” into a given process, for example by giving feedback to the customer at defined points. FLOW’s aim is to offer a foundation for tailoring by giving more aspects to the ones responsible for the project: existing and new experiences as well as direct communication and document-based communication are considered.

The simplicity of this notation enables a discussion in which not only process experts participate, but all project members. Because of this we currently investigate its value for consulting engineers.

References

[Beck, 2000] K. Beck, *Extreme Programming Explained*, 2000

[Birk, 2002] A. Birk, T. Dingsoyr and T. Stalhane, *Postmortem: Never Leave a Project without it*, 2002

[Boehm, 2002] B. Boehm, *Get Ready for Agile Methods, with Care*, 2002

[Buchloh, 2005] T. Buchloh, *Erstellung eines Baukastens für Experience Bases*, Hannover 2005

[Cockburn, 2001] A. Cockburn, *Writing Effective Use Cases*, 2001

[IABG, 1997] IABG, *V-Modell 97: www.v-modell.iabg.de*, 1997

[Lübke, 2004] D. Lübke, T. Flohr and K. Schneider, *Serious Insights through Fun Software-Projects*, Trondheim, Norway 2004

[Lübke, 2005a] D. Lübke and T. Flohr, *Experiences from the Conduction of a simulated Software Project driven by Quality Gates*, Maastricht, Netherlands 2005a

[Lübke, 2005b] D. Lübke and K. Schneider, *Agile Hours - Teaching XP skills to Students and IT Professionals*, Oulu, Finland 2005b

[Schneider, 2000] K. Schneider, *LIDs: A Light-Weight Approach to Experience Elicitation and Reuse*, Oulo, Finland 2000

[Schneider, 2005] K. Schneider and D. Lübke, *Systematic Light-Weight Tailoring of Quality Techniques*, Munich 2005