Agile Hour: Teaching XP Skills to Students and IT Professionals

Daniel Lübke and Kurt Schneider

University Hannover
{daniel.luebke, kurt.schneider}@inf.uni-hannover.de

Abstract. Agile Methods, like Extreme Programming, have increasingly become a viable alternative for conducting software projects, especially for projects with a very short time-to-market or uncertain customer-requirements. Using a technique called Agile Hours it is possible to convey many feelings associated with an Extreme Programming project. Within 70 minutes, a project is performed in which a product is built with Lego bricks. We applied this approach to (1) students and (2) IT professionals. By comparing the two groups, we found that both behaved comparable: we observed a number of interesting differences, although of minor importance. Both groups seemed to benefit from the Agile (Lego) Hours.

1 Introduction

In opposition to the established process-oriented software-methodologies, agile methods have increasingly become common for conducting software projects during the last years. These methods are built around very light-weight techniques [1] aimed at producing high-quality software in projects with very short time-to-market, with uncertain or even unknown customer requirements [2].

All agile methods share common values which are described in the Agile Manifesto [3]. The Agile Manifesto's authors "have come to value…

- …Individuals and interactions over processes and tools,
- …Working software over comprehensive documentation,
- …Customer collaboration over contract negotiation,
- …Responding to change over following a plan."

These values have proven to be useful in scenarios where requirements are rapidly changing and huge efforts to plan upfront waste resources [4-7].

The best-known agile method is Extreme Programming (XP) [1, 8] originally introduced by Kent Beck. XP consists of 12 practices, which support each other. These practices are project guidelines prescribing how to deal with requirements, how to manage code etc.

We started using the Extreme Hour when we were teaching agile methods in a software engineering lecture. The Extreme Hour was originally developed by Merel [9]. It is a very short simulation of an XP project. In this simulation, no software is

F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 517 – 529, 2005.

being developed but instead products are constructed by being drawn on paper. Later on, we modified the Extreme Hour and called the generalization "Agile Hours". For example, we developed it further by using Lego bricks instead of drawing. We called this particular variant the "Agile Lego Hour". In our terminology, Agile Hour is a more general term than Extreme Hour (drawing) and Agile Lego Hour (building).

We not only used the Agile Hours to introduce XP to (1) computer science students but had also the opportunity to teach XP to (2) IT professionals in an industrial Java User Group. In this paper we compare the behaviour of the two groups during the Agile Hours. On this empirical basis, we infer a number of lessons learned about the Agile Hour, and Agile *Lego* Hours in particular. We explored the options and potentials of Agile Hour variants. We present our observations from this exploration. Obviously, those observations need further confirmation through (controlled) experiments; we hope our findings will help to guide this continuing empirical work.

In the next section of this paper, typical problems of teaching XP in general are discussed. Afterwards, the Agile Hour is presented as our approach to deal with those problems. In the forth section, we discuss the differences between the Agile Hour and actual XP projects. Afterwards, we share common experiences of six Agile Hours we conducted. In the last two sections the differences between the students' and IT professionals' reaction to the Agile Hour and XP are analyzed.

2 Problem of Conveyance

2.1 Conveying XP Experience

XP favours and utilizes many social capabilities of the project's participants. Therefore, it is difficult to teach XP without practicing it. Because many things need to be experienced in order to fully understand them and realize their consequences, it simply is not sufficient to know what practices XP comprises, what they are called and what to do. XP, like other agile methods, needs to be experienced! In this case, we mean by experience (1) an observation combined with (2) associated feelings and (3) derived reasoning and conclusions. Experiences are stronger than (theoretical) knowledge, as they are more memorable and the reasoning can be used in future situations. Agile methods (like XP) evolved from programmers' *experiences* in the first place and can only be fully understood when combined with own experiences concerning their practices. Furthermore, many of the practices' dependencies can hardly be inferred in theory. However, they become obvious when the method is applied to a problem.

Moreover, XP practices are completely different from other established process models, like the waterfall-model [10] or the V-model used by the German government [11, 12]. Therefore, it is even more challenging to introduce XP to IT professionals who have used and got used to the above-mentioned traditional processes and their underlying assumptions. They often do not believe that some software projects could be run this way. Some have even learned to *resent* this option.

Besides the "softness" of this topic, time-constraints are further complicating the teaching of agile methods. In university, a lecturer often faces about 100 students in a software engineering lecture, all of whom are supposed to understand agile methods