AGILE SOFTWARE DEVELOPMENT

Michael Novikov and Nicolas Heuser

May 23, 2006

Contents

1 THE TIME BEFORE AGILE SOFTWARE DEVELOPMENT

Agile software development began, in the early 1990ies, as countermovement to the often-called heavyweight software development processes like the Waterfall Model or the V-Model. These methods are heavily regulated and regimented, slow and bureaucratic and counterproductive to the way actual developers perform optimal work. These methodologies try anticipating the complete development process in the planning phase and have severe problems with changing requirements, especially late in the development. These methodologies also contain forms of Taylorism, where the actual worker is considered as a replaceable part who is not allowed to choose the production method he prefers. In addition, the "One-best-way" principle is often incorporated, which means that manager decide on one exact sequence of operations and exclude any other possibility.

Many managers believed that software development does not fit into this scheme and searched for better ways by regarding the developer as a human being and by considering facts that do happen in normal software projects, like changes in the requirements and the appearance of so far improvident difficulties. There were already iterative methodologies available, but most of them only addressed the problem of better change management, and altogether did not go far enough.

Before 2001, the new methodologies were called "lightweight", until a conference in Snowbird where the main initiators of the movements adopted "agile" as an umbrella name for a whole family of methodologies, now named "Agile Software Development" (ASD).

2 ADAPTIVE VERSUS PREDICTIVE SOFTWARE DEVELOPMENT

If one had to divide all software development methods in adaptive and predictive methods, the agile methods would all be clearly on the adaptive side. The predictive methods are sometimes called "planned", which is a bit misleading as agile methods are not "unplanned". The focus of predictive methods is to plan the whole software project at the beginning in as much detail as possible. Predictability is a very desirable feature, as it implies the possibility to pre-calculate costs for a project.

ASD gives considerations to the fact that perfect predictability so not possible in common software projects. It tries to gain advantage in competition by giving the customer the possibility to change the software in the progress of development to adapt to market or technological changes.

3 WHAT IS "AGILITY"?

Etymology: Middle French, from Latin agilis, from agere to drive, act [Mer]

ãAgility is the ability to both create and respond to change in order to profit in a turbulent business environment." [CH02]

Jim HighsmithÕs definition of "agile" is very useful in the context of project management, as it states clearly that reacting on changes is a key value of ASD. The possibility to react to changes fast can be a competitive advantage as it happens in most software projects today, as customers have no to little computer experience and so mostly a narrowed view on the possibilities. Nevertheless, during the evolution of the program they realize the potentials and decide changes in the requirements. Here, the gap between the developer, with missing application-domain knowledge, and the customer, without computer experience, is becoming obvious. As the customer cannot

be forced to acquire computer knowledge, and the developer cannot adapt all application-domain knowledge perfectly, changes in requirements will stay inevitable.

4 CHAORDIC MANAGEMENT

Dee Hook, founder and CEO of the Visa credit card company, proposed the term "chaordic", as a mixture of the words "chaos" and "order". Hook studied organization forms that are neither rigid nor anarchic, but rather a blend of structure and chaos. In the 1990ies, Hook wrote "Birth of the Chaordic Age" in which he advocates self-organizing and self-evolving organizations instead of hierarchical bureaucracies. The declares the chaordic way as the natural way by comparing with natural phenomena, like the human brain, the immune system or any other living organism .[Dee06] Today the "chaordic commons" [cao] is a network of individuals interested in the development and implementation of new concepts of organization.

Representatives of agile software development methodologies belief that all organizations exhibit some kind of chaordic structure that prohibits a linear, predictive planning on project scope.

5 THE AGILE MANIFESTO

Individuals and interactions over processes and tools.
Working software over comprehensive documentation.
Customer collaboration over contract negotiation.
Responding to change over following a plan. [Agi]

It is a commitment to building better by adaptive methodologies in a highly cooperative manner. The authors of the manifesto emphasize that this statement does not mean that processes or tools are irrelevant, or that software projects do not need documentation. Software projects certainly do need contracts and have to be planned. It has only said this way to clarify the bias that agile software development has towards a more social way of realizing software projects.

6 AGILE SOFTWARE DEVELOPMENT

Jim Highsmith coined the notion "Agile Software Development Ecosystems" (ASDE) in his book of the same name. He wanted to avoid the notion "Methodology" to emphasize the differences to classic methods with their big strategic overhead and slow and bureaucratic body, and to express the large variety of new methods and their evolutionary nature.

In a harsh simplification, one could compare the Waterfall with Taylorism and ASD with a more social and technology combining world-view. The main aspect is to treat the developer as a human and to provide an environment where everyone can work self-responsive, in a demanding but exciting manner. This leads to a verifiable improvement in productivity in general [Kar].

Information exchange is supposed to be most efficient and effective in a face-to-face conversation. ASD considers this through floor communication, e.g. a conversation at a short break with a cup of coffee. The structure of the challenge and the size and composition of the development group are considerable factors, too. For the size of the group, the Ringelmann effect [Rin] must be observed and as general, group size should be as big as necessary and as small as possibly. Crystal even has complete different management approaches for varying group sizes.

The most known flavors of agile software development are Scrum, Extreme Programming, Crystal Clear, Adaptive Software Development and Feature Driven Development. Many different kinds of these methods exist, and they cover management and programming likewise.

7 EXAMPLES OF AGILE SOFTWARE DEVELOPMENT METHODS

7.1 SCRUM

Scrum was developed for the management of software development projects, but can be also used for managing other projects. The main concept consists of:

- A product backlog, a prioritized list of work to be done.
- A fixed set of items from the backlog to work on in the next iteration or sprint.
- A daily scrum meeting where the actual status gets discussed, future work is described and pending problems are mentioned.
- A "planning meeting" where items from the product backlog for the next sprint are defined.
- A post-sprint meeting to reflect about the last sprint.

The teams in scrum are self-organizing; everyone selects the next product backlog items to work on freely. A sprint is one iteration cycle, which is typically only about 30 days. After every sprint a working version of software is presented to the customers to show the evolution of the project. Despite the self-governance, a so-called "scrum-master" is assigned to help the team when problems arise during the sprint that could prevent the team from reaching the sprint goal and to hold off distractions of any form from the team. Scrum also encourages to wrap other methodologies, for example Extreme Programming, to aid the development process.

For further information, please refer to the JASS 2006 paper about scrum.

7.2 EXTREME PROGRAMMING

Extreme Programming (XP) is a programming methodology that was invented by Kent Beck, Ward Cunningham, and Ron Jeffries during their work on the Chrysler Comprehensive Compensation System (C3) project. XP builds on a set of five values, namely Communication, Simplicity, Feedback, Courage and Respect. XP requires the customer to be on place and mostly replaces a formal documentation of the system requirements. The programmer starts by writing tests first and then implements the simplest solution. The principles of no code ownership and courage allow the programmer to refactor the project later on to support more features, as they are needed. Pairprogramming is used to increase the quality of code by having to programmers work on one computer. The "driver" is the one working at the keyboard discussion the next steps and current thoughts with the second programmer. This technique also helps to share information very efficiently between developers.

8 CRITICS

One point of critic about ASD is that most methodologies assume skilled developers as team members. While this is certainly a problem, as not all developers have the same amount a skill in every area. The skill level is not even the main problem, which is the motivation of the developer and his commitment to the project. Skills can be gained and, e.g., pair programming helps to

spread knowledge between team members fast and to gain new programming skills. ASD wants to improve motivation of programmers by giving out reachable goals and letting the developer decide how to tackle problems, and which solution fits best.

References

[Agi] Agile Manifesto, http://agilemanifesto.org, 3/29/2006.

[cao] The chaordic commons, http://www.chaordic.org/ 3/29/2006.

[CH02] Cockburn and Highsmith. *Agile Software Development.* Addison Wesley, 2002.

[Dee06] "Transformation by Design" an interview with Dee Hook (http://www.wie.org/j22/hock.asp?page=1) 3/29/2006, 3 2006.

[Kar] Karasek Paradigm, http://www.workhealth.org/Adoben

[Mer] Merriam-Webster Online Dictionary: http://www.m-w.com/cgi-bin/dictionary?book=Dictionaryva=agile 3/29/2006.

[Rin] Ringelmann Effect, http://en.wikipedia.org/wiki/Ringelmann_effect, 4/1/2006.