A New DACS State-of-the-Art Report on Agile Methods

Patricia Costa, David Cohen, and Michael Lindvall
Fraunhofer Center for Experimental Software Engineering, Maryland

1. Overview

The so-called Agile Methods are creating a buzz in the software development community, drawing their fair share of advocates and opponents. The purpose of a new report from DACS is to address this interest and provide a comprehensive overview of the current State-of-the-Art as well as State-of-the-Practice for Agile Methods.

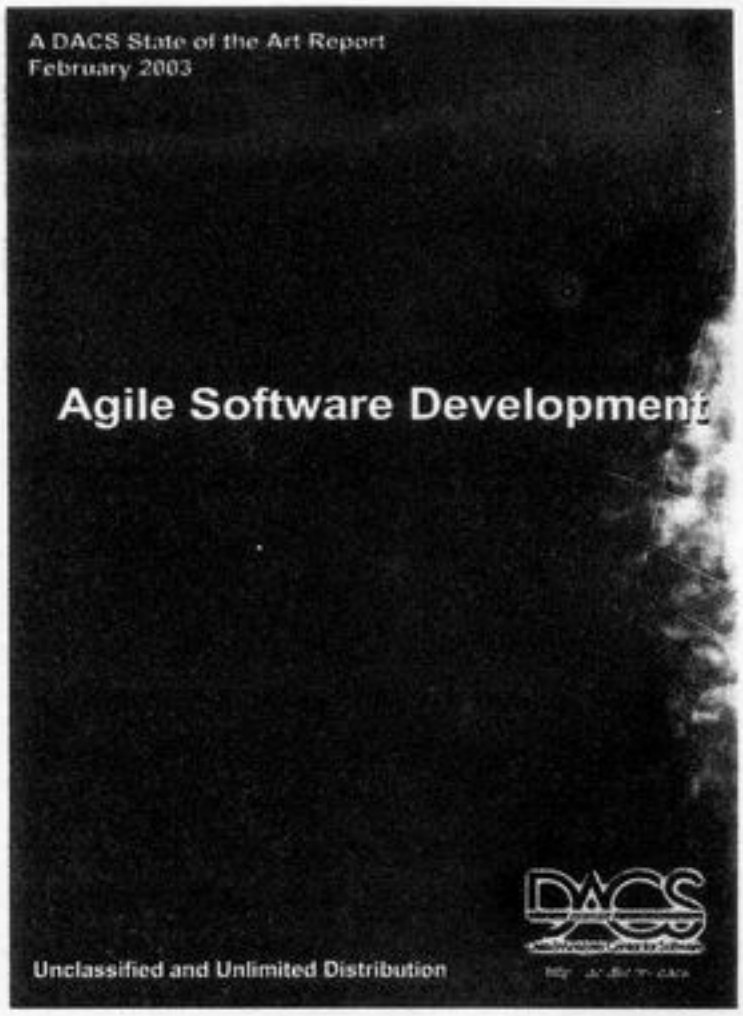
The report discusses the history behind Agile Methods, as well as the Agile Manifesto, a statement from the leaders of the Agile Methods movement. It looks at what it means to be agile, discusses the role of management, describes and compares some of the more popular methods, provides a guide for deciding where an agile approach is applicable, and lists common criticisms. It summarizes empirical studies, anecdotal reports, and lessons learned from applying agile methods and concludes with an analysis of various agile methods.

The target audiences for this report include practitioners who will be interested in the discussion of the different methods and their applications, researchers who may want to focus on the empirical studies and lessons learned, and educators looking to teach and learn more about agile methods. In this article we provide some of the history and reasoning behind Agile Methods.

The full report may be purchased in a bound hardcopy from the DACS Store: www.dacsstore.com or downloaded from the DACS Website: www.thedacs.com/techs/agile/.

2. Agile Methods

Agile Methods are a reaction to traditional ways of developing software and acknowledge the "need for an alternative to documentation driven, heavyweight software development processes" [1]. In the implementation of traditional methods, work begins with the elicitation and documentation of a "complete" set of requirements, followed by architectural and high-level design, development, and inspection. Beginning in the mid-1990s, some practitioners found these initial development steps frustrating and, perhaps, impossible [5]. The industry and technology move too fast, requirements "change at rates that swamp traditional methods" [7], and customers have become increasingly unable to definitively state their needs up front while, at the same time, expecting more from their software. As a result, several consultants have independently developed methods and practices to respond to the inevitable change they were experiencing.



Schwaber, who would go on to develop Scrum, one of the Agile Methods, realized that to be truly agile, a process needs to accept change rather than stress predictability [16]. Practitioners came to realize that methods that would respond to change as quickly as it arose were necessary, and that in a dynamic environment, "creativity, not voluminous written rules, is the only way to manage complex software development problems" [3].

Practitioners like Mary Poppendieck and Bob Charette began to look to other engineering disciplines for process inspiration, turning to one of the more innovative industry trends at the

article continued on page 14

time, Lean Manufacturing integrating Dr. W. Edwards Deming's Total Quality Management philosophy with the process. Deming believed that "people inherently want to do a good job, and that managers needed to allow workers on the floor to make decisions and solve problems" [14].

Independently, Kent Beck rediscovered many of these values in the late 1990s when he was hired by Chrysler to save their failing payroll project. Beck decided to scrap all the existing code and start the project over from scratch. A little over a year later, the system was in use paying employees. The project became the first project to use eXtreme Programming (XP) [7] which relies on the same values for success as Poppendiek's Lean Programming.

In the early 1990s, Cockburn found that at IBM "team after successful team 'apologized' for not following a formal process and for 'merely' sitting close to each other and discussing while they went." At the same time teams that had failed followed formal processes and were confused why it hadn't worked, stating "maybe they hadn't followed it well enough" [7]. Cockburn used what he learned to develop the Crystal Methods.

Practitioners recognized that new practices were necessary to better cope with changing requirements. And these new practices must be people-oriented and flexible [3]. Highsmith and Cockburn summarize the new challenges facing the traditional methods that eventually became Agile Methods:

- Satisfying the customer has taken precedence over conforming to original plans
- Change will happen - the focus is not how to prevent it but how to better cope with it and reduce the cost of change throughout the development process
- "Eliminating change early means being unresponsive to business conditions - in other words, business failure"
- "The market demands and expects innovative, high quality software that meets its needs - and soon" [3].

One of the models often used to represent traditional methodologies is Capability Maturity Model (CMM) [13]. The goals of CMM are to achieve process consistency, predictability, and reliability. Its proponents claim that it can be tailored to also fit the needs of small projects even though it was designed for large projects and large organizations [11].

Most Agile Development proponents do not believe CMM fits their needs: "If one were to ask a typical Software Engineer whether CMM and Process Improvement were applicable to Agile Methods, the response would most likely range from a blank stare to hysterical laughter [17]." One reason is that "CMM is a belief in Software Development as a defined process [6]. "For projects with any degree of exploration at all, Agile Methods developers just do not believe these assumptions are valid. This is a deep fundamental divide - and not one that can be reconciled to some comforting middle ground" [6].

Many Agile Development proponents dislike CMM because of its focus on documentation. A "typical" example is the company that spent two years working (not using CMM though) on a project until they finally declared it a failure. Two years of working resulted in "3,500 pages of use cases, an object model with hundreds of classes, thousands of attributes (but no methods), and, of course, no code" [6].

While Agile Development proponents see a deep divide between Agile Methods and traditional methods, this is not the case for proponents of traditional methods. Mark Paulk, the man behind CMM, is, for example, surprisingly positive about Agile Methods and claims that Agile Methods "address many CMM level 2 and 3 practices" [12].

Regarding the criticism about heavy documentation in CMM projects, Paulk replies: "over-documentation is a pernicious problem in the software industry, especially in the Department of Defense (DoD) projects" [12]. "Plan-driven methodologists must acknowledge that keeping documentation to a minimum useful set is necessary [12].

While many tired of traditional development techniques are quick to show support for the Agile Methods movement, others are more skeptical and view Agile Software Development as a step backwards from traditional engineering practices, a disorderly "attempt to legitimize the hacker process" [15]. In response Beck states: "Refactoring, design patterns, comprehensive unit testing, pair programming - these are not the tools of hackers. These are the tools of developers who are exploring new ways to meet the difficult goals of rapid product delivery, low defect levels, and flexibility" [7]. In response to the speculation that applying eXtreme Programming (XP) would result in a Chaotic development process (CMM level 1), one of the Agile Development proponents concluded that "XP is in some ways a 'vertical' slice through the levels 2 through 5" [8].

3. Conclusions

The question whether Agile Software Development is hacking is probably less important than whether agile and traditional methods can co-exist. This is due to the fact that many organizations need both to be agile and show that they are mature enough to take on certain contracts. One must remember that developing software for a space shuttle is not the same as developing software for a toaster [9]. In order to accommodate the right need for a Software Development project, it is necessary to select the right method for the task at hand.

One important factor when selecting a development method is the number of people involved. The more people involved in the project, the more rigorous communication mechanisms need to be. According to Alistair Cockburn, there is one method for each project size, starting with Crystal Clear for small projects and as the project grows larger, the less agile the methods become [2].

Other factors that have an impact on the rigor of the development methods are; application domain, criticality, and innovation [4]. Applications that may endanger human life, like manned space missions, must, for example, undergo much stricter quality control than less critical applications. At the same time, a traditional method might kill projects that need to be highly innovative and are extremely sensitive to changes in market needs.

In conclusion, the selection of a method for a specific project must be very careful, taking into consideration many different factors including those mentioned above. In many cases, being both agile and stable methods at the same time will be necessary. A contradictory combination it seems, and therefore extra challenging but not impossible. As Siemens states, "We firmly believe that agility is necessary, but that it should be built on top of an appropriately mature process foundation, not instead of it" [10].

About the Authors

David Cohen is a Junior Scientist at Fraunhofer Center for Experimental Software Engineering, Maryland. He specializes on the Software Development Lifecycle, including hands-on Agile Software Development experience, and User Interface Usability Modeling. Currently, David is working on development of collaborative, Internet-based tools to facilitate knowledge transfer between individuals and organizations and build experience bases supporting the Experience Factory Model.

Patricia Costa is a Scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland. She has a B.Sc. (1996) and a M.Sc. (1999) in Computer Science from the Federal University of Minas Gerais, Brazil and a M.Sc. (2001) in Telecommunications Management from University of Maryland University College. She has experience in Software Development and in the areas of Agile Methods, Knowledge Management and evaluation of Software Architectures. She is currently interested in using evaluation of Software Architectures as a tool to assess/ assure quality attributes like security and maintainability on software systems.

article continued on page 16

Dr. Mikael Lindvall is a Scientist at Fraunhofer Center for Experimental Software Engineering, Maryland. Dr. Lindvall specializes in work on Software Architecture evaluation and evolution and experience and Knowledge Management in Software Engineering, as well as Agile Software Development. He is currently working on tools and methods to quickly understand an architecture and identify the architectural deviations, as well as ways of building experience bases to attract users to both contribute and use experience bases. Dr. Lindvall received his PhD in computer science from Linköpings University, Sweden in 1997. Lindvall's PhD work focused on evolution of Object-Oriented systems and was based on a commercial development project at Ericsson Radio in Sweden.

Author Contact Information

David Cohen
Fraunhofer Center for Experimental Software Engineering, Maryland
4321 Hartwick Rd, Suite 500
College Park, MD 20742
dcohen@fc-md.umd.edu

Patricia Costa
Fraunhofer Center for Experimental Software Engineering, Maryland
4321 Hartwick Rd, Suite 500
College Park, MD 20742
pcosta@fc-md.umd.edu

Dr. Mikael Lindvall
Fraunhofer Center for Experimental Software Engineering, Maryland
4321 Hartwick Rd, Suite 500
College Park, MD 20742
phone: (301) 403-8972
mlindvall@fc-md.umd.edu

References

1. Beck, K., Cockburn, A., Jeffries, R., and Highsmith, J., "Agile Manifesto", www.agilemanifesto.org, 2001. 12-4-2002.
2. Cockburn, A., "Selecting a Project's Methodology," *IEEE Software*, vol. 17, no. 4, pp. 64-71, 2000.
3. Cockburn, A. and Highsmith, J., "Agile Software Development: The Business of Innovation," *IEEE Computer*, pp. 120-122, Sept. 2001.
4. Glass, R., "Agile Versus Traditional: Make Love, Not War," *Cutter IT Journal*, pp. 12-18, Dec. 2001.
5. Highsmith, J., *Agile Software Development Ecosystems*, Boston, MA: Addison-Wesley, 2002a.
6. Highsmith, J., "What Is Agile Software Development?," *Crosstalk*, pp. 4-9, Oct. 2002b.
7. Highsmith, J., Orr, K., and Cockburn, A., "Extreme Programming," *E-Business Application Delivery*, pp. 4-17, Feb. 2000. www.cutter.com/freestuff/ead0002.pdf.
8. Jeffries, R. "Extreme Programming and the Capability Maturity Model", www.xprogramming.com/xpmag/xp_and_cmm.htm, 2000, 12-4-2002.
9. Lindvall, M. and Rus, I., "Process Diversity in Software Development," *IEEE Software*, vol. 17, no. 4, pp. 14-71, Aug. 2000. http://fc-md.umd.edu/mikli/LindvallProcessDiversity.pdf.
10. Paulisch, Frances and Völker, Axel, "Agility - Build on a Mature Foundation," in *Proceedings of Software Engineering Process Group Conference - SEPG 2002*, 2002.
11. Paulk, M. C., "Extreme Programming from a CMM Perspective," *IEEE Software*, vol. 18, no. 6, pp. 19-26, 2001.
12. Paulk, M. C., "Agile Methodologies and Process Discipline," *Crosstalk*, pp. 15-18, Oct. 2002.
13. Paulk, Mark C., "Key Practices of the Capability Maturity Model, Version 1.1," Technical Report, Tech. Rep. CMU/SEI-93-TR-25, 1993.
14. Poppendieck, M., "Lean Programming" www.agilealliance.org/articles/articles/LeanProgramming.htm, 2001, 4-12-2002.
15. Rakitin, S. R., "Manifesto Elicits Cynicism," *IEEE Computer*, vol. 34, no. 12, pp. 4, Dec. 2001.
16. Schwaber, *K. Controlled Chaos: Living on the Edge*. www.agilealliance.org/articles/articles/ap.pdf, 2002, 4-12-2002.
17. Turner, Richard and Jain, Apurva, "Agile meets CMMI: Culture clash or common cause?," in *Proceedings of EXtreme Programming and Agile Methods - XP/Agile Universe 2002*, 2002, 153-165.