ISSN : 2231-0711

Extreme Programming versus CMMI – Conflicts and Compatibilities

Therese Clara V, Alagarsamy Dr. K.,
Madurai Kamaraj University College, India.

***Abstract*-Agile methods like extreme programming have assumed tremendous significance in the last few years. At the same time, since it has been getting clear that most project failures can be attributed to inconsistent and undisciplined processes, more organizations have started to rely on process maturity models. CMMI compliance is being demanded for projects where agile methods are employed. In this situation it is necessary to analyze the interrelations and mutual restrictions between agile methods and approaches for software process analysis and improvement. This paper analyzes to what extent the CMMI process areas can be covered by XP and where adjustments of XP have to be made. Based on this, the limitations of CMMI in an agile environment are described and further it is shown that level 4 or 5 are not feasible under the current specifications of CMMI and XP.**

***Keywords*— process models. CMMI, Agile methods, extreme programming**

I INTRODUCTION

Organizational maturity indicators like CMMI levels, SPICE ratings or specific ISO standards have become increasingly important for software development. Customers or organizations that set up a distributed project often rely on them when selecting suppliers, since the results of these assessments and audits can serve as a 'signal' for their process maturity [8, 19]. In large organizations there are policies which enforce that all parts of the organization have to achieve certain maturity levels.

At the same time, agile methods continue to gain currency. This has also been true for larger projects, e.g. Cockburn and Highsmith cite successful agile projects with up to 250 people [6] and even for outsourcing and offshoring projects [10, 24, 26]. This leads to the challenge that, on the one hand, organizations often rely on CMMI as an indicator for process maturity (which is supposed to translate into product quality), on the other hand agile methodologies like XP [3], Scrum [25], Lean Development [23] or the Crystal methods [3] get more prominent. It has been shown that projects that use agile methods with certain adjustments can achieve CMMI level 2 or even 3 [2, 17]. But from the various reports of successful agile projects it doesn't become clear how agile methods contribute to the fulfillment of process areas, where they have to be adjusted and where they are in conflict with CMMI goals.

Research should be conducted on how agile methods can be adapted to reach certain CMMI levels. This paper is meant as a starting point which reveals where adjustments have to be made. Therefore, this paper takes a qualitative approach to analyze in how far agile methods support or conflict with CMMI process areas, where adjustments have to be made and if organizations employing agile methods can reach conformity with certain CMMI levels. After analyzing XP, general statements and theses about the comparability and compatibility of CMMI and agile methods are derived.

A. Related Work

Several authors have discussed the compatibility of CMMI and agile methods. Paulk [21] analyzes how XP can help organizations to reach the SW-CMM goals. While his work gives good insights into the interrelations between XP and CMM, the use of the now outdated SW-CMM limits the results. The suggested approach extends his work and explicitly shows which process areas are in conflict with agile methods.

Kane and Ornburn [18] analyze which CMMI process areas are covered by XP and Scrum. Especially those areas related with process management are not considered by these two methods. Therefore, the authors propose tailoring of XP and Scrum to satisfy these goals. Unfortunately, most of the findings are not clearly derived. In addition, it is not discussed whether certain process areas are not addressed by agile methods or whether they are in conflict.

Finally, Turner and Jain [27, 28] show how CMMI can help to successfully implement agile methods. The difference to the suggested approach is that the researcher wants to analyze how agile methods support CMMI and not vice versa.

II AGILE METHODS

As an answer to the challenges of modern software development which in many cases cannot be tackled by 'traditional' processes, different 'lightweight' approaches have been established since the mid 1990's that can be subsumed under the brand 'Agile Methods' [3, 6]. They "allow for creativity and responsiveness to changing conditions" [8]by emphasizing customer participation, quick reaction to requirements' changes and continuous releases [7, 14]. Some of them are rather a collection of techniques and activities than complete process models with precise definitions of roles, products, activities etc. But there are some methods, e.g. extreme Programming (XP) [3] or SCRUM [25], which are widely employed in projects of various sizes. Some concepts and ideas from the agile space have even been introduced into 'heavyweight' process models [1]. The characteristics of agile methods are elaborately defined in the twelve principles behind the agile manifesto [4, 5, 9]:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity - the art of maximizing the amount of work not done – is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

These principles specify the four agile values [9] and provide a good summary of the intentions and ideas of agile methods.

III COMPATIBILITY OF AGILE METHODS WITH CMMI REQUIREMENTS

A. CMMI – an Overview

The Capability Maturity Model for Software (CMM) [22, 15] developed by the Software Engineering Institute (SEI) has had a major influence on software process and quality improvement around the world [20]. Based on the first version released 1991, the Capability Maturity Model – Integrated (CMMI) has been presented in 2000, integrating CMM for Software (SW-CMM), the Capability Model for Systems Development (EIA/IS 731) and the CMM for Integrated Product Development (IPD-CMM).

Software Process Improvement (SPI) assumes that a well-managed organization with a defined engineering process is more likely to produce software that consistently meets the users' requirements within schedule and budget than a poorly managed organization with no such engineering process. "In other words, the project failure is usually a process failure" [8]. CMMI – as SPI's "de facto method" [29] – describes managerial processes to attack software development difficulties at five maturity levels:

1. initial
2. managed
3. defined
4. quantitatively managed
5. optimizing

It is important to note that the CMMI process models do not contain prescriptive processes that can be used right out of the box. Instead, CMMI provides a way to assess the state of an organization's ability to build software in a repeatable, predictable way [8]. Applying CMMI as a means to increase process capabilities is an organization-wide challenge. Herbsleb et.al. show that the average time for an organization to move up one level is between 21 and 37 months [13]. Over three quarters of the organizations reported that implementing any key SPI activity took longer than expected. But the effort pays off since "software process management maturity is positively associated with project performance" [16].

In order to reach a certain level, an organization has to fulfill all process areas of that level as well as those of lower levels. A process area is a summary of all requirements for a certain topic, e.g. project management, organizational training or causal analysis and resolution. To satisfy a process area all of its associated goals – specific ones and generic ones – have to be met. Specific goals apply to a process area and address the unique characteristics that describe what has to be implemented to satisfy the process area. To meet a specific goal CMMI suggests a set of specific practices. A specific practice is an activity that is considered important in achieving the associated specific goal. Generic goals are called "generic" because the same goal statement appears in multiple process areas. In the staged representation, each process area has only one generic goal. To meet a generic goal, CMMI suggests a set of generic practices. Generic practices provide institutionalization to ensure that the processes associated with the process area will be effective, repeatable, and lasting [15].

B. An Approach to Analyze the Coverage of Process by Agile Methodologies

The goal is to determine which of the CMMI process areas are supported by agile methods, where adjustments need to be made and which process areas are in conflict. In order to do so we analyzed every process area and all of its specific goals in detail [11]. The specific practices are only expected model components, meaning that their use is recommended but not necessary. CMMI states that they can be replaced by alternative practices. In fact, agile methods often employ different approaches than those suggested by CMMI. Therefore we concentrate on the analysis of the goals, using the practices only as guidelines and always looking for possible alternative ways of implementing the goals. We also analyze the two generic goals ("institutionalize a managed process" and "institutionalize a defined process") and the generic practices, but only in general terms and not in conjunction with particular process areas. The reason for this omission is that agile methods do not directly address institutionalization practices. Institutionalization is a topic which has to be considered on the organizational level while agile methods only regard project level. Results in a detailed analysis of generic practices would be very limited.

For the coverage of specific goals, process areas and generic practices, a rating system is applied:

- Conflicting (–)
- Not addressed (0)
- Partially supported (+)

• Supported (++)
• Largely supported (+++)
"Largely supported" means that the agile method's practices, if employed correctly, satisfy the major part of the respective model component. "Supported" and "partially supported" describe a restricted coverage and "not addressed" reflects that there is no coverage at all. These ratings do not imply that the respective CMMI goals cannot be attained. They merely point out that additional practices have to be introduced to fully satisfy the CMMI requirements. "Conflicting" on the other hand indicates that the respective CMMI goal cannot be reached with the agile method being used. This rating is given if there are no possible extensions that do not interfere with the method's basic practices or the agile principles. To differentiate between "not addressed" and "conflicting", it was imperative to check whether the agile method could be extended to reach the CMMI goal without interfering with the method's basic practices or contradicting to the principles stated in the agile manifesto.

C. Applying the Approach to extreme Programming
The suggested approach is applied to XP and further shows the interrelations and conflicts between XP and the CMMI process areas and all of their associated specific goals. To not go beyond the scope of this paper the analysis is condensed. M. Fritzsche [11] provides a more detailed presentation and a discussion of Scrum.

1) Analysis of Process Areas and Their Specific Goals
Requirements management– Manage requirements (+++)
Understanding of the requirements is obtained through the integration of the customer into the team and the resulting intensive communication with the customer. The project participants' commitment to the requirements is obtained in the planning phase. Changes of requirements are quickly exchanged and discussed. Even if traceability of requirements is not an explicit goal of XP, it is supported by stories, tasks, functional tests that detect inconsistencies between project work and requirements, and by unit tests. XP's practice of throwing away story cards that already have been realized can prove to be problematic. To better implement this process area story cards should be kept. Thus, traceability can be extended by keeping record of previous story cards and old versions of the documentation.
Project planning (+++)
Establish estimates (+++)
Estimates for stories and tasks are established and can be corrected during the project.
The estimates' precision is increased through a short planning horizon due to short iterations.
Develop a project plan (+++)
The project plan is established through XP's release and iteration plans that evolve throughout the project. Therefore long term plans remain vague and only short term plans are detailed. Risks are identified, training needs are planned and the involvement of all relevant stakeholders is assured if XP is applied correctly.
Obtain commitment to the plan (+++)
Commitment to the release and iteration plans is obtained through the high involvement and responsibility of all team members.

Project monitoring and control (+++)
Monitor project against plan (+++)
Schedule and estimates are monitored by the tracker. Information on the project's progress is gathered by the use of measures. The intensive communication among the team members and with the customer helps to convey that information. Milestones are checked against the schedule by functional tests. The strict system of short iterations and the regular commitments to the plan make it easier to monitor the project against the baseline.
Manage corrective action to closure (+++)
Issues that demand corrective actions are informally collected and analyzed. Corrective actions can be adjustments of the method and also of the functionality that will be realized. In addition new iterations always offer good opportunities to make adjustments.
Supplier agreement management (0)
This process area is not addressed by XP. This method can be extended to fulfill the goals of this process area. However, involving suppliers could be problematic for agility if it hinders iterative development. There are cases where supplied components are needed to obtain functioning software at the end of an iteration. It can pose a critical problem if they are not available at that point.

Measurement and analysis (+)
Align measurement and analysis activities (+)
The only measurement objective is progress control. Measurements and analysis procedures are defined by the tracker. XP provides no specific guidelines for these tasks.
Provide measurement results (++)
The measurement data is obtained through intensive communication within the team. The tracker analyzes the data and conveys the results to the team using wall charts. The data is usually not permanently stored. However, there are many tools available for effort estimation and tracking for agile teams. By using these tools the measurement data and results can be stored permanently without too much effort.
Process and product quality assurance (+)
Objectively evaluate processes and work products (+)
XP doesn't demand an explicit evaluation of processes, work products and services against the applicable process descriptions. The only instrument of controlling that the method is applied in the right way is the coach who guides the team in the use of XP.
Provide objective insight (+)
Quality issues can be easily communicated in an XP team. The work of the coach supports this specific goal. However, there are no strict guidelines for the resolution of noncompliance issues and the establishing of records of quality assurance activities.

Configuration management (+++)

Establish baselines (+++)

Configuration items are code, design, tests and requirements. The use of a configuration management system is recommended since continuous integration relies heavily on it. Baselines are established regularly through functional tests. In addition, baselines are created at the end of each iteration.

Track and control changes (+++)

Changes are controlled and tracked through various practices like pair programming, tests, customer collaboration, etc.

Establish integrity (+++)

XP enforces continuous integration. Code is easy to read because of coding standards and therefore its own description. Audits are informally performed through pair programming, customer involvement and testing.

Requirements development (++)

Develop customer requirements (++)

The customer elicits requirements and specifies them in story cards and functional tests. The developers often support him in these tasks. The requirements specification however remains quite vague. Details have to be discussed directly with the customer during development.

Develop product requirements (++)

Customer requirements are refined into product requirements. These are specified using task cards. They remain relatively vague too.

Analyze and validate requirements (++)

An analysis of requirements is carried out in a well-defined way. The programmers consult the customer during requirements elicitation. In addition, the acceptance of changing requirements and the use of iterations allow constant analysis and validation of requirements. Operational concepts and scenarios are established using functional tests. However there is no in depth requirements analysis up front.

Technical solution (+++)

Select product-component solutions (+++)

Alternative solutions are explored at the beginning of the project through prototypes and later on through refactoring and iterative development.

Develop the design (+++)

A design as simple as possible is developed. Code is used as a design document. Design is carried out iteratively.

Implement the product design (+++)

XP employs a variety of implementation practices, e.g. refactoring, coding standards, pair programming. A product support documentation is developed if it is requested by the customer.

Product integration (+++)

Prepare for product integration (+++)

XP employs continuous integration and since integration steps are performed very often, a thorough preparation is critical.

Ensure interface compatibility (+++)

Interface compatibility is ensured by running all tests at each integration step.

Assemble product components and deliver the product (+++)

Component assembly and delivery is carried out. The use of continuous integration and direct customer involvement further helps to achieve this goal.

Verification (+++)

Prepare for verification (+++)

Verification is carried out through intensive testing. The preparation is therefore concentrated on this topic. A test framework should be used and hence according preparation activities executed. Furthermore XP employs a test-first approach. All tests have to be written before the code.

Perform peer reviews (+++)

Peer reviews are implicitly always part of XP. Pair programming, refactoring and the principle of collective code ownership imply constant peer reviews.

Verify selected work products (+++)

Methods for verification are mainly peer reviews and testing, which both are performed constantly.

Validation (+++)

Prepare for validation (+++)

Validation is performed in XP projects through customer participation and frequent releases. The main criterion for validation is acceptance by the customer.

Validate product or product components (+++)

The customer constantly validates the work done by the team. This is possible because he is integrated into the team. In addition he validates the deliveries at the end of each iteration. This may result in additional or changed requirements specified by the customer. The enormous influence of the customer improves the chances that the product is suitable for use in its intended operating environment.

Organizational process focus (–)

This process area is not addressed because it applies to the organization while XP only applies to a project. It even is in conflict with XP: like in other agile methods, adjustments are often done during a project. These improvements, however, are limited to the current project since they shall not be documented. Knowledge about improvements is linked to people. Other projects can benefit if people are moved between projects. But the problem is that in big organizations there are too many projects. In that case such a practice cannot let all of them benefit from a particular project’s experience. In addition the information is not permanent since people can retire or change organization. The conflict can be eased by establishing organization-wide repositories storing best practices of previous projects or by institutionalizing the exchange of lessons learnt between projects.

Organizational process definition (0)

Organizational training (++)

Establish an organizational training capability (++)

Training is carried out by XP during the exploration phase. Therefore an XP project requires organizational training capabilities. Pair programming and coaching can also be regarded as training, so XP further enhances the organization’s training capabilities.

Provide necessary training (++)
As stated above, training is carried out explicitly during the exploration phase and implicitly during the whole project through coaching and pair programming. Through the latter, there are however deficiencies regarding the establishment of records and the assessment of training effectiveness.
Integrated project management (++)
Use the project's defined process (0)
Coordinate and collaborate with relevant stakeholders (+++)
XP integrates and coordinates developers, customer, testers, and management.
Use the project's shared vision for IPPD (+++)
XP contributes a lot to the project members' integration and their close collaboration. This and the intensive communication within the team help to establish a shared vision.
Organize integrated teams for IPPD (0)
Risk management (+++)
Prepare for risk management (+)
XP doesn't explicitly state how risk management is to be conducted. But XP projects surely make some sort of preparation.
Identify and analyze risks (+++)
XP enforces the identification and analysis of risks during the planning phase.
Mitigate Risks (+++)
The flexibility gained by the use of short iterations is a potent instrument to mitigate risks.
Integrated teaming (+++)
Establish team composition (+++)
XP establishes a self-organizing cross-functional team in which all relevant stakeholders are integrated.
Govern team operation (+++)
Team operation is governed through a clear definition of the different roles, pair programming, collective ownership of the code and the focus on cooperation and communication.
Integrated supplier management (0)
Decision analysis and resolution (–)
Turner [27] points out that the ability to adapt quickly to new situations is preferred by agile methods to a formal evaluation process. XP identifies and evaluates alternatives informally and not in the way CMMI suggests.
Organizational environment for integration (+)
The issues of this process area are addressed at project level but not at the organizational level.
Provide IPPD infrastructure (++)
XP establishes the basis for this specific goal through the introduction of tools, intensive communication and cooperation. By promoting the abilities to communicate and cooperate as well as leadership skills the method further supports this goal.
Manage people for integration (+)
Leadership mechanisms are democratic within the development team. However the customer and the big boss have authority to decide on high level issues.
Organizational process performance (–)
XP focuses rather on individuals than on issues that are as process oriented as this process area. Turner [28] points out that the idea of measuring a process and maintaining baselines and models is in conflict with the agile manifesto.
Quantitative project management (–)
Statistical methods have their focus on defined processes and not on individuals since quantitative analyses need a static baseline. Therefore, statistical methods are in conflict with agile principles. Furthermore, they rely on the law of big numbers and on averaging out effects in large teams. Since most agile software projects are small the use of statistics is questionable.
Organizational innovation and deployment (–)
Process improvements and adaptations are made only within projects and not documented, so that they cannot be propagated to the whole organization. This topic relies heavy on "organizational process focus", a process area that is in conflict with XP.
Causal analysis and resolution (0)

2) Generic Practices

Establish an organizational policy (0)
Plan the process (0)
Provide resources (+)
This practice is conducted only regarding a few process areas.
Assign responsibility (+++)
The role model assigns responsibilities to certain team members. In addition the developers take responsibility for particular tasks during the project.
Train people (+++)
Training is conducted during the exploration phase. Furthermore pair programming and coaching is employed to train people.
Manage configurations (++)
A configuration management system is employed. The configurations of code, tests, design and requirements are managed. For protocols of test cases, measurement data, release and iteration plans configuration management is not planned.
Identify and involve relevant stakeholders (+++)
All relevant stakeholders are part of the team.
Monitor and control the process (++)
This generic practice is implemented for all project-related process areas due to XP's fulfillment of the process area "project monitoring and control". To realize it for all processes and not only for project-related processes, measures for monitoring actual performance of the process have to be established.
Objectively evaluate adherence (+)
The coach is XP's only instrument to support this generic practice. However by implementing
the process area "process and product quality assurance" which is not in conflict with XP it would be possible to fulfill this practice for all process areas.
Review status with higher level management (++)
Frequent releases enable reviews by the management.

Establish a defined process (0)

Collect improvement information (–)

Improvements are deliberately not documented by XP and therefore this generic practice cannot be implemented. This conflict could be solved by properly documenting process changes in a project and making them available to other projects in the organization. In addition, process improvement information might be easily captured during iteration planning and via postmortem analyses.

D. Coverage of Process Areas by Agile Methodologies

In 3.3., it was shown in detail which CMMI process areas are supported by XP and which are in conflict. In this section, a summary is given on the coverage of CMMI process areas by XP and Scrum.

All of the seven process areas of CMMI level 2 are attainable by both methods. From the fourteen process areas of level 3 only two are in conflict. Three out of the four process areas of level 4 and 5 are also in conflict. Of the twelve generic practices only one was rated as in conflict.

The results indicate that level 2 can be attained without major adaptations. The same is true for level 3 with the exception of two process areas. It is however practically impossible to reach level 4 and 5 with XP and Scrum without making changes to the methods that contradict agility.

Mainly those process areas that deal explicitly with process improvement (“organizational process focus”, “organizational process performance”, “quantitative project management” and “organizational innovation and deployment”) are in conflict with agile methods. Also the generic practice “collect improvement information” deals explicitly with process improvement and is in conflict. In addition “decision analysis and resolution” interferes with Scrum and XP due to the demand of a formal evaluation process. The major part of the process areas can be attained by agile methods. But often, the methods have to be extended by additional practices to fully satisfy the process areas.

There are only minor differences between the ratings of Scrum and XP. Scrum, not addressing development activities, gets lower ratings than XP in accordant process areas (“configuration management”, “technical solution”, “product integration” and “verification”). On the other hand, Scrum performs slightly better in process areas dealing with project management “measurement and analysis” and “integrated project management for IPPD”) and according generic practices (“provide resources” and “review status with higher level management”).

This analysis shows that XP and Scrum cover only project related, but not process related process areas.

E. Interrelations between Agile Methods and Process Maturity Models

CMMI evaluates an organization as a whole and its development processes. In contrast, an agile method is (a framework or sometimes only a fragment of) one individual development process. Thus, the concepts are not comparable per se. Their focus is different, but still they have interrelations. Paulk summarized that CMM is a method for software management whereas agile approaches are methods for software development [21]. They not only can coexist, but they even support each other [12].

It is quite convincing to say that CMMI is an appropriate way to improve processes also in an agile environment. Checking an agile method’s coverage of the process areas reveals shortcomings in the approach and thereby improvement potentials. However, process improvement with CMMI can only be carried out up to a certain degree since there are several process areas which are in conflict with agile principles. Some process areas of level 3 and most of level 4 and 5 are unattainable without sacrificing some agile bedrocks. This would weaken the agile method and eliminate several of its benefits. Also, such actions would be contradictory to the aim of CMMI, i.e. improving the process by making the agile method as good as possible and not turning it into a different kind of method which isn’t agile anymore. So it is concluded that the best improvement approach in an agile environment is to stop at CMMI level 3.

Implicitly, it can be concluded that CMMI levels have to be judged considering the process model employed in an organization. But also, like for every traditional process, refining the agile processes needs to be regarded as an ongoing, success-critical task.

IV CONCLUSION AND FUTURE WORK

Which CMMI process areas can be covered by Scrum and XP were analyzed in detail. Process areas where the methods have to be adjusted to fulfill CMMI goals were identified. Some process areas were in conflict with the two methods and agile principles in general. Most of the process areas can be fulfilled using agile methods. However some are clearly in conflict. Through the use of CMMI, shortcomings of agile methods can be identified. Therefore it can be concluded that process improvement with CMMI can also be carried out when using agile methods. However, since some process areas, mainly those of the maturity levels 4 and 5, are in conflict with agile principles, agile methods can be applied without any major adaptations up to level 2 and up to 3 with some minor changes described in this paper. Extending the project focus of agile methods to an organization-wide perspective would help to make use of the existing concepts of ongoing process-improvement.

If these concepts are employed in agile environments, agile methods will further gain acceptance. But today, an obstacle for process improvement with CMMI is the difficulty to carry out assessments of projects which use agile methods. The specific practices suggested by CMMI often differ from agile approaches. Assessors therefore encounter serious problems when trying to analyze a project. To remedy this situation a catalogue of practices and sub-practices that are typically used by agile methods to implement CMMI goals should be developed.

Here, only XP and Scrum were discussed. To make the results more general, further agile methods should be analyzed as well. In addition, concrete guidelines should be established which show how agile methods can be enhanced to fully cover all the process areas that are not in conflict. For this the present work can be seen as a starting point.

REFERENCES

[1] V-Modell XT Portal. http://www.v-modell-xt.de.
[2] D. J. Anderson. Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI Process Improvement at Microsoft Corporation. In AGILE, pages 193–201, 2005.
[3] K. Beck and C. Andres. Extreme Programming Explained: Embrace Change. Addison-Wesley, 2nd edition, 2004.
[4] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for Agile Software Development, accessed in 2006. http://AgileManifesto.org/.
[5] A. Cockburn. Agile Software Development. Addison-Wesley, 2002.
[6] A. Cockburn and J. Highsmith. Agile Software Development: The People Factor. IEEE Computer, 34(11):131–133, 2001.
[7] D. Cohen, M. Lindvall, and P. Costa. Agile Software Development: A DACS State-of-the-Art Report. Technical report, 2003. http://www.thedacs.com/techs/agile/agile.pdf.
[8] M. Doernhoefer. Surfing the net for software engineering notes. SIGSOFT Softw. Eng. Notes, 31(1):5–13, 2006.
[9] C. Dogs and T. Klimmer. Agile Software-Entwicklung kompakt. mitp-Verlag, 2005.
[10] M. Fowler. Using an Agile Software Process with Offshore Development, accessed in 2005.
http://www.martinfowler.com/articles/agileOffshore.html.
[11] M. Fritzsche. Agile Methoden im industriellen Umfeld. Master's thesis, Technische Universit¨at M¨unchen, 2005.
[12] H. Glazer. Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativity. CrossTalk: The Journal on Defense Software Engineering, (14):27–30, 2001.
[13] J. D. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, and M. C. Paulk. Software Quality and
the Capability Maturity Model. Commun. ACM, 40(6):30–40, 1997.
[14] J. Highsmith. Extreme Programming: Agile Project Management Advisory Service White Paper, accessed in 2005. http://www.cutter.com/freestuff/ead0002.pdf.
[15] S. E. Institute. Capability Maturity Model Integration (CMMI), Version 1.1 (CMMISE/ SW/IPPD/SS, V1.1). Technical report, Software Engineering Institute, Carnegie Mellon University, 2002.
[16] J. J. Jiang, G. Klein, H.-G. Hwang, J. Huang, and S.-Y. Hung. An exploration of the relationship
between software development process maturity and project performance. Inf. Manage., 41(3):279–288, 2004.
[17] T. K¨ahk¨onen and P. Abrahamsson. Achieving CMMI Level 2 with Enhanced Extreme Programming
Approach. In PROFES, pages 378–392, 2004.
[18] D. Kane and S. Ornburn. Agile Development: Weed or Wildflower? CrossTalk: The Journal on Defense Software Engineering, 2002.
[19] P. Keil. Principal agent theory and its application to analyze outsourcing of software development.
In EDSER '05: Proceedings of the seventh international workshop on Economics-driven software engineering research, pages 1–5, New York, NY, USA, 2005. ACM Press.
[20] M. C. Paulk. Using the Software CMM With Good Judgment. ASQ Software Quality Professional,
1(3), 1999.
[21] M. C. Paulk. Extreme Programming from a CMM Perspective. IEEE Software, 18(6):19–26, 2001.
[22] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber. Capability Maturity Model, Version 1.1. IEEE Softw., 10(4):18–27, 1993.
[23] M. Poppendieck and T. Poppendieck. Lean Software Development: An Agile Toolkit. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.
[24] R. S. Sangwan and S. P. Masticola. Model-Driven Rapid Application Development: A Framework for Agile Development in Outsourced Environments. Technical report, Siemens Corporate Research, 2004.
[25] K. Schwaber and M. Beedle. Agile Software Development with Scrum. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
[26] M. Simons. Internationally Agile, accessed in 2005.http://www.informit.com/articles/article .asp? p=25929.
[27] R. Turner. Agile Development: Good Process or Bad Attitude? In PROFES, pages 134–144, 2002.
[28] R. Turner and A. Jain. Agile Meets CMMI: Culture Clash or Common Cause? In XP/Agile Universe, pages 153–165, 2002.
[29] R. van Solingen. Measuring the ROI of Software Process Improvement. IEEE Software,21(3):32–38, 2004.