Open access • Book Chapter • DOI:10.1007/978-3-642-04133-4_8

Mapping CMMI Level 2 to Scrum Practices: An Experience Report — Source link

Jessica Díaz, Juan Garbajosa, Jose A. Calvo-Manzano

Institutions: Technical University of Madrid

Published on: 02 Sep 2009 - European conference on Software Process Improvement

Topics: Capability Maturity Model Integration, Empirical process (process control model), LeanCMMI, Process area and Agile software development

Mapping CMMI Level 2 to Scrum Practices: An Experience Report

Jessica Diaz[1], Juan Garbajosa[1], and Jose A. Calvo-Manzano[2]

[1] Systems & Software Technology Group (SYST), E.U. Informática
[2] Dpto. LSIIS, Facultad de Informática,
Technical University of Madrid (UPM), Madrid, Spain
yesica.diaz@upm.es, jgs@eui.upm.es, jacalvo@fi.upm.es

Abstract. CMMI has been adopted advantageously in large companies for improvements in software quality, budget fulfilling, and customer satisfaction. However SPI strategies based on CMMI-DEV require heavy software development processes and large investments in terms of cost and time that medium/small companies do not deal with. The so-called light software development processes, such as Agile Software Development (ASD), deal with these challenges. ASD welcomes changing requirements and stresses the importance of adaptive planning, simplicity and continuous delivery of valuable software by short time-framed iterations. ASD is becoming convenient in a more and more global, and changing software market. It would be greatly useful to be able to introduce agile methods such as Scrum in compliance with CMMI process model. This paper intends to increase the understanding of the relationship between ASD and CMMI-DEV reporting empirical results that confirm theoretical comparisons between ASD practices and CMMI level2.

Keywords: CMMI, Agile Software Development, Scrum.

1 Introduction

A wide range of large organizations rely on the Capability Maturity Model Integration (CMMI) as indicator for organizational maturity and they enforce that all their processes are a certain capability level of compliance. The reason is that improvements in software quality, budget and milestones fulfilling, and customer satisfaction usually have been associated with higher levels of CMMI compliance [1] [2]. These improvements have been reported for example by Galin et al. [3] who analyzed more than 400 projects during the 1990s about plan-driven software development methods where continuous CMMI-based SPI (Software Process Improvement) strategies were applied. However, medium and small organizations, usually featured by sparse resources, have a lot of difficulties to apply CMMI [4] [5] [6]. Some reported data prove that over 77 percent of process improvements have taken longer than expected, and over 68 percent have cost more than expected too [7].

At the same time organizations look for the improvement of their processes and they must respond continually to changing environments in a global market.

R.V. O'Connor et al. (Eds.): EuroSPI 2009, CCIS 42, pp. 93–104, 2009.

The rapid change increases frustration to the heavyweight plans, specifications, and other documentation imposed by plan-driven software development with maturity model compliance criteria [8]. Some authors assert even CMMI is not applicable to turbulent and volatile business environments [9] concluding that processes not only must respond to change but embrace it [10].

The competitiveness and evolution of the software market has led software companies to avoid heavy software development methodologies and to follow light software development methodologies, which are open for new changes. From these needs, Agile Software Development (ASD) [11, 12] emerged with the definition of the Agile Manifesto [13]. The Agile Manifesto is a statement of the principles that underpin agile software development, some of them are continuous delivery of valuable software, simplicity, on-site customer, and welcome changing requirements. ASD is mainly based on the improvement of the software development productivity, the human relationships of the development team, the tacit knowledge processes with little ware, adaptive planning, and lightweight. These values are preserved by introducing the customer as another member of the development team and by doing short time-framed software development iterations. These short iterations allow the checking of partial results of the work product and the introduction of new changes in a simple way. As a result, software development is more effective and adaptable; so agile methodologies have proved its effectiveness in projects with very changing requirements [14] [15]. ASD is growing mature for large projects, and this is demonstrated by its increasing put into practice at the industry [16, 17, 18], even for outsourcing projects [19]. In fact, the data reported in [16] show that over 69 percent of analyzed organizations are putting into practice agile practices on their projects.

But, what about CMMI compliant organizations that need to introduce light software development methods for adapting to turbulent markets? And, what about agile organizations whose clients require a certain CMMI level of compliance? These issues lead to the challenge for embracing CMMI-based SPI strategies and agile principles, as well as understanding the relationship between both approaches. This challenge may be addressed through an effort to stretch agile to fit CMMI analyzing the interrelations, constraints, and adjustments between agile and CMMI. Comparisons between CMMI and ASD have often been criticized comparing them like oil and water [20]. However the literature has summarized that CMMI and agile are compatible [20, 10] because agile methods are development process descriptions and CMMI is a reference process model that it is used for appraisals and improvements [21]. This means, CMMI tells us what to do, while agile methods tells us how to do it.

The primary purpose of this paper is to increase the understanding of the relationship between ASD and CMMI-DEV [22]. This paper reports empirical results that confirm the theoretical comparisons [23, 24, 25, 26, 27] between agile practices (in particular Scrum method) and three processes related to CMMI capability level 2. The paper is organized as follows. Section 2 analyzes background and related work. A mapping between CMMI specific practices and agile practices is described in section 3. Section 4 presents an internal CMMI appraisal in

a software development process in which agile practices are used. Finally, some conclusions and future work are presented in section 5.

2 Background

2.1 CMMI Overview: CMMI v1.2

CMMI for Development [22] is a reference model that consists of best practices that address development and maintenance activities applied to products and services. CMMI-DEV contains practices that cover project management, process management, systems engineering, hardware engineering, software engineering, and other supporting processes used in development and maintenance.

2.2 ASD Overview: Scrum

Agile methodologies provide the infrastructure (i) to evaluate the state of the product, (ii) to identify new changes in the development process, and (iii) to incorporate them in the final product by means of continuous integration. There are different agile methodologies such as Scrum [28] or eXtreme Programming (XP) [29]. Each one of them defines their own techniques for planning, estimating, or reviewing, but all of them are based on the same values defined by Agile Manifesto. Even, some of them share some practices, for example requirements in agile are captured as User Stories (US) [30]. The US objective is to reduce the cost of the requirement elicitation and management by means of scenarios written by customers without techno-syntax versus conventional methodologies based on formal requirements specification documents. These previous guidelines have offered a general vision of agile methodologies but this work has been focused on the Scrum methodology. Following Scrum is described in detail.

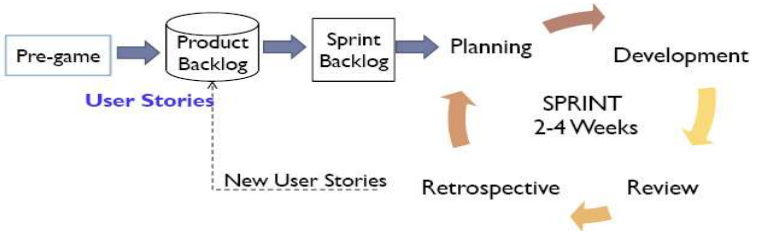

Fig. 1. Scrum Lifecycle

Scrum implements an iterative, incremental life cycle (see Figure 1) which involves three stakeholders: the *Product Owner*, the *Team*, and the *ScrumMaster* [28]; all together make up the *Scrum Team*. The Scrum life cycle defines a pre-game phase at the project beginning; planning, review, and retrospective meetings in an iterative way; and daily meetings during the whole iteration. The

pre-game phase consists in a light planning process where representative customers and members of the Scrum Team capture requirements as US; the result is the *product backlog*, a list of known US. Then US are prioritized and divided into short time-framed iterations called *sprints*. A sprint is a 2-4 weeks period of development time. Each sprint has a sprint *planning meeting* at the sprint beginning where the Product Owner and Team plan together about what to be done for the next sprint; the result is the *sprint backlog*, a list of US and tasks that must be performed to achieve the *sprint goal*, i.e., to deliver an increment valuable functionality of the final product. During the execution of each sprint, the team meets *daily in 15-minute meetings* to track the work progress answering three questions [28]: *What have I done since the last Scrum meeting?, What will I do before the next Scrum meeting?, What prevents me from performing my work as efficiently as possible?*

Anything that prevents a team member from performing his work as efficiently as possible is an impediment. The ScrumMaster is in charge of ensuring *impediments* get resolved; for it project adjustments could be necessary. At the end of the sprint, in the sprint *review meeting*, the Team asks the Product Owner whether the goals were met, the Product Owner could change US, add US, etc. Finally a *retrospective meeting* is held between the Team and ScrumMaster to discuss what was well and what could be improved for the next sprint; this is an estimate and tracking activity to achieve continuous improvement; i.e., retrospective meetings provide feedback to apply needed changes and adjustments for the next sprint.

2.3 Related Work

Existing literature has summarized that CMMI and agile are compatible [10, 20, 31, 23, 24, 25, 26, 27, 32, 33, 34, 35], even that hybrid approaches that combine both agile methods and methods based on the CMM[1] are feasible and necessary [36].

Only few works show how to achieve CMMI levels with agile practices, some of them are high level, theoretical and difficult to implement in a general full software product life cycle, and often do not provide specific details and examples. Theoretical comparisons between XP and CMM claim that XP does not fulfill CMM requirements but it may be possible to construct a process that fulfills CMM level 2 and 3 by adding sound practices to XP [34, 23, 33]. Vriens suggests that it is possible to achieve CMM levels 2 process areas using a combination of XP and Scrum as the base for the software development process [24]. Kähkönen and Abrahamsson [35] have reported empirical evidences when CMMI is used for assessing software development processes where XP practices are used. Afterward, some works haver assert that CMMI level 5 may be possible [32, 27]. Fritzsche and Keil [25], in turn, state that level 4 or 5 are not feasible under the current specifications of CMMI and XP, and describe the limitations of CMMI in an agile environment. Pikkarainen and Mäntyniemi [21] propose an approach for agile software development assessment and improvement strategies using CMMI;

[1] Some studies are related to the previous version of CMMI.

this approach is based on a mapping between CMMI specific goals and agile practices and supported by empirical evidences. However only two process areas are supported (Project Planning and Requirements Management) and only from a CMMI goal (not specific practice). Marcal et. al [26] describe a more detail mapping between CMMI Project Management Process Area to Scrum practices but do not provide empirical evidences.

Unlike these researches, our work tries to increase the detail of previous mappings between Scrum and CMMI, and to illustrate this mapping with a case study providing empirical evidences of the obtained results.

3 Mapping between CMMI Specific Practices and Scrum Practices

Software requirements elicitation, budgeting, and scheduling are very relevant process areas in software development. For it Project Planning (PP), Project Monitoring and Control (PMC) and Requirements Management (REQM) CMMI process areas were mapped with SCRUM practices.

3.1 Project Planning (PP)

According to CMMI-DEV, the aim of PP is to establish and maintain plans that define project activities. PP has 3 specific goals (SG) that enclose 14 specific practices (SP). A detailed description is carried out below:

- *SP1.1 Estimate the Scope of the Project.* Basically it consists in the identification of work packages in sufficient detail to specify estimates of project tasks, roles, responsibilities, and schedule. It is covered by the Scrum pregame phase where the product backlog and the sprints are defined; both items provide the resources for estimate the scope of the project.
- *SP1.2 Establish Estimates of Work Product and Task Attributes.* Estimate is carried out in two levels: product level and sprint level. So, Scrum establishes a first estimation in the pre-game phase and an iterative estimate in the sprint beginning (planning meeting). Estimates usually are based on size or complexity attributes. Some agile practices recommend the *Planning Poker*[2] estimation technique; it is based on the consensus of the participants (similar to Wideband Delphi) for estimating relative size of US. Some units might include story points [37] or function points.
- *SP1.3 Define Project Lifecycle.* This specific practice is fully addressed by Scrum because it defines the lifecycle shown in Figure 1.
- *SP1.4 Determine Estimates of Effort and Cost.* Again estimation is carried out in two levels: product level and sprint level. Product estimates are high level and less accurate and sprint estimates are low level and more accurate than the first ones. Scrum practitioners estimate the US effort in ideal

[2] http://www.planningpoker.com/

engineering days based on previous sprints (historical base of sprint backlogs), previous projects (historical base of product backlogs), capacity for the forthcoming sprint and the relative US complexity required to deliver the sprint goal. Burndown and Burnup models [37] facilitating the effort estimate.

- *SP2.1 Establish the Budget and Schedule.* During pre-game phase initial milestones (sprint goals), schedule (sprints), constraints and budget are setup according to the initial product backlog. Additional milestones or budget may be assigned to the project in each sprint during its planning. Corrective action criteria are identified during retrospective meeting. The Product Owner is an outstanding figure to implement these practices in a successful way.
- *SP2.2 Identify Project Risks.* In Scrum risks are captured as impediments (list of impediments). Their identification is not carried out in the initial plan or in a systematic manner. But this practice is partially satisfied in an iterative way, during daily meetings, and impediments are revised in retrospective meeting. The ScrumMaster is the outstanding figure in this identification process.
- *SP2.3 Plan for Data Management.* Any data generated by the project is stored in public folders or white-boards available to everyone [28], but there is no formal data management plan or procedure to collect this data [26]. Privacy and security are another weaknesses.
- *SP2.4 Plan for Project Resources.* During pre-game phase the staffing requirements and equipment list are defined. As the result the Scrum Team is established. During the sprints execution, the ScrumMaster is in charge of providing new resources it should be necessary.
- *SP2.5 Plan for Needed Knowledge and Skills.* Knowledge and skills needs are identified during pre-game phase, however the definition of mechanisms to provide knowledge and skills not found in the organization are considered as impediments and resolved during daily and retrospectives meetings.
- *SP2.6 Plan Stakeholder Involvement.* Scrum defines roles, responsibilities, and involvement of the stakeholders at the beginning and end or each sprint. This involvement is monitored by the ScrumMaster who is in charge of assuring the fulfilling of Scrum practices by all stakeholders.
- *SP2.7 Establish the Project Plan.* To start a Scrum project a vision and a product backlog are the basis for the project plan [28].
- *SP3.1 Review Plans That Affect the Project.* Plans reviews are carried out during planning and retrospectives meetings.
- *SP3.2 Reconcile Work and Resource Levels.* Work reconciliation occurs during planning meetings because product backlog is dynamic, so new estimations or schedules are possible.
- *SP3.3 Obtain Plan Commitment.* The commitment is obtained in an iterative way during face to face planning meetings in which stakeholders are involved.

3.2 Project Monitoring and Control (PMC)

According to CMMI-DEV, the aim of PMC is to establish and maintain plans that define project activities. PMC has 2 specific goals (SG) that enclose 10 specific practices (SP). The mapping described in Table 1 was carried out.

3.3 Requirements Management (REQM)

According to CMMI-DEV, the aim of REQM is to manage the requirements of the projects products. REQM has 1 specific goal (SG) that encloses 5 specific practices (SP). The mapping described in Table 2 was carried out.

Table 1. Mapping between PMC specific practices and Scrum practices

PMC specific practices	Scrum practices
SP1.1 Monitor Project Planning Parameters SP1.2 Monitor Commitments SP1.3 Monitor Project Risks	Daily and Retrospective meetings
SP1.4 Monitor Data Management	Not supported
SP1.5 Monitor Stakeholder Involvement	Retrospective meetings
SP1.6 Conduct Progress Reviews	Review meetings. Burndown and Burnup graphs
SP1.7 Conduct Milestone Reviews	Review meetings
SP2.1 Analyze Issues	Daily and Retrospective meetings
SP2.2 Take Corrective Action	Review meetings
SP2.3 Manage Corrective Action	Retrospective meetings

Table 2. Mapping between REQM specific practices and Scrum practices

REQM specific practices	Scrum practices
SP1.1 Obtain an Understanding of Requirements	User Stories (US) in an iterative way (sprints)
SP1.2 Obtain Commitment to Requirements	Planning meetings. Backlogs
SP1.3 Manage Requirements Changes	Planning and Review meetings
SP1.4 Maintain Bidirectional Traceability of Requirements	User Stories (US)
SP1.5 Identify Inconsistencies Between Project Work and Requirements	Pre-game and Planning meetings

4 An Experience Report: An Internal CMMI Appraisal

Once theoretical comparisons between Scrum and CMMI (level 2 for PP, PMC and REQM) were established, an internal assessment was carried out to confirm these hypotheses. An internal assessment against a CMMI reference model provided evidences about good agile practices, strengths and weaknesses for achieving a CMMI level 2 in agile contexts.

4.1 Case Study Description

The assessed project consisted in a software evolution of a product called Test and OPeration Environment (TOPEN) [38]. TOPEN is an acceptance testing tool built in-house that provides mechanisms for the definition and execution of operation and test cases through a domain specific language. The product evolution consisted in adapting TOPEN to test a biogas plant. The product evolution was developed following Scrum method in 6 sprints and 15 weeks. The Scrum Team was composed of 8 engineers: a Product Owner, a ScrumMaster, and a Team of six developers. An internal proxy customer was taken into account too. The Scrum methodology was applied as it is described following.

During the pre-game phase, US were first captured, together with the proxy customer, which formed a product backlog. The US were grouped in sprints of two weeks approximately. A planning meeting was established for every sprint. During the planning meeting, the sprint ending date is defined and the initial US are further elaborated together with the Product Owner and the Team in by means of a *planning game*. The planning game is technique that guides the estimating of the US involving all the Scrum Team. However, the developers found that the US estimations were too optimistic in the first planning games, which made several deviations during the first sprints. Through sprints developers learned more about Scrum practices, the needs of the customer and the product under development. As a consequence, the US estimations became more precise. After the planning game, the sprint backlog is formed. Product backlog and sprint backlogs were stored and managed through a tool named Rally[3]. Rally is a web based tool for managing user stories, tasks, backlogs, plan, releases, test cases, and defects.

During the sprint, daily meetings solved small problems in an agile way making technical decisions by themselves (self organizing teams). At the end of the sprint, a progress report was elaborated in the review meeting. The customer representatives validated the work products (documents, releases, or other artefacts), and thus the inconsistencies between their needs, plans and project work were continuously followed. Changes in the client needs were discussed, and the product backlog was updated correspondingly. Finally a retrospective meeting was established at the end of every sprint for analyzing strengths, weaknesses, problems, and improvements of the methods, the team and the project. The feedback obtained was applied to the following sprints.

4.2 A CMMI Appraisal Process Approach

Once the empirical case project has been described, the next step is the appraisal process description. We are selected the appraisal process defined by [21]. It is characterized by (i) appraisal teams of 3-4 members, (ii) appraisal time of 2-3 weeks, (iii) require considerable resources, (iv) medium intrusiveness, and (v) medium reliability and validity of the appraisal results.

[3] http://www.rallydev.com/

Three participants in the appraisal have scored each subpractice related to CMMI on a questionnaire; this questionnaire is supported by interviews with participants and reviews of the project documentation.

4.3 Results

Figure 2 and Figure 3 and Figure 4 show the results of the appraisal for some PP, PMC and REQM specific goals. Figure 2 shows the results of the appraisal for PP process area. Subpractices for SG1 are satisfied for this case study where Scrum method was applied. This process area is a challenge for the team because this case study was the first contact with Scrum method. However, since planning is an iterative process repeated at the beginning of the sprints, the team had the chance to improve the process practices in each sprint. So, the iterative planning enabled development teams to estimate more accuracy and answer to

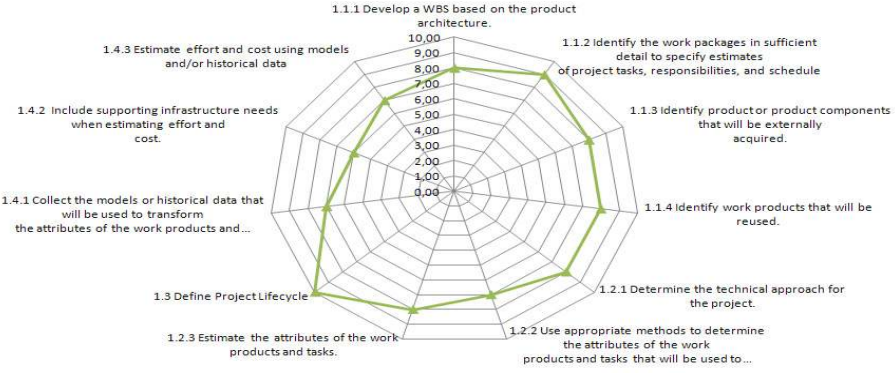

Fig. 2. PP - SG1 Establish Estimates

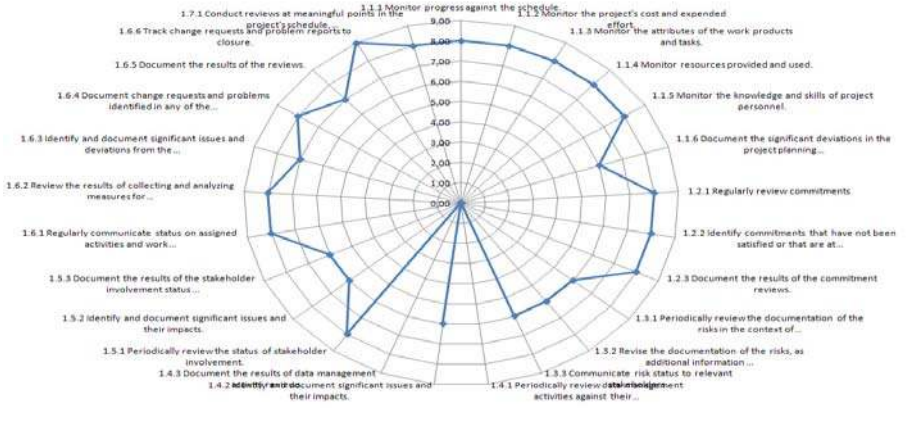

Fig. 3. PMC - SG1 Monitor Project Against Plan

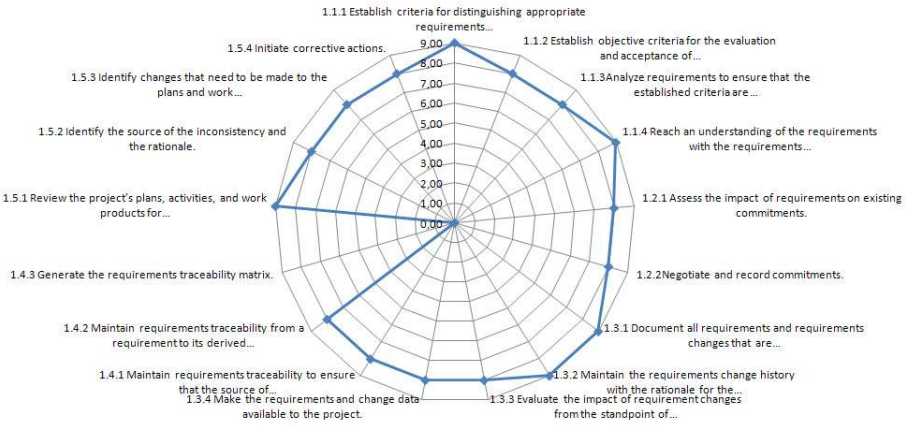

Fig. 4. REQM - SG1 Manage Requirements

changes quickly. As the project progressive, historical data of previous sprints were collected and used in order to estimate effort and cost.

Figure 3 shows the results of the appraisal for PMC process area. Subpractices for SG1 are largely satisfied because the Scrum lifecycle defines explicitly times for monitoring and control through daily, review, and retrospective meetings. Finally Figure 4 shows the results of the appraisal for REQM process area. Subpractices for SG1 are largely satisfied. Customers must not specify most of the requirement at the project beginning, so understanding of requirements is easier through iterative sprints and requirements change processes are flexible and largely supported by Scrum method.

5 Conclusions and Further Work

Agile methodologies are associated commonly to informal and lightweight documentation that do not emphasize process definition or measurement to the degree that models such as the CMMI do. However the literature has proved that CMMI model can be applied in a lightweight manner without incurring in excessive documentation. In particular, this paper has proved that Scrum processes can be considered valid under the CMMI paradigm. So, the appraisal has provided evidences that those process areas related to CMMI-DEV level 2 were largely covered. These results will be used for learning and selecting practices for the following agile projects.

The conclusion is that agile methodologies provide many good engineering practices, and together with CMMI, both approaches can achieve very positive synergies. Since Scrum method provides criteria to identify a minimum set of good practices to achieve CMMI capability level 2, small-medium organizations can take advantage of more flexible and lightweight methods to achieve a certain CMMI level compliance.

Acknowledgment

The work reported here has been partially sponsored by the OVAL/PM TIC2006-14840 project, the FLEXI FIT-340005-2007-37 (ITEA2 6022) project and UPM under their Researcher Training program.

References

1. Herbsleb, J., Carleton, A., Rozum, J., Siegel, J., Zubrow, D.: Benefits of cmm-based software process improvement: Initial results. Technical report, CMU/SEI-94-TR-013, Software Engineering Institute (1994)
2. Goldenson, D.R., Gibson, D.L.: Demonstrating the impact and benefits of cmmi: An update and preliminary results. Technical report, CMU/SEI-2003-SR-009, Software Engineering Institute (2003)
3. Galin, D., Avrahami, M.: Are cmm program investments beneficial? analyzing past studies. IEEE Software 23(6), 81–87 (2006)
4. Paulk, M.: Using the software cmm in small organizations. In: Proc. Joint 16th Pacific Northwest Software Quality Conf. and 8th Int'l Conf. Software Quality, Washington, DC, USA, pp. 350–360. IEEE Computer Society, Los Alamitos (1998)
5. Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., Murphy, R.: An exploratory study of why organizations do not adopt cmmi. Journal of Systems and Software 80(6), 883–895 (2007)
6. Pino, F.J., García, F., Piattini, M.: Software process improvement in small and medium software enterprises: a systematic review. Software Quality Control 16(2), 237–261 (2008)
7. Goldenson, D.R., Herbsleb, J.D.: After the appraisal: A systematic survey of process improvement, its benefits, and factors that influence success. Technical report, CMU/SEI-95-TR-009, Software Engineering Institute (1995)
8. Boehm, B.: A view of 20th and 21st century software engineering. In: ICSE 2006: Proceedings of the 28th international conference on Software engineering, pp. 12–29. ACM, New York (2006)
9. Lebsanft, K.: Process improvement in turbulent times – is cmm still an answer? Product Focused Software Process Improvement, 78–85 (2001)
10. Cohen, D., Lindvall, M., Costa, P.: An introduction to agile methods. Advances in Computers 62, 2–67 (2004)
11. Cockburn, A.: Agile Software Development: The Cooperative Game, 2nd edn. Addison-Wesley Professional, Reading (2006)
12. Abrahamsson, P.: Agile software development methods review and analysis. Technical report, VTT Electronics, 112 (2002)
13. K. Beck et al.: The agile manifesto, `www.agilemanifesto.org` (accessed, February 2009)
14. Dingsoyr, T., Dybå, T., Abrahamsson, P.: A preliminary roadmap for empirical research on agile software development. In: AGILE 2008: Proceedings of the Agile 2008, Washington, DC, USA, pp. 83–94. IEEE Computer Society, Los Alamitos (2008)
15. Dybå, T., Dingsoyr, T.: Empirical studies of agile software development: A systematic review. Inf. Softw. Technol. 50(9-10), 833–859 (2008)
16. Ambysoft: Agile adoption rate survey (February 2008), `http://www.ambysoft.com/surveys/agileFebruary2008.html`

17. Ambler, S.W.: Has agile peaked? let's look at the numbers (May 2008), http://www.ddj.com/architect/207600615?pgno=1
18. Flexi research project: Flexi newsletter (February 2008) ISBN 978-951-42-8586-8
19. Fowler, M.: Using an agile software process with offshore development (July 2006), http://www.martinfowler.com/articles/agileOffshore.html
20. Turner, R., Jain, A.: Agile meets cmmi: Culture clash or common cause? In: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002, London, UK, pp. 153–165. Springer, Heidelberg (2002)
21. Pikkarainen, M., Mäntyniemi, A.: An approach for using cmmi in agile software development assessments: Experiences from three case studies. In: Proceedings of SPICE 2006 (2006)
22. CMMI Product Team: Cmmi for development, version 1.2. Technical report, CMU/SEI-2006-TR-008, ESC-TR-2006-008, Software Engineering Institute (2006)
23. Paulk, M.C.: Agile methodologies and process discipline. The Journal of Defence Software Engineering (October 2002)
24. Vriens, C.: Certifying for cmm level 2 and is09001 with xp@scrum. In: Proceedings of the Agile Development Conference. ADC 2003, June 2003, pp. 120–124 (2003)
25. Fritzsche, M., Keil, P.: Agile methods and cmmi: Compatibility or conflict? e-Informatica Software Engineering Journal 1(1) (2007)
26. Marcal, A.S.C., Soares, F.S.F., Belchior, A.D.: Mapping cmmi project management process areas to scrum practices. In: SEW 2007: Proceedings of the 31st IEEE Software Engineering Workshop, Washington, DC, USA, pp. 13–22. IEEE Computer Society, Los Alamitos (2007)
27. Sutherland, J., Jakobsen, C., Johnson, K.: Scrum and cmmi level 5: The magic potion for code warriors. In: AGILE 2007, August 2007, pp. 272–278 (2007)
28. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice-Hall, Englewood Cliffs (2002)
29. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading (1999)
30. Cohen, M.: User Stories Applied for Agile Software Development. The Addison-Wesley Signature Series (2004)
31. Paulk, M.C.: Extreme programming from a cmm perspective. IEEE Software 18(6), 1–8 (2001)
32. Anderson, D.J.: Stretching agile to fit cmmi level 3 - the story of creating msf for cmmi®process improvement at microsoft corporation. In: ADC 2005: Proceedings of the Agile Development Conference, pp. 193–201. IEEE Computer Society, Los Alamitos (2005)
33. Glazer, H.: Dispelling the process myth: Having a process does not mean sacrificing agility or creativity. The Journal of Defence Software Engineering (November 2001)
34. Martinsson, J.: Maturing xp through the cmm. In: Extreme Programming and Agile Processes in Software Engineering (2003)
35. Kähkönen, T., Abrahamsson, P.: Achieving CMMI level 2 with enhanced extreme programming approach. In: Bomarius, F., Iida, H. (eds.) PROFES 2004. LNCS, vol. 3009, pp. 378–392. Springer, Heidelberg (2004)
36. Barry, B.: Get ready for agile methods, with care. Computer 35(1), 64–69 (2002)
37. Buglione, L., Abran, A.: Improving estimations in agile projects: issues and avenues. In: Proceedings of the 4th Software Measurement European Forum (SMEF 2007), May 9-11, pp. 265–274 (2007)
38. Magro, B., Garbajosa, J., Perez-Benedi, J.: A software product line definition for validation environments. In: Software Product Lines Conference, SPLC (2008)