Scrum and CMMI Level 5: The Magic Potion for Code Warriors

Jeff Sutherland, Ph.D.
Patientkeeper Inc.
jeff.sutherland@computer.org

Carsten Ruseng Jakobsen
Systematic Software Engineering
crj@systematic.dk

Kent Johnson
AgileDigm Inc.
kent.johnson@agiledigm.com

Abstract

Projects combining agile methods with CMMI[1] are more successful in producing higher quality software that more effectively meets customer needs at a faster pace. Systematic Software Engineering works at CMMI level 5 and uses Lean Software Development as a driver for optimizing software processes. Early pilot projects at Systematic showed productivity on Scrum teams almost twice that of traditional teams. Other projects demonstrated a story based test driven approach to software development reduced defects found during final test by 40%.

We assert that Scrum and CMMI together bring a more powerful combination of adaptability and predictability than either one alone and suggest how other companies can combine them.

1. Introduction

Successful software development is challenged by the supplier's ability to manage complexity, technology innovation, and requirements change. Agile and CMMI methods both address these challenges but have very different approach and perspective in methods applied.

Management of complexity requires process discipline while management of change requires adaptability. CMMI provides process discipline and Scrum enhances adaptability. This paper provides an analysis of the effect of introducing Agile practices into a CMMI Level 5 company.

2. Scrum and CMMI: a magic potion

Systematic, an independent software systems company, was established in 1985 and employs more than 400 people worldwide with offices in Denmark, USA and the UK. Solutions developed by Systematic are used by tens of thousands of people in the defense, healthcare, manufacturing, and service industries. Systematic was appraised 11 November 2005 using the SCAMPI[SM2] method and found to be CMMI level 5 compliant.

At Systematic CMMI Level 5 practices have reduced rework by 42%, maintained estimation precision deviation less than 10%, and assure 92% of all milestones are delivered early or on time. At the same time, extra work on projects has been significantly reduced.

More importantly, Systematic has transformed over twenty years of experience into a unified set of processes used by all software projects. Historical data are systematically collected and analyzed to continuously provide insight into the capability and performance of the organization.

The use of a shared common process makes it easier for people to move from project to project and share experiences and lessons learned between projects. Insight into the capability and performance of processes makes it possible to evaluate performance of new processes to performance of existing processes. And this forms the foundation for continuous improvement.

In short, Systematic was able to deliver what the customer has ordered on schedule, cost and quality using 69% effort compared to a CMMI Level 1 company [1, 2] prior to the introduction of Scrum.

CMMI Level 5 is increasingly a requirement from customers and key to obtaining large contracts, especially within defence and healthcare. Customers recognize that CMMI Level 5 gives high predictability and better-engineered product for scalability, maintainability, adaptability, and reliability.

CMMI provides insight into what processes are needed to maintain a **disciplined** mature organization capable of predicting and improving performance of the organization and projects. Scrum provides guidance for efficient management of projects in a way that allows for high flexibility and **adaptability**. When mixing the two, a magic potion emerges, where the mindset from Scrum ensures that processes are implemented efficiently while embracing change, and CMMI ensures that all relevant processes are

[1] ® Capability Maturity Model, CMM and CMMI are registered in the U.S. Patent and Trademark Office.

[2] SM Capability Maturity Model Integration, and SCAMPI are service marks of Carnegie Mellon University.

considered. Scrum now reduces every category of work (defects, rework, total work required, and process overhead) by almost 50% compared to our previous CMMI Level 5 implementation while maintaining the same level of process discipline .

Individually CMMI and Scrum has proven benefits but also pitfalls. An Agile company may implement Scrum correctly but fail due to lack of institutionalization, (see section 3) or inconsistent or insufficient execution of engineering or management processes. CMMI can help Agile companies to institutionalize Agile methods more consistently and understand what processes to address.

A company can comply with CMMI, but fail to reach optimal performance due to inadequate implementation of processes. Scrum and other Agile methodologies can guide such companies towards more efficient implementation of CMMI process requirements.

2.1. Systematic Lean experience

Systematic made a strategic decision to use Lean as the dominant paradigm for future improvements after achieving CMMI level 5. Lean has demonstrated notable results for many years in domains such as auto manufacturing, and due to its popularity, has been adapted to other domains, including product and software development. Systematic identified Lean Software Development [3] as the Lean dialect most relevant to Systematic.

Applying Lean Software Development, as a driver for future improvements in a company appraised to CMMI level 5, depends on the adoption of a lean and agile mindset in the implementation of the CMMI processes, and Systematic placed special focus on implementing the Lean change in the spirit of the Agile Manifesto.

Lean competencies were established, through handing out handout of books, formal and informal training, and walk-the-talk activities. Project Managers were trained in Lean Software Development, and Mary Poppendieck visited Systematic to present a management seminar on Lean Software Development.

This seminar established an understanding of the Agile and Lean mindset. The causal dependencies between the principles and tools in Lean Software Development were analyzed, by Carsten Jakobsen appointed change agent for Lean, and resulted in the model shown in Table 1. The model groups the thinking tools (T) and principles (P) from Lean Software Development according to causal dependencies, where elements to the right depend on one or more elements to the left. The model facilitated a way to prioritize what thinking tools to focus on. Left most tools were considered good candidates to start with.

The most important input for selection of what Lean tools to consider first. was an analysis showing improvement opportunities with a potential good cost-benefit. Internal studies at Systematic show that the cost of fixing a defect increases from 1.6 hours when detected in the coding phase, to 12 hours when detected in the testing phase and 23.7 hours when detected in the maintenance phase. Therefore improvements that could eliminate or move any defects to earlier phases have the potential for high leverage. We also observed that our focus on quality, gradually had led to longer test cycles.

	Value	Flow	Pull	Perfection
Engineering	*P6 Integrity* T19 Refactor T20 Test	*P2 Amplify Learning* T5 Synchronization T4 Iterations	*P2 Amplify Learning* T3 Feedback T6 Setbased development	*P6 Integrity* T18 Conceptual T17 Perceived
Management	*P1 Create Value* T1 Find Waste T2 Value Stream	*P4 Deliver Fast* T11 Queue Theory T12 Cost of delay	*P7 See the Whole* T22 Contracts T21 Measures T10 Pull	*P3 Defer Commitment* T7 Options thinking T8 Defer commitment T9 Decision making
People	*P5 Empower team* T16 Expertise	*P5 Empower team* T14 Motivation	*P5 Empower team* T15 Leadership	*P5 Empower team* T13 Self determination

Table 1 Lean Software Development arranged after causal dependencies

2.2. Systematic experience from pilots

The above analysis of Systematic improvement opportunities and Lean causal dependencies led to the decision to seek improvements based on the Lean Software Development principles of Build Integrity In, Amplify Learning and Deliver Fast. These Lean Thinking tools gave inspiration to consider Scrum and

early testing. In a period of approximately 4 months, two small and two large projects described and piloted Scrum and story based early testing.

2.2.1. Scrum. The first pilot was initiated on a request for proposal, where Systematic inspired by Lean principles suggested a delivery plan with bi-weekly deliveries and stated explicit expectations to customer involvement and feedback. The project had a team size of 4 and concerned software for a customer in Danish Government.

One of the main reasons that Systematic was awarded the contract was the commitment to deliver working code bi-weekly and thereby providing a very transparent process to the customer. During project execution, a high communication bandwidth was kept between the team, the customer and users. This was identified as one of the main reasons for achieving high customer satisfaction.

The delivery plan and customer involvement resulted in early detection of technological issues. Had a traditional approach been used these issues would have been identified much later with negative impacts on cost and schedule performance.

However, productivity of this small project was at the expected level compared to the productivity performance baseline for small projects. Another small project with a team size of 5 working for a Defense customer using Scrum shows a similar productivity and the same indications of high quality and customer satisfaction.

At Systematic, productivity for a project is defined as the total number of lines of code produced divided by the total project effort spent in hours. Data are attributed with information related to programming language, type of code: new, reuse or test.

Systematic has established and maintains a productivity performance baseline (PPB) for productivity compared to project size estimated in hours, from data collected on completed projects [4]. The data shows that productivity is high on small projects and declines with the size of the project. The productivity performance baseline in Systematic is divided into two groups: small projects less than 4000 hours and large projects above 4000 hours. Productivity of small projects is 181% the productivity of large projects.

When comparing the projects using Scrum to the current productivity baseline it is seen that productivity for small projects is insignificantly changed, but the productivity for large projects shows a 201% increase in productivity. As mentioned above, the large projects did additional improvements, and it is therefore not possible to attribute the benefit solely to Scrum. However the people involved all agree that Scrum was a significant part of this improvement.

There is a strong indication that large projects in Systematic using Scrum will double productivity going forward. Small projects in Systematic already show a high productivity. We believe that this is because small projects in Systematic always have been managed in a way similar to Scrum. However quality and customer satisfaction seems to be improved and we believe this is because Scrum has facilitated a better understanding of how small projects are managed efficiently.

2.2.2. Early testing. One large project with a team size of 10 worked on a military messaging system. This project was inspired from the Lean thinking tool "Build Integrity In" to investigate how to do early test, and as a result they invented an enhanced story-based approach to early testing in software development. The name "Story based" development was inspired from XP, but our approach included new aspects like: short incremental contributions, inspections, and was feature driven.

The idea of story-based development was to subdivide features of work, typically estimated to hundreds of hours of work into smaller stories of 20-40 hours of work. The implementation of a story followed a new procedure, where the first activity would be to decide how the story could be tested before any code was written. This test could then be used as the exit criteria for implementation of the story.

The procedure included a few checkpoints where an inspector would inspect the work produced, and decide whether or not the developer could proceed to the next activity in the procedure. These inspections are lightweight, and could typically be done in less than 5 minutes.

Many benefits from story-based development were immediately apparent. The combination of a good definition of when a story was complete, and early incremental testing of the features, provided a very precise overview of status and progress for both team and other stakeholders.

Developing a series of small stories rather than parts of a big feature creates a better focus on completing a feature until it fulfills all the criteria for being "done".

This project finished early, and reduced the number of coding defects in final test by 38% compared to previous processes.

Another project with a team size of 19 working on a module to a electronic patient record system, also worked with early testing. They ensured that test activities were integrated into development, with a strong focus on "seeing the whole" and understanding how the solution fit into the customer's domain. For

each week the project defined a goal to be achieved. The project ensured that test and domain specialists were co-located with the developers. This caused discussion and reflection between testers, developers, user experience engineers and software architects, before or very early in the development of new functionality. As a consequence the amount of remaining coding defects in final test were reduced by 42% compared to previous processes.

Based on these two projects, it was concluded that test activities should be an integrated activity through out the projects' lifetime, and Scrum inherently supports this, through cross-functional teams and frequent deliveries to the customer. Furthermore it was concluded that the story-based software development method should be the default recommended method for software development in projects.

2.2.3. Real needs. A customer sent a request for proposal on a fixed set of requirements. When Systematic responded, we expressed our concern that the scope and contents expressed in the requirements were beyond the customer's real needs.

Systematic decided to openly share the internal estimation of the requirements with the customer, for the purpose of narrowing scope by removing requirements not needed or too expensive compared to the customer's budget. The customer agreed to re-evaluate the requirement specification, and the result was that requirements and price were reduced by 50%.

This experience supports results in a Standish Group Study reported at XP2002 by Jim Johnson, showing that 64% of features in a fixed price contract are never or rarely used by end-users.

We believe that this illustrates how important it is to have a frank and open discussion with the customer, in order to find out what the real needs are. Success is not achieved by doing the largest project, but by doing the project that provides the most value for the customer, leaving time for software developers to work with other customers with real needs. This strategy is strongly supported by Scrum.

2.3. Adoption of agile methods

The result of the pilots were two-fold: it confirmed the general idea of using Lean mindset as source for identification of new improvements, and secondly it provided two specific successful improvements, Scrum and story-based early testing, showing how agile methods can be adopted while maintaining CMMI compliance. An important insight for Systematic was that adoption of these agile methods involved only small adjustments to existing processes. The main difference was to adopt a lean and agile mindset in interpretation of existing processes.

The evaluation of the results from the pilot projects led to the decision of adopting Scrum and story based early testing. These methods are now the default choice for new projects, and are integrated in the process descriptions at Systematic.

3. Guide for mixing CMMI and Agile

3.1. How CMMI can improve Agile

Our focus is on using CMMI to help an organization institutionalize Agile Methods. We have all heard Agile Methods described by some as just another disguise for undisciplined hacking and of some individuals who claim to be Agile just because they "don't document." We believe the value from Agile Methods can only be obtained through disciplined use. CMMI has a concept of *Institutionalization* that can help establish this needed discipline.

Institutionalization is defined in CMMI as "the ingrained way of doing business that an organization follows routinely as part of its corporate culture." Others have described institutionalization as simply "this is the way we do things around here." Note that institutionalization is an organizational-level concept that supports multiple projects.

CMMI supports institutionalization through the Generic Practices (GP) associated with all process areas. For the purposes of our discussion, we will look at the 12 generic practices associated with maturity levels 2 and 3 in the CMMI [5] and how they might help an organization use Agile Methods.

3.1.1. Establish and maintain an organizational policy for planning and performing Agile Methods (GP 2.1). The first step toward institutionalization of Agile Methods is to establish how and when they will be used in the organization. An organization might determine that Agile Methods will be used on all projects or some subset of projects based on size, type of product, technology, or other factors. This policy is a way to clearly communicate the organization's intent regarding Agile Methods. In keeping with the Agile Principle of face-to-face conversions at "all hands meeting" or a visit by a senior manager during a project's kick off could be used to communicate the policy.

3.1.2. Establish and maintain the plan for performing Agile Methods (GP2.2). This practice can help ensure that Agile Methods do not degrade into undisciplined hacking. The expectation is that Agile Methods are planned and that a defined process exists and is followed. The defined process should include a

sequence of steps capturing the minimum essential information needed to describe what a project really does. The plan would also capture the essential aspects of how the other 10 generic practices are to be implemented in the project. In Scrum, some of this planning is likely to be captured in a product backlog and/or sprint backlog, most likely within a tool as opposed to a document.

3.1.3. Provide adequate resources for performing Agile Methods (GP 2.3). Every project wants, needs, and expects competent professionals, adequate funding, and appropriate facilities and tools. Implementing an activity to explicitly manage these wants and needs has proved useful. In Scrum, for example, these needs may be reviewed and addressed at the Sprint Planning Meeting and reconsidered when significant changes occur.

3.1.4. Assign responsibility and authority for performing Agile Methods (GP 2.4). For a project to be successful, clear responsibility and authority need to be defined. Usually this includes a combination of role descriptions and assignments. The definitions of these roles identify a level of responsibility and authority. For example, a Scrum Project would assign an individual or individuals to the roles of Product Owner, ScrumMaster, and Team. Expertise in the Team is likely to include a mix of domain experts, system engineers, software engineers, architects, programmers, analysts, QA experts, testers, UI designers, etc. Scrum assigns the team as a whole the responsibility for delivering working software. The Product Owner is responsible for specifying and prioritizing the work. The ScrumMaster is responsible for assuring the Scrum process is followed. Management is responsible for providing the right expertise to the team.

3.1.5. Train the people performing Agile Methods (GP 2.5). The right training can increase the performance of competent professionals and supports introducing new methods into an organization. Institutionalization of the Agile Method being used requires consistent training. This practice includes determining the individuals to train, defining the exact training to provide, and performing the needed training. Training can be provided using many different approaches, including programmed instruction, formalized on-the-job training, mentoring, and formal and classroom training. It is important that a mechanism be defined to ensure that training has occurred and is beneficial.

3.1.6. Place designated work products under appropriate level of configuration management (GP 2.6). The purpose of a project is to produce deliverable product(s). This product is often a collection of a number of intermediate or supporting work products (code, manuals, software systems, build files, etc.). Each of these work products has a value and often goes through a series of steps that increase their value. The concept of configuration management is intended to protect these valuable work products by defining the level of control, for example, version control or baseline control and perhaps multiple levels of baseline control to use within the project.

3.1.7. Identify and involve the relevant stakeholders as planned (GP 2.7). Involving the customer as a relevant stakeholder is a strength of Agile Methods. This practice further identifies the need to ensure that the expected level of stakeholder involvement occurs. For example, if the project depends on customer feedback with each increment, build, or sprint, and involvement falls short of expectations it is then necessary to communicate to the appropriate level, individual, or group in the organization to allow for corrective action as corrective action may be beyond the scope of the project team. In advanced Scrum implementations, this is often formalized as a MetaScrum [6] where stakeholders serve as a board of directors for the Product Owner.

3.1.8. Monitor and control Agile Methods against the plan and take appropriate corrective action (GP 2.8). This practice involves measuring actual performance against the project's plan and taking corrective action. Direct day-to-day monitoring is a strong feature of the Daily Scrum Meeting, the Release Burndown Chart shows how much work remains at the beginning of each Sprint, and the Sprint Burndown Chart shows total task hours remaining per day. Scrum enhances the effectiveness of the plan by allowing the Product Owner to inspect and adapt to maximize ROI, rather than merely assuring plan accuracy.

3.1.9. Objectively evaluate adherence to the Agile Methods and address noncompliance (GP2.9). This practice is based on having someone not directly responsible for managing or performing project activities evaluate the actual activities of the project. Some organizations implement this practice as both an assurance activity and coaching activity. The coaching concept matches many Agile Methods. The ScrumMaster has primary responsibility for adherence to Scrum practices, tracking progress, removing impediments, resolving personnel problems, and is usually not engaged in implementation of project tasks. The Product Owner has primary responsibility for assuring software meets requirements and is high quality.

3.1.10. Review the activities, status, and results of the Agile Methods with higher-level management and resolve issues (GP2.10). The purpose of this practice is to ensure that higher-level management has appropriate visibility into the project activities.

Different managers have different needs for information. Agile Methods have a high level of interaction, for example, Scrum has a Sprint Planning Meeting, Daily Scrum Meetings, a Sprint Review Meeting, and a Sprint Retrospective Meeting. Management needs are supported by transparency of status data produced by the Scrum Burndown Chart combined with defect data. Management responsibilities are to (1) provide strategic vision, business strategy, and resources, (2) remove impediments surfaced by Scrum teams that the teams cannot remove themselves, (3) ensure growth and career path of staff, and (4) challenge the Scrum teams to move beyond mediocrity.

3.1.11. Establish and maintain the description of Agile Methods (GP 3.1). This practice is a refinement of GP2.2 above. The only real difference is that description of Agile Methods in this practice is expected to be organization-wide and not unique to a project. The result is that variability in how Agile Methods are performed would be reduced across the organization; and therefore more exchange between projects of people, tools, information and products can be supported.

3.1.12. Collect the results from using Agile Methods to support future use and improve the organization's approach to Agile Methods (GP 3.2). This practice supports the goal of learning across projects by collecting the results from individual projects. The Scrum Sprint Retrospective Meeting could be used as the mechanism for this practice.

All of these generic practices have been useful in organizations implementing other processes. We have seen that a number of these generic practices have at least partial support in Scrum or other Agile Methods. We believe that implementing these practices can help establish needed discipline to any Agile Method.

4. Conclusion

Using CMMI and Scrum together results in significantly improved performance while maintaining CMMI compliance. Scrum pilot projects showed significant gains in productivity and quality over traditional methods. These results led to an ROI based decision to more widely introduce Scrum and consider other Agile practices in Systematic. Scrum now reduces every category of work (defects, rework, total work required, and process overhead) by almost 50%.

For Agile companies, we described how CMMI Generic Practices can be used to institutionalize agile practices and we presented Lean Software Development as an operational tool to identify improvement opportunities in a CMMI 5 company.

Companies in defense, aerospace, and other industries that require high maturity of processes, should carefully consider introducing Agile practices and all software companies should consider introducing CMMI practices.

Our recommendation to the Agile community is to use the CMMI generic practices from CMMI Level 3 to amplify the benefits from Agile methods. Our recommendation to the CMMI community is that Agile methods can fit into your CMMI framework and will provide exciting improvements to your organization.

5. References

[1] Krasner and Houston, "Using the Cost of Quality Approach for Software," *CrossTalk,* November 1998.

[2] M. Diaz and J. King, "How CMM Impacts Quality, Productivity, Rework, and the Bottom Line," *CrossTalk,* March 2002.

[3] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Implementation Guide*: Addison-Wesley, 2006.

[4] M. K. Kulpa and K. A. Johnson, *Interpreting the CMMI: A Process Improvement Approach.* Boca Raton: Auerbach Publications, 2003.

[5] M. B. Chrissis, Konrad, and Shrum, *CMMI – guideline for process integration and product improvement*, 2002.

[6] J. Sutherland, "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects," in *AGILE 2005 Conference*, Denver, CO, 2005.