

Scrum and CMMI – Going from Good to Great: are you ready-ready to be done-done?

Carsten Ruseng Jakobsen
Systematic Software Engineering
crj@systematic.dk

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

Abstract

Projects combining agile methods with CMMI¹ combine adaptability with predictability to better serve large customer needs. The introduction of Scrum at Systematic, a CMMI Level 5 company, doubled productivity and cut defects by 40% compared to waterfall projects in 2006 by focusing on early testing and time to fix builds. Systematic institutionalized Scrum across all projects and used data driven tools like story process efficiency to surface Product Backlog impediments. This allowed them to systematically develop a strategy for a second doubling in productivity. Two teams have achieved a sustainable quadrupling of productivity compared to waterfall projects. We discuss here the strategy to bring the entire company to that level.

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 to achieve Toyota level performance – 4 times the productivity and 12 times the quality of waterfall teams.

1. Introduction

While monitoring the Scrum implementation in several projects in Systematic, significant better Scrum was observed in two projects. An analysis of these projects highlighted the importance of a proper balance between activities delivering a sprint and activities preparing or maintaining the product backlog.

Scrum is an iterative (empirical) development model, where it is anticipated that planning is an ongoing activity concurrent to the development activities.

Therefore in general Scrum can be considered to execute two processes at the same time: “Execute and Deliver Sprint” and “Prepare Product Backlog”. As a

team becomes better and better to “Execute and Deliver Sprints” their velocity increases, and imposes a similar need for increased speed of the “Prepare product Backlog” process.

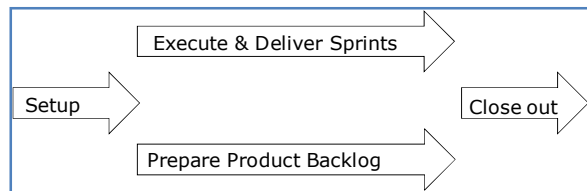


Figure 1 Scrum Process Overview

This paper presents how Systematic experienced that a proper focus on both these processes improved a pretty good scrum to a great scrum and the measures used to drive the change.

2. Going from good to great

2.1. The company

Systematic was established in 1985 and employs more than 500 people worldwide with offices in Denmark, Finland, USA and the UK. It is an independent software and systems company focusing on complex and critical IT solutions within information and communication systems. Often these systems are mission critical with high demands on reliability, safety, accuracy and usability.

Customers are typically professional IT-departments in public institutions and large companies with longstanding experience in acquiring complex software and systems. 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 SCAMPISM method and found to be CMMI level 5 compliant. During 2006 Systematic adopted Scrum and a story based early

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

SM Capability Maturity Model Integration, and SCAMPI are service marks of Carnegie Mellon University

testing approach to software development and achieved significant positive results that were reported in [X]. This work also resulted in experiences regarding how Scrum fit together with other CMMI driven processes, and these experiences were reported in [Y]

2.2. Adoption of Scrum in Systematic

The institutionalization of Scrum in Systematic happened over a period of approximately six months. The first Scrum pilots ended June 2006, and by the end of 2006 most projects had adopted Scrum. During this period Jeff Sutherland also visited Systematic for a management seminar, and to train the first 32 Scrum Masters.

2.3. Improving the Scrum process

From a CMMI perspective Scrum is just one process out of a set of processes used to execute a project. In a CMMI context all processes for development are monitored for effectiveness and efficiency. Therefore measures were also established on the Scrum process.

The choice of measures were highly inspired from Lean. We wanted a measure to help establish focus on a “Stop the line”-mindset to defects, to ensure defects are addressed immediately after they are identified. We also wanted insight into the flow of story implementation, that is how much waiting time is incurred when a story is implemented.

These considerations led to a number of measures where the most important are:

- 1) Fix time after a failed builds – are problems proactively handled?
- 2) Flow in implementation of story – is a story implemented without breaks in calendar time and context shift to implementation of other stories?

These measures were introduced by the start of 2007 in one business unit. In order to support the measure of fix-time, a standard build-server infrastructure was established for all projects. Data from build servers are automatically collected and stored in a shared database. Excel sheets were established to automatically collect data from this database and present the data in statistical control charts.

The measures for flow are supported by the fact that all developers in Systematic are using a standard checklist for implementing stories.

Projects in this business units added the following objectives to their projects:

- a) Bring fix-time after failed build in control with an average less than a working day
- b) Increase flow of implementation of story to greater 60%

None of the projects met these two objectives initially, but were committed to continually improve towards the objectives.

In august 2008 the productivity of two of these projects are compared to other projects in Systematic and shows their productivity to be 140% and 360% better than the average. The two projects participated in piloting of the use of cosmic function points (CFP) as a measure for size. Because the pilot of CFP is started in Q1 2008, the initial use of this measure may include some uncertainty due to application of a new measure, and hence the numbers are considered less confident than other measures.

On the other these numbers were consistent to the gut feeling of the management team for the projects, who believed that these projects showed hyper productive teams.

Based on this indication of high performance, it was decided to interview and analyze the projects, to identify reasons for their success.

An analysis and interview with these projects showed that they had:

- a) Already a pretty good scrum implemented, which was partly driven by focus on fix-time for failed builds, and supported with a good infrastructure for building and testing
- b) Focus on ensuring that work loaded into a sprint is truly ready, which was partly driven by focus on the flow of story implementation
- c) A clear understanding of how the product owner activities were performed by who and when

One of the projects were a fixed-price fixed-scope contract and the other was a contract based on time and material.

The two projects shows a “fix-time after failed build” to be in statistical control with an average fix-time of 1,9 hour and a maximal fix-time of 7 hours. The projects had improved “flow of implementation of story” from 32% in start of 2008 to 59% by the end of 2008.

We believe that the systematic use of these measures in teams already running a pretty good Scrum, combined with a clear understanding of how product owner activities are performed, are among the main reasons for the success of these two projects.

In the following we will present why and how these measures were used and discuss how they improved project performance.

3. Data driven detection of impediments

The two measures “fix-time-after-failed-build” and “flow-of-implementation-of-story” are established using the disciplines from CMMI and using statistical process control techniques. These techniques help to understand the natural variation in the measures, and thereby help to focus on the largest or most special causes of variation. These causes are addressed and resolved with an attitude based on Lean and agile values, where management in a respectful way supports the projects in eliminating them. The focus is on the system as a whole, and how to improve it based on the insight achieved through the measures.

3.1. Time to fix build

The main reason to measure how long time it takes from a build fails on the shared build server until next succeeding build, has to do with speed and quality. If a defect or a problem is not addressed immediately after it is identified, rework will accumulate and it will be difficult to deliver a sprint with high quality and maintain a high velocity.

These two projects focused very early on reducing the calendar time spent on test of the sprint delivery and reduced systematically the time for sprint test to 1-2 calendar days. The test of the sprint delivery can only be completed in this short time, if defects are fixed as soon as they are surfaced. The experience from these projects is, that it is a matter of what mindset you establish to defects.

A Lean mindset suggests that you address a defect immediately after it is identified as opposed to a mindset where defects are stored to be fixed later.

The measure “Fixtime after failed build” is the number of working hours from a defect is identified on the shared buildserver and until that defect is fixed and the shared build is successful. Applying this measure on the projects combined with an objective that the fix-time should be at most one working day, helped to build the Lean mindset of fixing a defect immediately.

In practice the measure is supported by an environment where the build-servers on a project automatically logs the status of a build to a shared database. Feedback to the project team on build status is handled immediately with CruiseControl. Accumulated data for all projects are also shown on a computer screen next to the coffee machine.

Periodically the data are collected by management and analyzed for statistical process control and included in the monthly project review with the project manager.

The measure helped establish focus on what the impediments are, by addressing special causes of variation, that is fix times for broken build that exceeds natural variation. Insight into the natural variation was established through the use of statistical process control techniques.

The figure below shows the fix-time for failed builds on one of the projects with an average fix-time of 1,6 hours and an upper control limit on 7 hours.

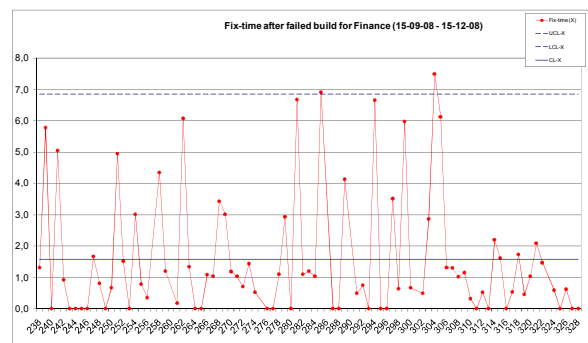


Figure 2 Time to fix a failed build

The graph, shows for illustrative purposes one data point exceeding the control limit with a fix-time of 7,5 hours. For each data point exceeding the upper control line it is asked whether there is a special cause, causing that particular fix of a broken build to take longer time. It is judged whether the cause is special and could be removed, or whether the cause should have been anticipated.

How the cause is categorized is not the most important part here. What really matters, is that these data points are systematically addressed and helps to systematically surface impediments and reflections on how to eliminate these impediments.

Such outliers found surfaced different impediments like:

- 1) The reason for the failed build is related to a special competence. The team member who possesses this competence the best is out of office for two days, and we will let him fix the defect when he is back in office
- 2) The disk on the build server ran full, and caused unanticipated rework
- 3) Misunderstandings to how the test environment was setup
- 4) A Commercial of the shelves (COTS) product failed

Uncovering these reasons, are used actively by the programme above the project. In the first case, it was re-evaluated how many team members to train in this special competence. In the second case the general configuration of build servers shared by all projects, were reevaluated

for disk capacity requirements. In the third example training in the projects infrastructure were re-emphasised.

The general experience is that the outliers are often caused by issues, that if not addressed will cause impediments for future sprints, and a measure like “fix-time for failed build”, will help to ensure that these impediments are identified and resolved.

3.2. Story Process Efficiency

In Lean, a steady flow is desired from customer requests a service and until that request is fulfilled. The flow in typical software development projects will often consist of at least three different types of potential waiting time:

1. Imposed waiting time from the contractual agreement: The amount of requested work in the contract exceeds agreed and anticipated production capacity or team size. This is the typical situation for fixed price/scope projects, and addressed by the Product Owner in Scrum who ensures that work is prioritized according to customer value.
2. Waiting time incurred as part of preparing work to be implemented in a sprint.
3. Waiting time incurred during implementation of a story in a sprint.

The contractual agreements with customers will vary, and may be mandated by legislation that makes it difficult to change. Improving imposed waiting time in contracts, can only be achieved in close collaboration with the customer. However the projects have full control to assure that once work is committed, then it is delivered in one smooth flow. To support that objective, flow of story implementation is measured.

In Systematic the work is decomposed from requirements in the contract, to a set of features. Each feature is decomposed into one or more stories, that will deliver customer value. Stories are allocated to a sprint and then implemented and delivered to the customer.

From a Lean perspective, we want to eliminate the waste associated with context shift or waiting. Therefore we strive to ensure that when work is started on a story, then it is implemented without any interruption or waiting time.

Assume a story is estimated to be 3 workdays of effort. However for various reasons it takes 9 workdays to implement the story. The flow of this story implementation is then defined as 3 days calendar

time of work implemented over 9 calendar days, a flow of 3/9 or 33%. This is measured for all stories.

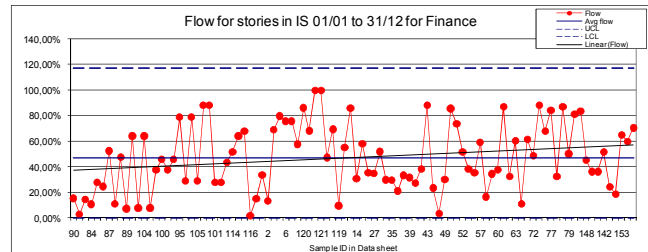


Figure 3 Flow of implementation of story

When we started measuring flow it was around 30%, but from 2007 to 2008 we have increased this to 59% for Q4 2008.

Efficient flow eliminates the waste associated with context shifts and handovers. In addition the team members find it more satisfying, that work initiated in a sprint, is sufficiently clarified to allow for a smooth implementation during the sprint.

4. Process Improvement

Based on these experiences the projects decided to make a recommendation for other projects, that would help them to achieve the same results.

4.1. Are you ready-ready to be done-done?

The two projects had focused on the flow measure through 2008, and they quickly understood that in order to establish a good flow within sprints, the product backlog has to be maintained continuously and concurrent to delivering of sprints. The difficult part for these projects was that the tasks involved in maintaining the product backlog required participation from at least key people involved in delivering the sprint.

The projects had established their own ways of ensuring that the product backlog were maintained, and to ensure that people were allocated to support both “Preparing Product Backlog” and “Execute and Deliver Sprint” activities.

Both projects had experienced how their increased velocity, demanded similar increased focus on preparing work on the product backlog to be ready for upcoming sprints. When the projects were interviewed how the high performance in their projects could be transferred to other projects, they suggested to establish a check-list to consolidate and support the activities to prepare work on the product backlog.

Ready for Implementation Checklist			
Feature: _____			
Product Owner: _____			
Architect: _____			
Lead Developer: _____			
Procedure / Primary role	Activity	Work Product(s)	Completed
	Customer requirements approved and finalized	PMA/RS	<input type="checkbox"/>
Prepare Feature for Commitment / Product Owner	Customer requirements assigned to the feature	PMA/RS, FDD	<input type="checkbox"/>
	Customer requirements sufficiently understood	FDD	<input type="checkbox"/>
	Technical design drafted (focus - feasibility)	FDD, EST	<input type="checkbox"/>
	Risks identified	FDD/EST	<input type="checkbox"/>
	Test design drafted (focus - testability)	FDD, EST	<input type="checkbox"/>
	Unknowns, assumptions, constraints, concerns identified	FDD, EST	<input type="checkbox"/>
	ROI (effort, size) established	EST	<input type="checkbox"/>
	Concept review conducted	ER	<input type="checkbox"/>
	FDD approved	DS	<input type="checkbox"/>
Clarify Feature for Development / Architect	Fit into system considered	FDD	<input type="checkbox"/>
	Feature decomposed into fit-to-sprint features	FDD	<input type="checkbox"/>
	Plan for unknowns/assumptions/constraints/constraints established	FDD, EST	<input type="checkbox"/>
	Estimate (effort & size) updated	EST	<input type="checkbox"/>
	Concept review conducted	ER	<input type="checkbox"/>
Prepare Feature for Implementation / Lead Developer	Unknowns, assumptions, concerns resolved	FDD	<input type="checkbox"/>
	Product requirements developed	PMA/RS, FDD	<input type="checkbox"/>
	Test design drafted (no uncertainties)	FDD	<input type="checkbox"/>
	Technical design drafted (no uncertainties)	FDD	<input type="checkbox"/>
	Decomposition into stories perfected	FDD	<input type="checkbox"/>
	Stories estimated (effort)	EST	<input type="checkbox"/>
	Concept review conducted	ER	<input type="checkbox"/>
	FDD approved	DS	<input type="checkbox"/>

888-64574 CSRS-0007 SB Version: 1.3.5 SDate: 24 Sep 2019 9

Figure 4 Feature ready-ready checklist

Systematic already have good experiences from using a story-completion checklist, to ensure that a story is done-done. The idea was to provide the product owner with a similar feature-ready-for-implementation checklist.

This checklist should ensure that work on the product backlog was properly and timely prepared for implementation in a sprint, and make it visible if work allocated to a sprint was not prepared sufficiently.

The projects observed that the existing process descriptions they had followed already described how to prepare work on the product backlog. What was needed to help other projects was a distillate of the process, formed as a checklist.

A draft checklist was established by November 2008, and is now being piloted. At the time of writing, the checklist has only been piloted for a few features, but the feedback from the projects has been very positive.

So far the main conclusions and results are:

- The use of the checklist gave appropriate focus on timely execution of preparation activities for work in future sprints.
- Due to timely execution of activities, it became easier to conduct estimation workshops with a broad representation of the

team well ahead of Sprint Planning. As a result the Sprint Planning meetings are now much more efficient, because the team knows what the features and stories are about.

- Planning Poker was integrated as part of the estimation workshop, and this has proven to be an efficient way of establishing consensus on scope and estimate of stories.

Even though the projects had achieved high performance without the checklist, they found that the introduction of the feature-ready-for-implementation checklist, consolidated the performance of the team.

Inspired from the common use of the term done-done to express that a story is fully completed, Systematic introduced the term ready-ready, to express that work from the Product Backlog has been sufficiently elaborated to be allocated to a sprint for implementation.

The Product Owner is asked “Are you ready-ready” and the Team is asked “are you done-done” – or in short to all “Are you ready-ready to be done-done”. When your project is ready-ready to be done-done you can deliver value in high velocity. Both ready-ready and done-done are supported with a checklist used by developer and Product Owner respectively.

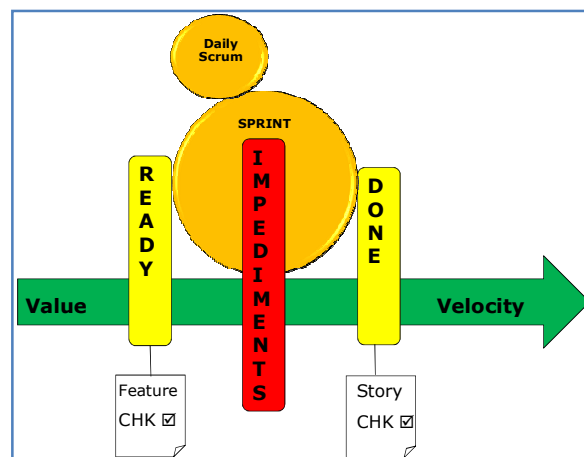


Figure 5 Scrum flow of work

4.2. Product Owner – who and when

The introduction of these checklists, illustrates that all roles in Scrum are important roles. In particular you need your Product Owner to prepare work for future sprints with at least the velocity that the Team is delivering current or past sprints.

An interesting observation from interview with the two teams, were how the Product Owner role was perceived. When any from the projects were asked how they handled clarification of work with the customer, everybody had a clear answer consistent within each project, describing clearly how Product Owner work was executed, by who and when and how decisions were made.

It turns out that Product Owner work was done with the participation of many key people, including the project manager, architect, scrum master and lead developers.

In one sense all these different people are a Product Owner team. This led to some confusion about who is the Product Owner on the project.

In general this role often defaults to the Project Manager, but in practice both projects relied heavily on a domain-expert on the team. On one project this was a user experience engineer, and on the other a software architect. Even though the Project Manager respectfully delegates most of the Product Owner work to other roles in the project, he is still considered the Chief Product Owner in the project.

In general three types of projects are seen in Systematic:

1. Software development projects
2. Products development projects
3. Customer managed project

In software development projects, the team is often heavily involved in product owner work, and the chief product owner the project manager.

In products development projects, a separate product owner team is assigned responsibility for the product, and other teams are assigned to implement specific features to the product. In this setup the product owner is found in the product owner team.

Finally, Systematic have a number of T/M contracts, where the customers manages the team directly. In this case the customer is the product owner.

The experience from these two teams is that one of the first things a project should organize, is the product owner role. Different settings may be applicable to different projects, and therefore care should be taken to ensure that everybody on the project understand how the product owner is organized.

5. Results

Since 2005 Lean has been used as the primary tool to improve the CMMI and Scrum way that Systematic works. Systematic previously reported how Scrum resulted in significant gains [X].

Inspired from Lean and CMMI, the projects were measured on fix-time for failed build and flow of story-implementation.

The measures were analyzed with techniques for statistical process control, which provides an insight into natural variation of the projects performances.

This insight was used to address special causes of variation, and systematically eliminate the reasons behind them.

Addressing outliers systematically shows directly in the measures with an average of fix-time of failed builds in 1,9 hours and an increased flow of story implementation of 59%.

The indirect consequence, is elimination of wasting time related to context shifting, and there is a strong indication that the productivity of these projects are 140% to 360% better than the average of other projects in Systematic.

A prerequisite that contributed significantly to these results, is that these projects had established a clear understanding of how the product owner work was organized within the project.

6. Conclusion

Using CMMI and Scrum together results in significantly improved performance while maintaining CMMI compliance. Scrum reduces every category of work (defects, rework, total work required, and process overhead) by almost 50%. We now have a clearly defined strategy to reduce all categories of work by 75% and have achieved that goal with a small number of teams. That success needs to be institutionalized in the company.

A lean culture with a disciplined approach, skilled people, and good leadership can systematically improve Agile velocity and quality using proven CMMI 5 level techniques of data driven assessment and organizational self-tuning. Systems can be measured and data magnifies learning. Careful attention must be paid to the human dimension because poor use of data will destroy productivity.

We have not completed our journey towards improved performance. The next phase will focus carefully on cross-functional team interactions and dynamics. Some Scrum teams have achieved 8 times waterfall performance using Agile organizational patterns implemented at the world's best companies. The authors are currently participating in a patterns research project involving many Scrum companies and this results of this work could take Systematic from very good to a great CMMI Level 5 Scrum.

7. References

- [2] M. B. Chrissis, Konrad, and Shrum, *CMMI Second Edition: Guidelines for Process Integration and Product Improvement*.: Addison-Wesley, 2006.
- [X] J. Sutherland, C.R. Jakobsen and K.A. Johnson, "CMMI and Scrum - a magic potion for code warriors" in proceedings for Agile 2007
- [Y] C.R. Jakobsen and K.A. Johnson, "Mature Agile - with a twist of CMMI " in proceedings for Agile 2008
- [3] D. R. Goldenson and D. L. Gibson, "Demonstrating the impacts and benefits of CMMI," *CrossTalk*, October 2003.
- [4] D. D. o. Science, "Maturity of customer and suppliers in the public sector," 2006.
- [5] J. Sutherland, "Agile Development: Lessons Learned from the First Scrum," *Cutter Agile Project Management Advisory Service: Executive Update*, vol. 5, pp. 1-4, 2004.
- [6] K. Schwaber, "Scrum Development Process," in *OOPSLA Business Object Design and Implementation Workshop*, J. Sutherland, D. Patel, C. Casanave, J. Miller, and G. Hollowell, Eds. London: Springer, 1997.
- [7] H. Ziv and D. Richardson, "The Uncertainty Principle in Software Engineering," in *submitted to Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*, 1997.
- [8] W. S. Humphrey, *A Discipline for Software Engineering*. Addison-Wesley, 1995.
- [9] P. Wegner, "Why Interaction Is More Powerful Than Algorithms," *Communications of the ACM*, vol. 40, pp. 80-91, May 1997.
- [10] P. DeGrace and L. H. Stahl, *Wicked problems, righteous solutions : a catalogue of modern software engineering paradigms*. Englewood Cliffs, N.J.: Yourdon Press, 1990.
- [11] M. Fowler and J. Highsmith, "The Agile Manifesto," *Dr. Dobbs*, July 13 2001.
- [12] Krasner and Houston, "Using the Cost of Quality Approach for Software," *CrossTalk*, November 1998.
- [13] M. Diaz and J. King, "How CMM Impacts Quality, Productivity, Rework, and the Bottom Line," *CrossTalk*, March 2002.
- [14] M. B. Chrissis, Konrad, and Shrum, *CMMI – guideline for process integration and product improvement*, 2002.
- [15] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Implementation Guide*: Addison-Wesley, 2006.
- [16] M. K. Kulpa and K. A. Johnson, *Interpreting the CMMI: A Process Improvement Approach*. Boca Raton: Auerbach Publications, 2003.
- [17] J. Sutherland, "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects," in *AGILE 2005 Conference*, Denver, CO, 2005.
- [19] J. O. Coplien, "Personal issues caused over 50% of productivity losses in the ATT Bell Labs Pasteur Project analysis of over 200 case studies," Personal communication ed, J. Sutherland, Ed. Lynby, Denmark, 2006.