Scrum and CMMI – Going from Good to Great

Are you ready-ready to be done-done?

Carsten Ruseng Jakobsen
Systematic Software Engineering A/S
Aarhus, Denmark
crj@systematic.dk

Jeff Sutherland, Ph.D.
Scrum Training Institute
Boston, MA, USA
jeff@scruminc.com

***Abstract*—Projects combining agile methods with CMMI 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. Our experiences shows 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.**

Keywords: CMMI; Flow; Lean; measures; product owner; Scrum

I. 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 on-going activity concurrent to the development activities. Therefore in general Scrum can be considered to execute two processes at the same time: "Execute and Deliver Sprints" and "Prepare Product Backlog". As a team becomes better and better to "Execute and Deliver Sprints" process their velocity increases, and imposes a similar need for increased speed of the "Prepare product Backlog" process.

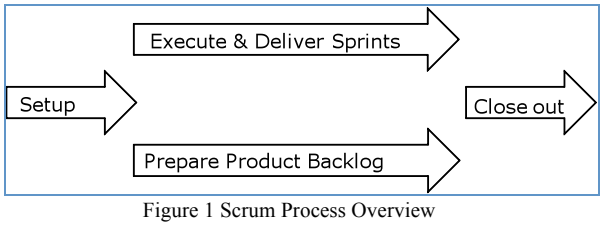


Figure 1 Scrum Process Overview

This paper shows how Systematic used both these processes to turn a good Scrum into a great scrum. We show specific techniques and measures used to drive this change.

II. GOING FROM GOOD TO GREAT

A. 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 SCAMPI method and found to be CMMI level 5 compliant.

During 2006 Systematic adopted Scrum and a story based early testing approach to software development and achieved significant positive results that were reported previously [1]. This work also showed how Scrum fit together with other CMMI driven processes, and these experiences were reported elsewhere [2].

B. Adoption of Scrum in Systematic

Scrum was institutionalized at Systematic 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.

C. Improving the Scrum process

From a CMMI perspective Scrum is 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 inspired from Lean [3] and from the objective to establish a stable flow of work.

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 (process efficiency of a story)

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 a standard checklist for implementing stories used by all developers at Systematic.

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

a) Reduce average fix-time after failed build to less than a working day

b) Increase flow of implementation of story to greater than 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 were 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 [4]. Because the pilot of CFP is started in Q1 2008, 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 hand these numbers were consistent with the management team subjective observations 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 good Scrum implementation, 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 was a fixed-price fixed-scope contract and the other was a time and material contract.

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 and had improved "flow of implementation of story" from 32% in start of 2008 to 59% by the end of 2008.

III. DATA DRIVEN DETECTION OF IMPEDIMENTS

The two projects used these measures to systematically identify impediments to meet the overall objective to be able to deliver high quality working code to the customer every month. Both measures are established using the disciplines from CMMI and analyzed using statistical process control techniques. These techniques help us to understand the natural variation in the measures, and thereby helps to focus on the largest or most special causes of variation [5]. By addressing these causes systematically the projects achieved the capability to perform complete test and release within 2 calendar days of each one month sprint.

The causes were addressed and resolved with an attitude based on Lean and agile values, where management in a respectful way supported the projects by eliminating impediments. The focus was on the system as a whole, and how to improve it based on the insight achieved through the measures. How this was done is illustrated with "Time to fix a failed build" in the next section.

A. Time to fix a failed build

The main reason to measure how long it takes from a build failure on the shared build server until the 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. Systematic introduced a story implementation checklist in 2006 in order to ensure an early testing mindset, and these experiences were reported in [1]. This checklist facilitates an individual focus, whereas the measure on time to fix a broken build provides the project team with a product/project level measure and focus.

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 remove 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 "Fix time after failed build" is the number of working hours from the time a defect is identified on the shared build server 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 log 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 builds that exceeds natural variation. Insight into the natural variation was established through the use of statistical process control techniques, as described in [5].

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.

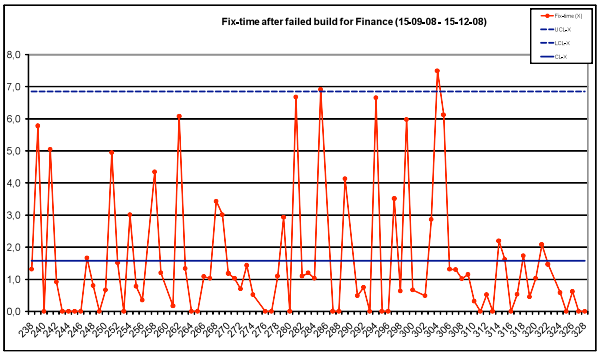


Figure 2 Time to fix a failed build

The graph, shows 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 help to surface impediments and reflections on how to eliminate these impediments.

Some outliers 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 off the shelf (COTS) product failed

These impediments were addressed by the project or the program management 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 project's infrastructure was re-emphazised.

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.

IV. PROCESS IMPROVEMENT

As the two projects improved and became better at implementing sprints they surfaced the problem that work prioritized for upcoming sprints was not sufficiently prepared in a timely manner. 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.

It was evident that when unprepared work was allocated to sprints, it incurred unanticipated waiting time and context shifts in the sprint. 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.

Therefore the team started to reject scope that was not properly prepared and analyzed how they could improve their process to ensure a good balance between the time spent on preparing future work and the time spent on implementing current sprint. The project tried to achieve a continuous flow of implementation of features and stories. When the projects were looked at in August 2008, it appeared that the projects had achieved a fairly good flow.

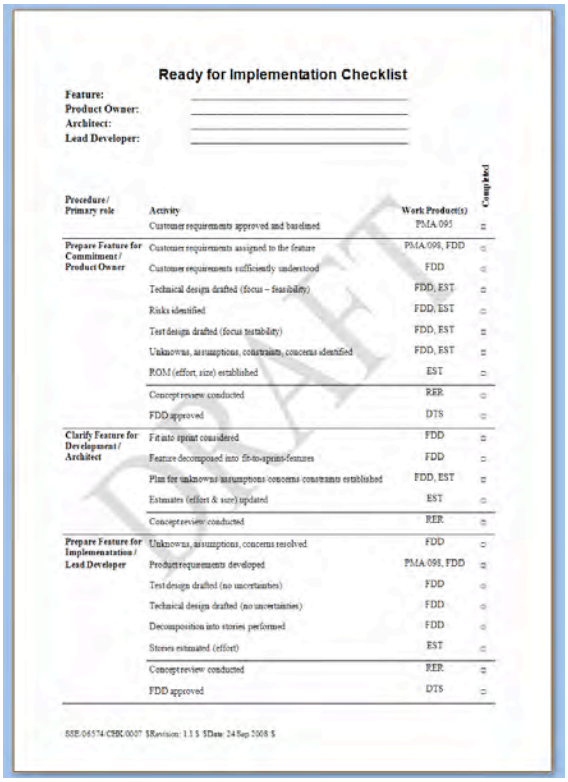
A. Are you ready-ready to be done-done?

The two projects had focused on the flow measure through 2008, and they understood that in order to establish a good flow within sprints, the product backlog must 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 key people involved in delivering the sprint.

The projects established their own way of ensuring that the product backlog was maintained, and ensured 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 asked how the high performance in their projects could be transferred to other projects, they suggested the following 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 baselined	PMA.095	☐
Prepare Feature for Commitment / Product Owner	Customer requirements assigned to the feature	PMA.095, FDD	☐
	Customer requirements sufficiently understood	FDD	☐
	Technical design drafted (focus – feasibility)	FDD, EST	☐
	Risks identified	FDD, EST	☐
	Test design drafted (focus testability)	FDD, EST	☐
	Unknowns, assumptions, constraints, concerns identified	FDD, EST	☐
	ROM (effort, size) established	EST	☐
	Concept review conducted	RER	☐
	FDD approved	DTS	☐
Clarify Feature for Development / Architect	Fit into sprint considered	FDD	☐
	Feature decomposed into fit-to-sprint-features	FDD	☐
	Plan for unknowns assumptions concerns constraints established	FDD, EST	☐
	Estimates (effort & size) updated	EST	☐
	Concept review conducted	RER	☐
Prepare Feature for Implementation / Lead Developer	Unknowns, assumptions, concerns resolved	FDD	☐
	Product requirements developed	PMA.095, FDD	☐
	Test design drafted (no uncertainties)	FDD	☐
	Technical design drafted (no uncertainties)	FDD	☐
	Decomposition into stories performed	FDD	☐
	Stories estimated (effort)	EST	☐
	Concept review conducted	RER	☐
	FDD approved	DTS	☐

$Id: 06574.CHK.0007 $Revision: 1.3 $ $Date: 24 Sep 2008 $

Figure 3 Feature ready-ready check list

Systematic already had 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 been piloted for a small number of 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 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 Product Owner and developer respectively.

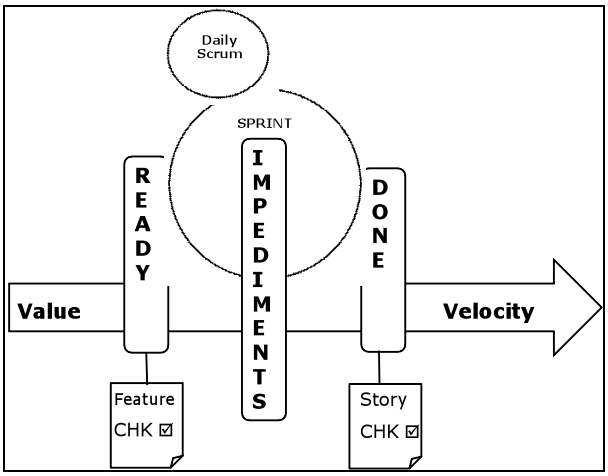


Figure 4 Scrum flow of work

B. *Story Process Efficiency*

The effect of ensuring that the product backlog is continuously maintained, and thereby work is properly prepared before it is allocated to sprints is evident when the measure of story implementation is analyzed during this period.

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% and was measured for all stories.

When we started measuring flow it was around 30%, from 2007 to 2008 it increased 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.

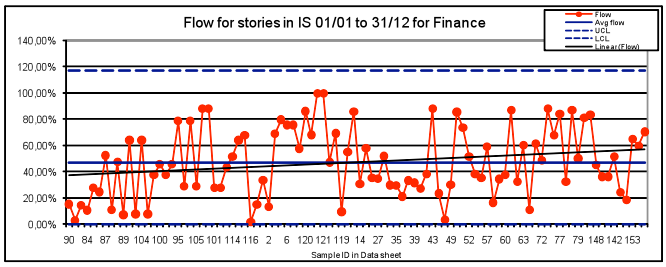


Figure 5 Flow of implementation of Story

V. RESULTS

Since 2005 Lean has been used as the primary tool to improve the CMMI and Scrum way that Systematic works. 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 project performance. 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 the two projects are significant better than the average of other projects in Systematic.

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

VI. 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 the results of this work could take Systematic from very good to a great CMMI Level 5 Scrum.

VII. REFERENCES

[1] J. Sutherland, C. Jacobson, and K. Johnson, "Scrum and CMMI Level 5: A Magic Potion for Code Warriors!," in *Agile 2007*, Washington, D.C., 2007.

[2] C. R. jakobsen and K. A. Johnson, "Mature Agile - with a twist of CMMI," in *Agile 2008*, Toronto, 2008.

[3] T. Ohno, *Toyota Production System: Beyond Large Scale Production*: Productivity Press, 1988.

[4] COSMIC, "The COSMIC Functional Size Measurement Method Version 3.0.1," Common Software Measurement International Consortium2009.

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