Mature Scrum at Systematic

Carsten Ruseng Jakobsen
Systematic Software Engineering
crj@systematic.dk

Jeff Sutherland
Scrum Training Institute
jeff@scruminc.com

Abstract

Systematic is a CMMI level 5 company, where the adoption of Lean led to a unique way of working based on synergies between CMMI, Lean, Scrum and other agile practices.

Lean provides the principles and values needed for high performance teams and organizations, as demonstrated by Toyota for more than 50 years. CMMI provides the process descriptions and support for what disciplines to consider in order to be successful. Agile approaches like Scrum provides best practices and methods for working according to the values of Lean and adopting change as it occurs. Lean, CMMI and Scrum are strong by themselves, but they can be combined to amplify each other – and that is what Systematic has done.

The experiences from combining Lean, CMMI and Scrum have led Systematic to identify examples of explicit guidance from CMMI that help to execute Scrum activities even better. These activities can be implemented based on Lean values and principles and by doing so Scrum can be augmented and matured to ensure that even larger and more complex projects in the future can and will benefit from Scrum.

1. Introduction

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

This paper presents how Systematic combined Lean, CMMI and agile during the past 5 years. What were the different steps and what are the experiences and lessons learned. Agile purists and small agile projects may find our experiences and suggestions non-agile or counter-productive and very large high maturity organizations may find our experiences too undisciplined.

The experiences are fit for Systematic, but may not be fit for your project or organization. Some companies may be in a similar context to Systematic, and find our experiences directly applicable to their challenges. Others may be in a different context and may find that the answers represented by these experiences are not applicable, and instead find inspiration in the questions asked leading to Systematic's experiences.

2. Why CMMI, Lean and Scrum

Experience. Lean established a new mindset leading to identification of significant improvements
Experience. Scrum implements many of the tools and principles in Lean Software Development

Experience. Scrum implements a subset of project management when interpreting CMMI in a Lean context

Systematic was appraised 11 November 2005 and 15 May 2009 using the SCAMPI[SM] method and found to be CMMI level 5 compliant. Implementing CMMI was the main driver for improvements until November 2005 and therefore as part of yearly strategic planning in the summer 2005, Systematic discussed how to ensure a continued flow of significant improvements after achieving CMMI level 5. The result of the discussion was that future improvements are expected to be implemented applying the CMMI capability established, but using primarily a Lean perspective on new improvements, may help identify new substantial improvements that may be overlooked in a focus on processes alone.

Lean has a proven track record for more than 50 years at Toyota and has been applied successfully to many other industries, and Mary Poppendieck has with her book "Lean Software Development" provided a solid basis for understanding how Lean is applied to software development, described through 7 principles and 22 thinking tools, [1].

In order to understand how to apply Lean Software Development, Systematic analyzed the causality of the principles and thinking tools, and the result is shown in Table 1. In the table thinking tools are labeled with a (T) and principles are labeled with (P) followed by the number and name of the principle or thinking tool as defined in [1]. Principles and tools to the right depend on principles and tools to the left. As part of the analysis it was also observed that the principles and tools addressed different perspectives. Some focused on the engineering practices, another part on the management perspective, and the rest focused on people. These categories are shown as rows below. Finally, the column headers are inspired from Womack and Jones book on "Lean Thinking" where Lean is defined in 5 steps: Value, Value Stream, Flow, Pull and Perfection, see [5].

	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 develop.	*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 Analysis of causal dependencies in Lean Software Development

The analysis shows that in the engineering dimension, the most dependent parts are the thinking tools "T17 Perceived Integrity" and "T18 Conceptual Integrity". "T18 Conceptual Integrity" is concerned with what is usually understood as good technical quality, whereas "T17 Perceived Integrity" is concerned that the customer receives what he *perceives* to have ordered as opposed to what was *actually* ordered initially.

This thinking tool addresses that the final product fulfills the *actual* needs of the customer and not only the *original* requirement specification. In most cases, the complexity and changes of organizations and technologies introduces the need for an adaptive approach where regular feedback (T3) is needed

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

to identify and absorb the changes necessary to achieve perceived integrity. To allow the customer to provide valuable feedback, the customer needs to see parts of the system as it evolves. In other words, it is necessary that the project is delivered in short iterations (T4) delivering parts of the project, in a way where the customer can relate and comment to it. To deliver high quality code in short iterations, the discipline of test (T20) must be well established, otherwise speed will soon drop as defects are identified and quality is built in afterwards.

The analysis shows similar dependencies for the thinking tools and principles related to Management and People. The model is not used to "implement" Lean, but as a mean to understand the primary tool for future improvements: Lean Software Development.

Using the analysis of Lean Software Development from above and analysis of historical data from all projects in the organization, it was decided to pilot a changed software development process with focus on early test and short iterations.

The improvement resulted in the introduction of a new development method in Systematic named Story Based Development, where work is ultimately decomposed into units called a story. The completion of a story is handled with a checklist, that by design ensures that test of a story is considered before the story is coded. Another key-aspect is that the work on a story is *inspected* a number of times to ensure that activities are carried out in the right order and are properly completed. The improvement also led to the introduction of Scrum, and it was realized that Scrum addresses many of the thinking tools and principles from Lean Software Development.

The results from adoption of Scrum and Story Based Development were reported at Agile 2007, SEPG2007 and HICCS2008[2] and how Scrum fit together with other CMMI driven processes were reported at Agile 2008 [3]. The core of these experiences stem from synergies between Lean, CMMI and Scrum.

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 CMMI is interpreted from the basis of Lean values and principles, a large part of project management activities can be implemented using Scrum. When mixing CMMI and Scrum, 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 considered with proper discipline.

Individually CMMI and Scrum has proven benefits, but also pitfalls. A company can comply with CMMI, but fail to reach optimal performance due to inadequate implementation of processes. Scrum and other agile methods can guide such companies towards more efficient implementation of CMMI.

An agile company may implement Scrum correctly, but fail to obtain the full benefits due to lack of consistent and sufficient execution of engineering or management processes within and across projects in the organization. CMMI can help agile companies to institutionalize agile methods more consistently and understand what processes and disciplines to address.

Systematic has gained valuable experiences in combining Lean, Scrum and CMMI that are relevant both for projects in a CMMI and Scrum context. In particular Systematic can point out where disciplines driven from CMMI, has created a positive synergetic effect to the implementation of Scrum. These activities may not be needed by small or simple projects, but these disciplines are in most cases founded in Lean values, in line with the intensions of Scrum and will be helpful for larger and more complex projects working with Scrum.

3. CMMI, Lean and Scrum combined

Experience. CMMI ensures that agile methods are institutionalized, including consistent implementation throughout the organization, role based training and continuous improvement

We believe the value from Scrum can only be obtained through disciplined use. CMMI has an organizational-level concept of institutionalization that can help establish this needed discipline based on a set of Generic Practices (GP) that are applied to all processes.

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." How these GP's are applied to Scrum can be found in [2]. The GP's will cover areas like, what is the policy for applying Scrum, how Scrum is performed is documented, relevant responsibility is assigned to Scrum Master and Product Owner, and people fulfilling these roles are trained for the roles.

Institutionalization will ensure that the organization as a whole is committed to Scrum, and that Scrum is performed completely and consistently on different projects, allowing for faster learning and improving of how the organization performs Scrum. Institutionalization will ensure that the Scrum evangelist in each project overall speaks the same gospel.

3.1. Overview

Experience. Scrum also needs some initial planning or setup activities
Experience. Project initiation is fast, and risk are not hidden by speculative planning

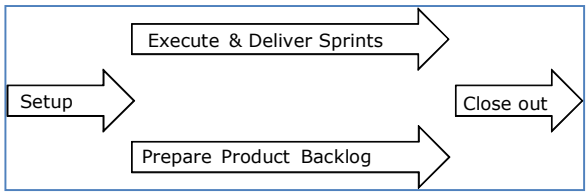

Figure 1 Overall lifecycle model

Using Scrum does not eliminate the need to setup or plan the project initially. Instead initial planning is done with a focus on establishing the project infrastructure, product vision and an initial product backlog based on a mindset to avoid planning based on speculation. This approach reduces the amount of initial planning significantly, but also makes explicit that once the project is setup, it will do two concurrent activities throughout the lifetime of the project. The part of the Product Backlog that is sufficiently prepared will be the foundation for planning and execution of sprints. The other part of the Product Backlog will iteratively be refined and make more of the product backlog sufficiently prepared for future sprints. Once the whole product backlog is produced, we often find that customers want a final acceptance test of the whole product, even though they have received working code iteratively.

3.2. Setup - Sprint Zero

Experience. Project Planning in CMMI is a disciplined and comprehensive Sprint Zero.

In order to run a good Scrum, it is vital to have good product backlog. CMMI provides very valuable insight into what activities to consider in order to establish the initial Product Backlog during

the Project Planning / Sprint Zero. The overall project is planned at a slightly higher level of granularity, and uncertainties are managed with risk management. The primary focus is to establish an initial Product Backlog. The initial Product Backlog is analyzed, and sprints are not initiated until uncertainty and risk are at a level where the projects overall objectives can be met and the draft version of plans and solutions are approved by senior management.

When Systematic adopted Scrum, the project planning process was updated to produce an initial product backlog. Expected CMMI practices include decomposition of work into manageable pieces that are estimated and analyzed for dependencies, planning of stakeholder involvement, and total project risk assessment. In addition to the product backlog a set of overall project plans are established.

Agile teams talk about a sprint zero to establish the foundation for the team to do efficient sprinting. Project Planning in CMMI can be perceived as a sprint zero that also produce a coherent set of plans, that will help improve execution of the product backlog. Such plans cover topics like, stakeholder management, milestone and delivery schedules, cost estimates, and quality.

The initial version of these plans are typically established within few weeks after project initiation and will focus on the most certain elements of the projects plan, leaving more uncertain parts to be elaborated as the project proceeds.

The project planning process, has been changed to primarily deliver a prioritized Product Backlog where the top elements are described in detail, but the lower elements are described in decreasing detail. It is an explicit decision to postpone much planning, and thereby accept that planning and execution will continue concurrently as the project evolves. Many of the CMMI project planning activities are in line with the intensions of Scrum, the main difference is that in CMMI these activities are elaborated and documented.

Experience: Planning and risk management activities reduces risk of product backlog

The first draft product backlog is assessed for risk by the team. Asking the team to estimate the products backlog using 3-point estimates for effort will reveal the most uncertain parts of the work in the Product Backlog. Conducting Risk Identification meetings will identify other important risks, and allow proactive mitigation to be initiated.

Experience. Two levels of planning and tracking: Project as a whole and each sprint.

The project is planned and tracked with plans for the overall project and active sprint(s).

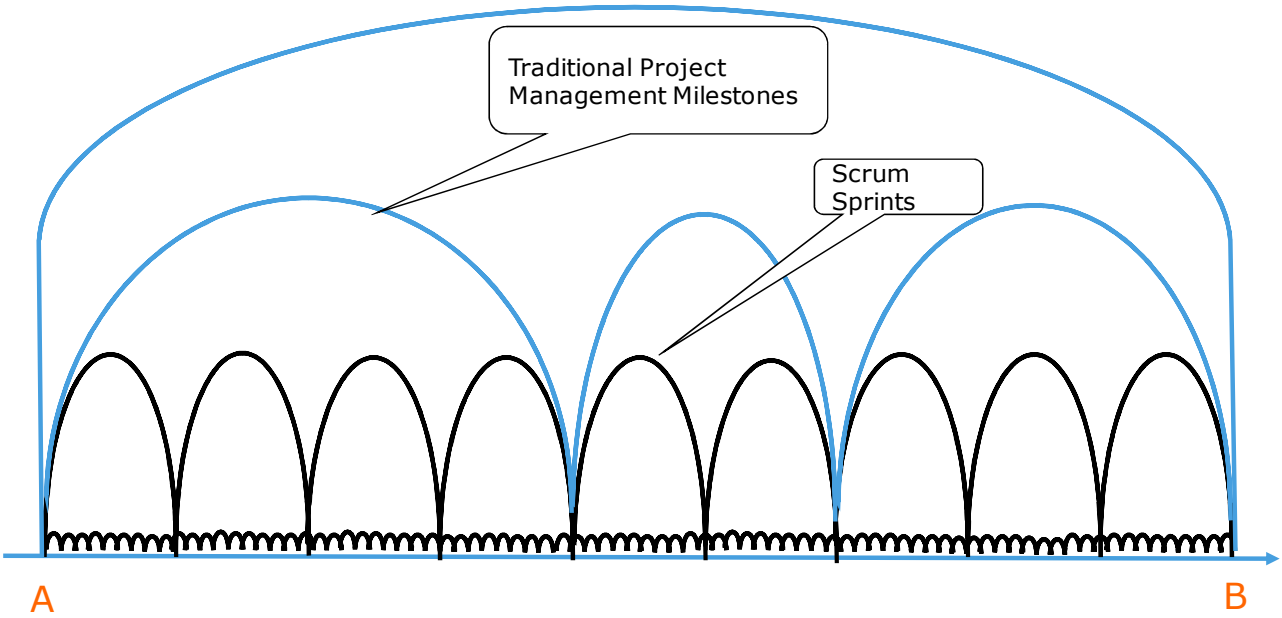

Figure 2 Two levels of planning and tracking

The plans for the overall project established with planning activities during setup provides a better context for defining sprint visions and goals. They allow the team to focus on the sprint, because they can rely on the project manager or Product Owner to track the progress the overall progress.

In general CMMI is used for planning and tracking of the overall project, and Scrum is used for planning and tracking at the sprint level. The CMMI and Scrum activities integrate primarily through the Product Backlog. Sprint Review becomes a heartbeat in tracking of the project and provides high quality feedback into tracking of the overall project plan.

3.3. Execute and Deliver Sprints

Experience: Risk management can proactively prevent impediments

Scrum has a strong focus on removing impediments as soon as they are identified, however CMMI risk management activities focus on proactively identify some of these impediments as risks, and through mitigation eliminate them before they occur as an impediment in the future. The distinction between risk and impediment is that risk describe a problem that may occur, whereas an impediment is problem that has occurred and is impacting planned progress.

Risk management activities are easily integrated with Scrum activities. During project planning the project plans and solution architecture are inputs for initial identification of risks. During project execution, bi-weekly meetings of 10-15 minutes are arranged, where the status of known risks is reported and new risks are identified. It is our experience that these risk management meetings should be kept outside the daily scrum meeting. New risks may be reported on the daily scrum, in which case the risk manager will just take a note.

Experience: Use checklist to ensure quality of stories

One of the important aspects of Systematic Story Based Development method was to ensure focus on early test. The quality of stories are generally ensured by focus on early specification of test and by getting somebody else to look at the work done.

Developing a story includes many different activities, that need to be structured to some degree. The Story Completion Checklist accomplishes these goals, by structuring activities and defining when work must be inspected by an inspector.

Experience: Automated test is a must in order to do one month sprints

When the sprint duration is one month, all tests must be automated to the extent possible. It is an integrated part of developing a story, to also implement the automated test verifying the story. Automated test are used on the teams shared repository and run every time a developer commits code to the shared build server.

Experience: Continuous integration must be supported with discipline for check in

In order to avoid chaos when developers continuously integrate with each other, we have defined the following criteria for check in of code to the integration repository: The test must run smoothly in the developers sandbox and the code must comply with the code standard checked with FxCop (a static code analysis tool).

Experience: Automated test and integration must be supported by a standard production line

Every project needs this infrastructure and therefore we have established standard production line setups, allowing projects to get started faster. Scrum promotes short iterations, e.g. one sprint per month, and in our experience this drives the need for a strong discipline in configuration management, test, integration and release. You can only deliver high quality sprint deliveries every month, if you address defects immediately they are identified as opposed to store and fix them later. In a 1 month sprint you should spent at most 3 calendar days to test, integrate and release. This is only possible if you build quality in, and test is expected to NOT find any defects.

To establish this capability CMMI helps with:

- Establish standards for production line, including standard setups for build- and test servers
- Establish discipline on criteria for integration
- Measures to objectively evaluate performance
- Disciplines to maintain integrity of configuration management system, builds, and releases.

Experience: CMMI facilitates data driven identification and elimination of impediments

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 are 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.

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

1) Fix time for 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?

The main reason to measure how long it takes from a build fails 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. The two measures can be considered a sub-process for fixing failed builds and coding a story respectively. The measures are regularly analyzed using statistical

control charts, and data-points outside the control-limits are evaluated. Analysis of such data often identifies project impediments.

Experience: Focus on "fix-time after failed build" drives good discipline in project

Using standard infrastructure setups allows for efficient data collection and analysis. In particular Systematic has been inspired from Lean thoughts on flow and jidoka (stopping the production line when an error is detected).

We want our projects to be able to deliver on a daily basis, and hence that unresolved failed builds are fixed within a working day. We use CruiseControl (a build management tool) to signal all developers when a build fails. We also monitor the objective by analyzing data from the build servers using control charts like the one shown below.

Many projects have achieved this one work day objective, merely by the focus on the measure. The objective is easy to understand, and presenting the information in CruiseControl and control charts has established a good habit of fixing broken builds immediately when they fail.

3.4. Prepare Product Backlog

Experience: *Explicit quality plans helps the team to build the right quality in*

One of the results of planning is a quality assurance schedule (QAS), where it is outlined what quality activities will be used to ensure the quality objectives are achieved and when. The QAS may specify

- What stories are subject to inspection
- What code is subject to review
- What documents are subject to what types of review
- What unit test and automatic test is produced
- What is included in the acceptance test

A typical QAS document is only a few pages long, but the above descriptions help a scrum team to elaborate and understand the definition of done.

Experience The Product Backlog must be groomed with at least the same speed as sprints are delivered

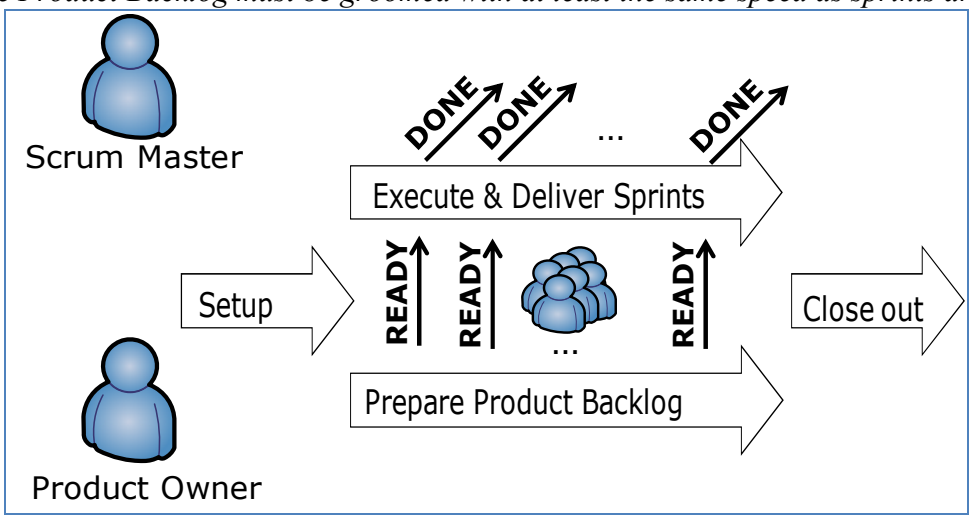

Figure 3 Balancing preparing and delivering activities

The general experiences from working with CMMI and Scrum is that it requires strong discipline to deliver high quality code after each sprint, while at the same time ensure sufficient planning activities to groom the Product Backlog.

The initial plan ensures a sufficient confidence for the project as a whole, and postpones detailed planning to be executed as the project executes. The activities postponed often include clarification or agreement to interpretation of requirements. The challenge is that these activities often requires participation from members of the sprint team, who are committed to deliver a sprint delivery.

The amount of time spent preparing features on the product backlog will vary over time, and it is difficult to make a fixed plan for how much time to spend. Instead the project track how much work is ready for future sprints concurrent to tracking the burn down chart for current sprint. This provides a visual feedback showing if planning activities are starved.

The reason we talk about whether a feature or story is ready for implementation, is that we inspired from Lean want to have a smooth flow of implementing stories or features from the backlog. We found that one of the main impediments for a smooth flow when executing sprints, was that the work allocated to the sprint was not sufficiently prepared or clarified. Therefore we will only allocate features or stories to a sprint when they are sufficiently clarified. Based on our CMMI implementation it is well defined when a feature is READY, and the work to bring from an initial state to the state READY is supported by a "Feature Ready for Implementation Checklist".

The Product Owner is responsible for ensuring that the Product Backlog is constantly groomed to keep at least an amount of 110% sprint velocity READY on the top of the Product Backlog. This assures that new sprints can be planned, and only will allocate work that is READY. By doing so we achieved significant better flow of story implementation in our projects.

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 defined as 3 days calendar time of work implemented over 9 calendar days, a flow of 3/9 or 33%.

When we started measuring flow of story implementation it was around 30%, from 2007 to 2008 it increased to approximately 60% 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. Recommendations

Systematic has experienced synergies between Lean, CMMI and Scrum that scales Scrum to be successful in larger or distributed projects and organizations. The experiences from Systematic can be turned into the a number of recommendations that may help you to a great Scrum. The recommendations are presented in the context of a stepwise adoption model for Scrum, that we believe many projects go through. Projects inexperienced with Scrum are often seen to only adopt parts of Scrum. This may have little or even negative effect. As Scrum is fully implemented and the projects build experience, they learn how to bring the full benefits out of Scrum, and will typically improve significantly. Instead of discussing the quantity of these gains, we will focus on what activities or disciplines we typically see a project master, as it brings the full benefit out of Scrum:

Scrum Type	Driven by
ScrumButt	Lack of discipline to implement all of Scrum
Pretty Good	Basic Scrum roles and meetings established, teams co-located and values mutual

Scrum	respect and shared commitment to sprint goal.
Good Scrum	Discipline in execution of Sprint and using DONE concept
Great Scrum	Discipline in grooming the product backlog using READY concept

4.1. ScrumButt

In order to avoid a partial Scrum implementation, Systematic have one interpretation of how to execute Scrum shared by all projects in Systematic and has gradually standardized on concepts. For example all projects are recommended to use a Physical Scrum Board for the sprint backlog and not a particular tool.

1. Use CMMI Generic Practices to ensure a full and consistent implementation of Scrum in all projects in your organization including training of Scrum Master and Product Owner to their roles. How this is done is elaborated in [2] and [3].
2. Use the Nokia test to identify gabs in your Scrum implementation, for specific descriptions and guidelines please visit http://jeffsutherland.com/scrum/nokiatest.pdf

4.2. Pretty Good Scrum

3. Co-locate project and teams to the extent possible
4. Establish your own sprint zero, and include activities in the items below
5. Use Risk Management to proactively address risks before they are identified as impediments
6. Transform requirements into a solution description described in a set of high level features on the product backlog. Prepare the product backlog by decomposing the highest prioritized features to stories allowing for efficient sprint planning.
7. Use 3-point effort estimates on elements of the product backlog during initial planning.
8. Analyze dependencies, stakeholders, risk on elements of product backlog.
9. Establish milestone and delivery plan (roadmap) and their initial relationship to product backlog.

4.3. Good Scrum

10. Decide and communicate on quality objectives including, what code and documentation to formally review to elaborate your projects definition of DONE.
11. Use Story Completion Checklist and inspection to maintain high quality of stories produced and ensure that DONE criteria is met
12. Establish standards for project “production line” including development, build servers, and test servers.
13. Automate test and nightly build, and measure performance.
14. Establish criteria for committing of code to integration.
15. Maintain integrity of configuration management
16. Use a checklist for Work Product Evaluation to ensure and validate that sprint deliveries are consistent.

4.4. Great Scrum

17. Define READY criteria for work on the product backlog.

18. Ensure visibility of the amount of work on the product backlog that is ready for Sprint Planning. The amount of READY work should at all times be kept to at least Sprint velocity of all teams
19. Product Owners work is not included in sprint plans but sprint is planned to support that team members can provide sufficient support to Product Owner activities.

5. Conclusion

Systematic has worked with Lean, CMMI and Scrum for several years, and found that they can be mixed in a way where strong synergies are achieved. The focus of this article has been to share how Lean and CMMI can bring synergies and support your adoption of Scrum and agile. The experiences were presented as recommendations for agile projects on activities or disciplines to adopt particularly in larger or distributed projects. These recommendations are based on experiences gained from projects at Systematic running Scrum in the context of Lean and CMMI. We believe that these experiences can be adopted by many agile projects, and are independent to whether your project or organization is working according to CMMI or Lean.

The experience from projects at Systematic is that to achieve the full benefit of Scrum you should use the concept of READY and DONE. Systematic identified the need for and value of the READY criteria, in our search for stable flow in Sprints. Adopting the READY concept makes it clear that the Product Backlog is divided into the following items:

1. In Sprint (work in progress – facilitated by Scrum Master controlled with DONE criteria)
2. Ready (Prioritized list of work ready for future Sprint Planning Meetings)
3. Preparing (work in progress – facilitated by Product Owner controlled by READY criteria)
4. New (Prioritized list of work waiting to be made ready)

It is important to understand that READY is the least amount of activity bringing an unspecified high granularity item on the backlog, into a state where it is sufficiently described to allow the team to implement the item without disruption.

Finally we advice other agile projects to map themselves against the Scrum adoption model presented above, and to apply the above recommendations stepwise from ScrumButt to Great Scrum.

6. References

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

[2] J. Sutherland, C.R. Jakobsen and K.A. Johnson, "CMMI and Scrum - a magic potion for code warriors" in proceedings for Agile 2007

[3] C.R. Jakobsen and K. A. Johnson, “Agile with a twist of CMMI” in proceedings for Agile 2008

[4] M. K. Kulpa and K. A. Johnson, *Interpreting the CMMI: A Process Improvement Approach, Second Edition*. Boca Raton: Auerbach Publications, 2008

[5] Womack and Jones, *Lean Thinking: Banish Waste and Create Wealth in your Cooperation,* Free Press, 2003.