Magic Pillars of High Performing Lean and Agile Development Experience

Carsten Ruseng Jakobsen
Systematic Software Engineering
crj@systematic.dk

Tom Poppendieck
Poppendieck LLC
tom@poppendieck.com

Abstract

Systematic Software Engineering works at CMMI level 5 and uses Lean Software Development as a driver for optimizing software processes. Previously reported pilot projects showed productivity on Scrum teams almost twice that of traditional teams with 40% fewer defects. Systematic has used Lean to improve their software development. Experiences from monitoring projects using Systematics optimized processes, has revealed an insight into key aspects in a project that are critical for successful execution. These aspects are rooted in Lean mindsets supported with agile practices. Our experiences show how to diagnose and drive the different key aspects. These experiences are easily transferred to agile companies not working with CMMI. The experiences also show important lessons learned on how to combine team retrospective learning with organizational learning.

1. Introduction

Since 2005, Systematic has used Lean principles and Lean Software Development to optimize how projects are executed. Initially this led to the adoption of Scrum and an agile development process with focus on early testing. Several years of monitoring projects using these processes has revealed that whenever a small number of pillars or hallmarks are ensured in a project, it will succeed.

The experiences presented represents the result of applying a subset of all the advice given in the three books "Lean Software Development", "Implementing Lean Software Development" and "Leading Lean Software Development". The experience is valuable because it indicates the 5 most important aspects in a typical software development project Systematic has found.

For a specific project it can be difficult to select what aspect to start with or focus on. Imagine a magic magnet swing over the books which is designed to attract the 5 most important aspects to focus on for that specific project. The experiences from Systematic during the past years, has been such a magnet, for a typical project in Systematic. Inspired and driven by Lean thinking, we have continuously improved our process. In retrospect, we realize that these improvements led us to select a few Lean mindsets, and establish simple practical objectives to visualize to what degree these mindset are implemented.

The most important objectives are the following:

1. Fix time of failed build must be less than a workday
2. Development of stories must have a flow of at least 50%
3. Defects must be found and fixed early so that final test and release for a typical sprint delivery takes less than 10% of the iteration (3 calendar days for iterations of 1 month duration)
4. Teams must be co-located, empowered and organized to achieve a size of 5+/-2
5. The velocity of elaboration of features (making them READY) must be at least the same as the velocity of implementing features (making them DONE).

Experiences from Systematic indicate that these 5 objectives have the properties:

- Successful projects will achieve the objectives.
- Troubled projects will fail on at least one of the objectives
- Objectives are meaningful to the team the and team can relate to them

This paper presents why these experiences work in Systematic, what mindsets from Lean and Agile they are rooted to and what agile practices support them. Finally we describe how Systematic has established this learning in a combination of project retrospective and organizational learning.

2. Lean inspired improvements

2.1. The company

Systematic was established in 1985 and today employs more than 450 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[SM] 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 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. Lean Software Development analyzed

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 [Z] 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 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 a first understanding of a Lean mindset. The causal dependencies between the principles and tools in Lean Software Development were analyzed, and resulted in the model presented in Table 1.

The model groups the thinking tools from Lean Software Development into categories: Engineering, Management and People. Furthermore the elements are arranged according to causal dependencies, where elements to the right depends on one or more elements to the left. These dependencies has been simplified into four phases named: Value, Flow, Pull and Perfection. The model facilitated a way to prioritize what thinking tools to focus on. Left most tools were considered good candidates to start with.

2.3. Systematic Lean experience

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 led to the adoption of Scrum and early testing.

In the period December 2005 – December 2006 Scrum and a development method with a strong focus on early testing was adopted. The following years has continued to focus on lean inspired improvements, and experiences from many projects has been accumulated. These experiences has identified that a few key-aspects can be identified for each row in the model shown in table 1. The five objectives described in the introduction, are key to successful implementation of the Lean thinking tools related to Engineering. The Prince2 described collaboration between Sponsor, SuperUser and Supplier is essential for the Lean Thinking Tools related to management and can be monitored with objectives on customer attendance to steering group and sprint review meetings. Finally the Lean Thinking Tools related to People depend on empowered teams with self determination. An indication of empowerment is to let senior management analyze the sprint goals defined for the teams, to determine whether the goals are true goals, or to what degree they are a list of tasks.

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

	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

.

Project improvement vs org. improvements

Improvement requires baselines

Baselines require measures (objective data)

Performance baselines are done using statistical methods, e.g. control charts

Who does the studies and makes the process changes

STOP READING HERE

3. Driving and measuring Lean mindsets

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 helps to understand the natural variation in the measures, and thereby helps 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 failed builds

The main reason to measure how long time it takes from a build fails on the shared build server until next successful 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 final verification testing 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 measure "Fixtime after failed build" is the number of working hours from when a defect is identified on the shared build-server 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.

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 <u>special causes of variation</u>, that is causes for broken build fix times that exceed 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.

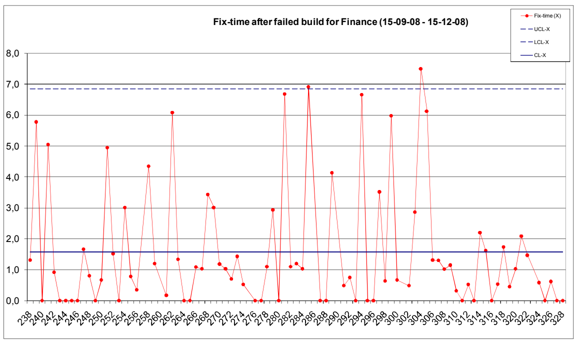

Figure 2 Time to fix a failed build

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 to surface impediments and motivate 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 posses 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 of how the test environment was setup
4) A commercial off the shelf (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-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.

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.

Systematic decomposes requirements in the contract, into 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.

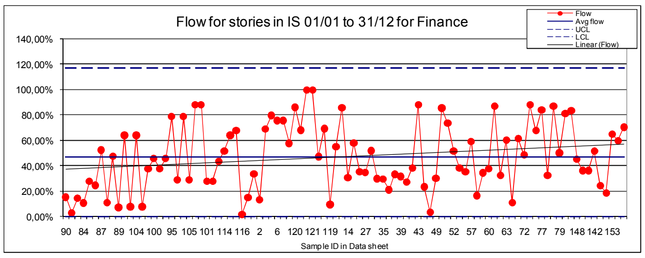

Figure 3 Flow of implementation of story

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. Agile hallmarks and adoption

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

5. Recommendation

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, Lean and Scrum together results in significantly improved performance while maintaining CMMI compliance.

A lean culture with a disciplined approach, skilled people, and good leadership can systematically significantly 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..