Lean as a Scrum Troubleshooter

Carsten Ruseng Jakobsen
Systematic
Århus, Denmark
e-mail: crj@systematic.com

Tom Poppendieck
Poppendieck LLC
Eden Prairie, MN, USA
e-mail: tom@poppendieck.com

***Abstract*— Systematic works at CMMI level 5 and uses Lean Software Development as a driver for optimizing software processes. Many of the optimizations described in this paper are the result of using A3 problem solving. What makes the Systematic experience unique, is the larger focus of the problem solving effort, at an organizational level, in which individual projects are used as experiments to try out countermeasures to address root causes. This is possible because Systematic, based on a CMMI focus, already employs a level of standard work across project and product engagements so that we can apply learning from an experiment on one project to future projects. Experience from the past five years has resulted in significant improvements to our processes including our Scrum implementation, and has revealed insight into five key measures to monitor projects. The experiences also show important lessons learned on how to combine team retrospective learning with organizational learning.**

Keywords: A3, continuous improvement, Lean, Scrum

I. 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 a focus on early testing. Since then a number of other larger improvements have been completed.

This paper presents through four case studies how the A3 problem solving tool from Lean, drives the thinking behind these improvements and how Systematic has established this learning in a combination of project retrospective and organizational learning.

II. SYSTEMATIC – SIMPLIFY CRITICAL DECISIONS

Systematic was established in 1985 and employs more than 450 people with offices in Denmark, Finland, US 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, police, public sector, finance and service industries. Systematic was found to be compliant with CMMI level 5 for the first time in November 2005. In 2005 Systematic decided future improvements were to be built on CMMI practices combined with a lean culture and mindset.

A. *Lean Culture and Agile Practices*

The first major improvements inspired from Lean included working in shorter iterations to get more feedback from customers and a focus on early test and immediate repair in development activities. This improvement resulted in the adoption of Scrum and a new feature-driven software development method with a focus on early test. It also demonstrated that disciplined work fulfilling CMMI practices is possible while at the same time being adaptive using agile methods. This was confirmed when Systematic were routinely re-assessed to CMMI level 5 in 2009. Systematic has demonstrated how it is possible to implement the CMMI disciplines within a Lean culture. Our processes are non-bureaucratic and are a transparent natural part of daily routines.

III. A3 PROBLEM SOLVING TOOL

A. *PDCA with an A3*

We develop software to help our customers solve problems. Along the way we need to avoid introducing problems created by the way we choose to pursue our software development. We strive to mistake-proof our process by paying attention to how well our work is going and repairing both code and practices as soon as possible. We keep our code clean using continuous integration to discover as many problems as we can as early as we can and fixing them immediately while the cost of fixing issues is low.

The cycle for improving practices is a bit slower. We repeatedly apply the plan, do, check, adjust (PDCA) cycle popularized by Deming and used extensively in Lean organizations. Specifically, we employ the A3 process described by Jon Shook in *Managing to Learn* [6]. The name A3 comes from the discipline of communicating the entire PDCA process on one side of a single A3 sheet of paper. This constraint forces us to present our thinking concisely so that it can serve as an effective discussion vehicle among everyone who can contribute to understanding or whose behavior needs to change to ameliorate the problem.

The A3 process focuses on the PLAN part of the PDCA cycle by addressing two questions. First, why does the problem we are addressing matter to the organization and to

those who need to change the way they think and work. If the people involved do not really care, we cannot expect their participation in helping to understand causes of the problem and in changing the way they work to make it go away. The second question of planning is to discover the root cause of the problem. This involves creating a mental model of cause and effect, perhaps using a 5-whys analysis, a fishbone analysis, or a causality network diagram. A candidate root cause is reached when we identify faulty thinking, false assumptions, or incorrect reasoning that produced decisions that caused the problem.

The second step in the PDCA cycle is the DO step. For each candidate root cause some experiments are devised to verify them as root causes. The goal is to observe the effect of correcting the faulty thinking we believe is the root cause of the problem. If the hypothesis reflected in our model is good, we should have a clear definition of what will change when we do our experiments and by how much.

The third step in the PDCA cycle is CHECK that the experiments confirm our root cause hypothesis. If our predictions of anticipated impact are significantly wrong we do not really understand well enough to make changes in our work practices and we need to go back to planning.

The fourth step in the PDCA cycle is to ADJUST our practices or policies or standards or checklists or guidelines to reflect what we have learned from the cycle. The full PDCA cycle can take a few hours within a team to months for issues affecting many teams or multiple projects and customers which intrinsically have much slower feedback.

B. *Application at Systematic*

Systematic, like many other organizations routinely compares actual performance to business objectives to create a list of important improvement opportunities or problems affecting many teams that it needs to solve. Prior to our adoption of Lean we attempted to assign a small group of the brightest people in the organization to tackle these problems. This did not work well. Even when we thought we identified good solutions, we were unable to get development teams and their leaders to buy-in and adopt our answers.

Significant progress began when we adopted the A3 process to engage people, particularly in the planning step focused on understanding the consequences and root causes. The typical pattern is that a VP or senior management identifies issues that impact many delivery teams to many customers. We then hold a workshop involving people from all the affected teams to work through the A3 steps starting with appreciating the impact and identifying candidate root causes. The workshop sessions collectively identify countermeasures, experiments that will tell them if their understanding is correct. After the initial workshops they try the experiments and get back together to analyze the results to see how they all can adjust their practices. We re-discovered that 'all of us are smarter than any of us.'

Systematic has introduced Lean-inspired practices like A3 problem solving and fish-bone analysis as part of implementing CMMI level 5 processes for optimization: Causal Analysis and Resolution (CAR) and Organizational Performance Management (OPM).

Systematic applies A3 problem solving at all levels. People, project teams or senior management solve significant problems or pursue opportunities when they are identified using the A3 problem solving tool.

Senior managers gain insight into the challenges within all projects and any significant deviation from business or project objectives through monthly project reviews. When they see that several projects are struggling with the same problem, this may trigger a management driven A3 problem solving involving a subset of the projects facing the issue.

Problem solving is always done by the people involved in the process being improved, and in the context where the process is used. This typically means by people on a project performing a work flow within that project. The managers role is to mentor the teams to ensure the completeness of their investigation, modeling, counter-measures and assessments [6].

When senior management initiates an improvement, the first step is to verify the existence of the problem or opportunity within the projects involved.

When senior management assigns an A3 problem solving effort and starts working with teams on the problem, senior managers will facilitate the problem identification and root-cause analysis among the projects. The different projects establish project-specific counter measures in response to their cross-project analysis of the problem or opportunity. Subsequently they check what worked and what didn't in the different projects and practices are adjusted for future projects based on this learning.

For a default standard process to be updated, at least one project must have shown the improved process to be better while executed in the improved project context. The people who use the process are responsible for seeing that the process works and improves.

IV. SYSTEMATIC EXPERIENCES

We present four examples of how we applied A3 at a cross-team level to improve our practices.

A. *Continuous Integration – Stop and Fix immediately*

During 2006 Systematic adopted Scrum and a new development method with a focus on early testing. These improvements were identified by matching business needs for shorter cycle time and improved defect containment with Lean Software Development tools.

1) Problem Impact

By the end of 2006 some projects occasionally found that they were unable to deliver as planned due to unexpected problems in the building or testing environment or due to fixing unanticipated problems with integration. The result was unhappy customers and excess costs due to late delivery created by late identification of integration issues.

2) Root cause

- **Why do we have unanticipated integration problems**?
  We identify issues related to build, test and integration too late.
- **Why do we identify these problems late**?
  Integration is both done too late and some issues discovered during development are postponed.
- **Why do we postpone**?
  All projects have their own build server, but it is not monitored. New developed code is checked in at the discretion of the developers, perhaps on a daily basis. Some projects do nightly builds but most don't.
- **Why do we not always keep the build working**?
  There is no policy or objective for how fast failed builds should be resolved.
- **Why do we not pay attention to always having a good mainline codebase**?
  We have the wrong fix-it-later mindset to integration problems instead of a mindset that any integration problem is real-time information on issues that must be resolved immediately.

3) Countermeasures

Aha, the crucial insight is that we need to change our mindset to integration problems. The negative effect of a wrong mindset is a process out of control. We started the improvement by defining a metric for fix time of failed builds, and we worked with the improvement, until the measure was used by all projects. The metric for fix-time of failed builds were plotted into statistical control charts, showing whether the process of fixing failed builds were in statistical control and the variation of the process. Our hypothesis is that the right mindset, stop and fix immediately, leads to a process in control. We looked at the data points making the process out of control, to better understand how to change our mindset.

- To establish a new mindset we decided to implement a new metric for the projects – **fix time for failed builds.** All projects were using CruiseControl. We established a central database to collect information from CruiseControl in all projects. Whenever a build is completed on the build server, the results are stored automatically in this database. This data collection allows project specific measures of actual average fix-time of failed builds. We immediately established a baseline for that metric. The baseline (presented and calculated as control charts) showed that the process was not in control.
- We expected that by making this metric visible to the teams, they would adjust their efforts to get it under control and make it shorter. We chose two ways to make the measure visible:
  1. We set up CruiseControl to make build status immediately available for all on the project.
  2. We showed the fix time status from all projects on a computer next to the coffee machine in the hallway providing daily opportunities to discuss with random people from projects why this mindset is important.
- An analysis of causes to long fix-times revealed that in most cases simple criteria for when code can be checked in to the build server would catch many defects earlier: Have you rebased locally, have you recompiled locally, did the unit test succeed locally … OK you may now proceed to the build server. We automated code standards checking with standard tools like FxCop at code check-in.

4) What did we learn from the results?

- In contrast to other project management metrics, we learned that the time it takes to fix a broken build is a number that most developers can immediately relate to.
- Sharing the real time numbers in the hallway next to the coffee machine was a great way to call attention to the topic. Within few months the mindset had changed in the projects. This could be observed from the metric. The number of builds with a fix-time of several days gradually disappeared.
- However, the mindset did not make it happen alone, but it gave people a good understanding of why they needed to change other practices, like criteria for when to check in code.

5) How did we adjust our way of working?

The major impact of this improvement was that people achieved a mindset that problems must be addressed immediately. The build scripts in all projects were standardized to support either a java setup or a .NET setup. A standard setup to collect the build status was established for each. Today the metric is mandatory for all projects, and the process is statistically managed with control charts. We know that if the largest fix-time is less than seven hours, then the process is most likely in control. The average fix-time will be somewhere between one- and-a-half to three hours.

Another important learning from statistical management of time-to-fix failed builds, is that long fix-times are almost always related to impediments for the team. Therefore extra attention is paid to the reasons for long fix-times of failed builds, on a daily basis, and during project reviews.

B. Ready-Ready and Story Flow concepts

The following is an example of an improvement initiated by an opportunity as opposed to a problem.

1) Opportunity impact

Senior management analysis of productivity in projects in Systematic in August of 2008 showed that some sprints in two projects achieved at least two times better performance than projects on average. Our opportunity was to determine if there were lessons learned from these high performing sprints that could be copied and sustained across all projects. In order to verify and understand the opportunity we conducted a series of interviews with eight people from the two projects and senior management.

2) Root cause

- **Why is such high performance not achieved on other projects**?
  Other projects have insufficient time allocated for product owner (PO) work.
- **Why is enough time not allocated?**
  Status and progress on PO work is not measured.
- **Why do we not measure PO Work?**
  The responsibility and organization of the PO role is unclear to most projects.
- **Why is PO work unclear?**
  Insufficient focus on readiness of work on Product Backlog.
- **Why do we not have ready criteria for PO work?**
  Sprint delivery has strong code focus, and this causes inexperienced projects to sacrifice activities to prepare product backlog.
- **Why is this not just a question of organizing the PO role in the projects**?
  The successful project sprints did not care about who is the product owner. They carefully designed what activities were to be carried out by who and when. This way they not only appointed the PO, they also described how the team collaborated with the PO. The successful projects did this to achieve a stable flow of activities within sprints.

3) Countermeasures

Aha, the key is to ensure that sufficient focus is on the activities to prepare the product backlog.

- To create focus, we decided to establish a management pull and metric to indicate if the product backlog is prepared sufficiently. Project managers were asked to define the metric, and they suggested a sprint readiness metric. The metric is the percentage of stories allocated to a sprint that is ready, and is supported by a checklist. We expected other teams adopting ready practices would double their flow as the high performing teams had.
- We measured the sub-flow for story implementation. The reason for ensuring that work allocated to a sprint is ready is to ensure that once a developer starts a story, the story can be developed in a smooth flow. Assume a story is estimated to be three workdays of effort. However, for various reasons it takes nine workdays to implement the story. The flow of this story implementation is then defined as three days calendar time of work implemented over nine calendar days, a flow of 3/9 or 33%.

4) What did we learn from the results?

- The Ready check list was piloted and resulted in the timely execution of preparation activities that otherwise often were postponed to a few days before sprint planning.
- Due to the timely execution of activities, it became easier to conduct estimation workshops with a broad representation of the team well ahead of Sprint Planning.
- 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 estimation of stories.
- As a result, the Sprint Planning meetings are now much more efficient, because the team knows what the features and stories are about, a topic that otherwise took a lot of time during Sprint Planning.
- When we started measuring flow it was around 30%. In Q4 2008 it had increased to 59%. 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.

5) How did we adjust our way of working?

Our policy now requires monthly reporting measures of %READY and flow by each project. The organizational objective is that at least 100% of stories selected for a sprint must be ready, and the implementation flow of each story must be at least 60%. These measures are good indicators that drive behaviors to let teams go twice as fast as they could before.

C. Project initiation

In December 2008 we observed that the past four projects spent two to three months to establish project foundations. This is significantly longer than our business objective of one month.

1) Problem impact

We observed that all project managers found it difficult to establish a project foundation in a timely manner after project initiation. In particular, planning project startup and good elicitation and management of requirements are very challenging. Sessions to analyze the root cause were conducted in three separate sessions with senior management, project managers, and key roles as participants.

2) Root cause

- **Why is the project foundation not established in a timely manner for new projects?**
  Project managers feel that initiating the project is like hitting a wall, and find no easy way to get through and no clear direction. Some activities done by the project manager during project initiation are not formalized or often done ad hoc. Examples on such activities include: how knowledge from a bid team writing a proposal is transferred to the project team executing it, how the appropriate competences are ensured in the project team, how initial setup with shared functions like Finance are established, and how to establish the initial product backlog.
- **Why is the initiation process difficult?**
  Our project start-up process is unclear. It has a focus on initiation inputs and outputs, but it lacks both essential milestones and the notion of co-ordinated support from shared functions. Furthermore, team competence needs are identified and met too late.
- **Why don't we have a usable process?**
  The project manager is typically under time pressure during project startup and has not had the opportunity to improve their process.
- **Why are project managers under such time pressure?**
  Often the project manager will close the previous project at the same time a new project is started. Project managers start a project every one or two years, and may not remember how the last project started or know what has changed. Starting a project involves substantial tasks: building a new team, learning a new customer, and involvement with most shared functions while at the same time establishing the foundation for the project. Without proper guidance and support, tasks to establish the project foundation are easily sacrificed.
- **Why are the project managers unable to handle these tasks efficiently**?
  There is a lack of essential milestones to report against so necessary support from shared functions during project start up is not triggered. Project managers and senior management discuss the staffing assigned to project for too long at the expense of late initiation of responses to competence gaps in the staffing assigned.

3) Countermeasures

Hmm, several candidate causes have surfaced: clear essential milestones, unified support from shared functions, and timely resolution of competence gaps in staffing provided. We decided to establish counter measures for all three.

- We established a project workflow checklist where work products are laid out in time to visualize what work products were due when and in what state.
- We established a project start up service, where a second project manager assists the project manager during startup, and acts as the point of contact to all other shared functions. This person has the latest information regarding best practices.
- Policies were installed to ensure that handover from sales is done formally within two weeks.
- Initial staffing can be discussed during the first week. From this point the discussion focuses on how to close competence gaps on the assigned staff.

4) What did we learn from the results?

- In January 2011 the latest six project start ups have established their project foundations within three to four weeks.
- The startup service concept is a huge success. All shared functions now provide a startup service, and project managers greatly appreciate the more coordinated service.
- Finally the experiments have contributed to the introduction of a competence gap score to support discussion of how well specific staffing fulfills the project's competence needs, and adjust project objectives accordingly.

5) How did we adjust our way of working?

The project startup workflow checklists, the unified start up service and policies are now a mandatory part of the processes in Systematic.

D. Feature clarification with customer

The following is an example on an improvement initiated by a problem identified in several projects across all business units in Systematic.

1) Problem impact

During 2008 the Vice Presidents observed many projects struggling with clarifying features in collaboration with the customer. Clarifications from the customer were late, leading to a decrease in flow, which we know causes schedule and cost overrun. The problems were expressed verbally by project managers during periodic project review meetings, and supported by the a drop in the Story Implementation Flow metric. Initially fifteen people from five projects were invited to a shared root cause analysis.

2) Root cause

- **Why are feature clarifications from customer late and causing low flow in story implementation?**
  In some cases, we ask the customer to clarify too much work in too much technical detail in areas where the customers have little competence. In other

cases, the project depends on a technical infrastructure provided by the customer.

- **Why do we ask customers to clarify too much technical detail?**
  The IT-infrastructure provided by the customer may be documented insufficiently or we may try to isolate work in a sprint from customer changes. However, we do not separate decisions regarding needs from decisions regarding solutions, but ask the customer to clarify both.
- **Why are upfront needs for documentation not identified and why do we try to avoid customer interaction for work in progress?**
  We trust that if we have questions about the customer's IT infrastructure, the customer can provide quick clarification. The customer wants to clarify or change work in progress, which causes a lot of rework. We would like to clarify work for sprints after the current sprint. We would like to get information from the customer to fit our needs using our internal development processes, but the customer would like to comment on the solution as it emerges.
- **Why are these needs for clarification not aligned and why are concerns for needs and solutions not separated?**
  We have not established a shared process with the customer regarding feature clarification. A shared process would describe the who, what, when, and how for feature clarification activities. For some projects the initial setup of the project resulted in an insufficient foundation causing more decisions to be postponed than can be achieved within agreed timeframes. Weak implementations of the product owner role, cause projects to ask for technical clarifications the customers are unable to provide, as opposed to clarification of needs. In other words, we are not establishing customer commitment for the desired collaboration. At the same time the implementation of the product owner role is unclear.
- **Why is customer commitment and implementation of the product owner weak?**
  Contracts increasingly depend on timely clarification and delivery from the customer to the project. Often the customer is unable or unaware of these expectations. The impact of this has not been visible to those negotiating and setting up project agreements with our customers. Different aspects of the product owner role are handled by different people on the project teams, but the teams have not explicitly clarified how the responsibility of the product owner role is allocated to the people on the team.

3) Countermeasures

The first understanding of the root cause focused on defining a shared process with the customer on feature clarification. The five projects involved executed counter measures to this understanding of the root cause as it was formulated in January 2011. When the effects of these counter measures were evaluated in March 2011, it was clear that the desired effect had not been achieved. The projects got together and re-evaluated the root-causes. The result of this reflection was that the root cause is related to building customer commitment to the close collaboration needed and to ensure that all projects establish a clear understanding of how the product owner role is implemented. Based on this new understanding of the root-cause it seems that what Systematic can do is to improve the product owner role with a stronger focus on:

- Aligned expectations on customer involvement and deliveries, for example by ensuring that the steering group meets shortly after contracting for a workshop to discuss the vision of the contract, and house rules for collaboration, like timely, open honest communication.
- Clear mutual understanding and agreement of who is involved and when on the customer side regarding specific clarifications.
- Clear mutual understanding and agreement regarding time frames for responses needed for efficient decision making.
- Clear understanding of how the different product owner responsibilities are allocated within the team.
- Shared process with customer for collaboration with the product owner and the team so that clarification needs are highly visible to responsible parties on both sides and so failure to perform is not ignored.

4) What did we learn from the results?

At the time of this writing, the effect of the revised countermeasures is unknown. However, this case study shows the importance of evaluating the effect of counter measures and to readjust if desired outcomes are not observed.

5) How did we adjust our way of working?

How this improvement will change our way of working is yet to be seen.

V. LESSONS LEARNED

The application of A3 problem solving is a powerful tool for an individual project, which is amplified when used across projects with the involvement of senior management. In the first case study, projects identified the need for an integration tool but senior management identified the need for a standard setup across projects. In the third case study, projects observed that part of the problem was related to how initial staffing was handled. This is difficult to change for a project but senior management could facilitate adjustments to the staffing policy.

Several years of Lean motivated improvements have created significant adjustments to our processes and identified these key metrics for monitoring projects:

1. Fix time of failed build must be less than a workday.
2. Development of stories must have a flow of at least 60% (Flow = estimated implementation time/actual implementation time.)
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 (three calendar days for iterations of one month duration)
4. Delivery teams must be co-located, empowered and organized to achieve a size of five plus or minus two.
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 five objectives have the following properties:

- Successful projects will achieve all 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 list of objectives represents five years of learning, but will change over time as new learning is created to improve performance further.

The disciplined use of A3 problem solving has helped Systematic to improve flow in its processes. The first improvements of flow were initiated in the release activities. Once flow was established in release activities, the A3 process was used to improve flow in test activities, development activities, project startup activities, and in customer activities related to contracting and ongoing clarification. These improvements have required a sustained focus over longer time, and have in return gained major benefits for Systematic.

VI. CONCLUSION

Many of the adjustments implemented are characterized by being desired but beyond the control of the projects. Had the projects driven these improvements on their own, they might have discarded the ideas for adjustments because of the need to involve senior management or VP's in the decision. When the problem solving is initiated by senior management, the negative impact of the problem is viewed both from the project perspective and also from the business unit or company perspective. In this larger perspective larger problems build high management commitment, because the impact of the problem and the benefit of the solution is visible in a larger scale.

Furthermore, such improvements are backed with more factual data, for example "We can see that the flow of story implementation increases from 30% to 60% when customers are able to clarify features on time, and all work allocated to a sprint is READY".

The most important learning from the improvements during the past five years, is the Lean concept of jidoka. Respect that those who do a particular part of work, are those who are best qualified to improve how this work is done. In essence do not ask people to speculate on a solution to a problem of someone else. Make people responsible for solving their own problems and ensure that management supports it. When management and senior management engage in helping people do their job better, instead of just asking them to do it better, it is soon evident to senior management what problems are shared among projects. Solving such problems will often create a change in the environment, where business is improved and customers and employees are more satisfied.

REFERENCES

[1] J. Sutherland, C.R. Jakobsen and K.A. Johnson, "CMMI and Scrum - 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] C. R. Jakobsen, J. Sutherland, "CMMI and Scrum – going from Good to Great" in *Agile 2009*, Chicago, 2009

[4] C. R. Jakobsen, J. Sutherland, "Mature Scrum at Systematic" published in "Methods and Tools" in 2009

[5] Steve Denning, The Leaders Guide to Radical Management, Published by Jossey-Bass, 2010, ISBN 0-470-54868-1, 2011,

[6] Jon Shook, "Managing to Learn: Using the A3 Management Process", Lean Enterprise Institute, Inc.; 1 edition (January 2008), ISBN 1934109207.