# Lean Mindset in Software Engineering: A Case Study in a Software House in Brazilian State of Santa Catarina

Mehran Misaghi[1] and Ivan Bosnic[2]

[1] UNISOCIESC, Joinville, Brazil
mehran@sociesc.org.br
[2] NeoGrid, Joinville, Brazil
ivan.bosnic@neogrid.com

**Abstract.** This article presents a literature review whose purpose is to identify the key characteristics of lean software development and its similarities and differences with agile methodologies. For concept proof, a case study conducted in a team of software developers is presented, where lean concepts were applied within the current process, previously based on agile methodologies. It was found at the end of this work that the indicator used by the team, percentage of the time spent on improvements and new features, had a significant increase, causing the team be able to add more value to the product, and to increase the level of quality. This article ends with the presentation of the steps required for the development of lean mindset in software engineering.

**Keywords:** Lean mindset, Agile methodologies, Scrum, Software.

## 1 Introduction

Modern societies depend every day more on diverse types of computer programs. Such programs manage our bank accounts, control the supply of water and electricity, monitor our health when admitted to hospitals, entertain us when we play video games, and provide many others critical services to the community. It was expected that, as they are dealing with services so fundamental to our lives, software projects were at a very high level of success.

However, according to [1], the practice of software development has been plagued with critically low success rates for decades. Meanwhile, demand for IT products and services do not stop growing and the situation seems to get into a chaotic situation with no solution. What has brought some optimism is the emergence of agile methodologies, which have shown that it is possible to obtain better success rates. The authors observed that there is a trend of improvement in the quality of the projects, but still the situation requires attention, because the percentage of projects that exceed the costs or terms remains almost as high as before.

[1] also emphasize that lean techniques have been increasingly applied to software development. Ideology and lean techniques to which the authors refer are the same

used in the Toyota Production System and Toyota Product Development. According to [2], the first step in the implementation of the lean software development is to understand these principles, because software development is a form of product development. Applying the concepts of lean manufacturing, used for a long time in traditional industry and especially in the automobile industry, to the process of software development is the challenge behind the lean software development.

This paper presents a case study conducted within a team of experienced software developers that have used agile methodologies in the past decade with great success. Since early 2012 the team has invested in implementing lean concepts in the process of software development, which has had a positive impact on monthly indicators presented to company management [3].

## 2 Lean Software Development

According to [4], the ideas of lean software have their origin in lean manufacturing and lean product development. These concepts, in turn, had their origins in the Toyota Production System and the Toyota System of Product Development.

According to [2], software development is a form of product development. The authors were first to introduce in 2003 the concept of lean software development. The main focus of their work was to identify lean concepts and how they could be applied to software development.

Although agile and lean software development both have been inspired by the lean concepts, [5] emphasizes that agile methods are applied only to software development, while lean is a much broader concept. According to [6], the lean philosophy is not just a set of tools. It affects all sectors of business, from human resources to marketing. From this work were established seven principles of Lean Software Development [2].

### 2.1 Principle One: Eliminate Waste

According to [7], the Toyota Production System has as one of its foci the total elimination of waste. The author states that everything that does not add value to the customer must be removed from the process. According to [1], this category includes a number of concepts that must be analyzed so that we can understand how waste indicated in the Toyota Production System can be identified in the process of software development.

- **Defects:** Defects are represented by themselves. Defects cause costly rework, which does not add value to the product. The lean software development has as one of its goals preventing defects.
- **Overproduction:** Unnecessary features. The cost of software is not contained only in writing the source code. This code needs to be maintained, documented, taught to the new team members, etc. For this reason, all the features embedded in the software should come from the real needs of the user, i.e., features that add value to the final product. According to [1], the study 'CHAOS study' Standish Group

showed that 64% of all the features are not used or are rarely used. This is a great waste of resources over time.

- **Stock:** Partially completed tasks. Here we consider requirements analyzed but not implemented, code that has not been tested or errors that have not been corrected. The lean philosophy does not admit the accumulation of uncompleted tasks. Instead, we try to adopt the unit flow that makes the task completed as soon as possible.
- **Transportation:** Switching between tasks. Interruptions and work alternated between very different activities affect productivity. Before starting work on a task, people need time to acclimatize to the problem and to understand the requirements. Any interruption causes this process to be restarted. This is one reason why the flow unit is so productive.
- **Further processing:** Unnecessary processes. This type of process is the most pure waste. It hinders productivity without adding any value to the final product. An example of this process is the creation of documentation that is not used by anyone, or even manual execution of tasks that could be automated.
- **Standby:** Delays. During the process of software development programmers often need to communicate with other project participants to ask questions and clarify certain requirements. If these participants are not available, there will be delays in delivery or implementation will be done without the proper information, which in most cases will generate rework. This rework is one of the most common forms of waste in the process of software development and should be avoided at any cost.

## 2.2 Principle Two: Integrating Quality

[7] states that it is not possible to inspect the quality of a product at the end of the production line. According to [1], traditional development methodologies make exactly this error: allow defects to be detected later by the team of quality assurance.

Lean software development, moreover, proposes a different philosophy. Instead of creating systems to control defects (nonconformities queues to be resolved), the process should be focused on the total elimination of defects and the consequent elimination of rows control [2]. To achieve such a degree of maturity in the process is only possible with the use of resources such as unit testing and continuous integration, among others.

## 2.3 Principle Three: Creating Knowledge

According to [2], one of the major flaws that software development plans aimed at is the idea that knowledge in the form of requirements exists separately from coding. Authors emphasize that software development is a process of knowledge creation and the detailed design, although it should be outlined before, it stands only during the implementation of the code.

[1] has put that knowledge should be stored in such a way that it can be easily located the next time it becomes necessary. People should not waste time learning something that has already been studied and put into practice by other team members.

### 2.4 Principle Four: Postpone Commitments

[1] assert that the best decisions are made when we have as much information as possible. If a particular decision needs not be made immediately, we should wait until we have more knowledge on the subject. According to [2], this item applies mainly to making irreversible decisions. The reversible decisions can be taken before, because they can be easily modified.

### 2.5 Principle Five: Delivering Fast

[8] teaches that we must begin with a thorough understanding of what adds value to the customer. Once understood the needs of the client, we create a workflow that seeks to make rapid and frequent deliveries of working software. According to [1], the importance of delivering fast is to get customer feedback as soon as possible. Thus, we avoid the requirements change just because they take too much time to be delivered.

### 2.6 Principle Six: Respect People

According to [2], thinkers and people engaged in the project are the largest and most sustainable competitive advantage that a company can have. This thought defines what people represent in a lean philosophy. Respecting people means trusting that they know the best way to perform a job and enables them to find ways to improve processes.

### 2.7 Principle Seven: Optimize the Whole

According to [2], improving a local process is usually achieved at the expense of the value stream in the entire process. This occurs when changes are made without considering the whole. This is known as sub-optimization, and an organization that implements lean concepts always tries to avoid it.

## 3 Case Study

The company chosen for this case study has a long experience in software development. Currently, it is ranked as the leading supplier of systems for the supply chain in Brazil. Furthermore, it has successfully implemented the agile software development, Scrum and XP during the last decade. In the last five years, the company has increased its interest in the concepts of lean software development, with the intention of improving the productivity of its teams [3]. This case study was conducted from September 2011 to August 2012. At this time, we had 12 people on staff. 8 people have had solid experience in software development (levels between full and senior). The rest were younger and some also trainees.

## 3.1 Lean Concepts in Practice

Several indicators have been used to monitor the productivity of these teams, and goals have been established to evaluate their progress. One of the main indicators evaluates the time that a team invests, during each software version, in improvements and new features. These tasks add value to the product and the increase of this indicator has been one of the goals of the company.

All other activities performed by the team are considered waste, even if some are needed so that the process can be managed correctly. Examples of some activities performed by the team are correction of non-conformities, participation in meetings, planning and others. When the time spent in correction of nonconformities (errors caused during the execution of software) increases, it is an indication that the product quality has worsened. Consequently, the team will have less time to invest in improvements and new features.

In an attempt to improve the indicators and increase the quality of the product, the team that was followed in this case study chose to adopt the concepts of lean software development. Each of the seven concepts explained in section 2 of this paper had a corresponding action based on [9,10,11,12,13].

### 3.1.1 Eliminate Waste

The problem of multitasking has been identified as a major cause of decreased productivity. People were constantly engaged in more than one activity, which took their concentration of the main tasks (implement improvements and new features). Some multitasking arose by the constant need to provide support to other teams about how the software works, but others were caused by the behavior of the team itself. That is, developers were involved in more than one task at a time, because there were no clear rules within the process about what should be the correct behavior in these cases.

To deal with the problem of multitasking, the team defined two new guidelines in the process of software development:

1. Each version of the software, one developer would be elected to handle support tasks requested by other teams. Thus, the rest of the team would be free to devote to the development of new features and improvements.
2. No developer would be involved in more than one feature at the same time. The aim was to implement the flow unit (continuous). Only after completing an activity the developer would dedicate to another, even if that meant some downtime.

### 3.1.2 Integrating Quality

The practice of automated tests, i.e., tests that do not depend on human interaction and ensure the correct operation of one or more software features, would be integrated into the process from the beginning. Experience had shown that leaving the development of tests for a later stage caused waste, because it created an inventory of tasks that hardly was handled.

#### 3.1.3 Creating Knowledge

All knowledge about the product should be available to all team members. To achieve this goal, the company implemented a collaborative tool for knowledge management, where everyone could contribute documenting the processes in which they were working. The knowledge could not be restricted to a group of more experienced developers.

#### 3.1.4 Postpone Commitments

Important decisions, especially those involving changes in the architecture of the system, were postponed until such time that the team had more knowledge on the subject and therefore more security in the process of decision-making. This practice proved to be very effective, because it avoided hasty decisions.

#### 3.1.5 Delivering Fast

Divide the project into smaller iterations between three to four weeks, enabled rapid delivery of functionalities, even partially completed. It was thus possible to obtain customer feedback more rapidly, and allow them to have a higher level of involvement in the evolution of the product. This practice is widely used in Scrum, one of the agile methodologies adopted by the team.

#### 3.1.6 Respect People

At all meetings of planning future versions of the software, all team members are heard. The final decisions take into account everyone's opinion and make the team commits to the estimates.

#### 3.1.7 Optimize the Whole

The importance of understanding the processes of the company was highlighted within the team. Workshops were made with other teams to clarify several questions about how the software was used in practice. This knowledge was useful for evaluating the impacts of development of a new feature on internal and external customers. The result of this approach was an improvement in usability and better acceptance by customers.

### 3.2 Data Collection

The company at which the case study was conducted has several tools to manage the process of software development. All developed requirements are recorded as well as the tasks and corrections of bugs. The data of this case study was obtained from tools used in the process of software development:

1. **Jira:** This tool is provided by the company Atlassian and is used for registration and monitoring of requirements, time recording and graphs tracking progress of versions;

2. **Confluence:** Also provided by Atlassian, tool is used for documenting functional and technical details of systems developed by the company. It is a collaborative software where all team members have access to edit documents.

Every day, the team members record worked hours. Each time recording is obligatorily linked to a task, which can be an improvement, a bug correction, a meeting, etc. Each of these tasks, in turn, is linked to a particular component. Currently the components are divided into:

1. **Product:** Groups all the hours spent on tasks that add value to the product, such as improvements and new features, development of automated tests, etc.
2. **Bugs:** The time spent on correction of nonconformities;
3. **Support:** hours are recorded in support activities provided to other teams;
4. **Management:** all tasks related to project management: meetings, planning, daily meetings, etc.

### 3.3 Analysis of Results

To analyze the results, we used the one-year period, from September 2011 until August 2012. The actions taken by the team and which were explained in the previous sections had its implementation in February 2012. Thus, it is possible to observe the evolution of the indicators analyzed in this case study, covering the phases before and after the implemented changes. Data were obtained from the BI (Business Intelligence) tool provided by the team of software quality. The percentage of time spent is monitored monthly, and information is divided into three groups [3].

The group "product" covers all the hours spent on improvements and new features. Corrections of bugs are classified as group "bugs", while in the group "others" are inserted all other activities performed by team. Table 1 shows the history of the percentage of time spent in each of the groups defined above.

**Table 1.** Data collection of time invested by component

| *Month* | *9/11* | *10/11* | *11/11* | *12/11* | *1/12* | *2/12* | *3/12* | *4/12* | *5/12* | *6/12* | *7/12* | *8/12* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Product* | 50 | 54.2 | 51.82 | 54.4 | 51.65 | 60.02 | 57.37 | 61.29 | 65.13 | 64.21 | 60.47 | 61 |
| *Bugs* | 13.9 | 13.7 | 10.8 | 10.2 | 9.4 | 8.6 | 8.5 | 8.6 | 7.4 | 6.7 | 6.6 | 5.9 |
| *Others* | 36.1 | 32.1 | 37.38 | 35.4 | 38.95 | 31.38 | 34.13 | 30.11 | 27.47 | 29.09 | 32.93 | 33.1 |

Through the graph shown in Fig. 1 it can be seen more clearly how the tracked indicators evolved during one year.

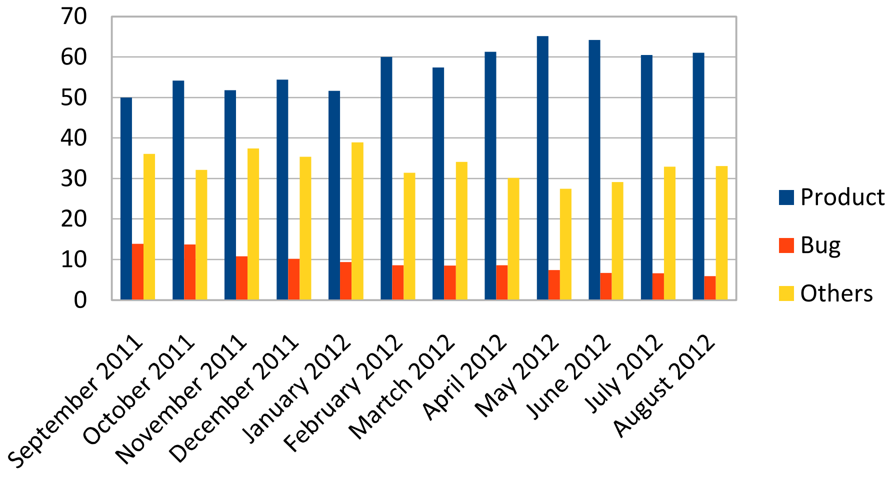

**Fig. 1.** Evolution of the percentage of time spent per component

### 3.3.1 Time Invested in Improvements and New Features

Through the collected data, it can be observed the increase of time spent on product relative to other components. While in 2011 the indicator stood at around 50%, from the changes implemented the same shall remain in the range of 60%. Therefore, we conclude that the indicator had an average increase of 20%.

Fig. 2 shows, in isolation, the evolution of the percentage of time spent on product. It is possible to observe that, as of February 2012, the month in which it started implementing lean software development; there was an average increase of 20% in this indicator.

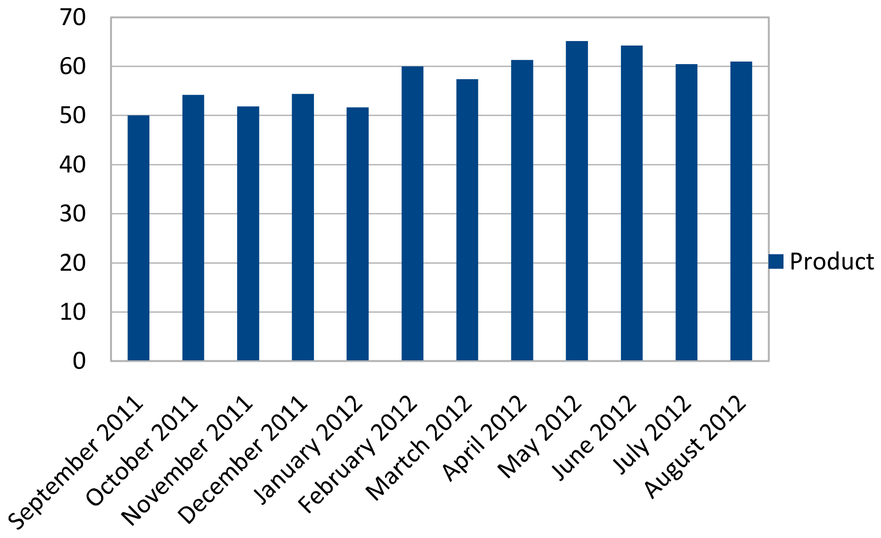

**Fig. 2.** Graph of the percentage invested in product

### 3.3.2 Time Spent on Bug Fixing

While there was an increase in the percentage of time spent on improvements and new functionalities, it was observed, on the other hand, a decrease in time spent on correcting bugs. Fig. 3. shows the evolution of this indicator.

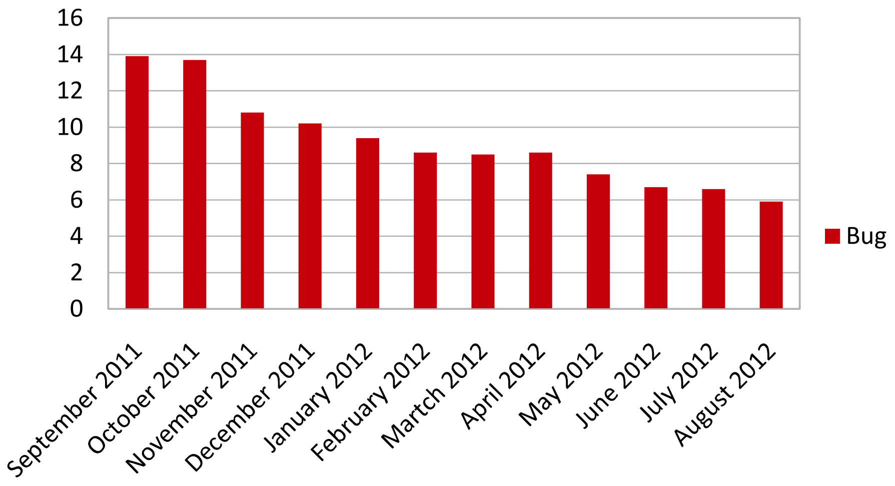

**Fig. 3.** Graph of percentage spent on bug fixing

It is important to note that the reduction of time spent on bug fixing was achieved with better product quality. That is, since the implementation of automated testing was incorporated into the process of software development, fewer errors were released and hence more time was available for investment in new features and improvements.

## 4 Lean Mindset in Software Engineering

According to [14], lean is a mental model of how the world works, lean is a mindset. For the impacts generated through implementation of lean principles in fact continue, there is a need to implement lean mindset.

[14] also emphasize that for presenting a mental model, we have to start with two questions: What is a purpose of a business? What kind of work systems are for accomplishing that purpose?

To understand how lean mindset work and how we can implement lean mindset, [14] propose five steps:

1. *The Purpose of Business*: emphasizes the principle Optimize the Whole, taking the Shareholder Value Theory to task for the short-term thinking it produces.
2. *Energized Workers*: is based on the work of Mihaly Csikszentmihalyi, who found that the most energizing human experience is pursuing a well-framed challenge [15].

3. *Delighted Customers*: urges readers to Focus on Customers, understand what they really need, and make sure that the right products and services are developed.
4. *Genuine Efficiency*: starts by emphasizing that authentic, sustainable efficiency does not mean layoffs, low costs, and controlling work systems.
5. *Breakthrough Innovation*: starts with a cautionary tale about how vulnerable businesses are—even simple businesses like newspapers can lose their major source of revenue seemingly overnight.

Fig. 4 shows in detail the various components of each step based on [14].

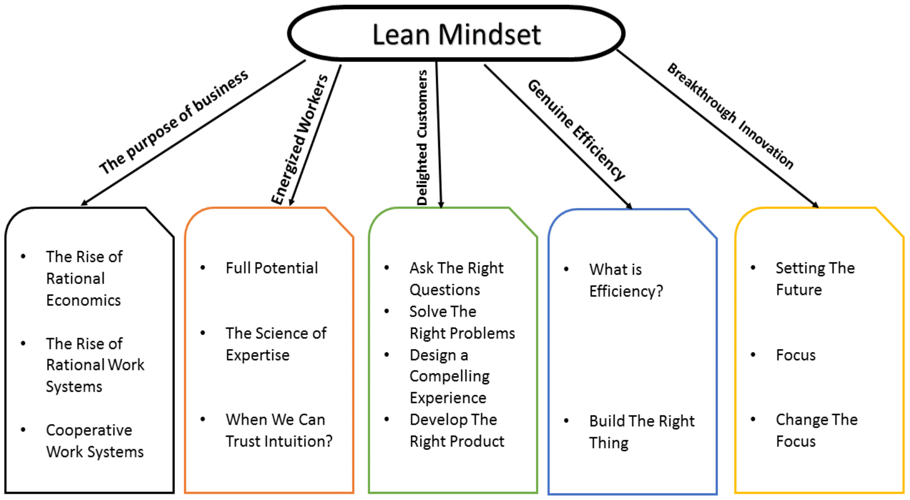
Lean Mindset
The purpose of business
Energized Workers
Delighted Customers
Genuine Efficiency
Breakthrough Innovation
• The Rise of Rational Economics
• The Rise of Rational Work Systems
• Cooperative Work Systems
• Full Potential
• The Science of Expertise
• When We Can Trust Intuition?
• Ask The Right Questions
• Solve The Right Problems
• Design a Compelling Experience
• Develop The Right Product
• What is Efficiency?
• Build The Right Thing
• Setting The Future
• Focus
• Change The Focus

**Fig. 4.** Lean Mindset Steps and Components based on [14]

For successful implementation of Lean Mindset, there is need for cooperation in various sectors of the whole organization, not just those directly involved with software engineering. We should view the organization as unique unit to ensure this success.

## 5 Conclusion

During the conduct of the case study and the subsequent analysis of the results, it was observed that the implementation of lean development software had several impacts on the development process adopted by the company. All the impacts were positive as they enabled the company to improve its software development process, adding more productivity and quality.

On the other hand, the specific objectives of this study, analysis of indicators of time invested in improvements and new functionalities and time spent on correcting bugs had their data collected and compared over a period of one year. Both indicators have improved, easily observed by the analysis of the results.

The elimination of waste was achieved with the elimination of multitasking, which had been identified as a major cause of reduced productivity. The practice of automated testing was responsible for integrating more quality to the developed software, while the implementation of a collaborative tool for knowledge management contributed to the creation of a unique knowledge base.

Our challenge is to define the criteria to implement lean mindset in software engineering, in our organization according to our need, with innovative ingredients.

## References

1. Hibbs, C., Jewett, S., Sullivan, M.: The art of lean software development. O'Reilly Media, Inc., Sebastopol (2009)
2. Poppendieck, M., Poppendieck, T.: Implementing Lean Software Development: From Concept to Cash. Addison-Wesley, Boston (2007)
3. Bosnic, I., Misaghi, M.: Lean Software Development: A Case Study in a Medium-sized Compnay in Brazilian State of Santa Catarina. In: ADIS-AC Proceedings, USA, pp. 163–170 (2013)
4. Shore, J., Warden, S.: The Art of Agile Development. Reilly Media, Inc., Sebastopol (2008)
5. Gustavsson, H.: Lean thinking applied to system architecting.Thesis. Department of School Of Innovation, Design And Engineering, Mälardalen University, Västerås, Sweden (2011)
6. Petersen, K.: Implementing Lean and Agile Software Development in Industry. Thesis - Department of School Of Computing, Blekinge Institute Of Technology, Karlskrona, Sweden (2010)
7. Ohno, T.: Toyota Production Software: Beyond Large Scale Production. Productivity Press, Oregon (1988)
8. Kniberg, H.: Lean from the Trenches: Managing Large-Scale Projects with Kanban. The Pragmatic Bookshelf, Dallas (2011)
9. Cohn, M.: Succeeding with agile: Software development using scrum. Addison-Wesley, Boston (2010)
10. Dyba, T., Dingsoyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology 50, 833–859 (2008)
11. Pressman, R.S.: Software Engineering: A Practitioner's Approach, 6th edn. McGraw-Hill, New York (2004)
12. Sommerville, I.: Software Engineering, 9th edn. Addison-Wesley, Boston (2011)
13. Vlaanderen, K., et al.: The agile requirements refinery: Applying SCRUM principles to software product management. Information And Software Technology 53(1), 58–70 (2011)
14. Poppendieck, M., Poppendieck, T., Kniberg, H.: Lean Mindset – Ask the Right Questions. Addison-Wesley, Boston (2014)
15. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. HarperCollins, New York (1990)