The Journal of Systems and Software 83 (2010) 1275–1287

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Software process improvement through the Lean Measurement (SPI-LEAM) method

Kai Petersen [a,b,*], Claes Wohlin [a]

[a] School of Computing, Blekinge Institute of Technology, Box 520, SE-372 25 Ronneby, Blekinge, Sweden
[b] Ericsson AB, Box 518, SE-371 23, Sweden

ARTICLE INFO

Article history:
Received 13 September 2009
Received in revised form 7 February 2010
Accepted 7 February 2010
Available online 16 February 2010

Keywords:
Lean software development
Software process improvement
Quality improvement paradigm

ABSTRACT

Software process improvement methods help to continuously refine and adjust the software process to improve its performance (e.g., in terms of lead-time, quality of the software product, reduction of change requests, and so forth). Lean software development propagates two important principles that help process improvement, namely identification of waste in the process and considering interactions between the individual parts of the software process from an end-to-end perspective. A large shift of thinking about the own way of working is often required to adopt lean. One of the potential main sources of failure is to try to make a too large shift about the ways of working at once. Therefore, the change to lean has to be done in a continuous and incremental way. In response to this we propose a novel approach to bring together the quality improvement paradigm and lean software development practices, the approach being called Software Process Improvement through the Lean Measurement (SPI-LEAM) Method. The method allows to assess the performance of the development process and take continuous actions to arrive at a more lean software process over time. The method is under implementation in industry and an initial evaluation of the method has been performed.

1. Introduction

Software process improvement aims at making the software process more efficient and increasing product quality by continuous assessment and adjustment of the process. For this several process improvement frameworks have been proposed, including the Capability Maturity Model Integration (CMMI) (CMMI-Product-Team, 2006) and the Quality Improvement Paradigm (QIP) (Basili, 1985; Basili and Green, 1994). These are high level frameworks providing guidance what to do, but not how the actual implementation should look like. The Software Process Improvement through the Lean Measurement (SPI-LEAM) method integrates the software quality improvement paradigm with lean software development principles. That is, it describes a novel way of how to implement lean principles through measurement in order to initiate software process improvements.

The overall goal of lean development is to achieve a continuous and smooth flow of production with maximum flexibility and minimum waste in the process. All activities and work products that do not contribute to the customer value are considered waste. Identifying and removing waste helps to focus more on the value creating activities (Cumbo et al., 2006; Womack and Jones, 2003). The idea of focusing on waste was initially implemented in the automotive domain at Toyota (Shingo, 1981) identifying seven types of waste. The types of waste have been translated to software engineering into extra processes, extra features, partially done work (inventory), task switching, waiting, motion, and defects (Poppendieck and Poppendieck, 2003). Partially done work (or inventory) is specifically critical (Middleton, 2001). The reason for inventory being a problem is not that software artifacts take a lot of space in stock, but:

- Inventory hides defects that are thus discovered late in the process (Middleton, 2001).
- Time has been spent on artifacts in the inventory (e.g., reviewing of requirements) and due to change in the context the requirements become obsolete and thus the work done on them useless (Petersen et al., 2009).
- Inventory impacts other wastes. For example, a high level of inventory causes waiting times. Formally this is the case in waterfall development as designers have to wait until the whole requirements document has been approved (Petersen et al., 2009). Long waiting times bare the risk of completed work to become obsolete. Furthermore, high inventory in requirements engineering can be due to that a high number of extra features have been defined.
- Inventory slows down the whole development process. Consider the example of a highway, if the highway is overloaded with cars then the traffic moves slowly.

* Corresponding author. Address: School of Computing, Blekinge Institute of Technology, Box 520, SE-372 25, Sweden. Tel.: +46 10 7140572.
E-mail addresses: kai.petersen@bth.se, kai.petersen@ericsson.com (K. Petersen), claes.wohlin@bth.se (C. Wohlin).
URLs: http://www.bth.se/besq, http://www.ericsson.com (K. Petersen), http://www.bth.se/besq (C. Wohlin).

0164-1212/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.02.005

- High inventory causes stress in the organization (Morgan, 1998).

Lean manufacturing has drastically increased the efficiency of product development and the quality of products in manufacturing (see for example, Cumbo et al., 2006). When implemented in software development lean led to similar effects (cf. Morgan, 1998; Middleton et al., 2005). Even though lean principles are very promising for software development, the introduction of lean development is very hard to achieve as it requires a large shift in thinking about software processes. Therefore, an attempt to change the whole organization at once often leads to failure. This has been encountered when using lean in manufacturing (Pascale, 1990) and software development (Middleton, 2001).

To avoid the risk of failure when introducing lean our method helps the organization to arrive at a lean software process incrementally through continuous improvements. The method relies on the measurement of different inventories as well as the combined analysis of inventory measurements. The focus on inventory measurement is motivated by the problems caused by inventories discussed earlier. Furthermore, inventories also show the absence of lean practices and thus can be used as support when arguing for the introduction of the principles. In the analysis of the inventories a system thinking method is proposed as lean thinking requires a holistic view to find the real cause of problems. That is, not only single parts of the development process are considered, but the impact of problems (or improvement initiatives) on the overall process have to be taken into consideration.

Initial feedback on SPI-LEAM was given from two software process improvement representatives at Ericsson AB (see Gorschek and Wohlin, 2006). The objective was to solicit early feedback on the main assumptions and steps of SPI-LEAM from the company, which needs triggered the development of the method.

The remainder of the paper is structured as follows: Section 2 presents the related work on lean software development in general and measurement for lean software development in particular. Section 3 presents the Software Process Improvement through the Lean Measurement (SPI-LEAM) Framework. Section 4 presents a preliminary evaluation of the method. Section 5 discusses the proposed method with focus on comparison to related work, practical implications, and research implications. Section 6 concludes the paper.

2. Related work

2.1. Lean in software engineering

Middleton (2001) conducted two industrial case studies on lean implementation in software engineering, and the research method used was action research. The company allocated resources of developers working in two different teams, one with experienced developers (case A) and one with less experienced developers (case B). The responses from the participants was that initially the work is frustrating as errors become visible almost immediately and are returned in the beginning. In the long run though the number of errors dropped dramatically. After the use of the lean method the teams were not able to sustain the lean method due to organizational hierarchy, traditional promotion patterns, and the fear of forcing errors into the open.

Another case study by Middleton et al. (2005) studied a company practicing lean in their daily work for 2 years. They found that the company had many steps in the process not being value-adding activities. A survey among people in the company showed that the majority supports lean ideas and thinks they can be applied to software engineering. Only a minority (10%) is not convinced of the benefits of lean software development. Statistics collected at the company show a 25% gain in productivity, schedule slippage was reduced to 4 weeks from previously months or years, and time for defect fixing was reduced by 65–80%. The customer response on the product released using lean development was overwhelmingly positive.

Perera and Fernando (2007) compared an agile process with a hybrid process of agile and lean in an experiment involving 10 student projects. One half of the projects was used as a control group applying agile processes. A detailed description of how the processes differ and which practices are actually used was not been provided. The outcome is that the hybrid approach produces more lines of code and thus is more productive. Regarding quality, early in development more defects are discovered with the hybrid process, but the opposite trend can be found in later phases, which confirms the findings in Middleton (2001).

Parnell-Klabo (2006) followed the introduction of lean and documented lessons learned from the introduction. The major obstacles in moving from agile are to obtain open office space to locate teams together, gain executive support, and training and informing people to reduce resistance of change. After successfully changing with the help of training workshops and use of pilot projects positive results have been obtained. The lead-time for delivery has been decreased by 40–50%. Besides having training workshops and pilots sitting together in open office-landscapes and having good measures to quantify the benefits of improvements are key.

2.2. Lean manufacturing and lean product development

Lean principles initially focused on the manufacturing and production process and the elimination of waste within these processes that does not contribute to the creation of customer value. Morgan and Liker (2006) point out that today competitive advantage cannot be achieved by lean manufacturing alone. In fact most automotive companies have implemented the lean manufacturing principles and the gap in performance between them is closing. In consequence lean needs to be extended to lean product development, not only focusing on the manufacturing/production process. This trend is referred to as lean product development which requires the integration of design, manufacturing, finance, human resource management, and purchasing for an overall product (Morgan and Liker, 2006). Results of lean product development are more interesting for software engineering than the pure manufacturing part as the success of software development highly depends on an integrative view as well (requirements, design and architecture, motivated teams, etc.), and at the same time has a strong product focus.

Morgan and Liker (2006) identified that inventory is influenced by the following causes: batching (large hand-overs of, for example, requirements), process and arrival variation, and unsynchronized concurrent tasks. The causes also have a negative effect on other wastes: batching leads to overproduction; process and arrival variation leads to overproduction and waiting; and unsynchronized tasks lead to waiting. Thus, quantifying inventory aids in detecting the absence of lean principles and can be mapped to root causes. As Morgan and Liker (2006) point out their list of causes is not complete. Hence, it is important to identify the causes for waste after detecting it (e.g., in form of inventories piling up).

Karlsson and Ahlströhm (2009) identified hinders and supporting factors when introducing lean production in a company in an industrial study. The major hinders are: (1) it is not easy to create a cross-functional focus as people feel loyal to their function; (2) simultaneous engineering is challenging when coming from sequential work-processes; (3) there are difficulties in coordinating projects as people have problems understanding other work-