린(Lean) 개념을 소프트웨어 개발 방법에 적용하기 위한 사례 연구: 낭비 제거의 가시화를 중심으로

How to Implement '*Lean*' Principles into Software Development Practice?: Visualization of Delays and Defects

황 순 삼 (Soonsam Hwang) 중앙대학교 경영학과, 교신저자
김 성 근 (Sung K. Kim) 중앙대학교 경영학과

요 약

소프트웨어 산업은 아직도 뿌리깊은 문제에 시달리고 있다. 어떻게 보면, 제조업과 같은 보다 성숙한 산업의 모범사례로부터 뭔가를 배워야 할 지 모른다. 제조업에서의 '린' 원칙을 소프트웨어 개발에 적용하는 것도 한 방안일 수 있다. 소프트웨어공학 문헌에서 이런 '린' 원칙은 시도해 볼만 한 것으로 언급되었다. 그러나 이들 원칙을 실제 어떻게 적용할 수 있을 가에 대한 구체적 방안을 제시한 연구는 별로 없었다. 본 연구는 '린' 원칙을 소프트웨어 개발에 적용하는 방안을 제시하고자 한다. 이 방안의 핵심은 낭비제거라는 린의 관점을 구체화하기 위하여 프로젝트에서의 리드 타임과 결함을 누적 흐름도(Cumulative Flow Diagram)을 통해 관리하는 방법이다. 또한 이 방안을 실제 프로젝트 사례에 적용함으로써 타당성을 검증하고 적용 방법에 대한 이해를 돕고자 하였다.

키워드 : 린 소프트웨어, 애자일, 스크럼, 누적 흐름도, 소프트웨어 개발

I. Introduction

Despite its half-century history, software industry still has many deep-seated problems. It lacks quality and manageability. Why is the industry still immature? Wouldn't there be something to learn from other mature industry? It was quite natural to look for best practices in manufacturing domain.

It is the concept of 'lean' that has attracted the most attention. The lean software development was first introduced in 2003, which is defined as translation of 'lean manufacturing' principles to the software development domain. In their book Poppendieck and Poppendieck (2003) indicated that these principles are applicable to software community as well.

There have been some researches for adopting lean principles into software practices. Sowmyan (1998) tried to verify the feasibility of lean software development. He insisted that software reusability is core in implementing the lean software as it helps

† 이 논문은 2011년도 중앙대학교 학술연구비 지원에 의한 것임.

to achieve improved quality and reduced time and cost. Poppendieck (2002) argued that underlying principles of lean provide a good framework for improving software development. However, these studies did not show specifically how to implement the lean principles into software development.

Another approach was proposed, which combines lean principles with a software development process. The process selected was *agile* which is a methodology based on iterative and incremental development. Poppendieck and Poppendieck (2003) indicated that lean has a series of good thinking, but it is hard to be applied into software development directly because lean lacks practical tools or techniques. Beyond their explanation that lean and agile are supplementary each other and, when used together, would produce good results, there was an attempt to prove the feasibility of connecting agile and lean through pilot project (Parnell-Klabo, 2006). Parnell-Klabo (2006) showed the rationale of applying lean principles in conjunction with agile. It led to 40% reduction in project duration and 10% drop in costs compared to waterfall baseline estimates while resulting in impressive increases in product quality. Though Parnell-Klabo's pilot effort is valuable in some aspects, it still has limitations. It could not obviously show in specific how agile exactly worked to realize lean software.

In this paper, we attempt to present a method of implementing lean principles into software development practice. Our approach mainly deals with the elimination of waste in software development. Key decisions were what to manage in order to reduce wastes and how to assist the managing of software development process. To this end we analyzed the main sources of wastes in software development and looked for a visualization tool by which the software development is better managed. In our approach project lead time and software defects are regarded as major targets of management and are visualized using Cumulative Flow Diagram. Further, we applied this method with actual project. The result reported here confirms that agile is positively effective on reducing defects.

From the following section, this paper will explore what the lean and lean software are about and the likelihood of connecting agile and lean as discussing their complementary relationship. It will then analyze the case study to show how effectively agile plays its role in implementing lean thinking and finally conclude with brief summary, limitations and future researches.

Ⅱ. Research Background

2.1 Lean and lean software

In this section, more detailed background of lean emergence and its core values are going to be discussed in order to see how important lean is in developing software in terms of waste elimination. Toyota Production System which is currently called as lean was introduced after the late 1940's in Japan. At that time, Japan was struggling with economic downturn as lost in the World War II. So, people could not afford to pay high costs for cars and even the market was not big enough to allow mass production which is appropriated for cutting costs down. Facing with the situation, Toyota had to find a way to make cars in small quantities, but keep them as cheap as mass-produced cars. To cope with those challenges, the Toyota Production System (TPS) was emerged as a solution and its core idea was to maximize quality, at the same time minimize costs and lead-time by eliminating wastes (Kim and Park, 2007). The introduction of TPS concept was suc-

cessful and its success made many researchers pay attention to it. One of them, MIT research group presented its importance with newly coined term lean to the world (Kim and Park, 2007).

"Lean is a mindset, a way of thinking about how to deliver value to the customer more quickly by finding and eliminating waste" (Hibbs *et al.*, 2009). Wastes are anything that does not contribute to the customer value but impede to quality and productivity (Petersen and Wohlin, 2010). There are 7 major wastes; over-production, inventory, motion, delay, transportation, over-processing, and defects (Shigeo and Dillon, 1989). To remove these wastes, lean conducts a key activity known as 'Value Stream Map' which breaks down whole process into a map of its individual steps and configure which steps are value-added and which are not. It then focuses on getting rid of wastes found whereas improving value-added steps (Hibbs *et al.*, 2009). After all, what only remained after applying lean were steps, time and people that add value from the eyes of customers (Poppendieck, 2002).

Lean thinking has proven its value and been applied beyond the manufacturing environment (Parnell-Klabo, 2006). As mentioned above, Mary and Tom Poppendieck were firstly aware of the probability of lean software development (Middletion, 2001). Lean software development is not a development methodology per se, but it offers 7 principles that are applicable in any environment to improve software development. The 7 principles are eliminate waste, amplify learning, decide as late as possible, empower the team, build integrity in, see the whole. The Poppendiecks foresaw that software industry would gain same results of what lean benefited in production as it helped productivity more than double compared to industry norms while errors and anomalies dropped simultaneously.

Sowmyan (1998) suggested that lean thinking is applicable to software development process. Hamiltion (1999) also stated, application of lean to software development reduces cycle time and improve overall quality. Morgan (1998) added that clearly implemented lean software causes dramatic improvements. But, the problem was lean principles in production could not be always applicable to software. Due to this fact, Poppendieck refined lean principles used in manufacturing and declared re-defined 4 basic principles which are most relevant to software development; eliminate waste, respect people, defer commitment and optimize the whole (Poppendieck, 2002).

Among the lean principles, Tierney (1993) contended 'waste elimination' is the heart of lean in software development. Wirth (1995) said, today's software is getting 'fat' so in the end results in 'complexity.' 'Complexity' is cholesterol which tacitly killing 'growth' and 'profit' in software development (Poppendieck and Poppendieck, 2007). It means the process of software development involves lots of unnecessary or non-value added activities and those wastes should be removed so as to avoid complexity. Wastes in software development include extra features, requirements, extra steps, finding information, defects, waiting, and handoffs (Poppendieck, 2002). Especially, in software, defects are one of the most harmful wastes in terms of product quality, costs, and speed. This is because undiscovered and not corrected defects will ruin project quality (Tierney, 1993). In addition, it will become far more expensive to fix defects as a project moves on (Middleton, 2001). As a result, waste elimination, particularly getting rid of defects is an important factor maximizing the value of the product finally delivered- the ultimate goal of lean.

2.2 Lean Software and Agile

So far, it has been discussed what the lean and

its core values are about, as well as the justification of applying lean to developing software. From this part, agile is going to be introduced in order to discuss the relationship with lean and then expected benefits that the application of connected agile and lean brings up to software development will be explored through existing research review. It will then address the ultimate meaning of this paper by pointing out limitations of existing studies.

Agile is a newly emerged approach on organizing software development. Agile methodology was disclosed to public by Agile Manifesto in 2001. The term 'agile' encapsulates diverse development methodologies that have the characteristics of so-called 'light weight methodology.' It is based on the concept 'Agility.' Agility applied to software development has the objective to embrace 'changes' through rapid and flexible response to them (Larman, 2004). "Agile development methods apply time-boxed iterative and evolutionary development, adaptive planning, promote evolutionary delivery, and include other values and practices" (Larman, 2004). This agile approach based on iterative and incremental improvements on developing software enhances value of products and ultimately user's satisfaction. Recognizing the usefulness of agile, there have been continuous efforts to investigate agile in relation with lean and their synergetic effects.

'The Art of Lean Software Development' (Hibbs *et al.*, 2009) explained the relationship between lean and agile by discussing common and different features they share. Above all, lean and agile have same goals; maximizing productivity of software development while optimizing the quality perceived by customers. Both not only positively think the changes in requirements involved over the whole course of the project, but put the highest value on satisfying the customer's real needs with delivered software. Each takes different view; agile has a narrower focus while lean prefers to take holistic view on software so that lean views agile as supporting practices of lean software development. Another prime difference is that lean more concerns about waste elimination while agile focuses on how collaboratively to work with customers and how rapidly to deliver software to them.

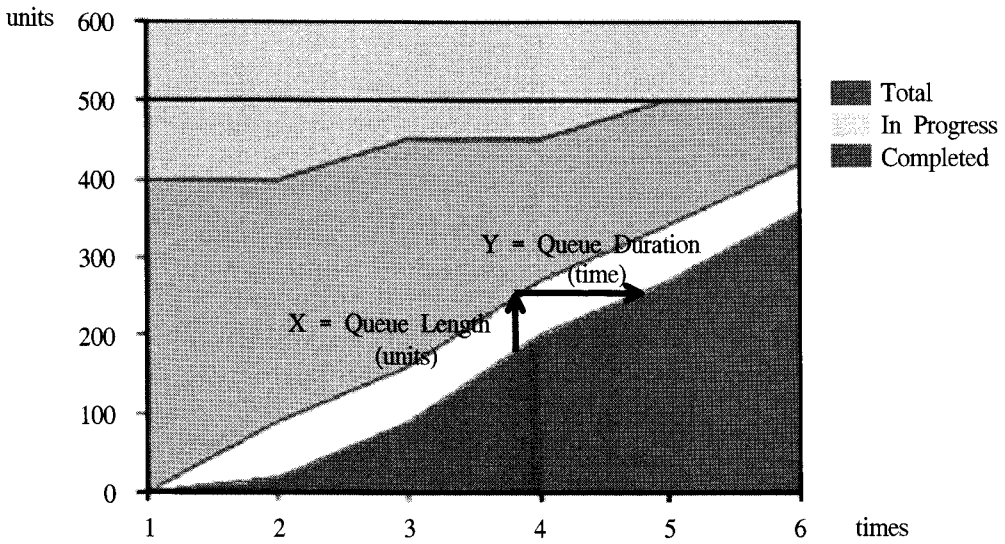
Focusing attention on their close relationship, efforts to explain the likelihood and appropriateness of agile-lean software development have continued. 'Lean Software Development; An Agile Toolkit' (Poppendieck and Poppendieck, 2003) suggested plenty of agile tools and practices required to develop lean software. It said lean has series of good thinking, but it is hardly applied to software development in direct ways since lean is lack of practical tools or techniques. Agile, however, has more specific tools or practices so that can help for implementing lean idea into software development. Beyond the explanation of their supplementary relationship and expected merits, there was an attempt to prove the feasibility of connecting agile and lean and synergetic work of them through pilot project (Parnell-Klabo, 2006). To empirically verify it, a pilot project targeted acquisition division at a large financial service company was performed. It firstly carried lean activity eliminating wastes out environment was better optimized with the 50% reduction in tasks. It then adopted agile technique, Scrum, to see how agile could be more effective in the lean applied environment. The result showed duration of software development and testing was decreased by 40% and resource costs were also dropped by 15%.

Even though the expectation of agile-lean software development is getting higher, only few researches have studied the field of connecting agile

and lean for the purpose of software development. Furthermore, empirical evidences or case studies backing it up are insufficient. In this situation, The Parnell-Klabo's experimental pilot is a valuable, but has issues. As discussed already, it did not present detailed Scrum role in achieving such outcomes. Moreover, the improvements in cost, duration, and quality are not aligned with the core value that lean pursues. In software development, what lean puts importance on is 'waste elimination.' However, those outcomes accomplished by the pilot attempt are the common goal discussed for a while from the past, not from the lens of lean. So, it is hard to understand the pilot success in the way of reflecting the lean idea.

This paper aims to show the effects of agile, especially Scrum on software development from the eyes of lean. Scrum manages a project with brief daily progress updates and iterative sprints development cycle. The reasons why Scrum has been chosen and used are followings. Scrum is the most applicable to any business environment with high variability and complexity (Parnell-Klabo, 2006). "Scrum is also simple-its process, practices, artifacts and rules are few, straightforward, and easy to learn" (Schwaber, 2003). These two features, 'simplicity' and 'high applicability' are main features enabling Scrum to be a strong tool in agile and this is because Scrum was chosen.

How we could identify the Scrum effect in the way of realizing lean core idea? It can be figured out through the Scrum influences on eliminating WIP (Work in Process) and Lead time. WIP is number of works to do and lead time is the time which is likely to take to finish works in process (Griffiths, 2007) Therefore in order to make a lean process in software development, one approach could be to break down work in smaller chucks to get things faster through the development process. Also lead time is regarded as wastes in lean production because long lead time creates high level of variation, increases costs, and hinders quality improvements (George, 2002). The number of works in process is caused to long lead time. These WIP



〈Figure 1〉 Cumulative Flow Diagram

and Lead time are clearly visualized in CFD (Cumulative Flow Diagram). In the following CFD diagram, the yellow area contains all the work which started but not yet finished, that is 'Queue.' The vertical Y indicates WIP while horizontal X indicates Lead-time (Griffiths, 2007). Visibly displaying how WIP and Lead-time changes during a project life cycle, CFD enables a project team to easily recognize any stagnant bottlenecks and perform better for overall process improvement. Therefore, CFD can be used as an excellent tool for waste elimination. This is because this paper selects CFD as a visual aid of Scrum effects.

Ⅲ. Research Methodology

With the need of finding a way of experimenting lean thinking applied to software development, two main options were identified. One was designing experimental project with controlled environment. It would be more aligned with the classic research model, but it is fairly hard to be arranged. It also could not duplicate full details of real projects. The other one was to choose real projects then observe them by gathering data and analyzing. However, some difficulties lie in the choice of a suitable subject and objective data gathering. It was not possible to apply the classical model on a project of any meaningful size or duration for this research. The choice of here was to use real case project. There were some requirements to apply this case study. The first one was that agile needed to be introduced at enterprise level so that plenty of agile projects exist. In addition, whole process of the projects must adhere to agile process. The second one was that objective data for defects and lead time must be collected and analyzed with an automatic tool.

Based upon the analysis of the real case study, we aim to show agile could be a good start of developing software in the lean perspective. Software development from the eyes of lean implies that software is built in the manner of mirroring key lean principle-waste elimination, waste would be defect in software. In the following case study, it is seen that agile, in particular Scrum is much more suitable than Waterfall for defect reduction. While Waterfall makes errors to be accumulated during the project life time since it follows traditional step-by-step project development procedure, agile reduces errors by frequently detecting them through iterative and incremental process of project development. This agile effect is shown through gradual decreases of WIP and Lead time with agile tool, Scrum. But, this paper is not going to emphasize the absolute necessity of using Scrum for obtaining the effects of lean, but to discuss that agile tool, Scrum could be effectively used for lean software development. For better understanding, two different projects, one was developed with Scrum and another was done with waterfall are introduced and analyzed in comparison.

3.1 Case Study

We selected a global internet portal company which has adopted Scrum as a standardized method of managing projects and actively applied it to the number of projects in many years. To manage those Scrum projects, the company developed Scrum project management system based on Bugzilla-bug tracking software of Mozilla. It automatically produces a visual figure called 'burn down chart' describing work status going on work process. "The burndown chart is a graphical display that is used in a number of agile tracking tools to visually depict the remaining work required for a milestone to be ach-

ieved" (Cooke, 2010). In the side of managing a project, it is helpful to keep a team on track. However, in the aspect of lean, it is no longer effective since the way burn down chart works is not aligned with the goal of lean. It means that burn down chart only focusing on the current state of sprint or iteration makes hard to see all the works proceeded from the beginning at a glance. So, it makes difficult to identify delays or defects causing 'waste' in the process and control them. Consequently, to achieve the goal of lean 'waste elimination,' an alternative tool visualizing holistic picture of the process is needed. CFD (Cumulative Flow Diagram) is an appropriate tool to supplement the weaknesses of burn-down chart. It enables to see work progress and the time required to finish as work items accumulated from the beginning of a project. In this research, we use a CFD tool developed by the company for the purpose of managing WIP and Lead time.

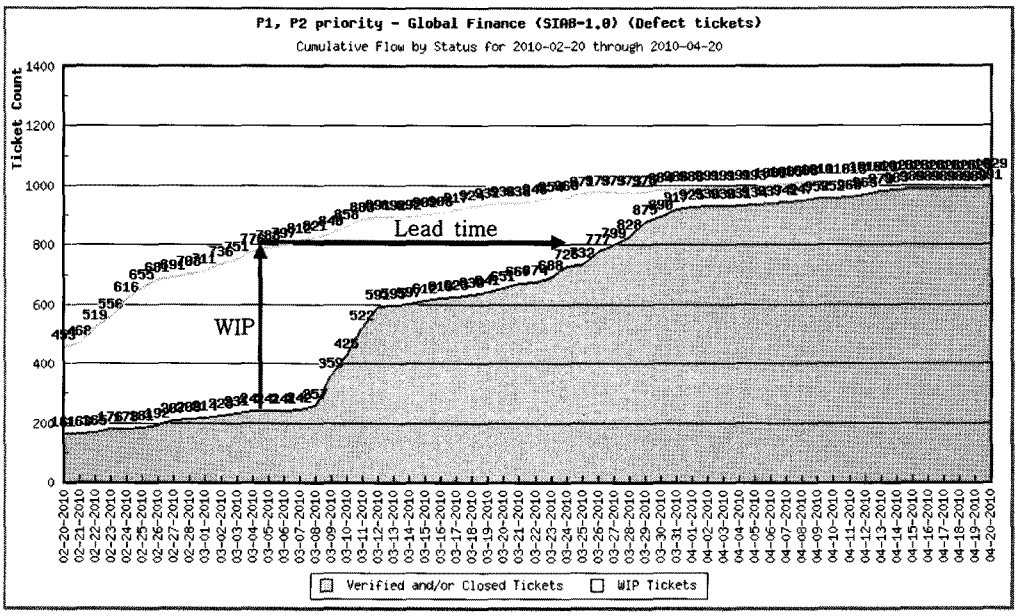
In order to see the Scrum effect, two real projects are chosen. The project named Global finance was completed under Waterfall approach whereas another project 'Global sports world cup' was completed with Scrum. We have carefully considered project intrinsic variables and chosen these 2 projects to meet an intention for this paper. First, developers in both projects used same programming language under same development environment. Second, the size and development period of the projects, projects complexities were similar. Third, each project team thoroughly conformed to their development methodologies; agile and Waterfall respectively. Both project teams use a bug tracking system, Bugzilla, which provides status data for reported defects. So we can analyze WIP and lead time of the two projects with CFD tool by collecting data from Bugzilla.

3.1.1 Case 1 (Waterfall Approach): Global Finance Site Development Project

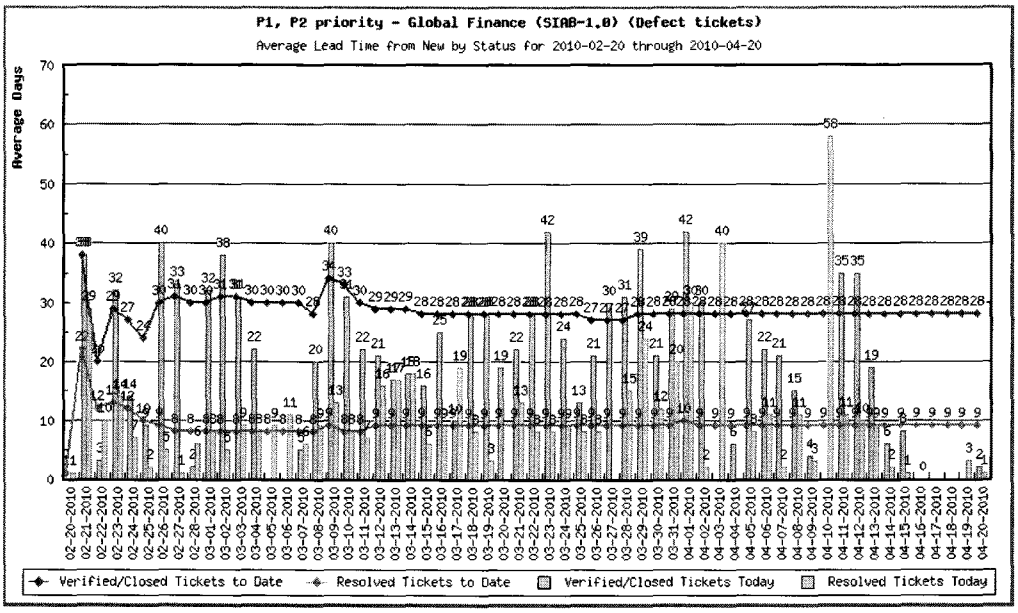
The objective of this project was to deliver a first "Site-In-A-Box" that can be used to replace existing finance sites over the world. A project team was divided into three sub teams which were responsible for 'Site development,' 'DB migration,' and 'Common infrastructure development' respectively. There were 15 team members. It followed the traditional project development cycle, 'Water fall.' The following CFD diagram is proportional of the whole project life time, 2 months test and release period. There were 2 times code freeze in total; first was 22 February 2010 when testing phase was initiated and second was 06 April, 2010. The following CFD is telling that there were sharp increases of WIP right after the first code freeze. The increased WIP then negatively affected to Lead- time. This phenomenon in which WIP and Lead time became larger made it difficult to stabilize the project for launch. In the end, as the <Figure 4> is showing, it took much time in order to resolve matters; average lead time used for both defect detection and correction was 28 days. As can be seen from the <Figure 2>, the number of defects sharply increased so that resulted in bottlenecks. Under waterfall way, most of defects occurred in design and development phases are usually detected in test phase. So, as can be seen from the <Figure 2>, the number of defects sharply increased in testing phase. In effect, during the process of fixing defects, it was found that it took longer time to handle defects which are simple but old.

3.1.2 Case 2 (Agile Approach): Global World Cup Project

The goal of this project was to open the site of



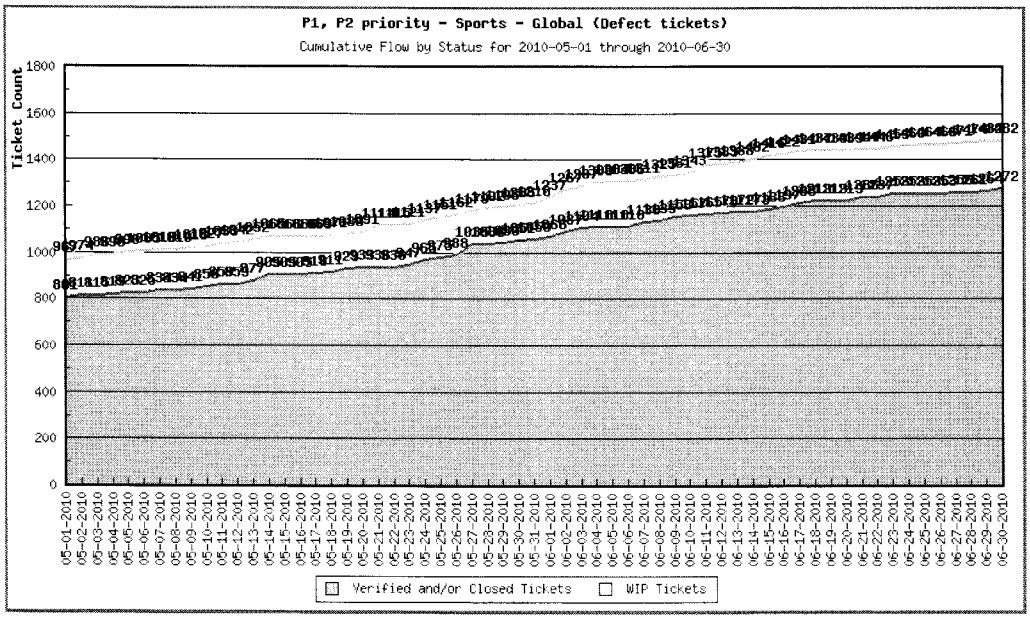
〈Figure 2〉 Cumulative Flow Diagram of Global Finance Project



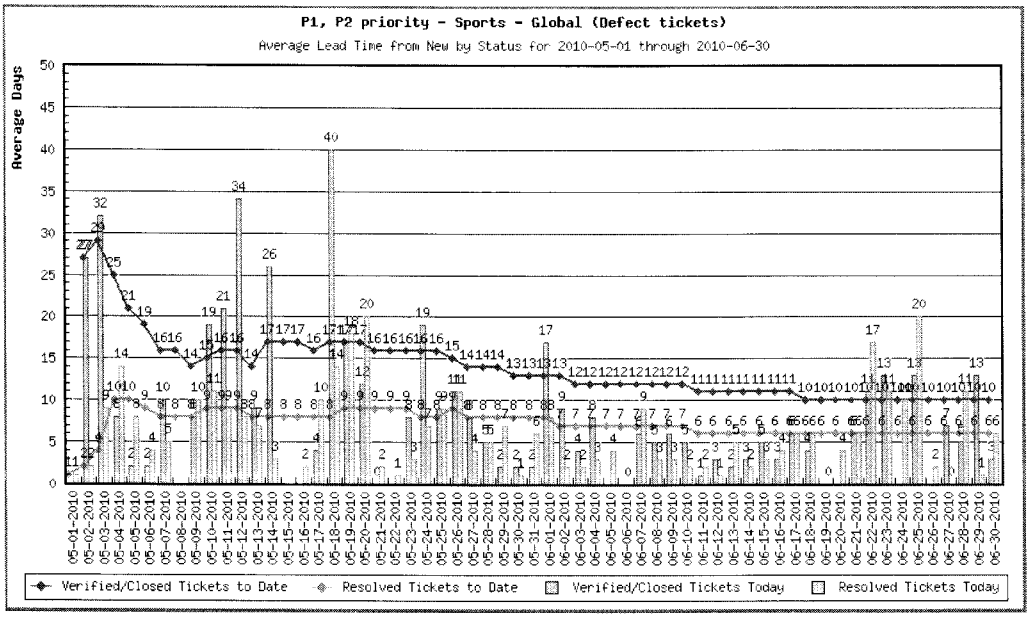
〈Figure 3〉 Average Lead Time Flow Diagram of Global Finance Project

FIFA World Cup held in June 2010 at every sports sites at the company. One team composed of total 12 project members was involved in this project.

As opposed to the above project, it was developed under Scrum and each Sprint was released by every 2 weeks. As the <Figure 4> describes, this project



〈Figure 4〉 Cumulative Flow Diagram of Global World Cup Project



〈Figure 5〉 Average Lead Time Flow Diagram of Global World Cup Project

developed with Scrum had always kept WIP at a consistent level and WIP was further improved depending upon the team competency. In addition, the <Figure 5> shows that repeatedly released Sprints were effective for lead time reduction as early detecting and fixing any defects happened. As you

can see, the lead time took around 12 days. It is fairly a lot shorter than the global finance project which took 28 days. A more noticeable point in the figure is the fact the average 12 days lead time had being decreased up to 10 days as the project moved more and more.

Through the real case of Global Sports World Cup project, it becomes evident that the project adopted Scrum was done in an effective manner on dropping WIP and Lead-time. How was that possible? In the Scrum framework, the project used to get monitored every 2 weeks in order to see what happened during the previous 2 weeks. By doing so, WIP (number of defects issued) was not much piled up, but resolved as soon as they occurred. Thus, Lead-time (time to spend on fixing defects) was steadily decreased. On the other hand, in the Waterfall based Global Finance Project, huge amounts of errors were detected and even stacked up since the team was not good enough to immediately correct them with limited team capability. So, increased WIP caused increases of Lead-time, eventually escalation of costs as it is more costly to resolve defects as time goes by. As a result, by observing and comparing the different results of the above projects, it is getting clear that Scrum adoption worked well in the way of implementing the core lean idea, waste elimination.

The core idea of developing lean software is to get rid of wastes occurred during the project process. Specially, defects cause costly reworks that directly lead to non-value added activities so are considered as the biggest waste factor (hibbs *et al.*, 2009). Therefore, it is very important to early detect and remove defects as soon as possible. Agile development was very helpful in early detection and fixation of defects since it is designed with short and iterative development process. The Global Sports World Cup project shows that defects accumulated throughout the project progress have been uniformly maintained. Moreover, the time required to remove defects has been gradually lessened. It would have positive learning effects to developers as detected defects are immediately reported to them.

Ⅳ. Conclusion

4.1 Brief Summary

The objective of this paper was to visibly prove the effects of Scrum on implementing core lean idea, 'waste elimination' in the whole course of software development. Through an empirical attempt carried out at a global internal portal company, it has figured out the fact that Scrum positively plays its role on reducing wastes by cutting WIP and Lead time down. As you can see from the case study, there was two-and-half times' differences between the highest level of WIP at each project; the maximum number of WIP in the Waterfall-oriented project was about 500 whereas WIP in the Scrum-adopted project was around 200. Additionally, the Lead-time came down more than 50% compared to the one in the project developed under Waterfall way. The decreases of WIP and Lead-time imply wastes in the project were lessened, after all the core lean idea was realized. That is, it shows agile could be a good tool for realizing lean software development which only contains theoretical principles.

This paper is worth in the following aspects. First of all, it has empirically approached to the topic of agile-lean software development which was mostly discussed in theoretical ways. In addition, this paper has value due to the fact that it presented the possibility that agile practices and tools can be lev-

ers on implementation of core lean principle via Scrum adoption. Also, the reduction of WIP and Lead-time in the research seems to positively impact on project cost, quality and productivity improvements. It is well matched with the proven goal of agile through lots of research or studies; project period reduction, quality improvements, and cost cutting. All the obtained findings in the research methodology and the way of performing the research seem to be a good guideline for future studies as they can be shared by others.

The well-conducted Scrum application, however, does not directly lead to the success of lean software development. A variety challenges such as smooth transformation of business environment exist for success and settlement of lean software development (Middletion, 2001). Lean principles have to be understood well by all members of a project, before implementing in a concrete, real-life situation.

4.2 Limitations and Future Research

This paper also faces challenges. At first, the case study showed mainly quantified outcomes with the use of CFD. But, the case study would have more useful if the process of utilizing agile tool for lean software development is more clearly demonstrated through team interview or close monitoring. Additionally, the case study itself may lack objectiveness for selecting projects in terms of control variables. In the research, we have considered project related variables (project development environment, size, duration, and project complexity) when selecting two projects for this case study although there were some differences between two projects in terms of project size and project complexity. Despite the fact that these intrinsic variables were not handled well, there would not be a significant issue, because the purpose of this case research lies not in scientific verification. Besides, few other case studies exist since companies rarely use agile at an enterprise level and manage lead time in quantitative ways.

From lean perspective, this paper has proposed agile is effective in eliminating the biggest waste factor, defects and lead time in developing software. But, it could not explain the validity of using agile in terms of actualization of other lean principles, but focused only on the first principle. In future study, we will endeavor to explore the role of agile in achieving rest of lean principles for software development.

References

Cooke, J., *Agile Principles Unleashed: Proven approaches for achieving real productivity gains in any organization*, IT Governance Publishing, the United Kingdom, 2010.

Dictionary.com, "Agility", Dictionary.com, LLC, 2011, online at http://dictionary.reference.com/browse/agility.

Fowler, M., "The New Methodology", MARTINE FOWLER, 2005, online at http://www.martinfowler.com/articles/newMethodology.html.

George, M., *Lean Six Sigma*, McGraw-Hill, New York, 2002.

Griffiths, M., "Creating and Interpreting Cumulative Flow Diagrams", 2007, online at http://www.leadinganswers.com

Hamilton, T., *A lean software engineering system for the Department of Defense*, Massachusetts Institute of Technology, 1999.

Hibbs, C., S. Jewett, and M. Sullivan, *The Art of Lean Software Development: A Practical and Incremental Approach*, O'Reilly Media, Inc., USA, 2009.

Kim, E. H. and J. H. Park, "Analysis of Lean Six Sigma Methodology for Postal Logistics", ETRI, 2007.

Larman, C., *Agile and iterative development: a manager's guide,* Pearson Education, Inc., Boston, 2004.

Middleton, P., "Lean Software Development: Two Case Studies", *Software Quality Journal*, pp. 241-252, 2001.

Morgan, T., *Lean manufacturing techniques applied to software development*, Massachusetts Institute of Technology, 1998.

Parnell-Klabo, E., "Introducing Lean Principles with Agile Practices at a Fortune 500 Company", *IEEE Computer Society*, 2006.

Petersen, K. and C. Wohlin, "Software process improvement through the Lean Measurement (SPI-'LEAN') method", *The Journal of Systems and Software*, 2010.

Poppendieck, M. and T. Poppendieck, *Implementing Lean Software Development: From Concept To Cash*, Addision-Wesley, 2007.

Poppendieck, M. and T. Poppendieck, *Lean Software Development: an agile toolkit*, Addison Wesley, Boston, 2003.

Poppendieck, M., "Principles of lean thinking", *Technical Report.* LLC, 2002.

Schwaber, K., *Agile project management with Scrum*, Microsoft press, Redmond, 2003.

Shalloway, A., G. Beaver, and J. Trott, *Lean-Agile Software Development: Achieving Enterprise Agility*, Addison Wesley, 2009.

Shingo, S. and A. Dillon, *A study of the Toyota production system from an industrial engineering viewpoint*, Productivity Press, New York, 1989.

Sowmyan, R., "Lean software development: is it feasible?", *Digital Avionics Systems Conference*, Vol.1, 1998.

Tierney, J., "Eradicating mistakes from your software process through Poke Yoke", *Software Quality Week*, Software Research Institute, pp. 300-307, 1993.

Wirth, N., "A Plea for Lean Software", *IEEE Computer*, 1995.

Information Systems Review
Volume 13 Number 1
April 2011

How to Implement '*Lean*' Principles into Software Development Practice?: Visualization of Delays and Defects

Soonsam Hwang* · Sung K. Kim*

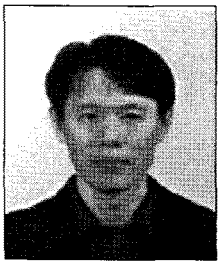
Abstract

Software industry still has many deep-seated problems. As a natural consequence, it may have to learn from best practices in more mature industry like manufacturing. An example is 'lean' software development which is defined as translation of 'lean manufacturing' principles to the software development domain. The principles include 'eliminate waste' and 'amplify learning.' It was much asserted that these principles are worth applying. Not much study, however, was done on how to practically implement these principles into software development practice. In this study we attempt to present a method in which project lead time and software defects are regarded as major targets of management and are visualized using Cumulative Flow Diagram. We further applied this method on actual projects. The result confirms that agile is positively effective on reducing wastes.

Keywords: Lean Software, Agile, Scrum, Cumulative Flow Diagram, Software Development

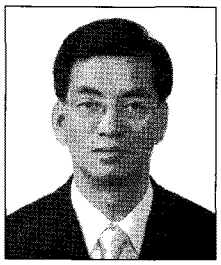
* Dept. of Business, Chung-Ang University

◐ 저 자 소 개 ◑



황 순 삼 (soonsam@empal.com)

중앙대학교 산업정보학과에서 학사, 동대학 경영학과에서 경영정보시스템으로 석, 박사 학위를 취득하였다. LG-EDS Systems, Handysoft, Yahoo Korea에서 근무하였다. 주요 관심분야는 개발방법론, 프로세스 개선, 소프트웨어 공학, 프로젝트 관리 등이다.



김 성 근 (sungkun@gmail.com)

미국 New York 대학교에서 정보시스템 전공으로 박사학위 취득 후 동 대학 전임강사를 거쳐 중앙대학교 경영학부 교수로 근무 중이다. 한국경영정보학회 회장 및 중앙대학교 전산정보처장을 역임했으며, 현재 국가정보화전략위원, 한국클라우드데이터센터포럼 의장, 한국 CIO포럼 대표간사로 활동 중이다. 관심분야는 Enterprise Architecture, IT Governance 등이다.

논문접수일 : 2011년 04월 12일 **게재확정일 : 2011년 04월 26일**
1차 수정일 : 2011년 04월 19일