UNIVERSITY OF GOTHENBURG

Bottlenecks in the Development Life Cycle of a Feature

A Case Study Conducted at Ericsson AB

Andrei Antanovich
Anastasia Sheyko
Brian Katumba

Bachelor of Science in Software Engineering and Management Thesis
Report No. 2010:012
ISSN: 1651-4769

University of Gothenburg
Department of Applied Information Technology
Gothenburg, Sweden, May 2010

Bottlenecks in Development Life Cycle of a Feature – A Case Study Conducted at Ericsson AB

Brian Katumba, Andrei Antanovich, Anastasia Sheyko

***Abstract*— Increase in lead time of software projects has been mainly attributed to long development cycles and changes in customer requirements. This has driven the development of a number of modern software development strategies to address the issue. Among them is streamline development - an in-house development framework at Ericsson AB inspired by lean principles. In this paper we explore the development life cycle of a feature to identify the possible bottlenecks - a term used in lean development denoting interruptions, re-work or any activities that hinder the development process hence an increase in lead time. This paper is based on a case study carried out at one of the development unit at Ericsson AB. The results presented here are after a qualitative interview study with one cross function team using streamline development framework. Using lean as a theoretical base, the results show that: task switching, competence, delayed replies and feedbacks, limited follow ups of evaluations, long communication chains, limited knowledge on the feature usage, unclear understanding of the development process, and lack of documents describing the architecture for the team are some of the possible bottlenecks that can increase the lead time of a software project.**

***Index Terms*— streamline development, feature development, software process improvement, software bottlenecks, lean**

I. INTRODUCTION

THE tendency of shifting from hardware to software has increased the market share of software in the business environment (Genuchten, 2007). This is leading many organizations to join the software industry; however software development organizations are still suffering from the problem of long development life cycles as well as handling changes in the customer requirements (Holmström, n.d). These are causing late deliveries to market and sometimes delivering products which do not meet the customers expected requirements. This has called for the different techniques to improve software quality and to reduce the time to market of software development.

Recently, lean principles have gained momentum in addressing the problem of long development life cycles, rapidly changing customer requirements, slow feedback and emerging of new technologies in software development industry. The lean methodology through its core principles of Providing the highest customer value, Maximizing flow, Eliminating waste has a significant effect in addressing the above problems which is leading to increased productivity in software organizations and reduction of lead time (Mehta, et al., 2008). The introduction of lean process principles which originates from the Toyota model of production was first identified by Womack and Jones (Womack, *et al.*, 1991 and 1990; Womack and Jones 1996a; Womack and Jones 1996b), in their five year study on why Japan's automobile industry was doing better than the American automobile companies. They revealed that Japan was using fewer resources than American companies which possibly made their cost of production low. In the same report, they said Japan was lean on resources in that they minimized wastes in the consumption of resources to maximize in their production without increasing the costs (Womack, *et al.*, 1991 & 1990; Womack & Jones 1996a; Womack & Jones 1996b).

The Software development industry has picked upon a few concepts from the Toyota model of production (Womack et al. 1990) hence the emergency of lean software process. It addresses the principle of reducing the development time by amplified learning, delay commitment, deliver fast, empower the team, build integrity in, see the whole, removing non-value adding wastes in the process (Poppendieck & Poppendieck 2003; Mehta, et al., 2008; Shalloway, et al., 2010). The principles of lean can be used as one of the best practice a company can acquire to improve its development process by reducing wastes as well as delivering fast (Shalloway, et al., 2010).

Based on a case study, this paper explores the development life cycle of a feature in one development team at Ericsson AB – one of the world's leading providers of telecommunication and data communication systems. At Ericsson AB, development teams use a process called 'streamline development (SD)' which is a process developed at Ericsson inspired by lean principles. The development team in this study uses the streamline process in order to reduce lead time and maximize the end-to-end flow so that the customer, in the end, will receive fast delivery of high-quality software products. To achieve this, each development team focuses on one feature at the time, i.e. a distinguished characteristic of a

software item (IEEE Std. 829-1998). Within the framework of 'streamline development' each development team is responsible for developing the feature in the shortest time possible and with as few interruptions as possible.
This research investigates what is involved in developing a feature, i.e. what is involved in the development life cycle of a feature. By interviewing the members in one development team we will identify 'bottlenecks' - a term used in lean development denoting interruptions, re-work or any activities that hinder the smooth end-to-end flow that is strived for to reduce lead time.

The question guiding this research is: What are the possible bottlenecks in the development life cycle of a feature? This question will be approached by taking a look at the different stakeholders involved in the development process, identifying the practices the feature is exposed for, identifying the queues i.e. bottlenecks and the milestones and activities during the process.
By answering this question, it will be possible to identify the possible bottlenecks in the Ericsson's development process, which will be used in the future to improve its end-to-end flow by removing queues and updating practices which will result into higher customer value as advocated in lean software development.

II. THEORETICAL BACKGROUND

A. Lean manufacturing

Since the early days of civilization humans were concerned with optimizing efficiency and decreasing waste to achieve better end results. Among theories aimed towards efficiency improvement are time and motion study, Taylorism and Fordism (Robins, et al., 2003; Kanigel, 1999; Tolliday & Zeitlin 1987). In the mid of 1940s, Toyota Motor Company stood in need of increasing production efficiency in order to stay competitive. At that time American automotive industry companies were approximately nine times more productive than Toyota Motor Company (Ohno, 1988). In order to find more efficient production ways Toyota looked at American method of production which was based on traditional thinking of mass production (Wu and Wee, 2009; Ohno, 1988). However this approach was not appropriate to Toyota because of its demand constraints (Ohno, 1988; Harvey, 2004). Using Fords theories, a new methodology for cars production was created and named Toyota Production System (TPS). Figure 1 shows 14 fundamental principles of TPS, divided into four groups by Liker (2004), that have to be considered to achieve sustainable and effective performance.
As shown on the figure 1 TPS foundation is long-term philosophy, which assumes prioritization of long term benefits over short term benefits. The goal of TPS is the "absolute elimination of waste" (Ohno, 1988; Towill, 2006; Wu and Wee 2009), where waste is anything that does not add customer value. According to Liker (2004), results that are not waste in TPS can be achieved by working in a process that follows certain principles. Principles that have to be followed in the development process to achieve waste elimination are: continuous flow, adding value from demand in order to avoid overproduction, eliminating overburden of process participants, preventing quality problems as soon as detected, using stable methods to maintain the predictability, using robust technology that support peoples' work, using visual control to highlight problems.

Two pillars of TPS Management philosophy are 'Respect for people' and 'Continuous improvement' (Larman & Vodde 2008; Towill, 2006). The former assumes that to add value to the organization respect for employees, partners and customers is required. The latter means that organization adopting TPS should create a culture of continuous improvement, by allowing developers to experiment, repeat and learn.

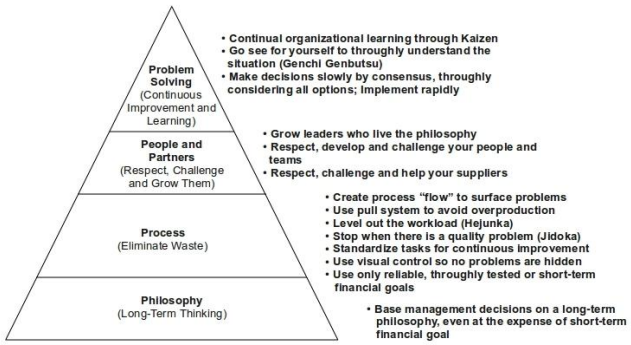

Figure 1: Model of the TPS (Liker, 2004)

B. Lean software development

TPS was formally documented around 1970 in Japanese and 1977 in English, however Western industries did not show interest in it until the oil crises, when imports from Japan started to threaten domestic manufacturers (Holweg, 2006). In 1990 as a result of International Motor Vehicle Program, with the goal of studying the position of automobile industry in the world economy, term 'lean' was introduced in the book 'The Machine that Changed the World' by Womack, Jones and Roos (Womack**,** et al., 1990; Harvey, 2004). This books sales remained slow for a year until the building crisis in Detroit in 1991 (Holweg, 2006).

Similarly in software development industry increasing globalization has brought out competitors with less costly development resources, amplifying cost pressure (Harvey, 2004). Software systems growing more complex and

development organizations becoming larger require development process improvements (Berry, 2003; Everett, et al., 2009). In a need for higher quality, lower cost, shorter delivery time of software, development organizations are looking for a process with regard to these characteristics in order to remain competitive (Harter, et al., 2000).

At the same time among benefits of implementing lean are lead time reduction, productivity increase and quality improvement (Kilpatrick, 2003). These are subsequently driving down the cost. Moreover lean proved to be universal and has been applied in such fields as health care, logistics and accounting.(Zidel, 2006; Baundin, 2004; Maskel & Baggaley 2003).

Despite uniqueness of lean, to apply it in particular field its ideas adoption to the new field is needed. In 2002 four lean manufacturing principles were translated by Mary Poppendieck into software development area (Poppendieck, 2002). As a result of further lean translation, seven most relevant to software development principles have been defined. For this research seven principles of lean software development are presented under the four initial categories defined by Poppendieck in 2002 (Poppendieck, 2002).

1) Add nothing but value.

a) Eliminate waste.

In order to add nothing but value and find wastes in software development, Poppendieck (2002) adopted seven wastes of manufacturing, identified by the creator of TPS (Ohno, 1988), into software development area. Overproduction, inventory, extra processing steps, motion, defects, waiting and transportation are all wastes found by Ohno (1988). According to Poppendieck (2002) features that lose its relevance and not used by the customer – extra features are the source of overproduction. Another waste is an inventory or partially done work. For instance unfinished or never used design documentation, ties up the resources and does not bring customer value (Poppendieck, 2002). Transportation is presented in software development in form of task switching. Searching for information process is a motion. Production of documentation with little use is an example of extra processing steps. Waiting and delays prevent customer from realizing what is valuable. The later customer gets software with necessary functionalities, the later he realizes that there are no need in some of previously requested .Defects introduced in the initial stages of development require more complex fixes as time goes by, therefore defects which are not caught by testers are wastes in software development.

b) Build quality in.

Shalloway (2010) claims that in lean software development quality should be build in both the code and the process. In order to build quality in code lean approaches simultaneous production of code and tests for the software and coding according to standards (Hibs, et al., 2009; Shalloway, 2010). It minimizes occurrence of initial error related defects (Hibbs, et al., 2009). Among the ways to reduce defects impact is to find them as soon as they occur (Poppendieck & Poppendieck, 2003). Existence of tests subsequently prevents defects introduction during changes implementation. Another way to reduce defects is frequent integration of small parts. This is because smaller parts are less vulnerable to defects than huge amount of code. Quality in process can be build by defining acceptance test early in the development process; it improves developers understanding about requirements and final product.

2) Center on people who add value

a) Empower the team.

Each member of development should be able to influence the process, make decisions on what to do and take responsibilities for these decisions. Poppendieck (2003) says that people involved in the process are “better equipped to make decisions”, however they should be supported and guided by management.

b) Create knowledge.

Shalloway (2010) argues that software development is more a discovery process than a building process, therefore creating knowledge that continuously perfect the product is essential. One way to amplify knowledge is to allow developers to experiment and improve relying on feedback obtained from iterations of the process. Learning to understand the process in which the work is done should be encouraged, so that the process can be improved by its users.

3) Flow value from demand

a) Deliver early and often.

Often and short iterations and subsequently earlier feedback allow developers to learn more and to discover more often, therefore make more product improvements and lead to better quality. With early and often deliveries, value can be delivered to the customer early on the development stage. It increases customer satisfaction, eliminates such waste as unnecessary features. For instance after getting the most necessary features customer may realize that he or she does not need others. According to Shalloway (2010) early releases provide early revenue, it can cover later development costs.

b) Delay commitment.

Delay commitment assumes giving a response in the moment when no more information can become available, still not too late to cause cost increase because of delays (Harvey, 2004). The main idea behind deferring commitment, is to allow changes in development project without negative consequences. Among the ways to delay commitment are: starting to discuss requirements from the most important to the customer and crucial for design solutions, constraining the implementation of design patterns to only those features that are current.

4) 4. Optimize across organization

a) See the whole.

Instead of each step optimization, Shalloway (2010), suggests to focus on flow in its entirety. Sub-optimization may hide systems constructional errors, for instance measuring performance of working individuals may not reflect overall system performance, since problems may be due to the way the whole system works (Poppendieck, 2003). Problems that can arise in transition process in between phases, such as poorly written design documents or integration errors, are most probably to be undiscovered when sub-optimizing.

Thus, successful application of lean in automotive industry, logistics and accounting was followed by its translation and usage in the software development field. Lean ideas have been transferred and formed into seven main principles that should be considered during software development improvement. Waste elimination, respect for people, pulling value form demand and optimization across organization are fundamental to lean in software development (Poppendieck, 2002).

III. ERICSSON CASE: BACKGROUND AND RESEARCH METHODOLOGY

A. Streamline development

Ericsson is one of the world's leading providers of telecommunication and data communication systems among software development organizations. For the purpose of this research, a study was carried out at one development unit. The purpose of the study was to follow the life cycle of a feature in order to identify possible bottlenecks that can be hindering or slowing down the development process. One feature was chosen and investigated throughout the development life cycle stages. At this development unit the work is organized in cross function teams where one of these teams was interviewed by the researchers while developing the feature. The team interviewed is using streamline development framework which is an in house development process.

The introduction of streamline development framework was an inspiration from the core value of lean and agile (Holmström, n.d; Tomaszewski, et al., 2007) i.e. providing the highest customer value, maximizing flow and eliminating waste. In this Ericsson considered the following agile and lean principles;"(1) satisfying customer needs through iterative development and continuous delivery, (2) welcoming changing requirements, (3) short timescale frequent deliveries, (4) motivated individuals in project building and face-to-face communication, (5) simplicity and (6) teams involvement on the reflection of the development process" (Holmström, n.d).

Streamline development was developed and tailored to meet particular needs for Ericsson i.e. to address the problems they experienced when using traditional software development process. Traditional software development methods are characterized with long development life cycles resulting in late deliveries to customers, failure to deliver the actual customer until late in the process (MacCormack et al.2003, Tomaszewski et al. 2007, Holmström, n.d). In addition, Ericsson was finding it difficult to handle customer changing requirements since they were exposed to changing market demands due to long duration of the projects (Holmström, n.d). In case of any customer changes in the requirements it would result into high cost of handling requirement changes. This is because; it was hard to deal with changes of already implemented functions (Tomaszewski et al. 2007).

In order to deal with the fore mentioned problems, SD was developed as a process and a working framework. Its main objectives are; to reduce the time to market, having a flexible development set up, more research and development during the project, having a better mechanism for scope setting and planning of the projects, creating a possibility to re-prioritize

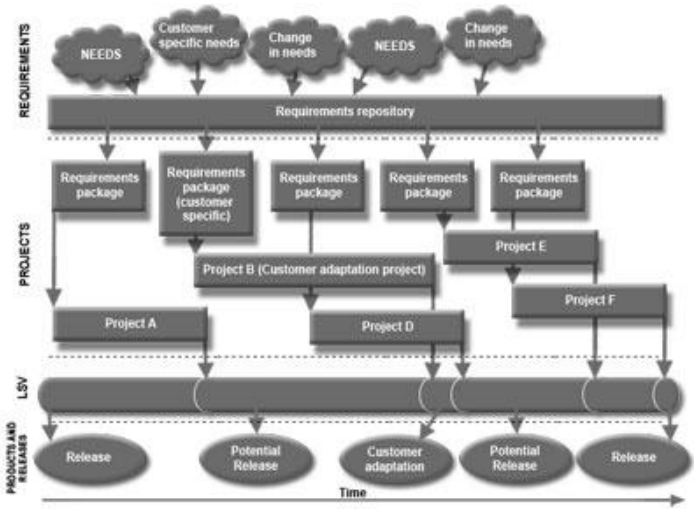

Figure 2. Streamline development

and re-plan at any time in the project and a greater focus on developing the right things.

As part of this study, the researchers followed the development life cycle of a feature which was developed using streamline development framework as shown in figure2.

In the development lifecycle of a feature using streamline development as shown in figure 2, a feature is exposed to different states and decisions from inception to release. As shown in figure 2, the project requirements are gradually and continually gathered in the requirement repository. These requirements come from either the customer, identified needs, change in needs, within the company or from standards. In the "early phase loop"; a phase of feature identification and prioritization in streamline development. The requirements are identified, prioritized and divided into requirement packages called feature projects as in figure 2. These projects last for an approximate of eight weeks. This phase is characterized by continuous and bi-weekly analysis of the features in order to know which feature needs to be prioritized first. This continuous analysis allows a possibility to re-prioritize and re-plan in the requirements, something that makes streamline development interesting as compare to the traditional development. Although it is the system owner responsible for this phase, the analysis is a combination of designers, testers, architects and PIDs inputs. The main goal here is to get which feature to start with without a consideration of resource constraints i.e. to come up with a feature priority list.

Once some features are prioritized and identified, they go into the "the activity and release planning loop" after "tentative" decision from the system owner. This phase is also characterized by continuous and bi-weekly analysis and coordination of input to be put in the projects as in figure 2. The main goal for this phase is to analyze and plan for the features, to get to know what resources they require i.e. what competence is needed for the feature, what tools. It is also in this phase the plan of integration and the releasing of the feature drawn. During this stage, decisions are made by the product activation team weather a feature should be implemented or not. It is the head of the product activation team responsible for this phase and is the one who gives a "GO" decision a term used in streamline development to mean, the feature is planned and there are resources for it to be executed or developed. In streamline development, a feature is given a "GO" decision when there is a development team also known as cross function team and there are available highly prioritized requirements (Tomaszewski et al. 2007).

When a feature is planned and given a GO decision, it goes to the" program execution phase". This is the phase when the feature is assigned to a specific cross function team. The cross function team (XFT) is composed of all the core competences ranging from design, architecture, test and system management (Holmström, n.d). It is in this phase when the actual execution or implementation of the feature occurs. After the GO decision from the product activation team, a "started" decision is made by the team leader to imply that the team has started working with feature. It is in this phase when the system manager introduces feature requirements to the rest of the team members after having participated earlier in the "the activity and release planning loop".

In the "program execution phase", a feature goes through two phases i.e. the feasibility and execution phase. The "feasibility phase" starts as soon as the "started" decision is made by the program execution manager (team leader). During the feasibility phase, the feature is further analyzed by the team and broken down into "anatomy" a term used in SD to denote the breaking down of the feature into smaller tasks. In the feasibility phase, the total test scope of the feature is also drawn and the roll-out plan (release) is made. The team studies the feature in details and confirms that they can implement it in a given time with the available resources. As a means of guarantee that team can implement the feature, a "commit" decision is made which is then followed by the "execution" phase. It is in the execution the testing and the designing of the feature is done. When the feature is fully implemented and fully tested, a "ready" decision is made, implying that the feature is ready to be integrated in the Latest System Version (LSV) as shown in figure 2. However in streamline development, there is always one version of the product, when a feature is integrated, it sets a basis for the next feature to be integrated in LSV (Tomaszewski et al. 2007).
After integration, a feature can be released to the customer or maintained in the LSV. While a feature is at this point in streamline development is said to be in the "release project" state.

All in all a feature is exposed to four states i.e. (1) the early phase loop where feature identification and prioritization is made and it is the system owner responsible for this phase. (2) The activity and release planning loop state where the planning and feature decision is, it is the head of the product activation team responsible. (3) The program execution state where the actual execution of the feature is done, here all the cross function team members participate, it's the team leader responsible for this phase. And (4) finally the project release state where the feature is integrated in the LSV or released to the customer, the release manager is responsible for it.

B. Feature team roles

For the purpose of this research, a study was carried out to know the possible bottlenecks a team may be exposed to during the program execution state. As earlier mention the program execution phase is where all the team members (cross-functional team) are involved. As in streamline development, a cross functional team has the opportunity to have all core competences such as design, test and system management with an architect, a team leader, system manager, function and system testers and designers as in figure 3. Each of these members has got core roles in the team as explained below. It is also important to note that the team takes full responsibility of the project during the program execution state. In this way, SD gives the teams more responsibility as well as freedom in their activities. This way of working creates an opportunity to share knowledge in the team and also the feedback loop in the team is shortened since all people are working together. The ability to re-prioritize and re-planning streamline development framework makes the team flexible to what is urgently required by the customer.

The role of the team leader is to coordinate the work of the team and to ensure that the development is going in the right direction. The team leader also checks that all team members are working according to the agreed process.

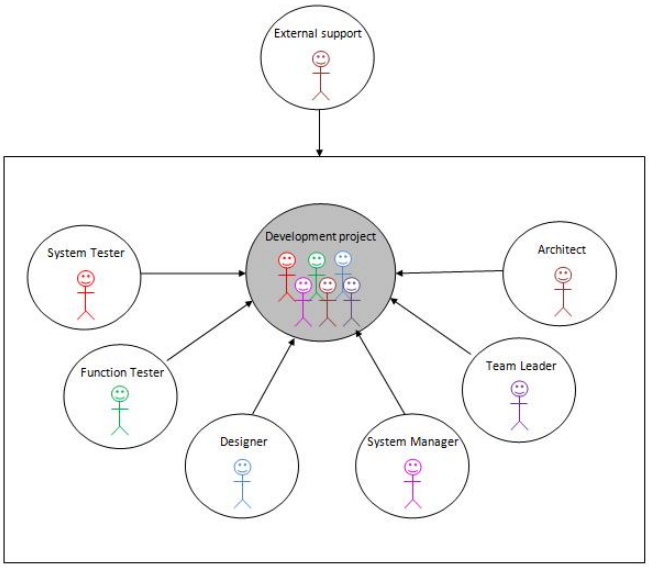

Figure 3: Feature team

Providing the team with the time plan which has to be followed or in case of necessary changes being modified according to the needs is also one of the core responsibilities of the team leader. The team leader also focus on removing all the barriers which can stop or negatively influence the team from achieving the goal and tries to find possible solutions to the problems.

System manager's main tasks include setting up the requirements for the feature which should be developed. Requirements either come from customers, standards or internally (for example implementing improvements to the functionality). However the system manager does not have a direct contact with the customers who ordered a feature, instead there is company's sales representative who is transferring customer's requirements to the system manager. When the system manager has got all the requirements, they have to be analyzed to make sure they are all testable and do not conflict with other features. In case of some uncertainties (vague requirements), all the questions are sent back to the internal sales representative who either has to clarify them or communicate back to the customer. And after the requirements are specified and clarified by the system manager, the customer has to approve it to ensure that there are no gaps in understanding the specifications and behavior of the feature.

On the other hand the architect's main role in SD process framework for this team is to help the team with support in architecture related questions. In this, the architect interacts with designers (as they are responsible for writing the code) and system managers (as requirements are main priorities in their work). Architect also helps to identify dependencies between the teams. If the team is implementing a feature which will influence another team's work, architect should identify possible dependencies and give support to the teams, mainly with guidelines on how to implement it in the correct way.

The main role of the designer in the project execution is to implement the feature in terms of actual code. As well as implementing the design test cases. As soon as the requirements are clear enough the designer should start the implementation.

Function testers in the beginning of the development process are responsible for validating the requirements specified by the system manager. They are also responsible for creating main flow and exceptional flow test cases. On the later stages when the implementation of the feature has started, function testers are interacting mostly with the designer. This kind of work style allows designers to react on the faults discovered by the function testers almost immediately. Even though designers do their own testing on the code they are focusing on testing boundary values (i.e. if the input is supposed to be in the range from 1 to 10, it is tested with 0 or 11). On the other hand, function testers; focus on doing tests inside those limits and trying to find faults.

The system tester is also another member of the team; the system tester studies the feature, write test analysis on what kinds of tests to conduct like capacity tests or hardware.
By analyzing what is involved in this way of working of the team, this study will identify the bottlenecks that can be hindering or slowing down the development process hence improve its end-to-end flow by implementing new metrics, removing queues and updating practices which will result into higher customer value as advocated in lean software development.

C. Research method

This research is approached by a case study method. A case study is "*an investigation of a contemporary phenomenon in depth and within a real life context, where the boundaries between the phenomenon and the context are unclear*" (Yin 2009; Walsham 1993). With the above definition, the study is based on real life experiences of the feature team using Streamline development framework to develop software features.
A case study method has been chosen because identifying bottlenecks in a process requires an in-depth investigation of the process from the beginning to the end to understand the underlying principles and the problem that may be involved. Similarly Yin (2009), suggests using a case study method, because it allows investigators to retain the holistic and meaningful characteristics of real life events such as individual life cycles, group behaviors, organizational and managerial processes which can augmented to fit the domain of this research. In addition, Dubé and Paré (2003) argue that, case study method is well suited for information system domain because it reveals patterns in an organizational environment. Although the question to this research does not explicitly talk about the patterns in the organization's environment, finding bottlenecks in the process requires researchers to find matching patterns in the data collected to give a clear analysis of the problems.

This study was carried out with the main aim of identifying the bottlenecks that may be hindering the feature development cycle from an idea until when the feature has passed the development stages. In this way, the study is expected to be used to improve the end-to-end flow by implementing new metrics, removing queues and updating practices in the development cycle.

Collecting data means to get the relevant information which can be used to address the research topic (Ricker T., et al. 1998). According to Yin (2009), there are six main sources of data: documentation, archival records, interviews, direct observations, participant observations and physical artefacts. For this particular study the interviews were taken as the primary sources of data and company presentations were the secondary source. In addition, company documentations and literature reviews helped in the validation of the results from the primary and secondary sources.
The information collected during the entire research served as evidence of the report's credibility. However on beforehand the un-disclosure agreement was signed in order to protect company's sensitive data from outside world.

D. Research process

a) Phase 1: Literature Review.

During the first phase of the research, the researchers identified the topic on which this research is based. After identifying the topic, it was important to get the context and positioning of the research in relation to similar studies. This called for a literature review on similar researches carried out. The literature review was based on articles and books which focus mainly on lean and agile principles. The purpose (see table 1) for this was to refine the research goals, to develop the realistic and relevant research question, select an appropriate method and identify potential validity threat of the research conclusion (Maxwell, 2005). In addition the literature review helped in the validation of the collected data in the interviews and presentations. This is argued by Bernd Heinrich (1984, pp. 151) that, even carefully collected results can be miss leading if the underlying context of assumptions is wrong".

b) Phase 2: Company Presentations

To get a clear understanding of the working process of the team members, the researchers were introduced to the working process of the feature team through company presentations. This took five weeks of which the researchers had a chance to spend at least one day at the research site each week. During this time, the researchers met the feature team members with a main aim of knowing their roles in the team as shown in table 1 and their perspective on the feature development cycle i.e. the architect presented the architecture of the feature, team leader presented how cross function teams work and the different roles of each team member, and system manager presented how feature requirements are handled. Still in the same five weeks, the researchers also met the Operational Development engineer who presented the streamline development framework and the process, methods and tools. Researchers were also got an overview of the technical product under development and its context, as well as were familiarized with the feature that was under development. It is important to note that on each presentation, there was time for interaction through questions and discussions.

Phases	*Activities*	*Data Sources/ Goals*
Phase 1	Literature Review	It was the first phase of the research; the researchers identified the topic on which this research was based. A literature review was carried out on similar researches. Especially on lean in software development. The purpose for this was to refine the research goals, to develop the realistic and relevant research question, select an appropriate method and identify potential validity threat of the research conclusion.
Phase 2	Company Presentations	Five presentations were carried out. The group was presented to Ericsson's product and the feature under investigation. During this time, Streamline development framework was introduced to the researchers by the Process engineer. The team which was developing the feature was also introduced to the researchers and the way how feature teams work by the Team leader. The way of handling feature requirements, and architecture of features was presented by the System manager and the operation architect respectively. The result of the presentations was to get insights about streamline development and ways of working in feature teams
Phase 3	Interview Studies	Six interviews were carried out with each lasting for approximately 1½ hours. The interviews were qualitative in nature and semi structured. The informants were; the team leader, the architect, designer, system tester and functional tester. All the interviewees belonged to one feature team. They were documented and transcribed.
Phase 4	Analysis	All the data collected was analyzed and groupped into four categories: organization of work and roles in a team, development process and phases, lean principles and reflection on SD work

Table 1. Research activities

c) Phase 3: Interview study

Interviews were considered to be the primary source of data of this study. Six interviews were carried out with each lasting for 1 hour and 30 minutes. Before the interviews it was essential to determine the right interviewees (Boyce and Neale, 2006) because interviews are perceived to be very effective when getting feedback or opinion on the developing processes from different perspectives, as well as activities, problems or other issues (Boyce and Neale, 2006). These were chosen based on the activities and roles in the process. Among the informants were: the architect, the team leader, system tester, designer, functional tester and the system manager. The reason for this, the researchers wanted to get different perspectives of the informants on the development life cycle of the feature. During the interviews the researchers shared the roles; one was conducting the interview while the other two were taking notes which were manually transcribed immediately after the interview. This structure was chosen because it's not easy to keep track of the interview at the same time take notes.

d) Phase 4: Analysis

Although literature reviews were taken first, it can be argued that data collected in this research emerged as an iterative process between theoretical conceptions and empirical

data (Klein and Myers 1999). This means that during empirical data collection, the researcher could review it in accordance to the lean principles as well as contacting the interviewees in case there was something which was not clear.

All the data collected was categorized into four categories i.e. 'organization work and roles in the team', 'development process and phases', 'lean principles' and 'reflection on streamline development framework'.These categories emerged during the interview process. The reason for this is that, knowing the organizational roles in the process would help the researchers understand the activities in the process during each phase and focusing on lean principles would further reinforce theoretical background on which this study is based on. It was also the idea of the researchers to have a category on reflection on streamline development for the purposes of knowing the team's percepective towards the process. Data gathered was used as an input in the analysis which however helped in answering the research question.

IV. RESULT

This chapter describes the results gathered from the interviews and presentations after Phase 3 (mentioned above), which will then be analyzed in order to get the answer to the research problem. To describe the results in the most efficient way it is categorized in the same way as interview questions:

- organization of work and roles in teams;
- development process and phases;
- lean principles;
- reflection on streamline software development;

The reason for having these categories is first to identify each member's role in a team, dependencies on each other and external factors. Second is to understand the development process which will allow mapping it and see the pattern in it.

Next two categories were created with the purpose to get the team members' understanding of lean and streamline software development. Each category includes perspectives of six roles in the team, i.e. team leader, system manager, operational architect, designer, system and functional testers' perspectives.

A. Organization of work and roles in teams

The idea of having cross functional teams is to provide each team with core competences which allow focusing on a specific feature and being responsible for it. The team which was investigated consists of: team leader, system manager, architect, designer, three function testers and system tester.

To improve the process flow in a team, the team leader has to identify dependencies of each team member this ensures that the development process is going in the right direction as well as every team member is working according to the agreed strategy. Providing the team with the time plan which has to be followed or in case of necessary changes being modified according to the needs is also one of the core functions of the team leader which is aimed to improve work flow and to have a good overview of the project. Finding dependencies and having a good overview will in a way empower the team leader to remove some barriers which can stop or negatively affect the team in sense of achieving the main goals of the project.

In the feasibility phase of development, one of the main contributions of the system manager includes analyzing, describing and formalizing the requirements in a clear way for the team. The source of the requirements, come either from customers, standards or internally. If the requirements came from outside the company (i.e. customer) the system manager has to contact sales representative in case of misunderstandings.

Although the system manager influences the feature mainly in the early phase and the feasibility phase, the architect influences the feature throughout the entire project. The architect mainly interacts with system manager during the feasibility phase. In this phase, the architect tries to find out if the feature can be implemented and how the feature will affect the whole architecture of the system. During the execution phase, the architect supports the designer with guidelines on how to implement the feature according with the architecture. He also answers questions the designer may have during the implementation. Architect's work depends on the complexity of the feature and the way the current feature influences the other parts of the whole system. In case of uncertainty he has to contact other architects or team members from other groups to map the dependencies and to transfer the knowledge to the right people.

The main role of the designer in the development process is to implement the feature in terms of actual code. Designer can start implementation as soon as the requirements are clear enough (this may happen during either feasibility or execution phases) and therefore late requirements directly impede his work. He also depends on function testers' and system tester's work, as some of the test results might cause rewriting the program code.

As some of the team members' progress is tightly linked to the code or complete functioning feature, designer's delay directly impacts the whole feature. System tester for instance is not able to start his testing procedures without having the feature itself or function tester cannot execute tests without actual code. However good collaboration between function testers and designer allows running test cases in parallel with designers work and give a feedback (trouble reports) immediately after faults have been discovered.

B. Development process and phases

Streamline development process has been adopted by the development unit at Ericsson for a number of reasons.

"…main reasons are… improving time to market, more efficient research and development, shorter time to commitment, allowing development setup to be more flexible, possibility to change priorities and plans at any time … to develop the right things…" (Interviewee)

For the team, feature development starts after the "Go" decision has been made. They start with conducting feasibility analysis of the feature. After the analysis, "Commit" decision has to be made and the execution phase begins, which has a number of milestones and checkpoints. When the feature is ready it is delivered to the latest system version (LSV). After that the team continues with final tests for the feature and ready to take over a new project.

Each team member is having different input on each phase of the development process, as explained in the previous section. Specific documents have to be created and milestones to be

met. Activities on each phase are highly linked to the results from previous stages; therefore any delays could cause even greater delays on the next phase.

Team leader has the main workload during the beginning and closer to the end of the project. On the middle stages of the project the work mostly consists of checking the progress and the fulfillment of all the milestones and solving problems which could arise within the team. Creating a feasible time plan and setting up the right goals by the team leader at the beginning will only ensure the success of the project.
System manager after the early phase where requirements have been created follows the execution process by doing reviews on each check point and helping team members to understand the feature in case of uncertainties.

"… Efficiency in work is highly depended on the requirements clarity and frequency of changes in the requirements which on its hand affects the performance of the project in general…" (Interviewee)

Operational architect is highly involved on all the phases of the development process particularly for those features which affects the current architecture of the whole system or depended on it. In the early phases he contributes by helping the team.

"… Helping to understand the scale of the feature, feasibility, requirements specification and suggesting the ways of most efficient implementations of the feature… "(Interviewee)

During the execution phase the architect works closely with designers in order to help with design decisions which should be in accordance with current architecture. For the architect it is a challenge to follow the dependencies on different teams work and to analyze the consequences which will affect the whole architecture after the feature is included in latest system version (LSV).
Function tester is mostly involved in the execution phase when he works in close cooperation with designer.

"…The main idea for function tester is to find situations which were not thought of by designers or other team members…" (Interviewee)

Streamline development allows working on the tests in parallel with the designer. That helps to increase the productivity of the function tester; however there are other factors which might slowdown the process, i.e. new testing framework which is more time consuming in relation to the execution of the test cases. As well as synchronizing different branches usually cause in delays due to differences in implementation approach.
Designers similar to function testers have the main workload during the execution phase. Apart from coding, a lot of documentation has to be created to explain/support the code. This documentation is mostly used by designers themselves or by function testers. According to the designer the efficiency of the work is mostly depended on the size, complexity of the feature and previous experience with projects in similar area. One of the typical problems is to understand the requirements of the feature which is completely new.
System testers are mostly involved in the beginning of the project (analysis of the feature and requirements) and closer to the end of the project when the feature is ready and system tests can be run. According to the system tester delays are seldom occur during the tests, but one of the reasons for delays could be due to updates of the testing tool to be used for a specific feature.

C. Lean principles

According to all the team members lean principles are reflected during the entire development cycle of the feature. They have positive impact on the development process in general and on the team work performance in particular. All team members agree that cross-functional teams allow immediate feedback from all the parties involved in the development process.

"…having all the team members close to each other is good as it speeds up the information exchange in the team…" (Interviewee)

Collaboration and communication between the stakeholders improved significantly comparing to the previous development process.
The shortage of particular resources within the team might lead to problems which were not feasible while using "waterfall" approach. According to one of the interviewees in some cases there is a tendency to move function tests to the design tests. This is potentially a negative move as designers will have to test their own code and might miss some faults.
Scaling the processes to fit different projects it is essential in lean development. This allows reducing "waste" and focus only on those activities which a necessary for specific projects.

"… Adopting the process to smaller features is the area which still can be improved… " (Interviewee)

However delays might arise in cases when some documents such as test analysis or test specification have to be approved by the line manager. As there is only one line manager who is responsible for doing it.
All the interviewees expressed a need to reduce on external meetings as they are causing delays by disturbing the implementation process since sometimes it is difficult to get on track of where they stopped before the meeting.
Usually the problems which arise during the project are reported to the management in the "conclusion exercises". However not all of the aspects are considered by the management hence looping up to the next projects.

D. Reflection on Streamline software development

According to the interviewees, adaptation of streamline development process has positively influenced the time required for releasing new features at the development unit. The team particularly seems to be in fond of activities which are carried out together by all the members, such as morning standup meetings, reviews, test analysis, etc.

As for the team leader, streamline development allows to understand the goals and activities in a better way, since it encourages communication and sharing knowledge among the team members.
"… Streamline process has a very good support for communication (information exchange) among the team members and among teams… it also allows having a better structure on how to work, how to share the work and it increases possibility of learning new things which is essential for improving teams' performance…" (Interviewee)

One interesting thing put forward during interviews was, the idea of dividing the requirements into packages and then later assign them to cross functional teams was good. Team members do not get stuck with a batch of requirements to work on, and according to one of the interviewee it was noted that having short projects is a good way to handle customer changes since the changes affect only specific feature and not the entire project.

V. DISCUSSION

This chapter is analyzing the findings which were gathered during the previous stages of the research. It brings forward the bottlenecks which were revealed when analyzing related theories, results of company's presentations and interviews. As the main purpose of the research is to identify possible bottlenecks in the development life cycle of the feature, this chapter is categorizing such bottlenecks and presents it not as problems, rather as challenges for the team and management.

1. Task switching

Streamline development encourages management commitment, for example regular meetings with the cross function teams or close contact and co-location with management during the process implementation. This can be both beneficial and can create some drawbacks. Lean philosophy describes this way of working as reducing the gap between the team and the management; it also encourages greater involvement of the entire organization in the production of software and in the process implementation (Shalloway, 2010). On the other hand, during the data collection it was mentioned by almost all the interviewees that, there are too many external meetings which sometimes results in delays throughout the whole process. In this case the team members have to leave whatever they are doing to attend the meeting. One of the interviewee noted that "*…you can be concentrating on some work, than you leave for a meeting, and when you come back it is difficult to catch up on what you were doing…*" This could be a possible bottleneck in the development life cycle of a feature, described in lean philosophy as task switching. Gathering thoughts and getting into the flow of new task requires time and decreasing developers' efficiency (DeMarcoand). Therefore task switching in lean considered as a waste. Since meetings require task switching and interrupt the development process, those which considered unnecessary by the team members can be linked to waste in lean philosophy. The more unexpected and unwanted meetings are held during the feasibility and/or execution phases in the program execution stage the more waste is created in the whole development process. Waste in turn is a potential bottleneck that impedes development of a feature and creates delays.
Possible solution mentioned by one of the interviewees, was instead of the whole team attending the meeting, it is enough to have just one or two team members to participate in it and later spread the information within the team. Therefore finding the right balance between the meetings is a great challenge for both: management who organizes it and team members who have to participate in it.
It is also important to note that not all meetings require task switching. Team's internal meetings, for example everyday's stand up meetings are much welcome by all the team members. It is during the internal meetings where the team members get to know each other's progress and have a chance of informing others about the problems that might have occurred in their work. This way of working allows the team to share knowledge and delay commitments since possible problems are identified as soon as they arise and therefore reduce the possibility of delaying the project.
Task switching affects all the stages of the feature

development cycle, as it does not matter what the team members were doing, any distraction could influence the performance and the efficiency of work.

2. *Documentation*

Documentation in this case has two ways of influencing the feature development cycle. On one hand it makes the development process easier to understand and the product easier to maintain, which can be linked to one of the principles in lean philosophy – knowledge sharing. On the other hand unnecessary documentation (documents which are not during the process or created for future reference) can be addressed as waste in lean principles.

The results which are described in the previous chapter show an example of how the documentation can improve the performance of some team members. Cross functional teams in streamline development assume having all the core competences within the team. However there is a role which is distributed over two or more teams, which is - operational architect. One of the tasks for operational architect is to provide the team with high level guide lines for the feature development process. The fact that architect should provide support for more than one team and the fact that there is no written architectural description (guidelines) available for the team, potentially could cause delays in the process. In case if the architect is not around and some of the team members require architect's support to continue the work or to make a decision, the person will have to wait as there are no documents where it is possible to refer to and get the answer. As it was previously mentioned streamline development process provides freedom for the development teams and in terms of architecture, it provides only guidelines on how to develop a feature. Supporting such freedom with documentation might improve the knowledge sharing and as a result influence the development life cycle of the feature in a positive way.

The other side of creating documentation is the fact that it is time consuming. In case if such documentation is not used by the team members contradicts with one of the lean principles – eliminating waste. Finding the right balance between the documentation available for the management and documentation created for the team is another challenge for the management which probably requires more attention.

3. *Lessons learned (Evaluation considerations)*

One of the lean principles which was mentioned previously in the research is perfection of the process, which is done via constant improvements. One of the conditions required to improve the development life cycle of the feature is to have the input from previous cycles.

According to the data from result section of this report, in the end of each program execution stage (this is where the feature is delivered), team members create a report where suggestions to the management are made on how to improve the development process for the next iterations. One of the reasons for that is to have team's opinion on the activities which are done and share the good experience among other teams. This supports the process improvement throughout the whole organization, by adopting one of the core lean principles such as knowledge sharing, which in this case drives forward another lean principle – perfection of the process.

However, according to the data in the result section, some of the evaluations are not always considered by the management and therefore the team continues to work on the next feature having the same problems unsolved. This could result into a bottleneck during the next feature implementation if the team happens to find the same problems faced in the previous cycle. Considering and implementing all the evaluation results is a great challenge for the management as it requires in-depth analysis of all the requests made by teams and consequences of such requests.

4. *Communication chain*

Requirements elicitation is one of the most influential steps of development with regards to eventual success of a program. Problems introduced during requirements elicitation are the most expensive if not detected quickly, since later corrections require the most rework and waste in form of defects (Walton, 1999; Poppendieck & Poppendieck 2003).

In the development unit observed in this research, system manager pre-studies the feature and analyzes the requirements in order to present it to the team in the most efficient way.

Management is the intermediate party between customer and system manager in the elicitation of requirements. Some interviewees suggested that the lack of direct contact between the system management and the customers can encumber the process of specifying requirements for the feature. According to interviewees, level of details is varying depending on representatives between customer and system manager. Some management representatives are able to transfer the idea and required functionality from the customers in sufficiently detailed and technical way, but some are not.

In case system management experiences problems with level of details of information transferred by management, intermediate party or the customer has to be contacted back.

Long communication chain between customers and system management introduces handoffs of information. In turn handoffs among functions can cause delays and increasing risks of information being misunderstood (Walton, 1999). Conveyance of information might cause a loss of knowledge,

as great amount of data remains with its carrier and never get handed off to others (Poppendieck & Poppendieck 2003). The longer the communication channel is, the higher are the chances to miss some of the details, which could be essential. Allowing direct communication of system manager with the customer could eliminate time and recourses consuming non-value-added-action in form of handoffs, therefore waste (Larman & Vodde 2009).

5. *Feature context*

According to the result section, some of the interviewees mentioned the absence of knowledge about what the feature is intended to be used for and in what context. Depending on these factors various architecture solutions supporting appropriate non-functional attributes can be proposed. Implementation also requires understanding of usage scope. With regard to this, feature can be coded in the most efficient way, using the most appropriate technologies.

Potentially, in order to implement what is needed developer can start guessing what the feature is intended to be used for. While delivering maximum value to the customer is the core philosophy of lean, guessing can prevent team from fulfilling this principle (Hurwitz & Demacopoulos 2009; Lean aerospace). In this way feature implementation can be technically correct, however suboptimal.

On the other hand developers can start searching for this kind of information by asking customer. As was already noted, customer reply can cause waiting. This method can potentially introduce waste and impede the development process, therefore should be avoided.

Customer involvement to the process can prevent this kind of complications. Hibbs (2009) in the book "The Art of Lean Software Development" highlights the importance of customer participation in the development process. Customer's representative who has a clear understanding of the vision, which has to be transferred to the team, could directly explain the intends if participating more in the development process. Since customer representatives participation in development can increase development cost, short and often meetings with representative are also suggested by Hibbs (Hibbs, 2009).

6. *Process perception*

Empowering the team is one of the core principles in lean software development which is adopted by the development unit. It allows the team to be more flexible in the development process by choosing tactics and development methods which are best suited for a specific case (Shalloway, 2010). Streamline framework proposes high level description of the processes which should be followed during the development of the feature, while still leaving a space for the process decisions on a detailed level. It is the team who knows in a better way what is the best for them, therefore it is the team which is responsible for the process (Shalloway, 2010).

During the interviews respondents were asked to describe the lifecycle of a feature they work on. The purpose of this question was to find out more about the process from the team perspective. The results show that team members whose work is not directly related to the management activities had difficulties with answering it. Questions about the phases or stages of the feature development process were also challenging for the interviewees. This has made an impression that not all the team members have a clear understanding of the development process. This is most likely to occur since the team investigated in the research is relatively new and team members are new to their roles. Despite this fact, the lack of common understanding of the process still indirectly influences the work. Without having a clear knowledge about the process, it is difficult to suggest improvements to it. Therefore most of the activities which are done throughout the development process assumed to be best and could not suggest further improvements to be done. This is one of the challenges faced by the management, dealing with which will make improvements suggestions coming from teams even more efficient and constructive.

7. *Competence*

In streamline development feature teams are organized as cross function teams with all the core competencies i.e. design, testing and management as explained in the SD section. This competence is used in development of the feature in the program execution state. The teams have full responsibility of the project during the program execution state. In this, SD gives the teams more responsibility as well as freedom in their activities to make decisions as recognized by Poppendieck (2003). This way of working creates an opportunity to share knowledge in the team and also the feedback loop in the team is shortened since all people are working together.

As means to check for core competence needed to develop a feature, in the "activity and release planning loop" resources are allocated to the feature, these may include tools, people and learning. However findings indicate that, delays are likely to occur due to lack of some competence in the team. One of the facts which may result into lack of competence may be attributed to the nature of the current volatile technological environment. New tools emerge time to time implying that people have to gain competence to operate them. An example can be revealed in function testing where a new testing tool TTCN was introduced; this caused a deduction on the number of test cases produced in one week from five to one or two.

The deduction in test cases, make it clearly that lack of competence in a team is a potential bottle in the development life cycle of a feature. As means of curbing the situation since streamline development there is continuous analysis and knowledge creating activities, the company tries to introduce courses on each new technology introduced; however adoption is a process which needs time and people's acceptance to the change. This can be argued for from the technological adaptation curve by Bohlen, et al. (1957), where some people may take less time to adapt new technologies while others may take longer time.

8. Feedback

The ability to exchange information in an efficient way between the development teams and other stakeholders in the development life cycle of the feature has a great impact on the lead time. Much as the team has got the core competence to develop the feature it still needs support and feedbacks from external parties. For instance feedback from the product activation team, the system owner, the line organizations and standard organizations is crucial for the feature team. According to the interviewees, getting information regarding the architecture from outside the team can lead to waiting. This kind of waiting creates a potential bottleneck in the development life cycle. Another analogous example described by interviewees are waiting for documentation approval from management or waiting for customer's feedbacks. These problems of waiting and delays can further be explained using lean philosophy.

In lean waiting is considered as a waste, due to the fact that it results into delays. Delays in turn prevent the customer from getting a product quickly. The time between giving question and getting an answer from any stakeholder is considered a waste (Hibbs, 2009). For instance Shalloway (2010) defines the time from asking customer a question until getting an answer as a common delay in software development. On the other hand, *"If people are immediately available, there is no delays and development continues at full speed"* (Hibbs, et al., 2009). Therefore waiting for documentation approval, waiting for feedbacks and waiting for anything during development is a bottleneck that can hinder the speed of the development cycle.

VI. CONCLUSION

This research has explored the development life cycle of a feature with the main purpose of identifying possible bottlenecks that are hindering the development process. The results presented here are after a case study with one of the development unit at Ericsson AB.

Using lean principles as a theoretical base the following bottlenecks were identified:

1. *Task switching*
   Task switching, which is seen as a waste in lean philosophy, has been mentioned by almost all the interviewees in this study. It hinders the development process in that team members lose focus when they change between tasks.
2. *Competence*
   When new things are introduced, there will be members whose competences are affected, due to lack of experience. For instance in this research it was noted that time required for function testing doubled after introducing a new testing tool.
3. *Feedback*
   While feedback among team members is fast, outside communication sometimes take long time. For example, some of the documents have to be approved by the management before the team can continue. In case if there is only one person who can approve (or give feedback) it can potentially result into delays.
4. *Evaluations considerations*
   In the end of each project the team gives an evaluation report for the management on suggesting improvements for future projects. However some interviewees mentioned that not all of the suggestions are considered, as a result the team continues to work, having the same problems unsolved, which might cause delays.
5. *Communication chain*
   Having a long hierarchy chain between teams, management and customers can lead to loss of information, for example communication between the system manager and a customer includes a product manager who sometimes might miss some details of the requirements.
6. *Feature context*
   Complete understanding of the usage of the feature can affect the speed of the development process. Developing the feature and not knowing its exact usage in the whole system increases the chances of misunderstandings the requirements.
7. *Process perception*
   Lean philosophy encourages full understanding of the process. Not having a clear understanding or enough knowledge about the process in which the development unit works makes it difficult to suggest improvements.

8. *Documentation*
   Documentation makes the development process easier to understand and the product easier to maintain, which can be linked to one of the principles in lean philosophy – knowledge sharing. An example could be the architecture description, which is supposed to provide only high level guidelines for the team to implement the feature. Having no document where to refer in case of uncertainty might cause a delay. As there is only one architect for two or more teams, sometimes it is impossible to get the immediate feedback or suggestion for solving the problem.

The bottlenecks identified here in this research can be taken as challenges which need to be addressed not only to the development unit in question but also other organization working with feature development. Addressing them and looking for possible solutions can be beneficial in that it may result into a decrease in lead time and also improve the end to end flow.

ACKNOWLEDGMENT

The author would like to thank Ericsson AB for having given us the opportunity to carry out this research. Special thanks are due to all Ericsson employees at the development unit involved in this study. Our special and dearest thanks go to our academic supervisor Helena Holmström Olsson from IT University of Gothenburg for her constructive feedback, meetings and support. We greatly appreciate your effort and valuable time you have put to this study.

REFERENCES

Abrahamsson. P, Salo. O, Ronkainen. J, Warsta. J. (2002). *Agile Software Development Method.* VTT Technical Research Centre, Finland.

Baundin, M., 2004. Lean Logistics: The Nuts and Bolts of Delivering Materials and Goods. New York: Productivity Press

Berry, R., 2003. Trends, challenges and opportunities for performance engineering with modern business software. IEE Proc.-Softw.

Boyce C., Neale P., (2006). *Conducting in-depth interviews: A Guide for Designing and Conducting In-Depth Interviews for Evaluation Input*. Pathfinder International

DeMarcoand Listet, Peopleuare,63.

Dube, L., Pare, G. (2003). Rigor in Information Systems Positivist Case Research: Current Practices,
Trends, and Recommendations. MIS Quarterly, 27 (4), page 597.

Everett, W., Keene, S., Nikora, A., 1998. Applying Software Reliability Engineering in the 1990s. IEEE Transactions on reliability

Genuchten van Michiel (2007). The Impact of Software Growth on the Electronics Industry: Eindhoven University of Technology/NXP Software

Harvey, D., 2004, *Lean, Agile,* "The Software Value Stream"

Harter., D.E., 2000. Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development. *Management Science, INFORMS*

Heinrich, B (1984). In a patch of fireweed. Cambridge, MA: Harvard University Press.

Hibbs, C., Jawett S., Sullivan M., 2009. *The Art of Lean Software Development*. USA: O'reilly Media.

Holmström, O, H (n.d), Acting Agile in 'Streamline Development. Department of Applied IT, Gothenburg University, Sweden

Holweg, M., 2007. The genealogy of lean production. J*ournal of Operations Management,* 25 (2007) 420–437

Hurwitz, D., Demacopoulos, K., 2009. The Case for Lean IT. CA, Transforming IT Management

IEEE Std. 829-1998, Standard for Software Test Documentation

Joseph A. Maxwell (2005). *Qualitative Research Design*: An Interactive Approach, Second Edition. George Mason University, Sage Publication, Inc

Kanigel, R., 1999 *The One Best Way: Frederick Winslow Taylor and the Enigma of Efficiency*, New York, Penguin Group

Kilpatrick, J. D., 2003. *Lean Principles*. Utah Manufacturing Extension Partnership

Klein, H. K. and Michael D. Myers. 1999 *A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems*, MIS Quarterly, Special Issue on Intensive Research.

Larman & Vodde, 2008, *Scaling Lean & Agile Development: Successful Large, Multisite & Offshore Products with Large-Scale Scrum,* Addison-Wesley, Ch.22

Liker, J., 2004. *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer.* New York: McGraw-Hill

MacCormack A, Kemerer CF, Cusumano M, Crandall B. 2003. Trade-offs between productivity and quality in selecting software development practices. *IEEE Software* 20(5): 78–85.

Maskell, B., Baggaley, B., 2003. Practical Lean Accounting: A Proven System for Measuring and Managingthe Lean Enterprices. New York: *Productivity Press*

Merton, R. K., Fiske, M., Kendall, P.L. (1990). *The focused interview: a manual of process and procedures (2nd ed.)* New York: Free Press

Mehta M , Anderson, D, Raffo. D, (2008) Providing value to customers in software development through lean principles. John Wiley & Sons, Ltd.

Ohno, T., 1988, *Toyota Production System:Beyond Large-Scale Production.* Translated by Productivity, Inc. New York: Productivity Press

Poppendieck, M., 2002, Principles of Lean Thinking, Poppendieck.LLC

Poppendieck. M., Poppendieck, T., (2003). Lean Software Development: An Agile Toolkit. Addison-Wesley Professional

Robbins, S.P. Bergman, R. Stagg, L. & Coulter, M., 2003, *Management*, 3rd ed. Sydney, Australia: Prentice

Shalloway. A, Beaver.G, Trott.J.R, (2010) Lean-Agile Software Development Achieving Enterprise Agility. Addison-Wesley

Tomaszewski, P., Berander, P., and Damm, L. (2008). From Traditional to Streamline Development – Opportunities and Challenges. Software Process Improvement and Practice. 13 (2), 195-212.

Tolliday, S. & Zeitlin, J., 1987, *The Automobile Industry and its Workers: Between Fordism and Flexibility*, New York: St.Martin's Press

Towill, D. ; Cardiff Univ., UK

Walsham, G. (1993). *Interpreting Information Systems in Organizations*. Chichester: Wiley & Sons.

Walton, M., 1999, Strategies for Lean Product Development. Massachusetts Institute of Technology.

Womack JP, Jones DT,Roos D. 1991 and 1990. *TheMachine that Changed the World*. Harper Collins Publishing: New York.

Womack JP, Jones DT. 1996a. Beyond Toyota: How to root out waste and pursue perfection.
Harvard Business Review. September 1, 1996.

Womack JP, Jones DT. 1996b. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. Simon and Schuster: New York.

Wu, S. ; Wee, H.M. ; Dept. of Ford Production Syst. of Mfg Div., Ford Lio Ho Motor Co., Chungli, Taiwan

Yin, R. K. (2009). *Case Study Research: Design and Methods* (4th ed.). Newbury Park: Sage Publications.

Zidel, T. G., A Lean Guide to Transforming Healthcare: How to Implement Lean Principles in Hospitals, Medical Offices, Clinincs and Other Healthcare Organizations. Milwaukee: ASQ Quality PressPublications