Factors That Restrict Speed in Streamline Development

Zana Leric
Magnus Södergren

Bachelor of Applied Information Technology Thesis

Report No. 2010:007
ISSN: 1651-4769

Academic Supervisor: Lars Pareto
Industrial Supervisor: Per Sundvall

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

Factors That Restrict Speed in Streamline Development

Magnus Södergren
magsod@ituniv.se

Zana Leric
zanal@ituniv.se

IT University of Gothenburg, Software Engineering and Management. Gothenburg, Sweden

CHALMERS | UNIVERSITY OF GOTHENBURG

Abstract

The fast moving area of mobile telecommunication forces companies to deliver functionality with increasing speed to customers. To deal with these challenges, Ericsson AB created a software process called Streamline Development(SD); which bridges the gap between long term thinking and agile development. SD is a process inspired by agile and lean methodologies, focusing on small efficient teams with an end-to-end responsibility of development, short life cycles, and continious code integration. The purpose of streamline is to take advantage of the benefits of shorter time to market, and to have a more flexible development set-up to aid the possibility to make changes at anytime. The goal is to get to faster development of products with higher quality.

This article investigates SD and searches for different factors that could have an impact on the development process of a feature and extend the lead time. The paper will try to answer which factors restrict speed in Streamline Development? The results are 34 factors that restrict speed in SD. These factors are divided into three areas and further divided into 12 categories.

The research was conducted as a case study at Ericsson AB as a part of the bachelor thesis project course at the IT-University of Gothenburg during the spring of 2010.

Categories and Subject Descriptors D.2.9 [*Software Engineering*]: Management: Productivity, Life cycle, Time estimation

Keywords: *streamline development, restricting factors, speed*

1 Introduction

The area of mobile telecommunication is moving rapidly forward and companies need to deliver functionality with increasing speed to customers. Frequently changing customer requirements and new emerging technologies are some of the major challenges companies need to take into consideration when developing long term strategies (Nerur et al., 2005; Olsson, 2008; Petersen and Wohlin, 2009; Tomaszewski et al., 2008). To deal with these challenges, Ericsson AB has developed a software process called *Streamline Development* which is a process created in-house at Ericsson AB. SD bridges the gap between long term thinking and agile development (Olsson, 2008).

Past research have mostly looked at the SD process from a high-level perspective and little attention has been given to the end-to-end flow in the life of a feature from conceptual idea to integration into finished product. Even though it is a fairly young process, implementation started 2007, it has been used in practice for a period of time now giving this research an insight into how SD works in a more mature setting. While some of the related research have looked into detail at SD, they have done so from the start of implementing SD in an organization. The problem of aquiring speed using agile processes in large organizations have been looked into by resent studies Olsson (2008); Petersen and Wohlin (2009).

This article looks at a department in a large organization that has been using SD for over a year and examines the SD process in detail from feature conceptualization to product release. The research question is *which factors restrict speed in SD?* To answer this research question, qualitative approach using grounded theory as research method was chosen. The research was conducted as a case study at the company Ericsson AB. By reviewing statistics and other documents related to already implemented features, the researchers have followed a feature throughout its life. Through several interviews with carefully chosen key individuals that represent the organizational framework, the researchers hoped to identify key factors that might lead to an increase in lead-time. This article shows that SD has not been easy to fully implement in practice into an existing organization with established processes. This article presents 34 factors in twelve categories divided into three areas that reduces speed in SD.

This paper is structured as follows: Section 2 describes SD in detail. Section 3 presents the research approach and method used. In section 4 the results are presented. Section 5 discusses the results and section 6 gives conclusions and suggestions for further work.

2 Streamline Development

When developing project in a traditional development way a project's lifespan runs for up to four years. Because the market moves forward quickly, the customer changes requirements before the product reaches the market due to the long duration of the projects (Tomaszewski et al., 2008). Projects done according to a traditional process has predefined scopes and product releases are done continuously. These projects are large and thus demand large development teams to carry them out (Tomaszewski et al., 2008). A product is released after requirement specification has been defined and developed. Because the projects are large, it takes a long time for a product to reach the market giving the customer plenty of time to change their mind making some of the requirements obsolete even before the product comes out on the market. Tomaszewski goes on describing that, whenever there is a request to change a product, a so called change request has to be filed. Change requests require a lot of maintenance and are therefore high in cost. A change request that is filed to already implemented work leads to decreased productivity, and decreased productivity in itself leads to more lead time that leads to less competitiveness for the company. The lifespan of projects developed according to the SD process are significantly shorter (around three months lead time) than the lifespan of projects developed in a traditional way(Olsson, 2008; Tomaszewski et al., 2008).

A SD project consists of three phases and must first go through a stage called early phases. In this stage, the projects initial time estimations are drawn up, as well as a study on how much of a need there is on the market for a product. When there has been decided that a project is conductible, a decision (called GO) is taken to send the project to a team for potential development. From here, the project goes in to a feasibility phase where new estimations are decided upon by the assigned team, and the execution of the project is discussed. When a team decides to start working on a project a so called Commit decision is made. This means that the project has gone out of the feasibility stage and into the execution stage. When a team has finished developing and testing a feature on their own brach called LLV(Latest Local Version), the feature is integrated into the LSV (Latest System Version) through integration and regression tests. After final testing, the feature is sent into a release project. The feature is now concidered ready for release. In the release project a product is getting ready for a potential release. Before a product is fully released into the market it is first tested out at one customer site. This is called a FOA (First Office Application). When a product passes a customer acceptance test, the product is ready for the market (Olsson, 2008; Tomaszewski et al., 2008).

Teams in SD are called cross-functional teams (XFT). These teams should be autonomous and each team member has an area of expertise (i.e. designer, tester, architect, etc.). Team leaders guides the teams throughout the development process. The team is fully responsible for the development of this feature and is free to choose in which way they want to develop.

SD has the benefits of that the products reach the customers faster and create a more positive customer response. This is the main motivation to why SD favours customer responsiveness compared to traditional development. Customer involvement is important since it helps to detect changing needs early on and thus reduces the waste risk. More frequent releases give faster feedback from customers. Only the highest customer demanded requirements are implemented (Tomaszewski et al., 2008). Tomaszewski points out that in SD there is only one product version at any given time which is another big difference from the traditional development way. This also reduces the maintenance risk since only the latest version needs to be maintained. The biggest difference between SD and traditional development is that in SD there is shorter time between identifying product needs and implementing them. And because there is shorter time between these two, there is less risk for changing market demands. A product that has been completed is added to the base product line, and that product line gives the basis for next project. (Tomaszewski et al., 2008).

SD was created with agile and lean inspiration in mind. Lean and agile development shares many of the characteristics that make out the SD process (Cockburn, 2007) Key principles such as; the focus on teams and individuals, short iterations with fast delivery and continuous code integration are some of the characteristics of agile development that are said to shorten the lead time of software development (Cockburn, 2002, 2007). Lean development shares the focus on shorter project life cycles which, according to middleton2001lean,schwaber1995scrum has shown to enhance the software quality of a product. He also discusses in his paper, the impact on an organization when switching to a lean principle way of working. The issues of switching mindset from traditional way of working into a more incremental style requires further change for the teams and roles involved with software development in an organization. Lean and agile practices, like SD require close, co-located teams to work together.

3 Methods

The research was conducted as a case study at a department at Ericsson. The study involved interviews with people working with SD. SD had been introduced 2007 to the company. However, the description of how SD worked with the company differed from paper to reality. This research was concerned with the findings of relevant factors that can possibly explain what in the development process of a feature is not done according to SD and what the cause of extended lead-time can be. The study involved interviews with six individuals, one seminar on architecture and, two introduction presentations on SD and the company. This was supported by relevant documents and statistics on features as well as the researchers own observations.

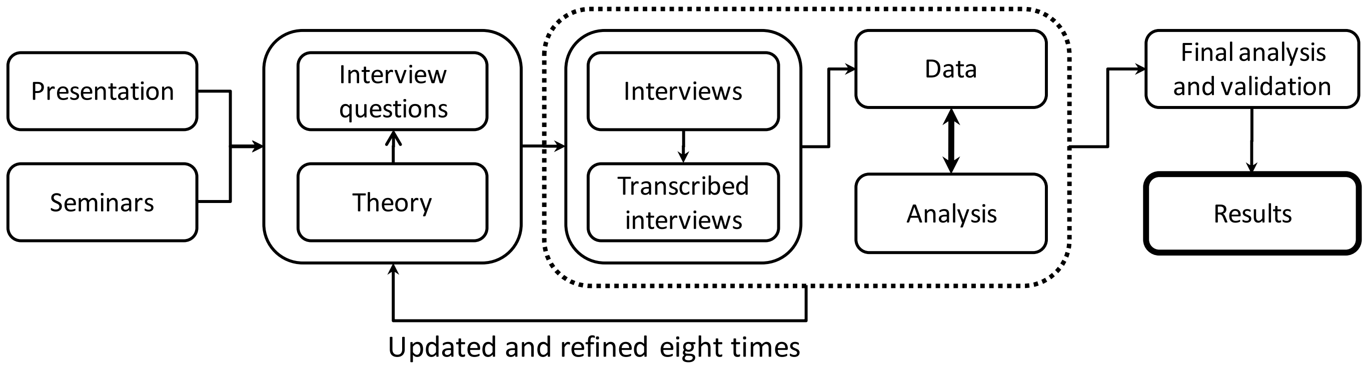

Figure 1: Method model

3.1 Strategy

The case study is a research strategy that is used for learning and understanding a particular setting or environment and aspects or phenomena within the environment (Eisenhardt, 1989). The reason for choosing this strategy is suitability in studying and mapping phenomenas within organizations. The methods used in case studies, such as document analysis, interviews and observations (Eisenhardt, 1989), fits the aim of this research from a methodological perspective, and therefore the case study strategy became the natural choice for this research.

3.2 Qualitative approach

Hancock (2002) explains qualitative research as: “Research which attempts to increase our understanding of why things are the way they are in our social world and why people act the ways they do, is qualitative research”.

Some of the characteristics of qualitative research are research involving observations of social environments, interviews based upon unstructured interviews using open-ended questions (Booth et al., 2003; Creswell, 2008; Silverman, 2009). Due to the fact that the researchers did not know much about SD and even less about what factors could restrict speed a more exploratory approach was needed; a qualitative approach felt like a natural choice as we could then gradually develop theories after making new discoveries about feature implementation using SD. Because we did not know the explicit problem, only that there was a problem, it was more logical to discover the problem by making observations at company site, and to conduct interviews with individuals that have experience working with feature development according to SD (Eisenhardt, 1989; Silverman, 2009; Strauss and Corbin, 1998)

Grounded theory

The aim of the grounded theory is to come up with new theories about a phenomenon by collecting and analyzing data about the chosen phenomena. Primary data collection techniques are interviews, observations and literature reviews on information relevant to the phenomena under research(Goulding, 1999). What is unique about the grounded theory is that the data collection and analysis are conducted simultaneously by using a process that is called *constant comparative analysis*. The data is analyzed and transcribed directly after it has been collected (Hancock, 2002; Silverman, 2009; Strauss and Corbin, 1997, 1998).

Since our focus was on interviews, the grounded theory approach seemed most appropriate. Interviews conducted were transcribed and analyzed directly after they have been conducted. The data collection and analysis happened simultaneously and the theory constantly evolved according to the new findsings after each interview (Hancock, 2002; Silverman, 2009; Strauss and Corbin, 1997, 1998).

3.3 Data collection and analysis methods

Selection of informants

Four interviews were conducted with key-individuals within four areas in the department under study; these together represented the SD feature development from the beginning of a project to the end. Two interviews were conducted with the team leaders that worked with the observed features.

Interviews

Two introduction presentations about SD and the company was given by Ericsson which served as basis for finding out how the organization worked and potential factors that could restrict speed in development. These presentations and a seminar on SD architecture served as basis for designing initial interview questions. Questions that came up during the presentations and the seminar were mapped into the areas the interviewees worked in. All informants where choosen based upon recommendations from the company.

Focus was on conducting semi-structured interviews with key-individuals both at Ericsson AB and University of Gothenburg. The interviews were semi-structured because there was a need to let the interviewee talk both freely but also to force the person to describe certain pre-defined areas that might be unclear to the interviewer at that point in time. The interviews conducted with the team leaders took place after the first four interviews in order to get a better overview over the SD process. The new factors that came up during the first four interviews were used in the interviews with the team leaders to see if any of the factors occurred during the actual development of the features. Any follow up questions that came up after an interview were sent by email to the interviewee in question.

All of the interviews were taped, transcribed and coded using Atlas.ti along with any document collected that could contribute to the research (Silverman, 2009). Interview transcripts was coded using software and this ensured that the interpretation of data fits well with the external reality at Ericsson AB.

The initial strategy involves thoroughly learning the area making use of a official process model provided by the company during one of the presentation to visualize the domain. This model served not only as a description of the area and help one to better understand the area but it also served as a tool in the interviews that the researchers conducted. The use of a model gave the interviewees a chance to have a more focused talk about complex domains and at the same time alleviate some of the issues we might have to follow and understand what the interviewee was referring to during the interview.

The four chosen areas to interview helped to better understand the framework of the research area and to identify findings correctly. These stages include: early phase, program activation board, program execution, and the release project. The taped interviews with these key individuals together with the use of a model of the area were used as to more efficiently make use of the interview times. Between each interview the model was updated and the questions prepared before each interview could be prepared with greater accuracy to improve a better data collection. From the beginning the researchers used internal documentation in the form of an power point documentation containing several illustrations of each stage or phase of the development process. This document together with the model proved to be very useful tools not only during the interviews but also in the analysis and preparation of each interview.

Coding

As in any semi-structured interview, not all information was relevant to the research question but with the use of a qualitative data analysis tool (Atlas.ti[1]) it proved to be a very easy to extract relevant data. Not only did the tool provide an easy way to ground the data but also confirm and correlate data through several interviews or other sources of data. These sources included one official final report of a project from a team and several printouts of statistics from the observed features.

Categorization

Many of the factors identified from the data were grouped by their relation to each other through actions or milestones in the development process. The following relations were discovered: communication between actors and between phases, internal or external issues regarding teams, and artifacts and their relation to phases in the SD process. Several models, excel-sheets, white board-sessions, and other ways of abstracting around the data were used in combination in this work. The most important tool during this phase of the analysis was Atlas.ti.

[1] http://www.atlasti.com/

Reliability and validity

The authors of this article used the following definitions given by Silverman of: reliability as "The extent to which findings can be replicated, or reproduced, by another inquirer."; and internal validity as "The degree to which findings correctly map the phenomenon in question."; and external validity as "The degree to which findings can be generalized to other settings similar to the one in which the study occurred."(Silverman, 2009). Taped interviews ensured that we could document everything that has been brought forward in the interviews more accurately. The risk of taking only notes could result in that certain information get's lost. To validate the truth of our data we compared our findings with internal Ericsson documentation on features. We also used research articles that reported similar research to ours to see if we could map our findings to to previous research, and see if there were similiarities or differences to our. further discussion of reliability and validity can be found in the discussion section of this article.

Other data collection

Research literature on SD was reviewed in order to select the most suited areas in SD to conduct interviews in. Other sources of data includes an official final report from a team after a finished project implementing a feature. The researchers also received a power-point document and a transcript of implementation hours for two features. A project report from one feature and feature statistics on the two selected features were also reviewed and used as input material to the interviews. These documents were also coded with Atlas.ti.

4 Results

Outcomes of our analysis are 34 factors in twelve categories within three areas (see Figure 2). Team is a core concept of SD and the focus is on strengthening the team by giving them the mandate to act autonomously. Teams were the first to be identified as one of the three. Communication is another core concept of SD. Lead in/Lead out consists of those activities in the early phase together with the work done by the Program Activation Board which leads in to a go-decision and a project for development by a cross-functional team. Lead in/Lead out also involves all activities from when a development team starts to make preparations for handing over the feature to integration, verification and release projects.

4.1 Teams

Team integrity

One of the cornerstones in SD is small and efficient teams that take an-end-to-end responsibility. This is recognized by Ericsson which lists several success factors for the implementation of SD: *Strive to allocate people to one team only. Keep teams together for at least 6-12 months and to have them co-located within the team.* The factors that restrict speed in the category of *Team* integrity and stability are:

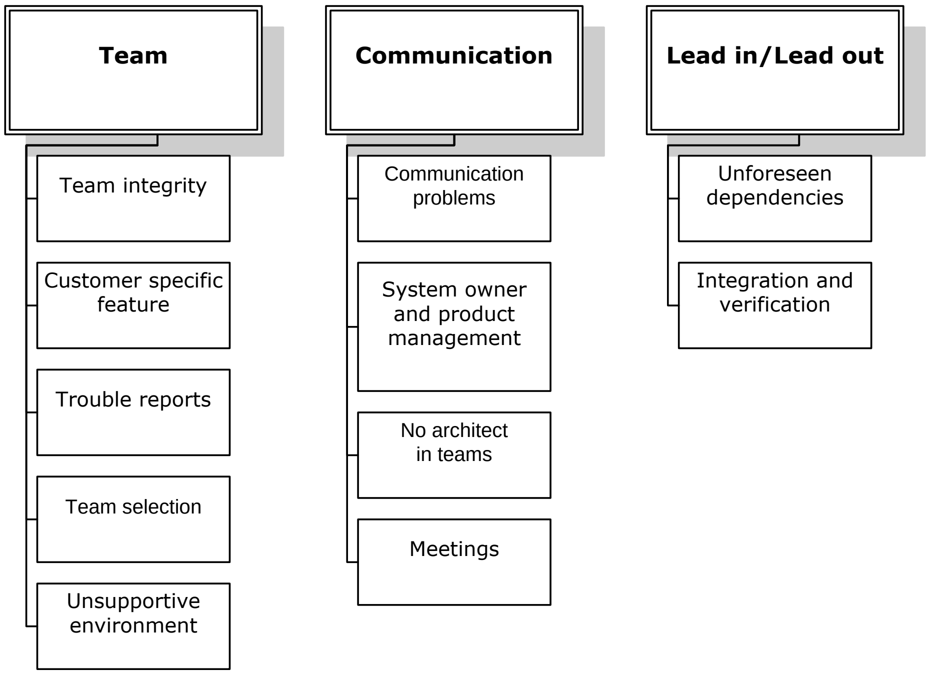

Figure 2: Areas and categories of factors

F.1 One respondent said teams usually work with 4-5 features at the same time.

F.2 The whole team does not always start at the same time.

F.3 The input from the functional test architect can sometimes come too late when team resources are unbalanced during a project.

F.4 Product Management makes changes to resources or to the feature after commit.

F.5 Changing team members during project.

F.6 Change of system test resource during project.

❝ Do not change the team during a feature. ❞ (Interviewee)

Customer specific features

The customer specific features are different in relation to features Ericsson AB specify themselves.

F.7 Customer specific features are often harder to break down in their dependencies.

F.8 It can be hard to merge the functionality and architecture into existing systems.

F.9 The customer usually comes with a lot of input during the project and this is considered as something that is disruptive.

❝ It's generally the customer specific features that are hardest to break down, because then you have to struggle with a solution for the customer and a good solution for our own system. ❞ (Interviewee)

Trouble reports

TR's can come at any time and having higher priority than what the team are currently working on, two factors regarding TR'r that hinder speed in SD are now presented:

F.10 TR's high priority (higher than the current feature the team is working on) makes for a disruptive element in a team's every day work.

F.11 Work on the current feature would have to stop entirely if it weren't for the split of the team and assignment of one or more persons to handle this work. The team leaders interviewed described how this is handled by permanently assigning one or more full time worker who is changed each month.

❝ We have selected one person to work with TR's. And then it is only that person who is interrupted and not the others. ❞ (Interviewee)

Team selection

There are different opinions whether or not teams should be assigned features that are outside the team's area of competence or not. Some people enjoy the learning experience of working outside a "comfort zone" and some don't.

❝ The organization has grown fairly much the last couple of years making us not reach the same level of competence throughout the organization... ❞ (Interviewee)

F.12 The competence of the next "free" XFT team doesn't always match the next feature on the Feature Priority List to be implemented and this might cause a delay in the implementation of that feature. And sometimes the opposite happens; the team is ready but the material is not.

Unsupportive environment

The test tool TTCN has been brought up several times by our interviewees and is considered a strong factor that hinders speed in SD. The tool is so hard to use, reducing the number of test cases produced daily from 4-5 to one, that teams choose to not use it or have to spend time planned for education on implementing test cases. An integrated development and testing environment is also mentioned as an issue causing delays in testing.

F.13 The test tool TTCN is so difficult to use that teams are choosing not to use it or do function tests. With JPacket teams could produce several test cases each day but with TTCN they can only produce one.

F.14 Teams want integrated design and testing environment and not separate as it is today.

F.15 Unreliable test results from testing environment. One interviewee said that tests done on seperate nodes were not conclusive because of different results.

> ❝ We still have a lot of problems with tools. ❞ (Interviewee)

> ❝ Spend some money to fix the environment problems because it cost every team a lot of time. ❞ (Interviewee)

Having no clear architecture causes added lead-time in projects through added work from unpredictable actions and responses from a system or its maintenance structure while working with it. Ericsson has acknowledged the need for an architect to be part of the team in order to provide the dissemination of high level architecture into the teams. The lack of a proper architecture is usually associated with legacy systems.

F.16 There is unclear architecture, especially in legacy systems. For instance, a small change in a legacy system may affect over a hundred test scripts that may have to be updated, causing a delay in testing. Ericsson has employed several architects that will be working closely with the teams to ensure that the architecture improves.

> ❝ The [department] does not have a real architecture today. ❞ (Interviewee)

4.2 Communication

Communication problems

This section mostly brings up communication problems outside the team towards other departments, either within Ericsson AB or externally.

F.17 Product Management sometimes fails to communicate an accurate description of a feature and when feedback from the team comes, it is often not what the Product Management had in mind.

One action towards closer communication to product management is going to be solved to some extent now when the Product Management will have two representatives located at the site. This management commitment is also recognized by Ericsson which lists; close collaboration with management during development, regular meetings, and co-location as some of the success factors for implementing SD.

> ❝ I had much better communication with the Program Management when I was sitting over there than I do now. I have noticed that we have become more isolated. ❞ (Interviewee)

F.18 Miscommunication during CPI-documentation production between development team and external company might lead to customer TR's based on wrong information after delivery.

The CPI documentation is written by an external company. One interviewee thought there was a problem with communication between the team and the CPI writers since the writers were not sitting with the team and had no insight. Several working hours were spent on phone calls giving very detailed information on what should be written about the feature in the CPI document. This was also identified as a source of even more extra work as incorrectly described features would produces erroneous trouble reports from customers, when the customers expected one described functionality and in reality getting another – making the customers act on false pretenses when writing the error reports.

> ❝ Yes it is the customers that suffer. They are the ones who read it [the feature descriptions]. And then they write trouble reports on faults they find. ❞ (Interviewee)

F.19 The release project has a hard time getting valuable information early. The release project would like to have high level description of a feature and its compatibleness with other system versions or surrounding systems to be able to work efficiently.

F.20 Test tools and their framework, developed in Shanghai and Hungary, changes without information reaching the teams.

F.21 One interviewee said that the feedback from the testing can take up to six weeks to reach the team and then that team may not consist of the same people anymore or the team members don't remember that feature anymore.

> ❝ The biggest problem is communication, without a doubt. ❞ (Interviewee)

> ❝ It is not so much who I should talk to, but that people are unavailable. ❞ (Interviewee)

System owners and product management

A System Owner is a person employed by Ericsson that is responsible for ordering a feature and responsible for approving a system's quality.

F.22 Not having the system owner co-located or closely working with the team causes uncertainty and added lead-time. The uncertainty is regarding if the implementation is going in the right direction and the added lead-time is from halting important decisions until the SO has been asked.

> ❝ but it is very important that you are close to the System Owner and the Product Management. . . so that you don't shoot wide of the mark but deliver exactly what they want. ❞ (Interviewee)

F.23 Not having close contact and regular meetings assisted by both PDU and PM being co-located can hinder the communication of overall high-level goals and increase the risk of miss-communicating the requirements correctly.

> ❝ A new thing is that we will have a couple representatives from Product Management on site and this is very welcomed. ❞ (Interviewee)

No architect in teams

Communication, dissemination of over-all system attributes, and early feedback on high-level architecture are desireable working routines at Ericsson AB and this factor influences all of them:

F.24 Having no architect in a team or not being co-located increases the risk for miscommunication high-level architectural goals and also reducing the feedback from the designers on the architecture towards the architect. It is difficult for an architect in Sweden to help a team stationed in Montreal.

> "I think so" (The respondents response to the question "Do you think it is a good thing to have an architect in the team?")

Meetings

The results show that meetings and problems relating to making them happen are factors that hinder speed in daily work.

F.25 The problem of finding a room to hold a meeting in creates problems. Team leaders have to guess meeting times several weeks and months ahead of time to be able to book appropriate rooms for important meetings at the end of a project.

F.26 The problem of getting key individuals together for one meeting is very hard.

> ❝ I would say that it is getting hold of people that is the difficult part. It is not so much who I should talk to but that people are not available. ❞ (Interviewee)

4.3 Lead in/Lead out

Unforeseen dependencies

Even though SD preferes that a team themselves decide how to implement a feature, the team is dependent on the work done prior to a commit-decision to be able to implement in an efficient manor. Taking into account the combination of focus on details whilst considering the over-all goals (i.e. high-level architecture). This section lists factors that reduces speed in the implementation of a feature in SD:

F.27 There is today no way of knowing how the reprioritization of a feature will influence the development chain. There is no way of overlooking the implications at present.

F.28 It is often that the feature description fails to take into account dependencies in a feature and correctly describe it from the beginning.

> ❝ It is general. There are very few features on which the estimates don't grow. Very, very few. ❞ (Interviewee)

F.29 Sometimes dependencies in features are identified even before requirements are set and this is written down in the material that is given as input to the Early Phase, but this is done ad hoc. This is seen as a challenge within Ericsson and they have identified that a better collaboration between node developments at system management level is required.

> ❝ So I don't think we have a formal process for it [identifying dependencies to other systems], it is more ad hoc. ❞ (Interviewee)

F.30 The quality of the early phase estimate has an impact on the lead-time of a features implementation.

F.31 It is often the case within DPI that customer requests and diverging interpretations of, for instance, protocols, cause additional lead-time.

> ❝ - Clear and defined requirements early -> quick start. ❞ Team Evaluation in Final Report

Integration and verification

This point is recognized by Ericsson which has listed speed restricting factors in integration and regression tests. Ericsson feel s that frequent anatom deliveries are needed to keep a low integration debt.

F.32 Features that are dependent on each other might be delayed at commit if testing on LSV cannot handle any more features because of failing regression tests on an earlier feature or if the feature is dependent on another feature not yet ready.

> ❝ Yes it can be. I mean we have dependencies between the features and the teams so sometimes it is not possible to test anything until another team has delivered their part. ❞ (Interviewee)

Late information to release project

Much of the research done in regards to SD does not take into account the later phases in the process of implementing a feature. Our research identifies these factors that slow down delivery of features to customer and that potentially can produce unnecessary work:

F.33 The content of a feature is decided or "locked" very late or its scope is changed by the product management and this information is needed by the release project. The dependencies of certain releases are not mapped beforehand and this causes delays in the RP. This factor makes the RP start on assumptions.

F.34 Documentation regarding the implementation of feature is not updated for each milestone forcing the RP to work with old and sometimes misleading information and this can delay IOT-tests when the interfaces for them cannot be specified and planned.

❝ The program work internally and doesn't want that much to do with the release project until they are done. Then they can say that they are ready to deliver. But the release project's different parts need quite a lot of information during the journey that we have problem getting here. ❞ (Interviewee)

❝ Our team has made the assessment that changing a "locked" feature costs too much. ❞ (Interviewee)

5 Discussion

5.1 Teams

Team integrity

Teams working on several features in parallel splits focus within teams (**F1**). When teams do not start at the same time (**F2**), the point of strong cross-functional teams in streamline gets lost. In one case the developers had already finished the implementation of the feature before the ST test cases were finished. The team was not assigned the same ST resource on full time for the whole project and that resource was changed during the project (**F5, F6**). Errors were discovered during NIV testing because the functional tests had not been conducted which is not optimal. The late input from the FT Architect came when delivery was only one week away and this made the feedback almost redundant (**F3**). When product management fails to communicate what they want in a feature (**F4**), it has implications on the integrity of the end-to-end responsible team. Having to re-plan and perhaps redo work creates a stressful environment. All these factors in this category produce stress for the team and do impact the speed of which a feature is developed and tested. It can also lead to higher development costs and lower quality in products.

One of SD's cornerstones is the strengthening of teams and Olsson (2008) also states this in several places. This is supported by the identified factors. Interviewees talk about the importance of having the whole team starts together on a feature and that those changes of team members during a project are undesirable. The point that Holmström makes about having semi-permanent teams is supported by our findings with an emphasis on not changing the team during a project (Olsson, 2008). The benefit of co-located teams is also brought up by our interviewees.

Customer specific feature

Customer specific features are generally harder to break down and identify dependencies within and this together with the challenge of finding a suitable solution for an existing system increases the lead-time during the feasibility phase (**F8**). The previously two mentioned factors in combination with strong voicing's and frequent input from the customer can potentially create unwanted disruptive situations and negatively affect speed (**F9**).

Trouble reports

Even though the benefits of a low fault density on the LSV cannot be denied, the disruptive and divisive force of the TR should not be overlooked (**F10**). The higher priority of a TR forces certain teams to split their teams and assigns one team member to a virtual TR-team to work with the TR a substantial part of their working day (**F11**). TR's reduces speed in SD.

Team selection

In a rapidly growing organization it can sometimes be hard to keep a desired level of competence throughout the company and this is reflected in the selection of teams to implement a feature (**F12**). If the next free XFT's competence does not match the highest prioritized feature, the implementation of that feature will be delayed – reducing speed in the development process.

Unsupportive environment

The test tool TTCN has been brought up several times by our interviewees and is considered a strong factor that hinders speed in SD (**F13**). The employees have not learned the new tool so it is hard to use (challenges), reducing the number of test cases produced daily from 4-5 to one, that teams choose to not use it or have to spend time planned for education on implementing test cases. An integrated development and testing environment is also mentioned as an issue causing delays in testing (**F14**). One interviewee pointed out that they had conflicting test results during a project and they decided to not test the feature on several nodes leaving some of the testing undone (**F15**).

5.2 Communication

Communication problem

Because the CPI-writers were not sitting with the team and had no insight, several working hours had to be spent on phone calls giving very detailed information on what should be written about the feature in the CPI document. This was also identified as a source of even more extra work as incorrectly described features would produces erroneous trouble reports from customers (**F18**). When the customers expected one described functionality and in reality getting another they acted on false pretenses when they wrote error reports.

The release project's momentum is halted by the lack of information from the implementation phase and this is a factor that potentially can be easy to solve by providing just a few lines of information regarding a features dependencies to other systems and which release-versions it will run on (**F19**).

When new updates of test tools and their framework are delivered, the persons that will use them have gotten little or no information regarding the content of a certain release (**F20**). This leads to questions such as: "What will there be in this delivery?" and "What have they changed in the framework?" This uncertainty cannot be in beneficial to improve speed in SD.

When teams get feedback from regression test loop on the LSV that can take upward to six weeks to be delivered (**F21**), much of the knowledge regarding that feature has almost been forgotten and the team members can also have been changed. This puts a heavier workload on teams to try and remember details regarding a feature than if the feedback was quicker.

System owner and product management

Communication is one of the strengths of SD Olsson (2008); Petersen and Wohlin (2009); Tomaszewski et al. (2008), but the improved communication mentioned in those articles is usually within the team or a discipline. The factors that influence the communication in a negative way are between different departments or phases. This is mainly due to the lack of co-location of teams to actors they depend on such as System Owner and Product Management (**F22**). Regular meetings can help reduce the risk of miss-communicating important information to teams and thusly reduce the risk of missing the overall high-level goals set by management (**F23**).

No architect in teams

This factor's (**F24**) impact on speed in SD has been reduced by the fact that Ericsson AB recently put architects to work more closely with teams in order to convey high-level architecture and dependencies into the teams. This will also provide the architects with feedback on their architecture producing a good synergetic effect.

Meetings

Even such an, perhaps initially trivial, issue as finding rooms to hold meetings has been recognized as an important hindering aspect of daily work (**F25**). Managers might have to book larger rooms several weeks in advance to hold important meetings in. These bookings are made by guessing a date several weeks in advance when a meeting might be appropriate. Another aspect identified is the problem of arranging meetings between key individuals with busy schedules (**F26**). A busy schedule is perhaps taken for granted these days but it might have adverse effects in creating a good environment for effective management of teams. One can perhaps try to identify which groups of key individuals that require regular meetings and try to set up meetings in advance.

5.3 Lead in/Lead out

Unforeseen dependencies

Petersen and Wohlin (2009) describes"Dependencies rooted in implementation details are hard to identify and not covered in the anatomy plan." This issue is also supported in our findings and maps to our category *unforeseen dependencies*. The mention of unforeseen dependencies is in regards to those software or hardware dependencies that a feature may depend on to work but has not been identified at an early stage in the life of a feature (**F27-F29**). Ericsson is now working with a document that they hope is an improvement to existing procedures. A document called an Anatomy Plan that maps dependencies between features on a feature level for two years in advance. Unforeseen dependencies discovered late in a features life creates extra lead-time and reduces speed in SD by re-planning and re-prioritization of projects (**F30, F31**). The data indicated that a key factor in reducing the unforeseen dependencies in features lie in the estimations made in the early phase and how well those estimations have been done (**F30**). It is generally hard to do cost estimations (Sommerville, 2007). At present date the estimates are usually increased by a factor of two going from early phase to commit decision.

Integration and verification

Petersen and Wohlin (2009) also describes their issue's cycle times may extend lead-time for package deliveries as if a package is not ready or rejected by testing it has to wait for the next cycle". This is supported by our research on several occasions and also by Ericsson AB that identifies that *frequent anatom deliveries to the LSV is needed to keep a low integration debt*. This corresponds to (**F32**).

Late information to release project

Teams sometimes "lock" features late in a project. This decision is sometimes taken on the grounds that it costs too much to change the content of a feature after it has been locked. It can also be tha the team feel that it is better to have the feature open to have greater freedom of its content. The release project needs information regarding what external system it interacts with and what releases of LSV it shall run on. The lack of information forces the RP to start their work on assumptions regarding the scope of a feature (**F33**). This means additional work load and reduced speed in getting the product to customer. The interfaces for the IOT-tests are also very hard to specify without this information (**F34**).

5.4 Validity and reliability

The use of Atlas.ti together with the method Selective Coding (Strauss & Corbin) to code the transcribed interviews gives this articles presented findings added validity. The unit at Ericsson where the research was conducted at has been developing according to SD for more than a year now and is quite familiar with that way of working. This helps eliminate some of the initial problems a transition to SD might entail. Several of the findings in this article may be generalized due to the fact that they are common in the implementation of other agile practices. The threat of not understanding the environment was reduced by first interviewing key individual from each phase of the development process. The interviewees were carefully selected by key persons within Ericsson in two sets to ensure a good and representative selection. The first sets of interviewees were especially selected to give answers regarding the overall way a feature travels through the idea-stage to the release project and to"frame"the second set of interviews. The last sets of interviews were with team leaders to get a more in-depth analysis and a more accurate data collection. All of the interviews were unstructured but through the use of models, charts, diagrams, and preparatory questions the authors tried to ensure a common understanding of the area questions were being asked.

6 Conclusion and further work

The purpose of this article was to identify factors that restrict speed in Streamline Development. By analyzing the results found in this research, we are able to conclude that the following factors, presented in their three areas, restrict speed in SD: *Team:* We found that when team members do not start a project at the same time the design might be done before any input has arrived from the functional test architect. When teams are working with 4-5 features in parallel and their resources change during a project, while still retaining their end-to-end responsibility, there is a possibility that the organization falls short of implementing SD to a desired extent. Team integrity play an important role in SD. If product management or system owners fail in communicating what they want in a feature it might lead to increased preassure and stress in a team, when they have to re-plan and redo work. A co-location of people one is dependent on is a key factor in solving this issue.

We have shown that customer specific features are harder to break down and hard to integrate in existing systems. One other important factor is the disruptive implications of TR's from a team's perspective. Splitting teams by assigning alternating resources to a permanent virtual TR-team.

In a organization that is expanding rapidly it is sometimes hard to retain a desired level of competence throughout the organization. This affects the speed of SD during team selection. The next available team's competence might not match the next feature in the pipeline. The feature with the highest priority. Leaving it to wait for a suitable team that are able to take on the workload.

One big factor was the TTCN test tool and old test scripts which might lead to a deterioration of testing in projects when teams decide the cost of using that tool to do tests are too high. This can impact speed in SD and quality in the product.

Communication: We identified that communication amongst team members works well. Poor communication between teams, product managers, release project, and system owners seems to have a large negative impact on speed. This issue seems to point to one main factor - distance. For some teams there are today no representatives from the product management present on the site at Ericsson where the research was conducted. This increases the risk of misunderstandings between what the product management expects from a team and what the team thinks the product management wants. This might lead to teams implementing features and missing the mark when it comes to high-level goals set by the product management. One other aspect is the importance of collaboration and synchronization on system management level is to identify potential dependencies during implementation of a feature between departments. Difficulties regarding related to sometimes having system owners in other countries reduces the important face to face communication that teams sometimes need. Another aspect of disseminating high-level goals is the importance of having an architect that works closely with the teams to ensure the team follows the over-all architectural plan.

We conclude that miscommunication during CPI-documentation exists and that it also can lead to customers writing TR's, further increasing the workload and reducing the speed, based on erroneous information. TTCN mentioned in section **5.4** also have implications for speed that are the result of bad communication. Content of framework och interfaces in new releases of the test tool are not communicated to the testers reducing speed when that information has to be tracked down. This is of the reasons that the production of test cases drops from 4-5 a day (with the old tool) to a single test case a day.

Communication also have negative influences further along in the development process. The release project suffers from not acquiring relevant information early because teams either want to "lock" a feature late or think it is to expensive to do this early. The release project needs information regarding what dependencies a certain feature has to external products and what system versions it will run on. This lack of information also affects the IOT-tests by delaying them when the release project lack the information to design the tests and interfaces towards other systems.

Lead In/Lead Out: unforeseen dependencies discovered after a commit-decision will reduce speed throughout the whole development chain and there is no formal process for handling it. Ericsson is dealing with this issue by using an anatomy plan. Our research also indicates that features might be held up if the regression test loop during integration and verification on LSV fails. One thing that surprised the researchers was the late feedback from the integration test loop that could take up to six weeks to come back to the team that implemented the tested feature. The team responsible for implementing that feature might not have the same members or for that matter, remember the feature. This in combination with high workloads makes it very hard for teams to quickly deal with late arriving TR's.

The release project is often delayed because of lack of important information early and have to start on assumptions and larger scope than might be necessary and also not being able to define interfaces for, and to plan, IOT-tests in time SD focuses very much on the team being autonomous and being able to plan their work on their own. This is the strength of SD and what our research have shown is that applying SD into a large organization with established routines is a challenge, even for a world leading telecommunications company like Ericsson AB.

Further research could focus on how team integrity and better communication of common goals impact the speed of SD. It could also be of importance to look into if the organization around the cross-functional team can work differently to achieve a smoother handover of features and responsibility throughout the life of a feature. Further, more research could go into the area of conflicting factors identified in this paper such as TR's versus team integrity. A more thorough estimation in the early phase with greater understanding of feature dependencies early versus more autonomous team.

Acknowledgements

The authors of this research would kindly like to thank Ericsson AB for providing valuable introduction to and seminars on SD. Helpful documentation and statistics provided by the company has been of great importance for this research. The authors would also like to thank the appointed supervisors for this research. Their knowledge and guidance has been invaluable. Also, a special thank to all people at Ericsson AB who kindly agreed to be interviewed, as well as all the people at the IT-University who helped and gave advice for this research.

References

Booth, W., Colomb, G., and Williams, J. (2003). *The craft of research*. University of Chicago press.

Cockburn, A. (2002). *Agile software development*. Addison weasly.

Cockburn, A. (2007). *Agile software development: the cooperative game*. Addison-Wesley.

Creswell, J. (2008). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Pubns.

Eisenhardt, K. (1989). Building theories from case study research. *Academy of management review*, 14(4):532–550.

Goulding, C. (1999). *Grounded theory: Some reflections on paradigm, procedures and misconceptions*. Citeseer.

Hancock, B. (2002). An introduction to qualitative research. *Trent Focus for research and development in Primary Health Care*.

Nerur, S., Mahapatra, R., and Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):78.

Olsson, H. (2008). Acting Agile in Streamline Development. *pending*.

Petersen, K. and Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, 82(9):1479–1490.

Silverman, D. (2009). *Doing qualitative research*. Sage Publications Ltd.

Sommerville, I. (2007). *Software Engineering. Eighth*. Addison-Wesley. ISBN 0-321-31379-8.

Strauss, A. and Corbin, J. (1997). *Grounded theory in practice*. SAGE Publications, Inc.

Strauss, A. and Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage Publications, Inc.

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.