CHALMERS

Evaluating the use of diagnostic services and the efficiency of the software development process

Master's Thesis in Product Development

JACOB DAOUD

Department of Signals and Systems
Division of Automatic control, Automation and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2014

Abstract

The number of diagnostic services increases whenever internal or external customers inquire additional system functionality. Consequently the list of services is constantly growing and therefore more efforts are put in to develop, implement and test the services. One concerning issue was whether or not all of the diagnostic services included in the requirement table are actually being used. A quantitative study was conducted by collecting and analysing a large amount of diagnostic data. This project work confirms that the majority of the diagnostic services are in fact used.

The main objective of this thesis was however to map the software development process and to identify areas of improvement. The particular part of the software development process which involves diagnostic services is mapped. This was done through a qualitative study involving several participants whom were specifically chosen for their role in the development process. Two different maps are described; one for internal and one for external designed ECUs.

The most common issues identified in this study concerning the software development process are categorized in the following way; *Diagnostic Service and Implementation*, *Requirement Management* and *The Waterfall Model*. These were identified as problems from a Lean perspective. The qualitative study used to map the software development process made it also possible to determine the main areas of improvement.

Sammanfattnig

Antalet diagnostiska tjänster ökar varje gång interna eller externa kunder kräver ytterligare systemfunktionalitet. Följaktligen ökar listan över tjänster ständigt och därför fler insatser sätts in för att utveckla, implementera och testa tjänsterna. En av frågeställningarna var huruvida diagnostjänsterna som ingår i kravlistan faktiskt används. En kvantitativ studie utfördes genom att samla in och analysera en stor mängd diagnos data. Detta projektarbete bekräftar att majoriteten av de diagnostjänster faktiskt används.

Huvudsyftet med detta arbete var dock att kartlägga mjukvaruutvecklingsprocessen och för att identifiera förbättringsområden. Det är i synnerhet utvecklingsprocessen som involverar diagnostjänster som har blivit kartlagd. Detta var möjligt genom en kvalitativ studie som involverade ett antal personer speciellt utvalda för sin roll i utvecklingsprocessen. Två olika kartor beskrivs, en för interna och en för externa utvecklade styrenheter.

De vanligaste problemen områden som tas upp i denna studie avseende på mjukvaruutvecklingsprocessen kategoriseras på följande sätt, *Diagnostjänster och implementering*, *Kravhantering* och *Vattenfallsmodellen*. Dessa identifierades som problem ur ett Lean perspektiv. Den kvalitativa studien som utfördes för att kartlägga utvecklingsprocessen möjliggjorde identifieringen av förbättringsområden.

Acknowledgment

I wish to acknowledge those whom it would not been possible to write this master thesis without, people that have supported me throughout the project, to only some of whom it is possible to give particular mention here:

Above all, I would like to thank my family whom are my source of energy and whom have always been there for me.

I am deeply grateful to my supervisor at Scania whom have guided me in the journey that is embedded systems.

I would like to thank my supervisor at Chalmers whom have truly cared for this project.

Thank you all for making this an incredibly rich experience.

Yours respectfully

Jacob Daoud

Contents

1 Introduction

The following chapter presents a background to this master's thesis by providing a context of which the problem is experienced. The purpose of this rapport is then explained along with three posed research questions. Finally, the limitation with the work is explained followed by a short company description.

1.1 Background

Product development associated with software is advancing, the systems that are being created seems to become more and more complex. To be able to compete in the world of automotive, the increasing demands on reliable products with shorter time-to-market have to be met. The development of products can become more reliable by improving the development process. The software systems at Scania are designed in a cross functional matter which not only makes it technically but also organizationally complex. Measuring long-term business success with the level of technology and process maturity as is assumed by most software assessment approaches is a risk. These two factors are not the best indicators of success and may even be quite misleading. Detroit had mature development and production processes in the 1960s; still it lost a great deal of its market share to the superior value and quality of the Japanese automobile of the 1970s and 1980s. Detroit made an attempt to reclaim its position in the market with their famous robotic-automation and other experiments within technology, without success. Even the German automaker Porsche with their famous technical expertise did not protect it from a grave fall from 1989 to 1992, in which the car manufacturer lost 72 percent of its unit sales (Middleton and Sutton, 2005).

The Japanese superior achievements in the automotive industry during the same period were not based on process definition or on high technology; it came from having an idea of what the industry itself is all about. They could not rely on technology as it was inferior to the U.S manufacturers; they instead adopted their lean worldview and practices (Womack & Jones, 1991).

It has been argued whether the lean paradigm will produce the same kind of results within software as it did in automotive. Though many people assume that the software industry is fundamentally different from others, it does share certain characteristics with most industries. Waste being added into production processes is one factor. Lean production puts a lot of attention into removing waste from industrial processes. The techniques for analyzing, identifying and reducing waste may be applicable in any system and industry. Waste is waste no matter what industry (Middleton & Sutton, 2005).

The Japanese term for continuous improvement is called Kaizen and is used as a tool in Lean. It is the process of making incremental improvements no matter how small. The purpose of lean is to eliminate all waste which adds cost without increasing value. A part of Kaizen is collecting and analyzing data in order to improve a process (Liker, 2004).The next step is to adjust to the changes made and improve the standards. Successful management on a day-

to-day basis boils down to one precept; maintain and improve standards. This means not only adhering to current technological, managerial, and operational standards but also improving current processes in order to achieve a higher standard (Imai, 2012).

1.2 Purpose and Research Questions

The purpose of this report was to map the part of the software development process that includes diagnostic services and identify any potential areas of improvement. It also meant finding possibly unused mandatory diagnostic services. To be able to draw any conclusions three research question needed to be answered. The first research question was posed in order to reveal potentially unused diagnostic services which at Scania would be regarded as waste. This since each diagnostic services requires effort to develop and integrate and hence a waste if not used. The other two questions focuses more on the process of developing diagnostic services and the problems that may occur. This also meant finding potential waste though now in the development process. Answering these three questions would identify both waste in the product and the development process and therfore be a base for future improvements.

The three research questions that were posed:

1. To what extent are the diagnostic services being used?
2. How is the software development process structured?
3. What areas of improvement may be identified in the software development process?

1.3 Method and Limitations

To be able to answer the research questions that are the bases for this thesis a thorough study was needed. A quantitative study method was performed to find out which diagnostic services that were not being used. In order to answer the other research questions, concerning the software development process, a qualitative approach involving several Scania employees was necessary.

The frequency analysis made on the use of diagnostic services was based on a limited amount of data and it was therefore difficult to draw definite conclusions from it.

To address research question number two and three a qualitative study involving ten people at Scania was made. This was considered enough to be able to discuss the two research questions.

1.4 Company description

Founded in 1900, Scania has become world leading truck and bus manufacturer. Though the company produces buses and marine-and industrial engines their

most important product is the heavy trucks (figure 1) which wheigh over 16 ton. The Headquarters and Research and Development are located in Södertälje, Sweden. Södertälje is also the location for one of the production sites that Scania have. Scania's objective is to be a profitable organization by putting the customer first and continuously improving quality (Scania AB, 2013)

Figure 1: Scania Trucks (Scania Image Archive, property of Scania CV AB (publ)).

2 Introduction to diagnostic communication

The master thesis was conducted in collaboration with the group Diagnostic Architecture and Product Data which is a part of the System Development department. The following chapter gives a wide and comprehensive view of the area that the master thesis addresses. A thorough understanding of the diagnostic communication is needed in order to comprehend the results of this thesis.

Electronic control units (ECUs) are micro controller-based modules which work according to the input-processing-output principle (IPO model). This means that they translate input signals into output signals. Two examples are the shifting of the gears in an automated transmission and the speed regulation of the wiper blades. To develop a specific ECU for a vehicle such as a truck, the functions of the ECU are first decided. There are different systems that can be controlled; mechanical, hydraulic or electronic components. Each component has the capability of implementing several groups of functions into one hardware. To make an input signal available for all functions and enable data communication among several ECUs, vehicle manufacturers created a serial bus system and Bosch created another type of serial bus system called, CAN. Today, CAN is the standard at Scania. An example of the effect of a serial system like this is that there is no need for more than one sensor for each function as the data can be used by all the ECUs. Otherwise, each ECU would require one sensor. Diagnostic communication has made it possible to detect errors which are stored in a memory. There are both on-board and off-board communications. The main difference between them is that off-board diagnostic communication will not work without the use of a communication protocol (Marscholik et al, 2008). This thesis work only considers off-board communication which will be presented in the next paragraph. The figure below (figure2) shows the placement of ECUs on a truck, the numbers are explained in table 1 on page 7.

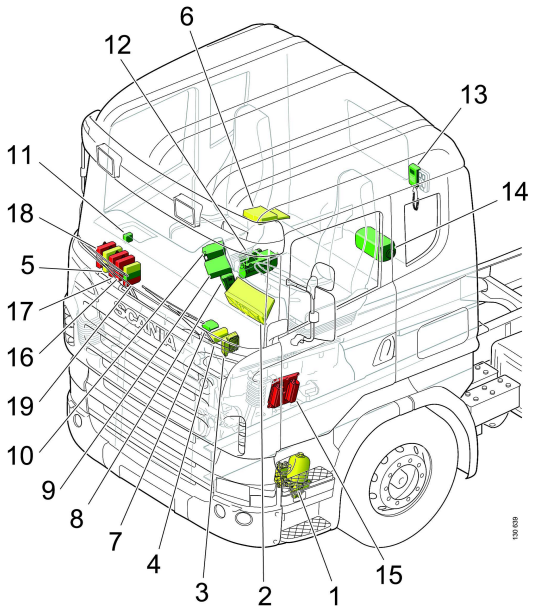

Figure 2: Placement of ECUs on a truck. Each ECU is presentend in table 1 (Scania Image Archive, property of Scania CV AB (publ)).

Data Communication

The information exchanged between humans and machines must be transformed or translated in both directions. This is the purpose of the transformer with implemented communication protocols. It enables human interpretations of the data sent by the ECU (figure 3).

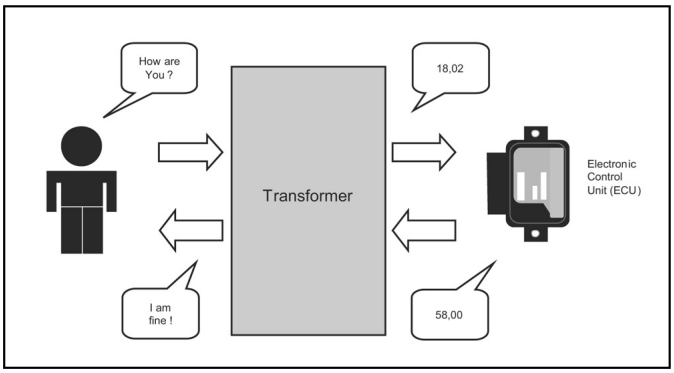

Figure 3: Human-Machine interpretation (C. Marscholik, P. Subke).

Off-board communication is created through machine to machine communication, using a PC as a sender and the ECU as the receiver (figure 4).The information that is being sent consists of different sorts of data commands and can be found in communication protocols.

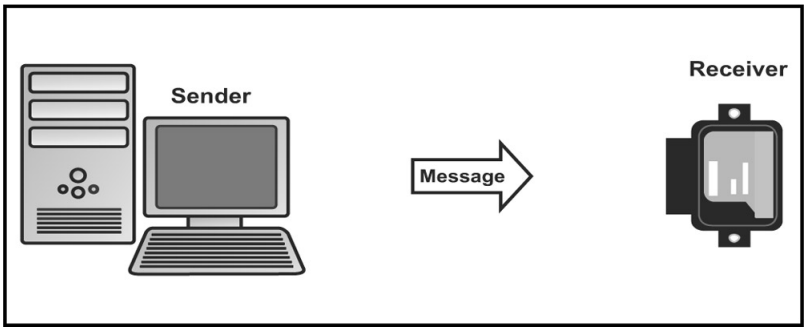

Figure 4: Communication between a sender and a receiver consists of data commands that are included in the used diagnostic service protocol (C. Marscholik, P. Subke).

This thesis work is primary about diagnostic services that are being sent (requested) although it should be understood that for every request there is a response (figure 5).

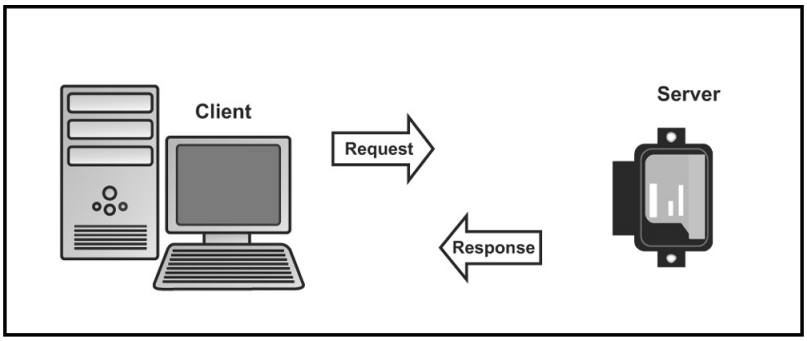

Figure 5: Request and response messages are sent back and forth. This can be compared to a question and an answer (C. Marscholik, P. Subke).

The information that is sent back and forth is structured into seven layers of communication. Each layer has its function of translating a message (from input to output information). It is called the OSI Reference Model, which is an ISO standard. However, not all companies have the seven layers structure. Scania's layers are presented later in this chapter.

Controller Area Network

Messages are sent to end up on the Controller Area Network (CAN) which is a serial network technology. It was first developed for the automotive industry but became a well-used bus in automation industries as well as other industries. The CAN bus is primarily applied in embedded systems. The network provides fast communication among micro controllers up to real-time requirements and eliminates the need for the much more expensive and complex technology of Dual-Ported RAM, (Voss, 2005). The ECUs in the system are programmed to write/send specific messages continuously in the CAN-network. They are also meant to read some of the messages send by other ECUs. The benefit of having the units connected together is that both the driver and the mechanic can receive more information than usually. It also makes the troubleshooting easier and faster. The CAN-network on a vehicle can, depending on its specification, include up to 20 ECUs (table 1). To minimize the risk for exhaustion Scania has chosen to distribute the ECUs over three CAN-buses. The most important ECUs such as BMS, COO, EMS and GMS are connected on the Red CAN-bus. The others are on either the green or yellow CAN-bus (figure 6). The diagnose program is connected directly to the green bus (Scania AB). The table shows the three CAN-buses and the included ECUs (table 1).

Applications

The CAN-network may be used for all sorts of purposes. The main advantage of a bus system like CAN lies in the reduction of expensive and maintenance intensive wiring and in the increased performance of a multiprocessor system. CAN is especially suited to be the serial communication system of choice when excellent performance is a must (Voss, 2005). It is used on passenger cars, trucks, buses, trains, aircrafts, factory automation, medical equipment and many more applications.

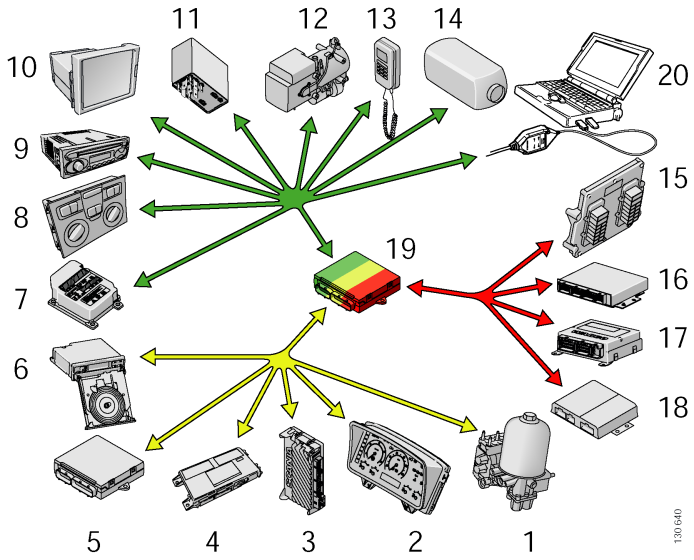

Figure 6: The ECUs on a truck are divided into three types of CAN-buses; yellow, green and red (Scania Image Archive, property of Scania CV AB (publ)).

	Function	Denomination	CAN - Color
1	Air pressure system	APS	Yellow
2	Instrument cluster	ICL	Yellow
3	Visibility system	VIS	Yellow
4	Locking and alarm system	LAS	Yellow
5	Body Work System	BWS	Yellow
6	Tachograph	TCO	Yellow
7	Crash safety system	CSS	Green
8	Automatic climate control	ACC	Green
9	Audio system	AUS	Green
10	Computer	PC	Green
11	Road transport informatics	RTI	Green
12	Clock and timer system	CTS	Green
13	Auxillary heater system - Air to air	ATA	Green
14	Auxillary heater system – water to air	WTA	Green
15	Engine management system	EMS	RED
16	Brake management system	BMS	RED
17	Suspension management system	SMS	RED
18	Gearbox management system	GMS	RED
19	Coordinator	COO	RED

Table 1: Example of functions and ECUs on the CAN network

2.1 Scania's Diagnostic Communication Model

The layer reference model that Scania has applied is presented in the following chapter. Every layer is described from the diagnostic program to the diagnostic codes being on the CAN – network (figure 7).

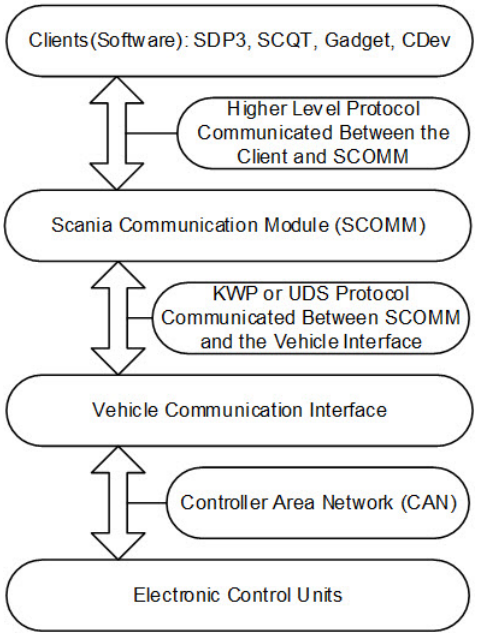

Figure 7: This model describes how the diagnostic services are transferred from the diagnostic program to the server of each ECU.

2.1.1 SDP3

Scania's software for diagnostics is called Scania Diagnose & Programmer 3 (SDP3). It is a computer based tool for troubleshooting and programming of the CAN-based part of the electrical system on trucks, buses and industrial and marine engines. SDP3 reads the vehicle specification in order to present the chassis-specific information. It also contains information about all fault codes which are required for troubleshooting the vehicles. SDP3 is connected to the vehicle's diagnostics socket which is located on the green CAN bus (Figure 8). This means that the client communicates directly with the control units on the green CAN bus. The control units on the red and yellow CAN buses are linked through the coordinator (COO). During the thesis work, SDP3 has been interesting because it can tell which diagnostic services are used when performing diagnostics on a vehicle (Scania AB , 2013).

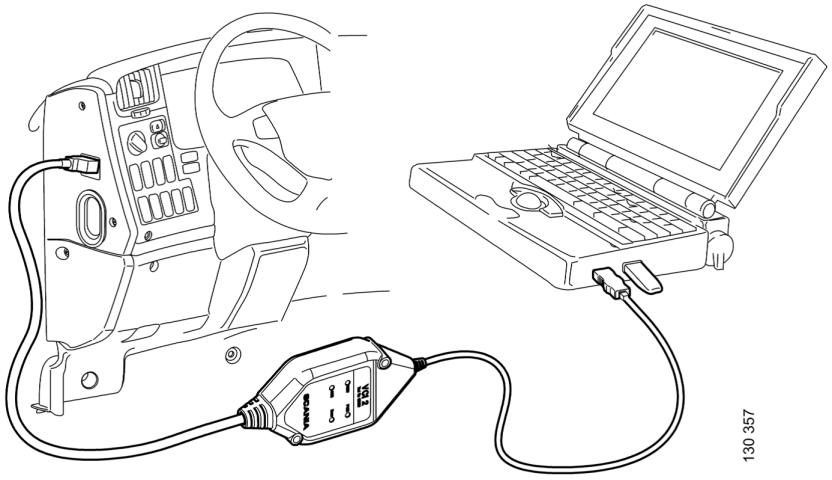

Figure 8: A PC with the diagnostic program SDP3 installed is needed to troubleshoot a truck (Scania Image Archive, property of Scania CV AB (publ)).

2.1.2 Scania Communication Module (SCOMM)

SCOMM is a central diagnostics communication component at Scania which is used whenever a PC program needs to communicate (in KWP or UDS) with an ECU. Its purpose is to translate or transform one type of code to another.

2.1.3 Vehicle Communication Interface (VCI)

VCI is the interface used to connect vehicles or/and industrial and marine engines with a computer. It may be done through a wireless network or with a USB cable. The image shows two different generations VCIs , the third generation provides a wireless option (figure 9).

Figure 9: Two different generations VCI. With the VCI 3 there is the option of wireless connection (Scania Image Archive, property of Scania CV AB (publ)).

2.1.4 The communication protocols

The exchange of diagnostic data from a client to ECUs and back is only possible if a diagnostic protocol is implemented and utilized. Scania requires diagnostics services to be implemented in ECUs according to the Keyword Protocol (KWP) or the Unified Diagnostics Services (UDS). The protocols are a set of requirements that an ECU needs to handle (ISO documents, ISO 14230 entitled "Road vehicles - Diagnostic systems - Keyword protocol 2000" and ISO 14229-1 entitled Unified Diagnostic Services). A diagnostic service consists of request messages which engineers at Scania send or a response message which is an answer from the ECU to the tester. Both types of services have unique service identifiers (IDs or SIDs). They contain a byte coding and hexadecimal values. There are three different types of messages (figure 10).

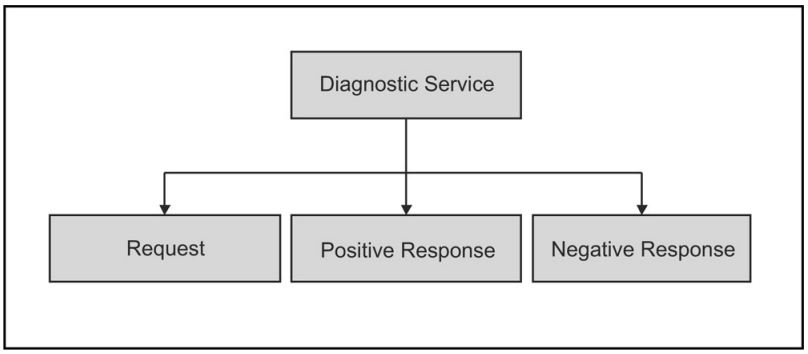

Figure 10: Diagnostic services consists of three types of messages; request, positive response and negative response (C. Marscholik, P. Subke).

Example on common protocol services

DiagnosticSessionControll(10hex)
This service is used to enable the diagnostic session with the servers that are needed to be diagnosed. A session enables a specific set of services and/or functionality in a server. Certain conditions needs to be fulfilled to start a new diagnostic session. An example of such a condition is safety, for a session to be started, the vehicle must be shut down.

EcuReset(11hex)
This is used when clients requests a reset in the servers. Here the vehicle either needs to be motionless or have a speed below a certain limit.

Example on a KWP specific service

ReadEcuIdentification(1Ahex)
This is a service used to request identification data from the server (ECU).

Example on a UDS specific service

CommunicationControl (28hex)
This is a services that is used when the client switches or turns-off the sending or receiving of some messages, like application communication messages of a server.

3 Lean Software Development

This chapter describes well-established theories in product and process development that have had a great influence on this work. The terms "Agile" and "Lean" are not always clearly defined in different software development literature (Wang et al, 2012). To fully understand this project it is important to comprehend "lean software" and how it differs from "Agile" practices. A description starting with the origin of the concept, the lean principles and ending with lean practices is given.

It all began in the early 1980's when the world recognized something special about Japanese quality and efficiency. When Japanese cars were compared with American it became evident that the Japanese cars lasted longer and required fewer repairs. Later it was also clear that Toyota designed autos faster, with more reliability and at a competitive cost. Another impressing factor was that every time Toyota showed a weakness and seemed vulnerable the company suddenly solved the problem and made an even stronger comeback. The reason Toyota is continuously delivering products with high performance is due to operational excellence. Based on a couple of tools, such as, just-in-time, kaizen, one-piece flow, jidoka and heijunka that have become famous by Toyota's success. These techniques led the way to the "lean manufacturing" revolution. But the tools would not have an effect if Toyota did not know how to implement them. It is about continuously being a learning organization by having a business philosophy that is based on knowledge of human behaviour-and motivation, as well as good leadership and strategies (Liker, 2004).

Muda, the Japanese word for "waste", is often used in lean contexts and is defined as any human activity that does not create value. Waste includes; mistakes which require do overs, defected products that cannot be sold, steps in the process that is not needed, unnecessary transportation and movement, waiting on delivery from other parts of the company and products and services that does not meet customer needs. The Toyota Production System is a manufacturing approach that laid the foundation for lean production and has dominated manufacturing the last decade. The Toyota Way is described with 14 principles divided in four categories; Philosophy, Process, People and Problem Solving (Figure 11).

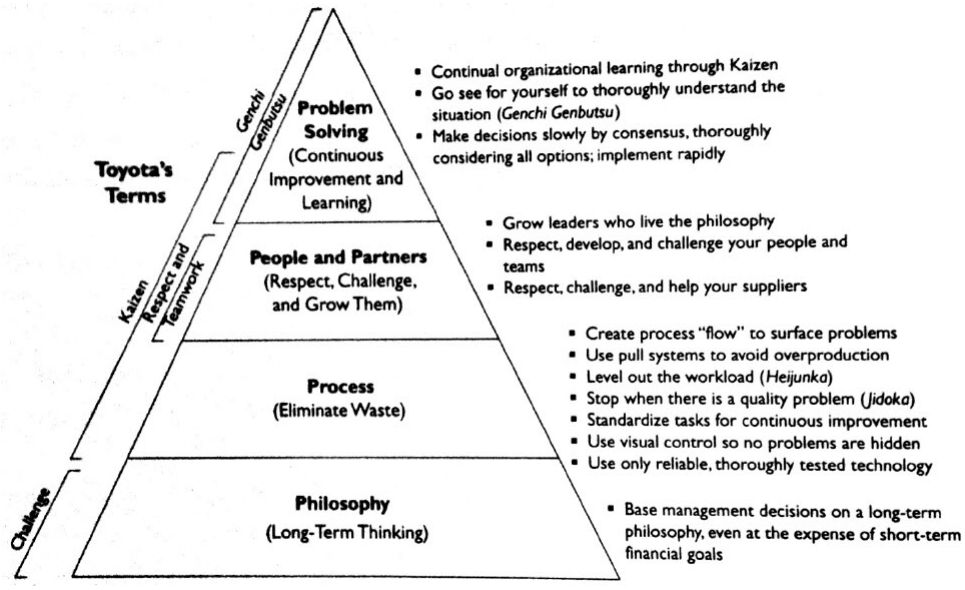

Figure 11: The traingle shows the 4P model; Philosophy, Process, People and Partners and Problem Solving (Liker, 2004).

3.1 The Lean Software Concept

Software is a product that is purchased either as a standalone product like a game or a computer program, or as a part of another product embedded in hardware. Most useful software is embedded in something larger than its code base. The software development is in other words a sub process of the product development in which the software is embedded. Software changes continually and modifying production software tend to ad complexity and increase expenses. Lean thinking enables companies to define value, map value-creating steps and perform these more effectively (Wang et al, 2012).

The same as in lean manufacturing the primary focus and principles of lean software is the identification and elimination of waste. Since waste is defined as anything that does not add value, the first step is to develop a keen sense of what value really is. When value is understood the next step is to work on really acknowledge what waste is (Poppendiech and Poppendieck, 2012). In software development, there are seven types of waste; *extra features*, *waiting*, *task switching*, *extra processes* , *partially done work*, *movement*, *defects* and *unused employee creativity* (table 2). The objective of this thesis work was to identify waste and so the part of lean that is about optimization and continuous improvement is not included.

Manufacturing	Software Development
In-Process Inventory	Partially Done Work
Over-Production	Extra Features
Extra Processing	Relearning
Transportation	Handoffs
Motion	Task Switching
Waiting	Delays
Defects	Defects

Table 2: The table presents the seven wastes in manufacturing and software development (Poppendiech and Poppendieck, 2012).

3.1.1 The Seven Principles of Lean Software Development

Principle 1: Eliminate Waste

Lean software is about removing all non value-adding work, to do that, the first step is to recognize it by separating value and waste. In the world of software it is often not that simple as customers do not really know what they want. Inventory in manufacturing is considered as waste. It ads complexity and the aim is to store as little in inventory as possible. Software inventory contributes to waste in the form of partly done work. Inventory gets lost, grows without anyone knowing it, problems do not get dealt with and it ties up money. In software development waste is found in requirement management and involves text-and-fix cycles that often take twice as long as what was planned. It could for example be about coding starting long after the requirements are set. Coding usually creates the need to change the requirements or ad new ones.

But a far worse source of waste in software development is when a feature is not used regularly which is a common problem. It is only about 20 percent of the functions in software that are in fact being used. This leads to unnecessary high costs because of features that are not needed at all. These extra features will make coding more complex, besides the efforts that is required to implement extra code, with time it will be expensive to maintain. Everything needs testing whether if it is used or not, as long as it is included in the code.

Principle 2: Build Quality In

Product development, in terms of lean, demands discipline. The goal is to eliminate the need for tests by focusing on getting the code right from the beginning. This kind of work is highly demanding and requires a skilled organization to achieve. Companies can fix defects in two ways, either to control the properties to prevent defects or to take care of them after the fact. To keep up with defects, companies use some sort of program that tracks all the detected problems. Instead, companies could eliminate this list by building in a unit test or an acceptance test in the code. The code will not be forwarded in the process until it passes the test. An organization with discipline should improve building in quality in the beginning rather than having impressing test groups

and routines later on. The verification step is important but if it creates a lot of iterative work due to defects then the product development process is not good enough.

Principle 3: Create Knowledge

The software development process is a knowledge-creating process that makes the waterfall development model unfitting. If the waterfall model is applied, the requirements need to be frozen before the coding starts. Even if companies write exceptionally detailed design documents, in practice, most new information occurs during coding. Companies that try to predict outcomes struggle as software development is an unpredictable process. Creating a plan that makes the future more accurate will only lead to decisions being made to early which lock the course of the project making it difficult to change at a later stage.

Principle 4: Defer Commitment

Deferring commitment means scheduling a point in the process when there is no way of going back and actually freezing what have been done and move on. This moment should come as late in the process as possible, of course most of the decisions should be reversible so that changes will not affect the product or the process. When there is a lot of uncertainty this moment is usually set early in the process to get tough decision out of the way. The approach should be to leave the irreversible decision to be made as late in the process as possible while experimenting with different types of solutions. Planning is important but plans are overrated.

Principle 5: Deliver Fast

Companies that have the advantage of delivering fast often have fewer costs to think about compared to their competition. They understand the customers and have developed a quality product. Software development does not have to be slow and careful as believed for a long time. Quality can be achieved in two ways, you can per long the process and be careful or you can develop people to always try to improve their process. This is about designing products that has quality built in and working in a way that responds faster to customer needs than the competitors.

Principle 6: Respect People

Designing products that satisfies customers often involves respected leaders. Leaders that foster engaged people who enjoy working on successful products and creates a team that focus on just that, designing quality products. An organization needs to trust its employees with the responsibility of meeting the goals that are set and let the people decide what and when to do things. The

employees should be trusted of knowing their job the best and improve their part of the process when flaws are detected; this needs a sense of freedom, freedom to take a step back to watch the process they are involved in. Continuous improvement philosophy should be a part of every organization that develops software.

Principle 7: Optimize the Whole

If an organization wants to be lean the whole process needs to be optimized. Optimizing small parts of the process will perhaps only degrade the overall process. Dividing the process into smaller pieces seems natural for some, like when solving a problem by decomposing complexity. Each part of the process will then be measured and optimized alone and then put together. This, which is expected to make the whole system more optimized, has shown to have an opposite effect. Improving piece by piece will make the whole system perform worse than before. This happens because the many measurements, the real goal of the improvement gets lost in the procedure. Having fewer measurements that only involve the ones that will have the biggest impact on the whole process is the most effective way of achieving process optimization.

3.2 Traditional and Agile Software development

A classic approach to software development is referred as the waterfall model (figure 12) which defines different phases. The model involves every software development stage starting with planning and finishing with deployment. The model assumes that the segments are isolated but that is rarely the case. There are usually problems that occur along the way such as; new requirements, bugs that are found or if something is wrong with the system design. This is when communication between units working together becomes crucial. Projects are often toughest to handle in the end of the test phase. The last testing team will keep finding more bugs than anticipated. The later a problem is detected, the higher the costs (Stober and Hansmann, 2010).

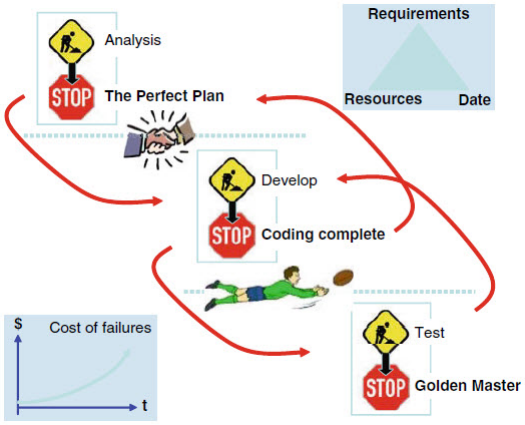

Figure 12: The principles of the waterfall model are illustrated in this figure. Decision are being made and assignments are tossed from one group to another (Stober and Hansmann, 2010).

Where the waterfall approach fails the agile software development tries to make amends. Both lean and agile are fairly used terms in software today. Lean has been explained in the beginning of this chapter but Agile has not been introduced. Although lean software is the focus of this master thesis, agile software development is applied at Scania and therefore it is important to understand how the two different principles relate to each other.

Lean is sometimes said to be a close relative of agile, or that they have nothing to do with each other. It is not totally clear and confusion about it exists when trying to compare (Coplien and Bjørnvig, 2010).

Agile software development is not only a set of rules that need to be applied in order to be successful. It is about, leadership, project management, work practices, techniques, and tools. A project that is agile do not use a standard process pattern but adapts to the shape of the process which is determined early on and by previous projects. It grows and evolves with time. Agility is combining thoughts and customizing the need of the team for the project. Different teams work in different ways, even within the same company. Applying agile development means relying on practitioners to shape the actual product development unlike classic product development, which is probably defined by top management. Letting decisions be made by employees further down the ladder will reflect how the teams actually are doing. It has to do with the desire to establish a development process that is flexible and efficient which results in quality products (Stober and Hansmann, 2010).

There are both similarities and differences between Lean and Agile development. Agile focuses on what is going on in the moment of a project and less on the beginning. At the same time, Agile does not pay enough attention on the beginning, on long-term planning and profitability, or achieving standardization. Both Lean and Agile are about eliminating defects that are discovered

throughout the product development process. The Toyota way is based on standardization while Agile is about working continuously with inspection and adaptations (Liker, 2004). Lean is also about thinking and taking actions to avoid unnecessary mistakes and especially avoiding rework, while Agile is more about doing and making rapid decisions (Coplien and Bjørnvig, 2010).

4 Diagnostic service analysis

To what extent are the diagnostic services being used

A quantitative analysis was used to study the diagnostic services, in hopes off concluding how many services actually are in use, was based on performed diagnostic sessions using SDP3. This client stores information every time a diagnostic session is completed. Logs are created containing information about requested and received messages during a diagnostic session (figure 13). Therefore accessing the logs was the key to creating a frequency analysis of the diagnostic services that are used. Hence, making it possible to know which services are not used. The log files were collected from different units within Scania to cover a long range of different kind of diagnostic services.

Figure 13: The picture shows an ongoing diagnostic session where the PC is connected to the truck (Scania Image Archive, property of Scania CV AB (publ)).

To be able to interpret the information in more than 100 files efficiently a program which could handle the data was created. The program searched all the files looking for services and created a frequency analysis. This way not only the least used services were known but also the most used ones. The program would need to count the number of times each service occurred on each file and summarize the result of all the files combined.

The output of this program became a frequency table in Excel, showing the number of times each diagnostic services was invoked.

The diagnostic services that are included in the KWP and UDS protocol

are the ones that the program used as reference data. The services in the log files are written in hexadecimal format of one byte. Most services also have sub services in which case means that it contains not only one byte hexadecimal but up to three. This is marked red in figure 14, a screenshot of a log file. The main service is 1A, which means *ReadEculdentification*. The second byte is the sub service, 0x81, *ECUIdentificationScalingTable*.

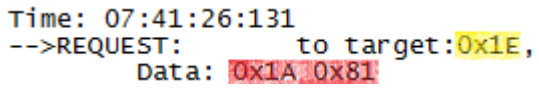

Figure 14: This is two lines of a log file. Data higlighted with red indicates which diagnostic service is used.

4.1 Frequency analysis of used diagnostic services

The outcome of the program was a table which shows the number of times each diagnostic service was invoked. Files from the units; development, production, test and aftermarket were all analysed. The results of the analysis with both of the protocols included are presented in the following table (table 3). The table is organized after the main services and the ones that are red highlighted have not been invoked.

Diagnostic Service	KWP	UDS	Invoked
0x10	✓	✓	11260087
0x10 0x01		✓	16552
0x10 0x02		✓	384
0x10 0x03		✓	40
0x10 0x81	✓		5621275
0x10 0x85	✓		6494
0x10 0x86	✓		5613834
0x10 0x87	✓		942
0x10 0x8A	✓		10
0x10 0x90	✓		88
0x10 0xF9	✓		457
0x11	✓	✓	10079
0x11 0x01	✓	✓	4368
0x11 0x02		✓	5
0x11 0x03	✓		5704
0x12	✓		61568
0x12 0x00	✓		0
0x12 0x00 0x00	✓		0
0x12 0x04	✓		0
0x12 0x83	✓		0
0x12 0xFF	✓		61516
0x14	✓	✓	235862
0x17	✓		2108475
0x18	✓		50663
0x18 0x02	✓		50117
0x18 0x03	✓		379
0x18 0x11	✓		0
0x18 0xF0	✓		0
0x18 0xFF	✓		167
0x18 0xFF 0xFF	✓		167
0x21	✓		260111
0x21 0x01	✓		213
0x21 0x02	✓		249729

Diagnostic Service	KWP	UDS	Invoked
0x1A	✓		97400
0x1A 0x70	✓		2
0x1A 0x71	✓		2
0x1A 0x72	✓		2
0x1A 0x73	✓		2
0x1A 0x80	✓		48069
0x1A 0x81	✓		40853
0x1A 0x87	✓		562
0x1A 0x88	✓		313
0x1A 0x89	✓		252
0x1A 0x8A	✓		222
0x1A 0x8B	✓		221
0x1A 0x8C	✓		214
0x1A 0x90	✓		216
0x1A 0x91	✓		236
0x1A 0x92	✓		225
0x1A 0x93	✓		222
0x1A 0x94	✓		222
0x1A 0x95	✓		222
0x1A 0x97	✓		232
0x1A 0x98	✓		5
0x1A 0x99	✓		5
0x1A 0x9A	✓		2
0x1A 0x9B	✓		2
0x1A 0x9F	✓		3814
0x1A 0xA0	✓		214
0x1A 0xA1	✓		0
0x1A 0xA2	✓		214
0x1A 0xA5	✓		214
0x1A 0xA6	✓		214
0x1A 0xA7	✓		214
0x1A 0xA8	✓		0
0x1A 0xA9	✓		0
0x1A 0xAA	✓		0
0x1A 0xAF	✓		209
0x29	✓		91
0x29 0x01	✓		0
0x29 0x02	✓		91

Table 3: The table presents the number of times each diagnostic service is invoked in the log files.

Diagnostic Service	KWP	UDS	Invoked
0x19		✓	951393
0x19 0x01		✓	0
0x19 0x02		✓	460057
0x19 0x03		✓	0
0x19 0x04		✓	0
0x19 0x06		✓	491291
0x19 0x06 0x01		✓	2819
0x19 0x0A		✓	37
0x22	✓	✓	400065
0x22 0x00 0x01	✓		121
0x22 0x01 0x01	✓		0
0x22 0x01 0x02	✓		0
0x22 0x01 0x03	✓	✓	0
0x22 0x01 0x04	✓	✓	0
0x22 0x01 0x05	✓		0
0x22 0xF1 0x80		✓	417
0x22 0xF1 0x81		✓	520
0x22 0xF1 0x82		✓	165
0x22 0xF1 0x86		✓	2677
0x22 0xF1 0x87		✓	8001
0x22 0xF1 0x88		✓	8000
0x22 0xF1 0x89		✓	5336
0x22 0xF1 0x8A		✓	7837
0x22 0xF1 0x8B		✓	8001
0x22 0xF1 0x8C		✓	8011
0x22 0xF1 0x90		✓	8001
0x22 0xF1 0x91		✓	8011
0x22 0xF1 0x92		✓	7836
0x22 0xF1 0x93		✓	7366
0x22 0xF1 0x94		✓	16144
0x22 0xF1 0x95		✓	7535
0x22 0xF1 0x97		✓	16020
0x22 0xF1 0x98		✓	5452
0x22 0xF1 0x99		✓	5450
0x22 0xF1 0x9A		✓	5100
0x22 0xF1 0x9B		✓	5090
0x22 0xF1 0xA5		✓	4127
0x22 0xF1 0xB0		✓	2505
0x22 0xF1 0xB1		✓	2151
0x22 0xF2 0x00		✓	5
0x22 0xF2 0x01		✓	93

Diagnostic Service	KWP	UDS	Invoked
0x27	✓	✓	10954693
0x27 0x01	✓	✓	11114
0x27 0x02	✓	✓	9511
0x27 0x03	✓	✓	5607014
0x27 0x04	✓	✓	5325034
0x27 0x05	✓	✓	210
0x27 0x06	✓	✓	206
0x27 0x07	✓		181
0x27 0x08		✓	181
0x30	✓		76167
0x30 0x00	✓		0
0x30 0x01	✓		74707
0x30 0x02	✓		3
0x30 0x03	✓		3
0x30 0x04	✓		9
0x30 0x05	✓		119
0x30 0x06	✓		0
0x30 0x07	✓		0
0x30 0x08	✓		0
0x30 0x09	✓		0
0x31	✓	✓	5626909
0x31 0x01		✓	311
0x31 0x02		✓	69
0x31 0x02 0x00		✓	0
0x31 0x03		✓	3749
0x31 0xFF 0x00		✓	0
0x31 0xFF 0x01		✓	0
0x32	✓		0
0x33	✓		273
0x3B	✓		13394
0x3E	✓	✓	372
0x3E 0x01	✓		342
0x3E 0x02	✓		0
0x85	✓	✓	5
0x85 0x01	✓		2
0x85 0x02	✓		2

Diagnostic Service	KWP	UDS	Invoked
0x28	✓	✓	239
0x28 0x00		✓	0
0x28 0x01	✓	✓	0
0x28 0x02	✓		75
0x2C	✓		0
0x2C 0x01	✓		0
0x2C 0x02	✓		0
0x2C 0x04	✓		0
0x2C 0x81	✓		0
0x2C 0x82	✓		0
0x2F	✓	✓	3026330
0x2F 0x00		✓	0
0x2F 0x03		✓	0

Diagnostic Service	KWP	UDS	Invoked
0x87		✓	0
0x87 0x01		✓	0
0x87 0x03		✓	0
0x87 0x11		✓	0
0x87 0x12		✓	0
0x87 0x13		✓	0
0xB1	✓	✓	233559
0x24		✓	119389
0x2E	✓	✓	122526

4.1.1 Seldom used services

The tables were reviewed and the services that were not invoked were questioned. Most of the services that were not invoked were cleared as being optional and would not be further investigated. The ones that were are presented in the following paragraph. These were mandatory services which were the main reason why they were chosen to be followed up on.

KWP services

DisableNormalMessageTransmission(28hex)
EnableNormaMessagesTransmission(29hex)

The *DisableNormalMessageTransmission* service (figure 15) is used by a client to stop the normal (non diagnostic) message transmission from the vehicle server or servers. The servers being addressed respons then with a positive message or, if unable to stop the message transmission, respond with a negative response message. The normal message transmission is re-enabled when the session is switched to the standardSession or by using the *EnableNormalMessageTransmission*

The *EnableNormalMessageTransmission* service is be used by a client to indicate to the servers that normal message transmission can be resumed. The service is used in combination with the the previous described service *DisableNormalMessageTransmission* to once again start the normal message transmission from the servers. As an exception to the general protocol rules for application layer services, the service can be used as an unconfirmed service. If the parameter responseRequired = No in the service request message, then the servers being addressed in not supose to send neither a positive nor a negative response message.
The result of the frequency analysis mad show that only sub-service 0x02 is being used in both cases. The question that arouse was, why do both of them need to be implemented or be included in this requirement document at all? These two services were flagged and were reviewed again to confirm whether the sub services are being used or not. Further research revealed that 0x01 is not being used that often. According to the SCOMM team, the 0x02 sub-service is only used when talking to all the ECUs at once. This is a usual the way of working when using these diagnostic services, however the other sub-service in question, 0x01 is used when talking to a specific ECU. If it is important to be able to act on one ECU witch these two services enables then both of the sub services are needed. On the other hand, if this rarely done then it should be considered deleting from the requirements as it is unnecessary.

Hex	Description	Cvt
01	**Yes** The server(s) shall send a positive or negative response to the request message.	**M**
02	**No** The server(s) shall not send any response to the request message.	**M**

Figure 15:

ReadDataByCommonIdentifier(22hex)

The *ReadDataByCommonIdentifier* service (figure 16) allows the client to request data record values from the server. A data record is identified by a common data identifier value. The format and definition of a data record is system supplier specific, and may include analogue input and output signals, digital input and output signals, internal data, and system status information if supported by the server. The sub service named diagnosticProtocol 0x01 0x04 is an ID that asked the ECU what type of communication protocol it uses, if it is UDS or KWP. Information that is valuable for many reason and it is a mandatory service that is not being used. It has been clarified by the SCOMM team that another way of requesting the information about which protocol the ECU uses are being used. To conclude, according to data of the frequency analysis and the information given from different groups within Scania this identifier should be reconsidered removing from the service protocol requirements documents as it is not being used at all.

0104	**diagnosticProtocol** Indicates type of diagnostic protocol. 00_{hex} KWP 2000 SSF 14230-3 01_{hex} UDS ISO 14229-1 02_{hex} -FF_{hex} reserved	**M**

Figure 16: ReadDataByCommonIdentifier from the KWP protocol

DynamicallyDefineLocalIdentifier(2Chex)

The *DynamicallyDefineLocalIdentifier* service (figure 17) allows the client to dynamically define in a server a data identifier that can be read via the *ReadDataByLocalIdentifier* service at a later time. The intention of this service is to provide the client with the ability to group one or more data elements into a data super set that can be requested via the *ReadDataByLocalIdentifier* service. The server shall maintain the dynamically defined data record until the definition is cleared or upon power down or reset of the server. All existing dynamic local identifiers in the server must be possible to be defined at the same time.
The sub service 0x2C 0x04 clearDynamicallyDefinedLocalidentifier is a mandatory service that should be an optional as it is not being used. Further investigations was made with the help of the SCOMM team which showed that this

sub services but can not be confirmed as unused.

04	**clearDynamicallyDefinedLocalIdentifier** This value shall be used to clear the specified dynamic data identifier. Note that the server shall positively respond to a clear request from the client, even if the specified dynamic data identifier doesn't exist at the time of the request. However, the specified dynamic data identifier is required to be within a valid range.	**M**

Figure 17: ReadDataByLocalIdentifier from the KWP protocol

InputOutputControlByLocalIdentifier(30hex)

The *InputOutputControlByLocalIdentifier* service (figure 18) is used by the client to substitute a value for an input signal, internal ECU function and/or control an output (actuator) of an electronic system referenced by an inputOutputLocalIdentifier of the server. The user optional controlState parameter shall include all information required by the server's input signal, internal function and/or output signal. The server sends a positive response message if the request message was successfully executed. It is up to the system supplier if the positive response message shall include controlState information which possibly is available during or after the control execution.

The diagnostic service 0x30 not showing up on the frequency analysis was regarded as quite strange which lead to another analysis with logs that was created during an extra diagnostic session. The result of the second session made it clear that *InputOutputControlByLocalIdentifier* service is in fact being used.

06	**executeControlState** This value shall indicate to the server that it is requested to execute the ***controlState*** parameter(s).	**M**

Figure 18: InputOutputControlByLocalIdentifier from the KWP protocol

StopRoutineByLocalIdentifier(32hex)

This service is used to stop an executing routine, referenced by a *routineLocalIdentifier*, in the server. The routine would have been started with the service *StartRoutineByLocalIdentifier*. The parameter *routineExitOption* in the request message may contain supplier defined data. The response message must contain the parameter routineExitStatus indicating if the routine stopped in a normal or abnormal way. The parameter *routineExitStatus* may also contain supplier defined data. This was also important enough to make a second check, it was also provoked and it appeared in the log. It is used as promised.

UDS services *ControlDTCSetting(85hex)*

The ControlDTCSetting service is used to stop and start the setting of diagnostic trouble codes, DTCs, in the servers. It is only the setting of DTCs

that is stopped, safety checks and other important tests can be carried out normally. When setting of diagnostic trouble codes is stopped it can also be used to be turned on, using this service or when the session is switched to the defaultSession. This service is equal to the KWP protocol service *ReadDataByCommonIdentifier(22hex)* and has the same conclusion to whether it should be removed or not.

InputOutputControlByIdentifier(2Fhex)

The *InputOutputControlByIdentifier* service (figure 19) is used by the client to substitute a value for an input signal, internal server function and/or force control to a value for an output (actuator) of an electronic system. In general, this service is used for relatively simple (e.g. static) input substitution / output control whereas the service RoutineControl is used if more complex input substitution / output control is necessary. This service was also confirmed to being used during a second session.

Hex	Description	Cvt
00	**returnControlToECU** This value shall indicate to the server that the client does no longer has control over the input signal(s), internal parameter(s) or output signal(s) referenced by the ***dataIdentifier***. ***controlState*** in request: 0 bytes. ***controlState*** in response: as the dataIdentifier's ***dataRecord***.	**M**
03	**shortTermAdjustment**	**M**

Figure 19: InputOutputControlByIdentifier from the UDS protocol

4.2 Answering Research Question 1

To what extent are the diagnostic services being used?

The quantitative data analysis that was generated during the thesis work consisted of a table of diagnostic services. The table shows both the frequently and the seldom used diagnostic service. Among the unused services (chapter 3, principle 1), only the mandatory was a concern to the company. The services that were said to be mandatory and were not being used was flagged and further investigated. Many of these were cleared as there was solid knowledge of them actually being used. These particular services were coincidentally not invoked and therefore lacked data. There were a group of services that required more effort to clear as used. These had to be provoked to appear on an additional diagnostic session. Finally only three diagnostic services were left standing as unused. Two of those services were sub services and only used when communicating with one ECU at a time which almost never happens when using the main service in question. Usually a sub service communicates with all ECUs at once. That leaves one service which was confirmed to be an unused service. It is a service used to determine whether or not the ECU has UDS or KWP implemented protocol which is proven using another method. Result of the analysis was positive for Scania as it was evident that most of the services that is developed, implemented and tested are in fact being used. In other words, there was not a lot of waste in terms of extra features to be found. However, this does not mean that all of the services are necessary. The frequency analysis also showed that their where services thought to be unused but that actually were. This came as a surprise, which indicates lack of knowing the services included in the protocols. The next step would be to study the functionality of each service and the sequence order that they are being used in.

Analysing the diagnostic services was rewarding in more ways than one, it made it possible to compare the two protocols. It became clear during the interviews that different individuals preferred one protocol over the other. Scania is debating if it is time for a transition, from KWP to UDS. More on having two protocols, the comparison and the identified problems related to them are presented in chapter 6.1.

5 Identifying the Development Process for Diagnostic Services

How is the software development process structured?

The software development process consists of several parts which are required when designing an ECU system or a diagnostic program. Every organization applies a specific software process adapted to their needs. The life initial cycle-model also known as the Waterfall model is a widely used process model. The Waterfall model includes a set of phases:

1. Specifications: The functions of the system are defined in detail.
2. Design and implementation: The structure of the system is designed and specific attributes are set. These are implemented using programming.
3. Integration and testing: The implementation made by different individuals/units are integrated and tested.
4. Operation and maintenance: The software is delivered to the customer and modified to repair error discovered when using the program (Sommerville, 1996).

Scania's agile product development model has some similarities to the waterfall model and therefore is it explained. The result of mapping the development process is presented here. To be able to detect waste a mapping of the process had to be done. This was also beneficial in that there would be a visual overview of the process that previously did not exist.

5.1 Qualitative Data Analysis

In order to answer the second and third research question (answered in chapter 6), mapping the product development process and identifying problems, another approach was chosen. The embedded software process at Scania is agile and creative. These may be essential qualities when working with software development but it makes a process that much harder to define. Several interviews and a workshop were held with key individuals involved in the product development process. The two different ways of eliciting information complemented each other. Where the interviews could be static, the workshop encouraged open discussions.

5.1.1 Exploratory Research

Personal interviews were held with different stakeholders, each stakeholder covering a part of the process, from requirements to testing and release. Once a list of different stakeholders was available ten interviewees representing different groups were selected. There were three major stakeholders; System owners, who acts as project leaders or coordinators for the ECUs; SCOMM engineers who applies the diagnostic services in SCOMM and Method engineers who are responsible for creating functions in SDP3 and program interface. Interviewees belonged either to the embedded systems department or to a project/product planning support organization. The average interviewee had many years of experience of either purchased or internally designed ECUs. The interviews lasted for about one hour each with an open questions guide. The following questions are some of those which were asked:

1. What is your role in the development process?
2. Can you describe a fairly ordinary day of work?
3. Who is involved in the development of diagnostic services?
4. How would you describe the development process?
5. What would you recognize as wastes?
6. Which part of the process produces most of the iterations?
7. What would improve the process?

5.1.2 Confirmatory Research

The information that was gathered from the interviews needed to be confirmed. A workshop which involved a gathering of several engineers from different projects was therefore held (figure 20). The purpose was to confirm what was said in the interviews and if possible collect new information by creating a discussion around the subject. The idea was that if people could speak freely about certain problems in their daily work then others would join in to elaborate.
Two different process maps (figure 20 and figure 21) were presented during the workshop. The participants were divided into two groups; one group discussed the development of purchased systems and the other internal designed systems.

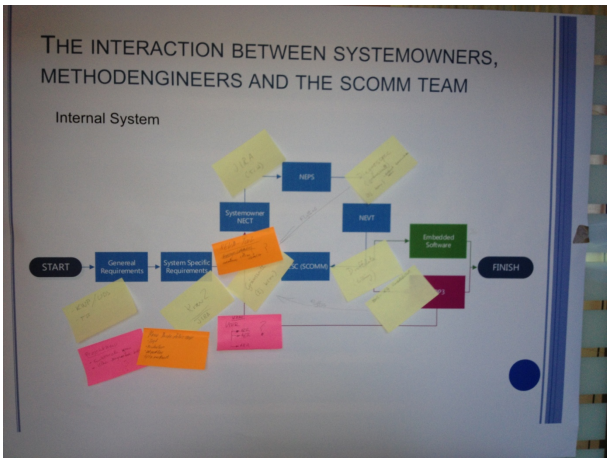

Figure 20: The result of the workshop was summarized with two posters. This poster described internal design systems

The groups put up post-its and drew on the posters to make sure that the information said within the group would be documented (figure 22).

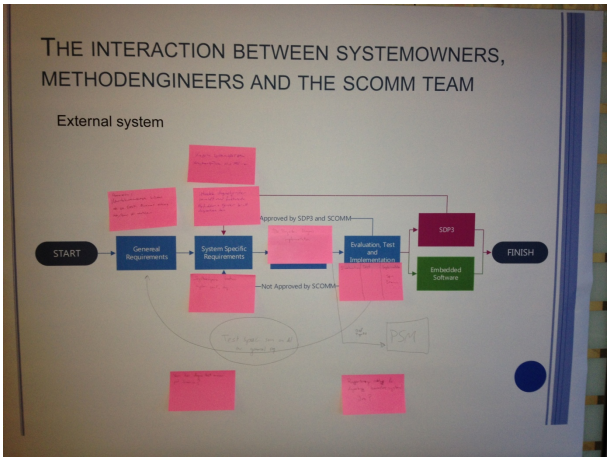

Figure 21: This poster described internal designed systems

5.2 The Development of Diagnostics Services in External ECUs

An ECU designed by an external supplier is called external ECU. An example of an external ECU is the air pressure system (APS). APS was one of the external systems that were studied when trying to first map the development process of diagnostic services. This process had not earlier been mapped which made it somewhat troubling for the stakeholders to describe.

Moreover, information concerning the way the different stakeholders worked and communicated together was now more known. The information involved requirements, specified functions, implementation and testing of the services. A general process map was created which was based on information given by the participants.

Process description

General Requirements : The process starts with a decision being made to either update an ECU or create a new one. Obviously the later brings more uncertainty to the project as the properties of the ECU are unknown. All types of possibly required specifications from different units are then gathered. If the project is about developing a new version of an already established ECU then lessons learned from previous projects are also added. The group that handles the basic diagnostics adds their requirements which can either be from the KWP or the UDS protocol. Requirements from production i.e. the services needed to properly produce the ECUs are also added. The aftermarket puts their requirements involving the services needed to identify problems and repair the vehicles in to the mix. Finally, newly recognized technologies are also added.

System Specific Requirements : The system owners have a long list of requirements that covers all angles and areas which needs to be specified. Some of the requirements are simply thrown in just in case. This approach is used due to the degree of unknown ECU properties. The next stage is specifying the requirements table which demands more knowledge about the product. The system architecture and functions are set here. This part of the process involves many necessary changes and the company strives to make these as soon as possible to front load the project. As the ECU functions becomes clearer the effects on other systems also becomes clearer. This leads to requirements being added by other system owners that are involved.

Implementation: System owners will then include the ECU suppliers that will be involved. The supplier, with a close collaboration with the system owner, has the task of implementing the services in the ECU. The supplier delivers several prototypes depending on the iterative process between the supplier and Scania.

Evaluation, Test and Implementation : The iterative process mentioned in the previous paragraph includes the collaboration between system owner, supplier

and the groups that do the testing. The prototypes are sent to Scania to be tested and implemented in SCOMM. It is usually a work in progress to get this right. Often the services are wrongly or not fully implemented in the ECU which in turn leads to the supplier sending multiple prototypes. This procedure goes on until the communication services are fully implemented in the ECU.

YS(SDP3) : Next in line to make sure that the prototype has been correctly implemented are the method engineers. They test to confirm if the results from the previous implementations can be utilized in the diagnostic program SDP3. If problems arise then the prototype will either be sent back to the SCOMM team or directly to the system owner who in turn will contact the supplier to make the changes required by the method engineers. The process map is presented on the next page (figure 22).

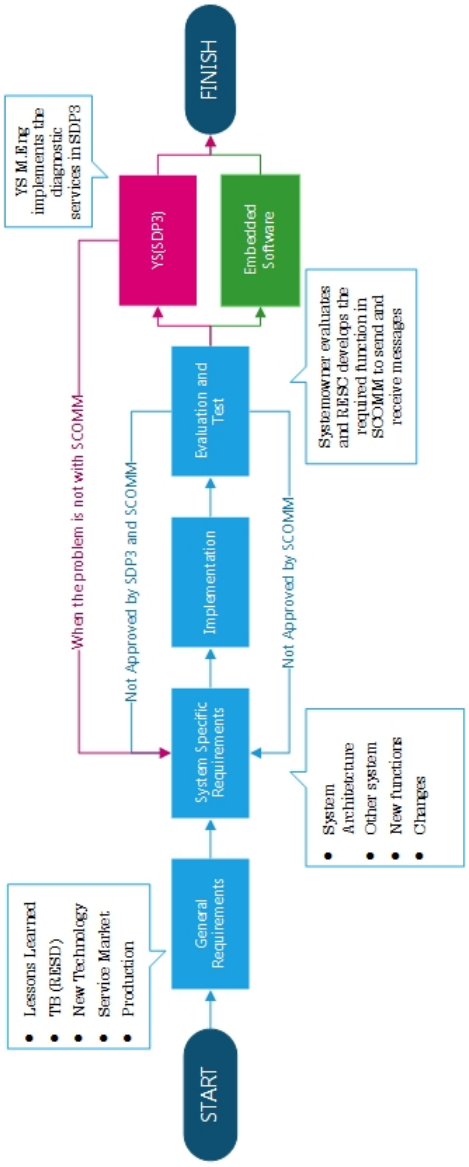

Figure 22: A process map of external developed ECUs.

5.3 The Development of Diagnostics Services in Internal ECUs

The systems that are fully designed and implemented by the Scania are called internal ECUs. They are often the most important systems, like the Engine Management System (EMS). The process of designing an internal ECU differs slightly from the external. Due to this, internal system process was also mapped.

Process description
The beginning and the end of the process are similar to an external system development process. The main difference between an internal and an external process is that there is no external supplier in the internal development process. Instead, there are several departments working together to implement the services that are required in the hardware. The groups work closely together and iterate the implementation until the method engineers takes over and make sure it works in SDP3. The difference can be seen when comparing the process maps. In this case, it was engine system groups "NE" that were mapped (figure 23).

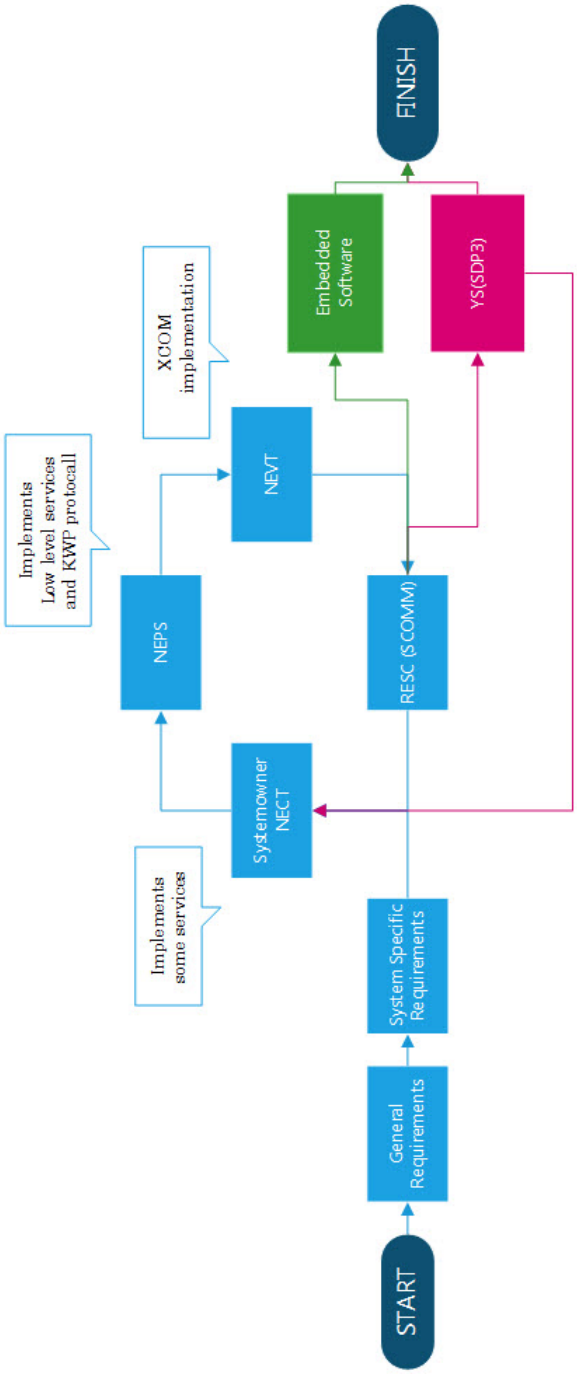

Figure 23: A process map of internal developed ECUs.

5.4 Answering Research Question 2

How is the software development process structured?

The software development process at the department of embedded system is a highly cross functional process. Even the people working there had a hard time describing the process due to that no established process map of this particular part of the larger R&D process had ever been constructed. The process would only involve diagnostic services and the development of SDP3. It meant investigating the daily work of people working at different phases of the process; beginning, middle and end. Result of the interviews was two process maps giving a general description of both external and internal designed ECUs.

The process has a fuzzy beginning when the knowledge of the system is premature. All kinds of different requirements are set to be on the safe side. The system properties are then specified together with the service requirements which are implemented in the hardware by either an external supplier or by different group within the company. The implementations are later tested against SCOMM's capabilities and later forwarded to the method engineers whom need to assure that everything is working fine when actually using the services in SDP3. There are iterations made every time a capability test of some sort fails. This is visualized as iteration loops in the process maps presented in chapter 5.2 and 5.3.

While the process was being mapped, the interviewees were deliberately given time and opportunity to reflect upon areas that could be improved. They were asked to think of changes that ought to be made to achieve a more efficient product development process not only concerning their issues but others too. Researching and creating a map of the software development process was essential to be able to answer research question three.

6 Identified areas of improvement at Scania

What areas of improvement may be identified in the software development process?

To answer the third research question the process of designing an ECU diagnostic program needed to be mapped and analyzed (chapter 5). This was possible with the help of experienced Scania engineers who were asked certain questions involving their role in the development process. They were also asked to think about different kind of problems that could occur during an ordinary day work. This was then analyzed together with observations made during the five month period at the company. The following paragraphs describe the main issues when working with diagnostic communication services.

6.1 Diagnostic Service and Implementation

The first and the most obvious obstacle standing in the way of a more efficient process is the use of two different protocols. If a company aims to be superior in diagnostic communications, logic dictates that only one protocol is to be used. Scania may be going through a transition from KWP to UDS because of UDS being a more standardized communication protocol. According to the frequency analysis made, the UDS protocol has the higher ratio of services that is actually being used. When read and observed the protocols differed in more ways than one. The KWP protocol includes many optional services and requirements with unclear definitions of when to be used. To exemplify, some of the conditional services have "only required if defined" as statement. This statement could be further clarified to better understand the requirements.

One example of this is: 0x1A 0xAA ScaniaECUCalibrationOrParameterVersionNumber#3 (figure 24)

AA	**ScaniaECUCalibrationOrParameterVersionNumber#3** This id shall contain the **Scania ECU calibration or parameter version number #3**. Only required if defined.	C

Figure 24: A diagnostic service with unclear requirement definition.

On the other hand, it has been expressed by the participants in this study that the KWP is beneficial when designing external ECUs. It was clear that there were different opinions concerning which protocol that should be used at Scania.

"It is easier for the supplier to implement services from the KWP protocol as they are used to it and it is better established for our ECUs"

Too much is being tested all at once after the services have been implemented. In addition, the testing related to quality is often given low priority and the

resources are sometimes used to test the same things twice. The diagnostic services are a cluster of different services, used for different purposes. For example, different services are needed in testing, development, production and after-market. The different services may also be divided into two groups; either system specific services which dictates functionalities in an ECU or diagnostic services. Dividing these will enable the individuals working with diagnostic services to work independently, without any delay caused by others. Nowadays everything is tested at once which creates excess lead time. One interviewee said:

"A specification for testing the services needs to be designed to prevent misunderstanding. Divide the testing and test the basic diagnostic services and system specific for themselves"

Moreover, when the method engineers test codes they sometimes encounter services that are not needed in SDP3. For example one interviewee expressed it like this:

"Old versions should be deleted automatically when testing new ones by packaging non relevant code"

According to the participants there are sometimes too much irrelevant code and that it would be easier if unnecessary parts of the code were removed. When implementing the diagnostic services, at different stages, it is often not implemented in the right way in consideration to the next customer in the process. This creates unnecessary iteration which postpones the time table and requires more effort to be put in. It is said to be one of the main problems when working with external suppliers. However it was also stated that the fault is not always with the external supplier:

"The supplier should be able to test whether or not the implementation will work for us."

Almost every participant in this study expressed that there are a lot of manual work in the software development process. Different groups work with different file formats and every time there is a translation of format the code needs to be written again. The stakeholders that participated in this study felt that this procedure should be done automatically somehow, generating an output file when choosing an input file.

"There are too much manual labor in this process. We translate one format to another by hand. This should be automatically done."

6.2 Requirement Management

In the rapid world of the automotive industry, it is now more important than ever that system development projects get a good start. The pressure is increasing on the development process due to fast-changing technology and global competition. Creating effective requirements depends on the organizations ability to lead the company and to prepare for potential obstacles that the company may encounter in the future (E. Hull et al, 2011).

To start with, there is a need to document steps taken regarding product solutions. This would benefit future Scania employees and help clear out any confusion regarding the development process. Information that allows individuals to easily trace, understand past decisions and actions within a project. A company like Scania with a large R&D department within Embedded System makes it difficult to keep track on information or even know where to look for people that possess the sought out knowledge. Even if there were detailed step-by-step descriptions of the documents they are not being handled properly. This was expressed by one interviewee:

"It is not good enough having requirements in form of Word documents, their should be a requirement management program."

The requirement specifications have in some cases been regarded as not good or explanatory enough. One comment from an interviewee was:

"The protocols are to weak and open for interpretation

This has later its effects on the ECU supplier that depend on the set of requirements which have been provided and are expected to be implemented. External systems have often said to be delayed due to lack of information regarding the properties of the ECU. This was confirmed by more than one interviewee:

"Misunderstandings with the supplier is often due to undefined requirements."

Another part of this procedure is contacting the stakeholders to ensure that everyone that should be involved in defining the requirements are participating. Both the system owners and method engineers agreed on the fact that method engineers are late in their involvement.

"The method engineers requirements are set later in the project due to lack of knowledge about the design."

The last thing to mention related to the requirements is the fact that they are sometimes added very late because of tenuous information available about the end of line properties.

"The end of line requirements are poorly executed. For example, if the vehi-

cle has three or four axles."

6.3 Waterfall model

The Waterfall process model (chapter 3.2) in a software product is seen as a linear development flow, from concept, requirements, design, code and test. This way of doing things was introduced in 1970 when systems had requirements that seldom changed unlike nowadays when there are both internal and external stakeholders. Users were not involved and therefore could not provide feedback that would lead to properties changing. This model assumes that the requirements that are set are stable and completely frozen already from start. Working accordingly to the Waterfall model in product development is still a common and widely used method in companies where software is developed. Problems that are usually associated to this way of working are for example complications that arise when change is required and mistakes are detected too late in the process. The model is associated with high costs and workload due to changes and iterations. Issues in the Waterfall model process are summarized here:

- Writing and approving documents for each development phase. These involve high effort and costs.
- Responds badly to changes
- Iteration are considered to be a lot of rework
- Test and implementation for a system are often placed in the end of the project

The processes described in chapter 5.2 and 5.3 are product development processes regarded as agile software development but with a pinch of Waterfall characteristics. These processes do not cope well with change either. Changes that are made generate a lot of rework and lead to quality problems because of the testing being late in the process. It is logical that the later the testing, the more amount of defects will be found. It has indeed a negative influence on the development process. This was expressed:

"We should move back the testing, in other words, have it earlier in the process."

The main reason for failure when working accordingly to the waterfall model has been identified as requirement management. This has been confirmed at Scania and is one of the major difficulties discussed in this report (chapter 6.2). The correction of poorly implemented diagnostic services does not only come in to late in the process but the description of the problem is not detailed enough. One interviewee said:

"The description of the errors that are sent are sometimes not detailed enough."

If a company wishes to follow a linear model like this one, then the requirements are far more important then they usually are as it is costly to change the requirements later on in the process. It was brought up during the workshop that the communication between individuals from different units needs improving. An idea was to create a team that would be responsible for each ECU which would improve the cross-functional collaboration.

"Create a team that would work closely together with the system owner, including one individual from each unit, SCOMM, CFC, SPIN, SDP3. One team per system"

6.4 Answering Research Question 3

What areas of improvement may be identified in the software development process?

The stakeholders that were involved in the study all agreed that the process needed improving though it came easier for some to mention problems associated with it. The interviewees that had difficulties mentioning problems said that they were so focused on their own work that they did not have the time to consider a process such as this which required seeing things from a larger perspective. Eventually everyone stated their opinions on what needed to be changed to achieve better products in less time.

The first and easiest identified obstacle that prevents from achieving an effective development process is that there are two different communication protocols being used. This means that time and energy are spent on developing, implementing and testing diagnostic services in two different ways. This has led to an ongoing discussion at Scania about only using one protocol, the UDS, in the future.

The services that are being implemented are not being handled in a certain order. This creates confusion among the different groups within the company as well as for the external supplier whose job is to implement the required services. To avoid this confusion the services could be sorted according to the area of use (development, test, production, after market, etc.).

This makes it easier for those involved to better understand the services and how they should be tested. Defining how the testing of the services should be done is expressed as something that is missing today. A test specification should be included with the requirements. Different groups handle service codes in different formats. This creates a lot of rework each time the codes are handled by a new group.

Requirement management is seen as a part of the process that is continually difficult to handle. It is hard to track changes and revision numbers due to that the requirements tables are written in Word or Pdf. The requirements may lead to misunderstanding between groups in case they are ambiguous. Requirements added late in the process are a consequence of insufficient knowledge about the ECU or the end of line requirements which affects the function of the ECU.

When new requirements are added late in the process necessary changes are made, which in turn leads to iterative work. These loops of iterations are demanding and postpone the time schedule by weeks. Having both good communication and collaboration between groups facilitates this procedure and is a must if Scania wants to avoid working according to the waterfall model and more like the agile process they aspire.

7 Discussion and Conclusion

In the following chapter the results achieved during this thesis work are summarized, conclusions are drawn and possible future actions are presented. The affect that lean and process improvement has on a product development process such as software is also discussed. Finally the frequency analyses on the diagnostic services together with the identified improvement areas are analyzed.

7.1 Lean applied on a software development process

Scania is a well-oiled machine that produces quality products and will most likely be doing so many years to come. The software development process presented in this thesis works well as the company relies on every single individual to come through, making his or her deadline. There is no doubt that Scania always tries to improve their product development process. In order to achieve better products a company may either slow down and be careful or develop people to continually improve their process. The company is careful not to be very process oriented since the software development process is a highly creative process and defining it too rigidly would strip it from its essence, creativity. Incremental changes are made every day though it involves mostly the products and not as much the whole software process. When improving the process there is a tendency to break it down into smaller parts and improving part by part. Because of the size of the company and the cross-functionality involved this is believed to be the logical and most effective way of improving the process. What happens is, every unit measures their own part of the process against costs, schedule, scope, and quality and customer satisfaction. This has been proven to have the opposite effect. When processes are assessed using too many measurements the evaluation may lead to misleading results.

The interviews that were held were enlightening in more ways the one, besides the answers to the questions, expressions and thoughts was also noted. It was quite easy for the interviewees to answer questions related to their own work and clarifying things that helped out with the mapping of the process. It was evident that questions related to identifying factors that were regarded as problematic in the product development process were more difficult to answer. One said:

"I haven't thought about it, I'm to busy trying to finish the tasks that was handed to me"

This was not surprising at all, not everyone is hired to think about processes and how they could be improved. But not to question the work process isn't realistic, everything can be improved. Sooner or later all the interviewees had many "problems" that they had encountered, it just took longer for some to come up with them. The embedded system organization at Scania has informal work procedures with employees empowered to work as freely and creatively as they want, as long as they deliver. It is therefore important to improve the

organization with care and without any rush.

7.2 The identified problems

Having several diagnostic protocols is most likely a temporary solution for Scania as the company is going through a transition period, from KWP to UDS. Some units are well prepared for this kind of change while others will struggle a bit. There are fewer diagnose communication services in the more compact UDS which is a standardized protocol in the automotive industry. The suppliers must also adapt to this change. The communication between Scania and the ECU supplier will be crucial in order to ensure quality products. No matter which protocol is used, the analysis made in this project shows that most of the mandatory diagnostic services are in fact being used. Hence, the implementations of the services are not being made in vain though this study only investigates the services that are used and not how they are used. In other words, this does not mean that everything is fine regarding the service requirements just because the majority of them are being used.

No matter which protocol that is used, the implementation process should be questioned and reviewed. Most interviewees mentioned the subject of manual code writing several times. The codes are translated manually from one format to another due to that different groups use different formats. During the interviewees it became clear that there is a need for more automatically generated information, especially when translating codes. It is an ongoing project; each group is working on creating easier and quicker way of sharing implemented diagnostic services. Both protocols are standard protocols but KWP is more adapted to Scania and the ECUs used there, which means that services are added to the protocol when needed. Before adding a service, the customers must be identified first to really understand what the services are for. The information about which services that were not invoked was not the only valuable information that came from the frequency analysis, it also showed that some diagnostic services were surprisingly used. This indicates that somewhere along the way, the purpose of some diagnostic services have become uncertain. Requirements that are not fully defined or fully understood are considered to be waste.

The SCOMM team who are supposed to make it possible to send and receive message through SCOMM is testing if it is possible at all, something that is said to be an extra undertaking by the team.

"Nobody does the testing of the implementation made by the external supplier, it has become our job"

A closer collaboration with the external supplier is important to make sure it is really understood how to implement the diagnostic services so that it could be enabled in SCOMM. Creating test specification to send along with the requirements is one way of doing this; another is letting the supplier try out SCOMM capabilities on their own. The goal should be to build in quality to make sure it

works when arriving to the SCOMM team instead of putting effort in to testing the prototypes. A tracking system is used to find defects but the focus should be on avoiding having anything to fill that tracking defect system with.

Building quality in the code means that not allowing for the verification stages to become the start of test-and-fix defects cycle (chapter 3, principle 2). The purpose of creating a unit test and an acceptance test in the code is that the supplier cannot deliver a prototype ECU with badly implemented diagnostic services. Furthermore, there are different phases in the software development process where tests are made and run, the purpose should be to prevent defects, not to find them. It is important to have a verification point where on occasionally a defect is found, but if it becomes a routine then there is something wrong with the implementation process.

A big part of lean that companies may not think as much about is that it requires the right philosophy and people. During this thesis work at Scania significant qualities were witnessed in the way that they develop skilled workforce at the company, knowing that buying expertise may be done by every competitor in the world. The attitude encountered at Scania during this project was quite surprising, employees at Scania are proud of their work and to be employees at Scania. The positive attitude of the employees facilitated my thesis work immensely. All of the interviewees and workshop participants were gladly there to help and provide the information needed. It was evident that the positive commitment along with the vast knowledge available among the employees is what seems to make Scania the successful company it is today.

8 References

Books

P. Middleton, J. Sutton, *Lean Software Strategies: Proven Techniques for Managers and Developers*, Kraus Productivity Organizations, Ltd, 2005.

J. Womack, D. Jones, D. Roos, *The Machine That Changed The World*, New York, NY: Harper Perennial, 1991.

J. Liker, *The Toyota Way*, McGraw-Hill, 2004.

M. Imai, *Gemba Kaizen: A Commonsense Approach to a Continuous Improvement Strategy*, Second Edition, McGraw-Hill, 2012.

W. Voss,*A Comprehensible Guide to Controller Area Network*, Second Edition, Cooperhill Media Corporation, 2005.

C. Marscholik, P. Subke, *Road Vehicles: Diagnostic Communication: Technology and Applications*, Laxmi Publications, 2008.

J. Womack, D. Jones, *LEAN THINKING:Banish Waste and Create Wealth in Your Corporation*, Simon & Schuster UK Ltd, 2003.

T. Poppendieck, M. Poppendieck, *Implementing Lean Software Development: From Concept to Cash*, Poppendieck, LLC, 2007.

T. Stober, U. Hansmann, *Agile Software Development: Best Practices for Large Software Development Projects*,Springer, 2010.

J. Coplien and G. Bjørnvig, *Lean Architecture for Agile Software Development*,A John Wiley and Sons, Ltd, Publication, 2010.

E. Hull et al, *Requirements Engineering*, Third Edition, Springer-Verlag London Limited, 2011.

Articles

Wang et all, *"Leagile" software development: An experience report analysis of the application of lean approaches in agile software development*,Free University of Bozen/Bolzano, Italy

E. Gottesdiener, *Requirments by Collaboration: Getting It Right the Firtst Time*, The Atlantic Systems Guild, 2003.

P. Laplante, C. Neil, *The Demis of the Waterfal Model is Imminent and Other*

Urban Myths", Penn State University, 2004.

I. Sommerville, *Software Process Models*, Computer Department, Lancaster University, UK, 1996.

Miscellaneous

H. Gustavsson, *Lean Thinking Applied to System Architecting*, Mälardalen University, 2011.

D. Bergsjö, L. Almefelt and J. Malmqvist, *Supporting Requirements Management in Embedded Systems in A Lean-Influenced Organization*, International Design Conference, 2010.

K. Petersen, C. Wohlin, D. Backa, *The Waterfall Model In Large-Scale Development*, Blekingen Institute of Technology, 2009.