LEAN THINKING APPLIED TO SYSTEM ARCHITECTING

Håkan Gustavsson

Akademin för innovation, design och teknik kommer att offentligen försvaras fredagen den 18 mars 2011, 14.00 i Beta, Mälardalen University, Västerås.

Fakultetsopponent: Professor Gerrit Muller, Buskerud University College
Abstract

Software intensive systems are an increasing part of new products, which make the business impact significant. This is especially true for the automotive industry where a very large part of new innovations are realized through the use of software. The architecture of the software intensive system will enable value creation when working properly or, in the worst case, prevent value creation.

Lean thinking is about focusing on the increase of customer value and on the people who add value. This thesis investigates how system architecting is performed in industry and how it can be improved by the use of Lean thinking. The architecting process does not create immediate value to the end customer, but rather create the architecture on which value in terms of product features and functionality can be developed. A Lean tool used to improve the value creation within a process is Value Stream Mapping (VSM). We present a method based on VSM which is adapted to enable analysis of the architecting process in order to identify improvements.

A study of architecting at two companies shows what effect differences such as a strong line organization or a strong project organization have on the architecting process. It also shows what consequence technical choices and business strategy have on the architecting process. In order to improve the understanding of how architecting is performed a study including architects at six different internationally well-known companies have been interviewed. The study presents the practices that are found most successful. The context of the different companies as well as the architecting practices are compared and analyzed.

The early design decisions made when developing software-intensive systems are crucial to the outcome of development projects. In order to improve the decision making process a method was developed based on Real Options. The method improves the customer focus of critical design decision by taking the value of flexibility into account.

This thesis provides a toolbox of knowledge on how Lean thinking can be applied to system architecting and also presents how architecting is performed in industry today.
Software-intensive systems are increasingly part of new products, which leads to significant business impact. This is especially true for the automotive industry where a majority of new innovations are realized through the use of software. The architecture of the software-intensive system will enable value creation when working properly or, in the worst case, prevent value creation.

Lean Thinking is about focusing on the increase of customer value and on the people who add value. This thesis investigates how system architecting is performed in industry and how it can be improved through the use of Lean Thinking. The architecting process does not create immediate value to the end customer, but instead creates the architecture on which value, in terms of product features and functionality, can be developed. A Lean tool used to improve the value creation within a process is Value Stream Mapping (VSM). We present a method based on VSM which is adapted to enable analysis of the architecting process in order to identify improvements.

A study of architecting at two companies shows what effect differences such as a strong line organization or a strong project organization have on the architecting process. It also shows the consequences technical choices and business strategy have on the architecting process. In order to improve the understanding of how architecting is performed, a study was carried out, including interviewing architects at six different well-known international companies. The study presents the practices that were found to be most successful. The context of the different companies as well as the architecting practices are compared and analyzed.

The early design decisions made when developing software-intensive systems are crucial to the outcome of development projects. In order to improve the decision-making process a method based on Real Options was developed. The method improves the customer focus of critical design decisions by taking the value of flexibility into account.

This thesis provides a toolbox of knowledge on how Lean Thinking can be applied to system architecting and also presents how architecting is performed in industry today.
Going through the process of getting a PhD is a long and winding road, similar to raising a baby. At first, even before the baby is born, the parents think a lot about what becoming a family will be like. This is very similar to before starting work for a PhD. You think you know exactly what it will be like and what you will do, but in the end it is like nothing you could imagine.

As an industrial PhD student I sometimes miss the academic atmosphere found at the university. When visiting Mälardalen University I have always found support and inspiration from the members of the BESS research group. It has been wonderful to make this PhD journey together with Peter Wallin and Stefan Cedergren.

This work has been supported by the Knowledge Foundation, the Swedish Agency for Innovation Systems (VINNOVA) and Scania CV AB, for which I am grateful. I also owe my thanks to many people at Scania who have given me support and interesting discussions. I am grateful to my former manager, Nils-Gunnar Vågstedt, and current manager, Tony Sandberg, for always giving me time to discuss my research and taking care of the project's administration.

My steering committee has done a great job in keeping me on track and on time. My former industrial supervisor and co-author Jan Sterner was great support at the beginning of the journey. During the latter part I was lucky to have a great partnership with my co-author, Ulrik Eklund. My assistant supervisors, Joakim Fröberg and Christer Norström, have provided great support throughout my journey. Discussions with Christer Nordström early in the project were particularly good inspiration and motivation. This journey would not have started if it were not for my supervisor Jakob Axelsson. He was one of the main reasons I started this journey and has guided me to the end, always challenging, supporting and continuously improving my skills. You have been amazing!

I would like to thank my parents for giving me the courage to accept the challenge of pursuing this journey, but recently the biggest fan of my research has been Ebba. When I explain my recent findings to her she sometimes screams with happiness. Her enthusiasm is always there, and even when I discuss research methodology she waves her arms in joy.
Going through the process of getting a PhD is a long and winding road, similar to raising a baby. At first, even before the baby is born, the parents think a lot about what becoming a family will be like. This is very similar to before starting work for a PhD. You think you know exactly what it will be like and what you will do, but in the end it is like nothing you could imagine.

As an industrial PhD student I sometimes miss the academic atmosphere found at the university. When visiting Mälardalen University I have always found support and inspiration from the members of the BESS research group. It has been wonderful to make this PhD journey together with Peter Wallin and Stefan Cedergren.

This work has been supported by the Knowledge Foundation, the Swedish Agency for Innovation Systems (VINNOVA) and Scania CV AB, for which I am grateful. I also owe my thanks to many people at Scania who have given me support and interesting discussions. I am grateful to my former manager, Nils-Gunnar Vågstedt, and current manager, Tony Sandberg, for always giving me time to discuss my research and taking care of the project’s administration.

My steering committee has done a great job in keeping me on track and on time. My former industrial supervisor and co-author Jan Sterner was great support at the beginning of the journey. During the latter part I was lucky to have a great partnership with my co-author, Ulrik Eklund. My assistant supervisors, Joakim Fröberg and Christer Norström, have provided great support throughout my journey. Discussions with Christer Nordström early in the project were particularly good inspiration and motivation. This journey would not have started if it were not for my supervisor Jakob Axelsson. He was one of the main reasons I started this journey and has guided me to the end, always challenging, supporting and continuously improving my skills. You have been amazing!

I would like to thank my parents for giving me the courage to accept the challenge of pursuing this journey, but recently the biggest fan of my research has been Ebba. When I explain my recent findings to her she sometimes screams with happiness. Her enthusiasm is always there, and even when I discuss research methodology she waves her arms in joy.
Needless to say, Ebba is my newborn daughter who has been the best motivation to finish on time. Lastly, this work would not have been possible without the love and support of my fiancée Cecilia, and I am looking forward to continuing our journey together. You are the light of my life.

Håkan Gustavsson
Huddinge, February 2011.
List of Included Papers

Paper A


The case study at Scania and Volvo Cars was conducted in cooperation with Ulrik Eklund. The paper was written by the author with support from Jakob Axelsson.

Paper B


The design of the study was made by the author. The remaining work was done in close cooperation with co-author Ulrik Eklund.

Paper C


The case study was conducted by the author and the paper was written with support from Jakob Axelsson.
Paper D


The development of the method was conducted by the author and the book chapter was written with support from Jakob Axelsson.
Additional publications

Theses

Journal

Conference papers


Workshops

Preface

I have taken this opportunity to present myself and the background to this work. I have worked with vehicle electronic systems integration and architecture at Scania in Södertälje since 2002. At Scania, each truck and bus produced is customer ordered and unique, but is based on the same architecture. In order to reduce complexity, the interfaces need to be simple and flexible and this often requires a trade-off. When I started working as an architect, I thought decisions were made solely on the basis of technical and financial aspects. It soon became clear that the technical issues are often the easy part of the job. The organizational issues, such as where competence is allocated or how responsibilities are shared, are often much more complex.

As my only experience of architecting comes from working at Scania, I thought I needed to learn more in order to improve our way of working. This idea led to the start of my research journey. Working at Scania, it is hard not to be affected by the company’s core values, which are very influenced by Lean. Scania has applied Lean to its production for 20 years, and for more than 10 years in research and development. This background resulted in the question of how we can improve our work further by applying Lean Thinking to system architecting.

I hope this work will provide academia with knowledge of how architecting is performed in industry and how Lean can be applied to architecting. I believe that the methods found can be used in industry for comparison and inspiration regarding process improvements.
Table of Contents

Chapter 1. Introduction .........................................................................1
  1.1 Background ......................................................................................2
  1.2 Research scope .................................................................................9
  1.3 Thesis outline .................................................................................12

Chapter 2. Related work ......................................................................13
  2.1 The architecture..............................................................................14
  2.2 Business domain.............................................................................16
  2.3 People and organization .................................................................17
  2.4 Architecting support .......................................................................19
  2.5 Lean development ..........................................................................26

Chapter 3. Research methodology ......................................................31
  3.1 Research design..............................................................................32
  3.2 Validity...........................................................................................34

Chapter 4. Research results .................................................................37
  4.1 Paper A: Improving the system architecting process through the use of Lean tools ....................................................................................38
  4.2 Paper B: Architecting automotive product lines: industrial practice........................................................................................................38
  4.3 Paper C: A comparative case study of architecting practices in the embedded software industry ............................................................39
  4.4 Paper D: Evaluation of design options in embedded automotive product lines.....................................................................................40

Chapter 5. Discussion ...........................................................................41
  5.1 Lean architecting ............................................................................41
  5.2 Identifying best practice .................................................................42
  5.3 Industrial impact.............................................................................43

Chapter 6. Conclusions and future work............................................45
  6.1 Summary of results.........................................................................45
  6.2 Future work ....................................................................................46

References....................................................................................................49
Table of Contents

Chapter 1. Introduction .........................................................................................1
  1.1 Background ..................................................................................................2
  1.2 Research scope ..........................................................................................9
  1.3 Thesis outline ............................................................................................12

Chapter 2. Related work ..............................................................................13
  2.1 The architecture ..........................................................................................14
  2.2 Business domain .......................................................................................16
  2.3 People and organization ...........................................................................17
  2.4 Architecting support ..................................................................................19
  2.5 Lean development .....................................................................................26

Chapter 3. Research methodology ......................................................31
  3.1 Research design .........................................................................................32
  3.2 Validity ........................................................................................................34

Chapter 4. Research results .................................................................37
  4.1 Paper A: Improving the system architecting process through the use of Lean tools ........................................................................................................38
  4.2 Paper B: Architecting automotive product lines: industrial practice ..........................................................38
  4.3 Paper C: A comparative case study of architecting practices in the embedded software industry .......................................................................................39
  4.4 Paper D: Evaluation of design options in embedded automotive product lines .........................................................................................................40

Chapter 5. Discussion .........................................................................................41
  5.1 Lean architecting .........................................................................................41
  5.2 Identifying best practice .............................................................................42
  5.3 Industrial impact ..........................................................................................43

Chapter 6. Conclusions and future work ............................................45
  6.1 Summary of results .....................................................................................45
  6.2 Future work ................................................................................................46

References ....................................................................................................49
Introduction

Product development involving software-intensive systems is becoming more and more complex, both organizationally and technically. Most large companies are offering their products to a global market and development is often conducted in different countries. Global presence leads to an increased number of variants and more competitive market. A global product development organization is challenged by geographical distance, cultural differences and, more practically, different time zones. To stay competitive models are launched more frequently, leading to a demand for shorter development cycles and a shorter time-to-market. The development cycle can be shortened by improving the process – making more with fewer resources. The development cycle can also be shortened by increased reuse of technology and components– making more variants with fewer parts. Or as stated by a Japanese development manager: “No-change development is the best development”.

Software-intensive systems are often cross functional, which leads to more and closer cooperation with suppliers and between different organizational units. The increased complexity of the products through a larger number of variants and models places high demands on the interfaces between the different parts of the system. The architecture of those systems is therefore important in order to handle the complexity and to cope with market demand. In 1951, the architect of the DC3 aircraft series highlighted some of the essentials of successful aircraft development. One of the fundamental elements is argued to be the adaptiveness of the development process: “The ability to cope with the unexpected. Since no amount of planning or technique can bring all factors under control, there must be an ability to capitalize on good luck or minimize the effects of hard luck.”

Today, it is even more important to reduce risk in the early phases of development in order to prevent projects running over time and budget. This thesis aims at improving how system architecting is performed by the analysis of industrial practice and through the development of new methods.
Chapter 1. Introduction

Product development involving software-intensive systems is becoming more and more complex, both organizationally and technically. Most large companies are offering their products to a global market and development is often conducted in different countries. Global presence leads to an increased number of variants and more competitive market. A global product development organization is challenged by geographical distance, cultural differences and, more practically, different time zones. To stay competitive models are launched more frequently, leading to a demand for shorter development cycles and a shorter time-to-market. The development cycle can be shortened by improving the process – making more with fewer resources. The development cycle can also be shortened by increased reuse of technology and components– making more variants with fewer parts. Or as stated by a Japanese development manager:

“No-change development is the best development”.

Software-intensive systems are often cross functional, which leads to more and closer cooperation with suppliers and between different organizational units. The increased complexity of the products through a larger number of variants and models places high demands on the interfaces between the different parts of the system. The architecture of those systems is therefore important in order to handle the complexity and to cope with market demand. In 1951, the architect of the DC3 aircraft series highlighted some of the essentials of successful aircraft development. One of the fundamental elements is argued to be the adaptiveness of the development process:

“The ability to cope with the unexpected. Since no amount of planning or technique can bring all factors under control, there must be an ability to capitalize on good luck or minimize the effects of hard luck.”

Today, it is even more important to reduce risk in the early phases of development in order to prevent projects running over time and budget. This thesis aims at improving how system architecting is performed by the analysis of industrial practice and through the development of new methods.
1.1 Background

Architectural changes to distributed embedded systems are either evolutionary or revolutionary [40]. The main purpose of this research project is to understand the evolutionary architecting process and its contribution to lean values. The lean philosophy is basically common sense that is packaged so it can be applied to different domains. The project will thereby provide an understanding of the process and the value of the deliverables coming out of the architecting process.

1.1.1 System architecting

The context of architecting includes many disciplines, ranging from human science to computer science [64]. Even the architecture of software-intensive systems is a very large concept. The level of architecture studied in this work is focused on a complete system level. The architects at this level are responsible for the overall system, rather than just one sub-system. The main industrial users of the results from this project are systems architects and their managers, but the results will also be highly relevant to developers of software-intensive systems. The academic reader is probably conducting research in the field of software engineering or technology management.

In order to approach the field of system architecture scientifically, it is necessary to define the terminology as precisely as possible, and we will therefore now introduce definitions of some key concepts.

**System** is defined by Rechtin and Maier [61] as a set of different elements so connected as to perform a unique function not performable by the elements alone.

---

![Figure 1 Examples of software-intensive systems.](image-url)
Software-intensive system [1]: Any system where software contributes essential influences on the design, construction, deployment, and evolution of the system as a whole. Examples of such software-intensive systems are industrial robots and vehicles (Figure 1).

Architecture [1]: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

Architecting is the process (Figure 2) of shaping the architecture to meet customer demand by balancing requirements, guiding principles and product vision.

One architect interviewed in this project replied as follows to the question: How do you know if the architecting process is working well? “When new functionality can be absorbed by the architecture without the need for large changes.”

A similar view is given by Coplien and Bjørnvig [23] when explaining how architecture adds value to the end-user. Good architecture shortens the time between understanding user needs and delivering a solution by eliminating rework [23].

According to Maier and Rechtin [61] one of the most widely applicable heuristics in systems architecting is: “Simplify. Simplify. Simplify.” Another is “Build in and maintain options as long as possible in the design and implementation of complex systems. You will need them.”

![Figure 2 Attributes that affect the architecting process.](image-url)
Similar heuristics was found in a case study at Scania [39]. A development team illustrated how alternatives are evaluated on the basis of three architectural principles:

- Simple is best.
- Smallest number of variants.
- Minimal interface between modules.

1.1.2 Automotive embedded systems

Today, most innovations made within the automotive domain are driven by electronics. In 2006, Volvo Cars [28] estimated the value of electronics in a high-end car to be 30%. The increased use of electronics also means that the organization involved in the development of embedded systems is growing. Figure 3 shows the increase in employees within embedded systems at Scania as well as the stagnating effect of the automotive crisis in 2008-2009. According to a study by Hoch et al. [44] the total value of electronics in automobiles was expected to rise from 25% in 2006 to 40% in 2010.

Automotive customers demand new functionality with every new product release and the time-to-market is constantly shortened. One example of new functions is Advanced Driver Assistance Systems that help the customer to drive the vehicle safety. Those systems typically use information about the surroundings to increase road safety.

![Figure 3 The accumulated increase in employees within embedded systems at Scania relative to 2003.](image-url)
This is done by using sensors to identify nearby objects or communication with other vehicles or infrastructure to obtain more information. The increased interaction between various components and the wider boundaries of the system increases its complexity and demands flexibility for easy integration.

The building blocks of an automotive electrical and electronic (E/E) system consist of electronic control units (ECUs) executing the software modules that implement the functionality. ECUs are connected to communication networks. As shown in Figure 4, the communication networks are usually divided into sub networks and the communication between those is done through gateway ECUs connected to a backbone. Different sensors and actuators are connected to the ECUs, depending on how functions are allocated to the ECU.

When designing an automotive E/E system, there are many different attributes to consider, such as functional requirements, energy management and the wiring harness [37].

![Figure 4 A typical vehicle communication network.](image-url)
1.1.3 Lean

The concept of Lean derives from the production methods developed by Toyota in the 1950s. Since then, Lean philosophy has been applied to diverse areas of operation. The ideas originating from Toyota are also used in the Six Sigma quality system [41]. Lean development is a way of thinking and a system of management used to create customer value [91]. The value creation starts at the suppliers and goes through factories into product features and out to customers. The concept of Lean production has today moved from manufacturing into various sectors, such as maintenance, purchasing, logistics, and on to product development, which is the topic of this thesis. In a comparison of the product development process made by Morgan [62] it was found that Toyota outperformed its US competitors in both quality and time-to-market.

Software development has also been inspired by Lean, resulting in Scrum, Agile [82] and Lean Software Development [70]. In this thesis, Agile is used to refer to the software development practices of the Agile Manifesto\(^1\).

Mapping the practices to the different functions (Figure 5) of a company that produces software-intensive systems shows how Lean is a much larger concept than Agile, for example. Agile is applied only to the development of software and Scrum is used mainly within R&D. Six Sigma can be used to improve processes in most parts of the company, but does not discuss cultural issues [41]. The Lean philosophy is not just a set of tools; instead it affects all parts of the company, from human resources to marketing. This is also the reason why Lean applies so well to architecting, because architecting is a cross-functional activity.

It is important to note that none of the seminal work in Lean Product Development was carried out by people working inside Toyota or by native Japanese people. This means that the available knowledge should be considered more as a western interpretation of the Toyota Way, though the success of the Lean philosophy is undeniable.

Lean development focuses on creating re-useable knowledge - knowledge that contributes to the profitability of future operational value cycles and that can ideally be used for many projects [91].

\(^1\) http://www.agilemanifesto.org/principles.html
Baines et al. [7] present the result of a systematic literature review of what is meant by the term Lean in product development. One finding is that the definition of Lean is drifting and moving from waste reduction towards value creation. Another result is that value is added in product development when useful information is produced, but value needs to be defined precisely.

Figure 5 Implementation of development practices in the organizational parts of a company.

1.1.4 Personal experience - Lean in Japan

During a study trip to Japan I visited a number of companies and personally experienced the culture that gave birth to the Lean philosophy.

Within Japanese culture there is great dedication to following rules and avoiding errors. Lean literature often mentions how work should be standardized in order to ensure quality. My experience is that it is easy to make a standard, but very hard to get everybody to follow it. Visiting a Toyota factory, I noticed how everybody at the plant indicated with their hands that they are looking right and left before crossing a street. Finally they pointed straight ahead before crossing. During the entire two hour stay at the factory I did not see anyone breaking this standardized way of crossing. Independent of the number of people crossing, everybody did it according to the standard. This practice is called pointing-checking (yubi-
sashi-kakunin) and is a common Japanese practice for dealing with safety checks.

In development, other methods are used to ensure quality. The main method I experienced during different company visits was the extensive use of checklists. Checklists are used during all different steps of the development process. The checklists reflect the engineering knowledge accumulated over time. When a failure is found during test activities the relevant checklist is updated. In this way, checklists are used to ensure that errors never will be repeated, as well as to transfer knowledge. When visiting one company they explained that they not only test to the specification, but also to twice the limit of the component.

Japanese companies invest a great deal in training new employees. New students are therefore educated during their first years at the company [63]. One company mentioned that 70-90% of the time was spent on education during the first year and 50% during the second year, another company mentioned that even math was taught. One possible explanation as to why companies invest so much in their employees is the Japanese concept of lifetime employment. If engineers move, they usually move from original equipment manufacturer (OEM) to supplier; it is rare to move from supplier to OEM. Unlike in the west, a salary decreases on changing employment [36].

Lean literature argues that companies should establish long term relationships with a small number of suppliers, so you can really know them, and they rely on you for business [91]. Studies made by Fujimoto and Clark [18] shows that Japanese OEMs involve suppliers to a much higher degree than in the US and Europe.

The companies we visited also seemed to be working closer to the supplier than western companies. This is supported by a study of patent applications [57] made by Japanese automotive OEMs and their suppliers. It shows that the ratio of shared patents applications made by Toyota is twice as high as its Japanese competitors. This study indicates that Toyota works very closely with its suppliers in the early phases of development. It also shows that Toyota, in this regard, is different from its Japanese competitors. Research and Development in Japanese companies are often geographically separated (Figure 6). Corporate research seems to include what is commonly defined as research and advanced engineering. The manufacturing we visited includes development and production. Development is located by the production site in order to support production.
Introduction

During my visits, two explanations were given as to why research is separated:

- To attract top students they need to be close to the top universities.
- To keep research disconnected from production and development.

Having a separated organization could entail complications for the architecture of the system. During our study trip the companies we visited showed a great deal of interest in how to achieve reuse and product modularization, although there was no hard evidence found to show that the level of reuse is low.

1.2 Research scope

The purpose of this research is to improve how systems architecting is performed within software-intensive systems. A more specific purpose is to find the success factors for different methods used within the industry.

The overall goal of the project is to investigate how system architecting is performed in the automotive industry and how it can be improved by the use of Lean Thinking. To achieve this goal, architecting will be studied in various industrial settings in order to find a successful use of methods and areas for improvement. Lean Thinking will be studied to find how it can be applied within the architecting of software-intensive systems. New methods that can improve decision making when developing software-intensive systems will be developed and evaluated. The results of this research are increased knowledge in this field. Case studies of the system architecting process at the different companies will result in an increased understanding of the system architecting process. The analysis of the processes will provide the companies with inspiration for improvements and ease future academic
studies within the field. Four research questions (RQ) are stated in the following sections and their relationships to business, architecture, process or organization [90] are mapped in Figure 7.

Figure 7 The relationships between the research questions (RQ).

1.2.1 Research question 1
Lean Thinking aims to improve the development process by creating a cadenced flow. Understanding the process and the methods that are used and available is important in order to improve the processes. The architecting process described in the documentation is generally not the same as the real process. The real process needs to be mapped to find what artefacts are produced and for what customer. If unnecessary iterations and artefacts can be eliminated, the process will be faster and more efficient. One hypothesis to be tested is whether Value Stream Mapping is a suitable method.

How can an architecting process be mapped in order to identify improvements?

1.2.2 Research question 2
To be able to compare different practices, one needs to understand the context of the specific architecture under observation. The context might be influenced by lifecycle, procurement strategy, organization, volume, and guiding principles. Different methods can help you find the best solution, but some will be more effective than others. To better understand how solutions are reached one needs to study how engineering tools are really used, for
example, and how tasks are performed. Depending on the context, different methods will be more suitable than others and also affect what tasks that needs to be done. When you know what tasks are needed you can start improve in order to create an efficient flow through the architecting process.

What tasks are performed in the process of architecting automotive embedded systems?

1.2.3 Research question 3

Just because a method is used it does not mean it will be successful. How roles are distributed throughout the organization and how information is communicated is probably important. According to Cedergren [17] product development is to be considered successful if its products not only satisfy the needs of its customers, but also creates value to its stakeholders at large. The methods used within the architecting process should then be considered successful if the results are valuable to its stakeholders and the architecture fulfills the needs of the customer Success might also depend on issues such as what authority and responsibility is given to the involved stakeholders. The answer to this question can be used as a guideline for when to use different methods.

In what context are the methods used within the architecting process found successful?

1.2.4 Research question 4

Embedded systems are evolving and changes are continuously being introduced. A Lean architecture needs to be flexible in order to reduce the number of variants [63]. To cope with those changes, the system needs to be designed with the right amount of flexibility. A product that has an architecture that can absorb new functionality will be able to react quickly to new customer demands and thereby provide customer value. RQ 4 aims at developing methods that will aid the architect when making architectural decisions.

How can one value the flexibility needed to withstand an uncertain future in automotive embedded systems?
1.2.5 Contribution

The main contribution of this thesis is to present how Lean Thinking can be applied to system architecting. The contribution is presented in paper A, B, C and D:

- In Paper A, Value Stream Mapping is adapted to be suitable for identifying potential improvements to the architecting process. A case study presents, in general terms, what types of waste and improvements could be found.

- The different tasks performed when architecting automotive embedded systems are presented in Paper B. To understand how different methods are suitable in different contexts, a case study is conducted.

- The contexts of the different companies, as well as the architecting practices, are compared and analyzed in Paper C.

- To improve how decisions are made in the early phases of development, a method and process is presented in Paper D. The method shows how flexibility can be valued.

1.3 Thesis outline

The thesis contains an introductory part and a collection of the articles mentioned previously. The introductory part is divided into seven chapters. Related work is presented in the next chapter. This is followed by the research methods used to study each of the questions presented in Chapter 1. The research results and their relation to the appended papers are described in Chapter 4. The appended papers are summarized in Chapter 5. Finally, the results are discussed in Chapter 6 and conclusions and future work is proposed in Chapter 7.
Chapter 2. Related work

This section describes research in the field covered by this thesis and provides a frame of reference for the concepts used. The architecture reflects the business goals of a company. Product development involves many stakeholders that have an interest in the system during its entire life-cycle. The development process will include stakeholders from departments such as purchasing, aftermarket and sales. How those stakeholders are organized within the organization will affect what solutions are chosen and also the system architecture. Changes in the concerns architecture (Section 2.1), business (Section 2.2), process or organization will, according to van der Linden et al [90], have an impact on other concerns. The competence and skills of the individual people working in the organization will influence how the work is performed (Section 2.3).

Architecting is affected by various support functions surrounding the development activities. Computer aided tools are necessary to handle the large amount of information needed to make the right decisions and to preserve knowledge of previous decisions. Processes are needed to aid the architecting activities and the process can be improved using different approaches, as discussed in Section 2.4.1. There are many different methods available; the methods most suitable for architecting are presented in Section 2.4.2.

Figure 8 The context of Lean Architecting.
2.1 The architecture

Architectural trends in the automotive domain are currently changing, but there has been a philosophy of “one function – one ECU”. There have been various attempts to resolve these issues. Academic projects like DECOS [69] have proposed standardized system architectures. The EAST-EE project [27] proposed an architecture description language to support component-based development. In 2003, a number of automotive OEMs and suppliers launched the Automotive Open Systems Architecture (AUTOSAR) with the aim of creating an industry de-facto standard for automotive software. The main motivations behind AUTOSAR are very similar to those addressed by Ommering and Bosch [24], where the basic arguments for software product lines are listed; size, complexity, quality, diversity and lead time reduction. It is also pointed out that it is hard to succeed in combining components from different companies without a common global architecture. To address those topics, AUTOSAR both defines an architectural framework and a supporting component framework to achieve an instantiation between the basic hardware and the application software. Large parts of the automotive industry are now adapting to this standard [42] and AUTOSAR will change how software is integrated into embedded systems within the automotive industry [3]. The transition from a proprietary architecture to AUTOSAR involves many stakeholders, ranging from developers and testers to purchasing.

Most automotive software is now bought at a fixed price associated with a specific hardware. AUTOSAR will enable sales of pure software components; it will therefore change the acquisition process and the terms on which software is priced. The supplier structure of the automotive domain is in this case both a driver and a challenge.

Architectural changes in distributed embedded systems are either evolutionary or revolutionary [4]. Evolutionary changes to the system are continuous improvements and increasing functionality. Revolutionary changes are made when large fundamental changes are needed. A common scenario occurs when the technical debt [25] of the system has increased so much that the cost of evolutionary changes become too expensive. Due to the technical debt in terms of “spaghetti code” or undocumented system architecture, the cost of changes is increasing. Revolutionary changes could also be caused by market changes or when adapting to a new standard, as is the case with AUTOSAR.

One way to capture the knowledge stored in existing architectures and the vision of future needs is a concept known as Reference Architecture [20].
Reference Architecture is a standard or template based on the previous best practice of the domain. The evolution of an automotive Reference Architecture is described by Eklund et al. [30].

Another way to manage the architecture is through the use of Software Product Lines. The term Software Product Lines is defined by Clements and Northrop [19] as:

“A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.”

The practice of Software Product Lines involves long-term strategy and the management of the core assets in order to increase reuse and decrease time to market. Industrial uses of Reference Architectures [20] are in some cases very similar to the use of Product Line Architectures [9].

The use of Software Product Lines has a great deal in common with the traditional manufacturing of mass-produced products and, in this way, is nothing new. What differentiates Software Product Lines from component-based development is the larger scope, involving both business and management aspects.

Blackenfelt [10] discusses why modularization should be applied to product development. He concludes that the freezing of interfaces should be the last step in the modularization project. Similarly to software product lines engineering, it is argued that the product will benefit from the investment after the introduction of a few family members or variants (Figure 9).

![Figure 9 Total product cost in relation to the number of variants](93)
2.2 Business domain

The business domain of a company has a large impact on the architecture. Factors such as the type of customers, sales volume and the lifecycle of the product will influence the architecture. A company that builds a small number of variants of one system with low complexity and few developers is not likely to invest anything in developing the architecture. Companies who develop software-intensive systems have the common challenge of managing the complexity of mechatronic systems. The systems addressed in this work often have a lifecycle of 20-30 years and long term service contracts.

The strategy of using one common architecture for all variants seems to be a viable solution in order to keep cost and complexity at a low level, but this is not always the case. In the case of Bosch [81, 84] it was found that they needed to implement a low-end and a high-end architecture in order to stay competitive in different market segments.

The long term strategy of the company has great implications for the architecture. If the strategy is to lead technology development in one field, they need to keep the knowledge in-house or perform co-development with strategic partners. A component developed in-house or in close co-operation is much more likely to fit into the existing architecture than a purchased component. The procurement strategy of make or buy has great implications for architecture and the project’s success [35]. The complexity of the component in comparison to customer value is one way to make the trade-off. Figure 10 illustrates a common strategy of keeping components with high complexity and high customer value in-house to guard the expertise. Simpler components with low customer value are found to be more beneficial to outsource because they are not viewed as core competence and are easier to specify.

Commercial vehicles and passenger cars are part of the automotive domain and similar in many ways, but even within one industrial domain there are many differences. The main purpose of commercial vehicles is transportation of goods, but the transport task differs with each customer and market. The customer requirements on the vehicles are very different. A commercial vehicle must manage to run 300,000 km per year; breakdowns do not just influence the driver, but also the delivery time of the goods being carried. Commercial vehicles have a lot in common with passenger cars, much of the functionality is found in both segments [100]. The differences in business aspects affect the architectural requirements.
2.2 Business domain

The business domain of a company has a large impact on the architecture. Factors such as the type of customers, sales volume and the lifecycle of the product will influence the architecture. A company that builds a small number of variants of one system with low complexity and few developers is not likely to invest anything in developing the architecture. Companies who develop software-intensive systems have the common challenge of managing the complexity of mechatronic systems. The systems addressed in this work often have a lifecycle of 20-30 years and long term service contracts. The strategy of using one common architecture for all variants seems to be a viable solution in order to keep cost and complexity at a low level, but this is not always the case. In the case of Bosch [81, 84] it was found that they needed to implement a low-end and a high-end architecture in order to stay competitive in different market segments.

The long term strategy of the company has great implications for the architecture. If the strategy is to lead technology development in one field, they need to keep the knowledge in-house or perform co-development with strategic partners. A component developed in-house or in close co-operation is much more likely to fit into the existing architecture than a purchased component. The procurement strategy of make or buy has great implications for architecture and the project's success [35]. The complexity of the component in comparison to customer value is one way to make the trade-off. Figure 10 illustrates a common strategy of keeping components with high complexity and high customer value in-house to guard the expertise. Simpler components with low customer value are found to be more beneficial to outsource because they are not viewed as core competence and are easier to specify.

Commercial vehicles and passenger cars are part of the automotive domain and similar in many ways, but even within one industrial domain there are many differences. The main purpose of commercial vehicles is transportation of goods, but the transport task differs with each customer and market. The customer requirements on the vehicles are very different. A commercial vehicle must manage to run 300,000 km per year; breakdowns do not just influence the driver, but also the delivery time of the goods being carried. Commercial vehicles have a lot in common with passenger cars, much of the functionality is found in both segments [100]. The differences in business aspects affect the architectural requirements.

2.3 People and organization

In 1951, Arthur Raymond, architect of the DC3 aircraft, stated that in order to be successful the design area should be close to both testing and prototyping [72]. In 1968, the origin of what later would be known as Conway’s Law [21] was presented: “Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure.”

More recently, Coplien [23] addressed this issue in software development with the following technique: “Be attentive to domain partitioning. In particular do not split a domain across geographic locations or architectural units.” This technique is often seen implemented in the automotive domain where development is divided into the architectural units of the vehicle, although this was done long before the introduction of software.

The architects carry knowledge across different functions. The geographic and organizational location of the architects is therefore important. Knowledge sharing between architects is affected, whether they are colocated or separated. The organizational structure of a company often mirrors the product architecture of the products it produces. Henderson and Clark [43] explain how this tie between product and organization causes difficulties when the architecture need to be changed. One example of how the performance of the organization is affected by the architecture is given by Reinertsen. He argues that developers will be more effective if they are organized around the modular structure of the product [74]. Unphon and Dittrich [86] conclude that one must consider the organization and business domain when adopting a product line architecture. In a study of eight different software development organizations [87], it was found that the
architecture is maintained and evolved through face-to-face communication rather than documents. Coplien and Bjørnvig argue that developers and testers should be friends and there should be ongoing testing support during development [23].

The size of the R&D organization, as well as the size of the system development organization, will also affect the role of the architect. In a small organization everyone will be an architect. Eventually, when it is not practical for everybody to talk to each other someone will take on the role of an architect. Or, as stated by Rechtin; “architecting is a consequence of system complexity” [73]. As the number of architects grows they will also need coordination. Coplien and Harrison [22] have developed organizational patterns suitable for software development. Three of them are presented below:

**Engage customers.** This stresses the importance of the development organization ensuring and maintaining customer satisfaction. This can be achieved by encouraging communication between customers and the different roles in the development organization.

**Function owner and component owner.** To ensure responsibility, every function and component should have a dedicated owner.

**Developer controls the process.** The developer should be at the centre of the process and also be able to change the process.

The type of organization will affect how communication is carried out and how decisions are made. The power centers of an organization also affect how work with the architecture is done. Nedstam [66] presents the great differences between how work is done in an organization with strong line management and in an organization with strong projects. This has also been found to be true at the companies studied in Paper B of this thesis.

Kruchten [58] suggests that the productive time spent by architects can be sorted into three categories: internal (architecture design), inwards (input from outside world) and outwards (providing information) communication and that they should roughly have the ratio of 50% internal, 25% inwards, 25% outwards. In order to work effectively as an architect it is important to understand the organization and to know who is the right person to answer the current question. Different companies have different cultures and the culture varies depending on the geographic location.

In a survey of 279 IT architects in the Netherlands, Farenhorst et al. [32] conclude that architects are lone decision makers; not very willing to share architectural knowledge, but eager to consume. A study of decision-making
in Swedish and German teams [65] concludes that Sweden has flatter organizational hierarchies and that the decision-making process is less formal. This is well-known to people working across those borders, but could cause unnecessary tension if forgotten. Rechtin [61] highlights the importance of architects understanding the cultural characteristics of the organization in order to be successful.

2.4 Architecting support

The architecting process needs to fit into an overall product development process and the work should be supported by methods and sub-processes. Tools will be used to document and analyze the architecture. Those tools will mostly provide support, but often also constrain and limit the way architecting is done. The architect will need to make choices and sometimes, more importantly, communicate the pros and cons of different solutions. Visualizing different possible solutions (Figure 11) is therefore important even for the solutions that are not chosen.

![Figure 11](image)

**Figure 11 One way to visualize different possible solutions.**

Different architectural frameworks exist in order to effectively describe and communicate architectures. Architectural frameworks are often used to describe the details of architecture. Greefhorst et al. [38] have performed a comparison of many of the existing frameworks. They found that there are many differences between the existing frameworks and conclude that the differences partly depend on different original goals and context. Describing an architecture is also standardized in IEEE 1471 [49]. The frameworks used by the defense industry, for instance, are shaped by a long systems engineering tradition and driven by requirements.
2.4.1 Process and process improvement

The importance of embedded systems, together with a growing organization developing a system that is becoming more complex every day, makes the performance of the development process crucial. High performance is the result of efficiency and effectiveness in each activity [16]. Effectiveness is defined as doing the right thing and efficiency as doing the thing right. The architecting process involves many stakeholders who all produce knowledge needed to develop the architecture. Liang et al. [60] present a process based on architectural knowledge for software architecting. Architecting of software systems are described in many pieces of literature; Eeles [29] presents the process of software architecting of IT system [29]. Most available architecting processes are software oriented, though the architecting method CAFCR [64] is one exception. This has a focus on what the internal and external customer wants on a system level of embedded systems. Through comparison of available processes, Hofmeister et al. [45] have developed a generic process (Figure 12) for creating and maintaining an architecture. Inspired by Scrum [82], the process emphasizes the need for a backlog to keep track of issues found in the architecture.

Different methods can be used to assess the process and to find improvements. Quality systems such as the Integrated Capability Maturity Model (CMMI) and SPICE can be used to perform process assessment. The results of a systematic literature review show that there is a great variety of maturity models and that the trend is towards more specialization in the models for specific domains [97]. In the European automotive domain, automotive SPICE is the most common model. Axelsson [5] presents a maturity model for architecting embedded system product lines based on CMMI.

Process improvements (Figure 13) can be made through radical (Kaikaku) and continuous (Kaizen) improvements. Continuous improvements are often based on a broad effort involving everyone in the organization, while radical improvements are top-down initiatives with the goal of dramatic results [98]. As with all large changes, the risk involved increases when making radical improvements. The same analogy can be made with the information batches flowing through the organization, large batches increase the risk. Small batch sizes reduce the variability of flow, reduce risk and accelerate feedback, which increases motivation [75]. This is very similar to the key drivers of SCRUM [82], which are to increase the speed of development and to add energy and focus.

It is hard to differentiate between Lean and Agile. The two practices share the same values; they both empower people to achieve results and are keen to adapt and improve the processes to fit current needs. One difference between the two is the scope of implementation. Lean is applied to all the different parts of a developing organization and Agile is focused on software development. Differences between implementations of Lean software development are more likely to occur because of cultural or organizational factors, compared to Agile software development. A common criticism of Agile is that architecting is insufficiently emphasized and that the architecture emerges during development [56]. However, a literature review...
Model (CMMI) and SPICE can be used to perform process assessment. The results of a systematic literature review show that there is a great variety of maturity models and that the trend is towards more specialization in the models for specific domains [97]. In the European automotive domain, automotive SPICE is the most common model. Axelsson [5] presents a maturity model for architecting embedded system product lines based on CMMI.

Process improvements (Figure 13) can be made through radical (Kaikaku) and continuous (Kaizen) improvements. Continuous improvements are often based on a broad effort involving everyone in the organization, while radical improvements are top-down initiatives with the goal of dramatic results [98]. As with all large changes, the risk involved increases when making radical improvements. The same analogy can be made with the information batches flowing through the organization, large batches increase the risk. Small batch sizes reduce the variability of flow, reduce risk and accelerate feedback, which increases motivation [75]. This is very similar to the key drivers of SCRUM [82], which are to increase the speed of development and to add energy and focus.

![Figure 13 Making radical (Kaikaku) and continuous improvements (Kaizen) to improve performance.](image-url)

It is hard to differentiate between Lean and Agile. The two practices share the same values; they both empower people to achieve results and are keen to adapt and improve the processes to fit current needs. One difference between the two is the scope of implementation. Lean is applied to all the different parts of a developing organization and Agile is focused on software development. Differences between implementations of Lean software development are more likely to occur because of cultural or organizational factors, compared to Agile software development. A common criticism of Agile is that architecting is insufficiently emphasized and that the architecture emerges during development [56]. However, a literature review
Related work

[12] of architecting within Agile development concludes that there is no empirical research to support or contradict this assumption.

A systematic review of agile software development methods [26] found that most empirical research studies focused on extreme programming. Very few studies were found on Lean software development or Scrum. Kettunen has performed a comparative study of manufacturing methods and Agile software product development. He concludes that the there are not many profoundly new ideas compared to earlier manufacturing methods [56]. Coplien [23] makes a comparison of Lean and Agile and claims that Agile is about doing and Lean about thinking and doing. He also claims that Lean focuses on process and Agile focuses on people, although seminal work [62, 79] on Lean says otherwise.

In the literature, there is little work on how Lean can be applied to the process of developing software-intensive systems. Recently, Lean has been applied to the overall systems engineering process of INCOSE [67]. Poppendieck and Poppendieck [70] present how Lean can be applied to the software development process. In their work, typical wastes to be found are hand-offs between individuals, switching between tasks and adding extra features.

Browning et al. [14] discuss how process modelling of a product development process is conducted and present a simple framework. Furthermore, they argue that process modelling should be tailored to that environment.

In [80] a model for evaluating the degree of leanness of manufacturing firms is presented. This model was based on the initial research done by Karlsson and Åhlström [51] who developed a method for measuring the change progress in production.

Value Stream Mapping is presented as a way to find waste. There are many different techniques available for process modeling, but Value Stream Mapping (VSM) is different to other process modeling tools due to its focus on value creation [14]. Value Stream Mapping (VSM) was initially a tool for improving the manufacturing process [77] and has been shown to be effective within manufacturing [48]. The method is now also used within many other disciplines. This method is explained in detail and adapted to the architecting process in Paper A of this thesis.
2.4.2 Analysis methods

In order to make an adequate design decision, one must consider numerous factors. There are obvious aspects such as size, cost and performance, yet other less tangible factors are very important; factors such as customer preferences, development cost, production volume and time to market. All these factors – and many more – influence the final decision.

Architectural decisions are made when selecting components and allocating them to subsystems that then are combined into a system. These decisions can be made on different levels with various impacts and predictability [33]. This section explores the available architecting methods.

SWOT analysis (Figure 14) is one common way [15] of evaluating design alternatives and enables a clear visual view of the trade-off. Trade-off curves (Figure 15) are used [39] to visualize the design space and enable efficient communication of knowledge. A practical example could be to compare the accuracy versus the cost of different sensors in order to make the best choice. An extension of trade-off curves using Data Envelopment Analysis [68] presents how resource utilization of different user functions can be evaluated. An example of one of the more commonly used [2, 50, 78] formal evaluation methods is Pugh's evaluation matrix, which was developed by Stuart Pugh in the 1980s [71].

The Design Structured Matrix (DSM) [13, 89] has been used to evaluate architectures in various cases. Larses [59] uses the balanced scorecard to balance the important perspectives in system design for the complete E/E system, in combination with a cluster analysis using DSM. He found that the combination of quantitative and qualitative data can provide good decision
support. Problems that were found were the lack of input data and the need to consider procurement and change aspects when reusing the architecture.

The Architecture Trade-off Analysis Method (ATAM) is a method for evaluating different architectural approaches and was developed by the Carnegie Mellon Software Engineering Institute [54].

The goal of ATAM is to assess the consequences of architectural decisions in the light of quality attribute requirements and to perform an analysis in a repeatable manner. Each stakeholder has different quality attributes that they consider to be the most important ones. The top level attributes are typically attributes like safety, performance, maintenance and maintainability but the number of attributes can vary from case to case.

A utility tree is created with input from all stakeholders. The utility tree is only constructed by the architects and the project leader and will therefore only show the architects’ view of what is important to the system. The next step is to perform a brainstorming for scenarios. The scenarios are made up by all stakeholders. The scenarios are comparable to the leaves of the utility tree.

Each stakeholder is given a number of votes, typically 30% of the total number of scenarios, and then votes for what each stakeholder considers to be the most important one. The result of the vote is then compared with the result from the utility tree. If the result is the same, it is quite certain that the most important attributes are being considered in the architectural decision. If not, the view of the most crucial attributes for a successful architecture differs between system architects and other stakeholders. In this case, some kind of reasoning is necessary between the system architects and other stakeholders in order to conclude which are the most important parts.

ATAM is a structured method that is tailored for analyzing architectures, which ensures that the right questions are asked. The main result of ATAM is an identification of the potential architectural risks. The greatest benefit of the method is that it can provide a common understanding of the importance of different quality attributes.

ATAM does not provide any support for evaluating different design alternatives. An extension of ATAM made by Wallin et al. [88] provides support for evaluating different design alternatives with the use of paired comparison of scenarios.

The Cost Benefit Analysis Method (CBAM) is an extension of the ATAM and was also developed by the Carnegie Mellon Software Engineering Institute [53]. It uses the quality attributes derived from the ATAM, but also
Related work

considers cost when reasoning around the most suitable architecture. A second iteration of CBAM also takes the uncertainty of the used figures into account, but it does not consider flexibility and architectural evolution.

Using options theory is one approach to dealing with the high level of uncertainty when making design decisions in the early phases. The theory derives from finance, where an option is the right but not the obligation to exercise a feature of a contract at a future date [46]. Real Options are discussed in detail in Paper D.

The benefits of using a structured method are widely accepted in academia, but various studies [2, 39, 78] indicate a very low industrial usage. The proposed solution is to present success stories and to further investigate the needs of the industry [78]. An article published in the Journal of Engineering Design attempts to answer the question of why industry ignores design science [34]. The article claims that industry solves problems by using the knowledge of experienced engineers, which is often faster than using a structured method. One of the answers presented is that many structured methods require information which is often not present or very resource consuming to generate. Ken Hurst [47] presents the following reasons for why a structured method should be used:

- Time wasted in pursuing wrong alternatives to the detailed design stage is avoided.
- Causing visible decision-making helps to ensure the process is repeatable.
- The ability to evaluate the thought processes of others is developed.
- The designer can defend decisions made in discussions with managers or clients.
- A designer with no previous experience can carry out a sensible evaluation of alternative concepts.
- The process of concept selection stimulates new concepts or encourages a combination of concepts.

Ulrich and Eppinger [85] present a similar list of benefits and emphasize that the use of a structured method provides customer focus and a more competitive design.
2.5 Lean development

Lean development focuses on creating re-useable knowledge - knowledge that contributes to the profitability of future operational value cycles and that can ideally be used for many projects [91]. The concept of Lean production was defined in the literature by Womack et al. [96], but derives from the working methods developed by Toyota in the 1950s.

Lean methods focus on increasing customer value and on the people who add value. A Lean-based company encourages its employees to perform continuous improvement and to learn. This is done by cross-functional and parallel work and a high degree of standardization in order to improve and to share knowledge across the organization. Lean production is achieved by the careful planning of a production line in order to optimize the production flow to meet customer needs. Each assembly station is arranged to minimize unnecessary motion and material transportation. Each assembly station is assigned defined tasks to be finalized at a specific time in order to achieve a balanced flow throughout the production line. A balanced flow means that the results are delivered on time without waiting or over-production.

An important starting point for lean product development is to view product development as a process, and like any other process there are repeated cycles of activity [63]. From a process perspective, there are many activities that are shared between different development projects. An increased flow is achieved by eliminating the waste in a process, thus new products can be brought to the market at a higher pace.

There are two main differences between manufacturing and the early phases of product development. Firstly, in product development the flow does not consist of materials, but more often of information and knowledge in different forms. The different organizational and geographical locations of the stakeholders influence how this knowledge is shared. Secondly, the product development process does not consist of one flow, but instead iterations are frequent and different concepts are developed in parallel. For coping with the rapid changes made in product development, Ward [92] makes an analogy to surfing: in order to be in control you need to constantly adjust, changing direction and shifting from wave to wave instead of trying to control the waves. Creating a cadenced flow of information is one way to be able to react to the changes. One practical example is how short meetings are used at Scania to quickly distribute information to the organization (Figure 16).
Kennedy et al. argue that Toyota standardizes their knowledge into checklists and reviews all their design against these standards. Those checklists are updated after every project. Product development consists of two value streams (Figure 17) [55]:

- The product value stream is unique to each project. Project X is not started until the alternative designs have been evaluated and decided upon. When the project starts, the risk should be very low. Knowledge acquired during and after the project is fed back into the knowledge value stream.

- The knowledge value stream consists of knowledge generalized for visual flow across projects and organizations. Checklists and A3 documentation are used to carry this knowledge. Architectural knowledge such as patterns and guidelines should be part of the knowledge value stream.
Allen Ward [92] claims that 20% of the time spent in product development is value adding time. Non-value-creating time such as administration work occupies 20% and the remaining time is waste. This fact would suggest that optimization is possible if we identify the wasteful activities. It is common to define seven types of waste [63] and value stream mapping is one method to identify the waste within any process.

According to Allen Ward [92], the most frequent waste in development is waste of knowledge. He divides knowledge waste into three categories: scatter, hand-off, and wishful thinking. Scatter is described as actions that disrupt the flow of knowledge. This disruption can be due to communication barriers and the use of inappropriate tools. An example of knowledge waste created by hand-offs is to functionally move people around rather than assigning them to one task from beginning to end. Waste due to wishful thinking is, for instance, testing according to specification rather than testing to learn about the limits of the product. An example of waste in terms of discarded knowledge is testing to specifications (rather than testing to failure), which throws away the opportunity to find out when and how the design actually fails [91]. Knowledge from testing would then be fed back to the knowledge value stream (Figure 17) as A3 documentation and engineering checklists. Liker [63] points out the difficulty of transferring tacit knowledge compared to explicit knowledge. Explicit knowledge, such as mathematical equations and historical facts, is often easier to store. Tacit knowledge is often more diffuse, similar to what is taught through
apprenticeship. Toyota creates their learning network through activities such as technology demonstrations, checklists, know-how databases, mentoring and lessons learned [62].

<table>
<thead>
<tr>
<th>Seven wastes</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overproducing</strong> more or earlier than the next process needs</td>
<td>Batching, unsynchronized concurrent tasks</td>
</tr>
<tr>
<td><strong>Waiting</strong> for materials, information, or decisions</td>
<td>Waiting for decisions, information distribution</td>
</tr>
<tr>
<td><strong>Conveyance</strong> - Moving material or information, or decisions</td>
<td>Waiting for decisions, information distribution</td>
</tr>
<tr>
<td><strong>Processing</strong> - Doing unnecessary processing on a task or an unnecessary task</td>
<td>Stop-and-go tasks, redundant tasks, reinvention, process variation-lack of standardization</td>
</tr>
<tr>
<td><strong>Inventory</strong> - A build up of material or information that is not being used</td>
<td>Batching, system over utilization, arrival variation</td>
</tr>
<tr>
<td><strong>Motion</strong> - Excess motion or activity during task execution</td>
<td>Long travel distances, redundant meetings, superficial reviews</td>
</tr>
<tr>
<td><strong>Correction</strong> - Inspections to catch quality problems or fixing an error already made</td>
<td>External quality enforcement, correction and rework</td>
</tr>
</tbody>
</table>

Table 1 Applying the seven wastes within product development [63].
Chapter 3. Research methodology

In this research project the system architecting process of several large companies has been studied. Traditional research is often done through the use of quantitative methods. In natural science, experiments are used to validate a model through measuring a series of samples in a controlled environment. Experiments are also applied to software engineering and are used to investigate the qualities of different programming languages or code inspection techniques [94]. However, experiments on real industrial processes are difficult because of the large number of dependent variables. The variables are often very hard to control and to measure. The cost of performing an experiment that captures the complexity found in an industrial setting would be very high. The methods considered in this research are surveys and case studies.

The typical feature of a survey is, according to Robson [76], the collection of a small amount of standardized data from a relatively large number of individuals in a known population. Disadvantages of surveys sent to different organizations include the risk of misunderstanding and the risk of the respondents not taking the exercise seriously [76]. Different companies perform architecting in various ways and there are many different factors that have an influence. Many of those factors are thought to be soft factors [31] that are hard to find through a survey.

System architecting is, as previously described, a cross functional activity, which makes it very difficult to measure or control the process. According to Yin [99], case studies are especially suitable when the boundaries and context are not clearly evident, and case studies can be both quantitative and qualitative [94]. System architecting has boundaries to many other processes and is influenced by its context. Case studies are often used to investigate real industrial processes and therefore suitable for this research. Evidence can be collected from different sources in a case study. Documentation or archival records can be used if they are retrievable [99], but access to architectural information in industrial settings is often found to be limited. This is primarily because of commercial issues, but also due to the limitation of available documentation. Another source of case study evidence can be
Chapter 3.  Research methodology

In this research project the system architecting process of several large companies has been studied. Traditional research is often done through the use of quantitative methods. In natural science, experiments are used to validate a model through measuring a series of samples in a controlled environment. Experiments are also applied to software engineering and are used to investigate the qualities of different programming languages or code inspection techniques [94]. However, experiments on real industrial processes are difficult because of the large number of dependent variables. The variables are often very hard to control and to measure. The cost of performing an experiment that captures the complexity found in an industrial setting would be very high. The methods considered in this research are surveys and case studies.

The typical feature of a survey is, according to Robson [76], the collection of a small amount of standardized data from a relatively large number of individuals in a known population. Disadvantages of surveys sent to different organizations include the risk of misunderstanding and the risk of the respondents not taking the exercise seriously [76]. Different companies perform architecting in various ways and there are many different factors that have an influence. Many of those factors are thought to be soft factors [31] that are hard to find through a survey.

System architecting is, as previously described, a cross functional activity, which makes it very difficult to measure or control the process. According to Yin [99], case studies are especially suitable when the boundaries and context are not clearly evident, and case studies can be both quantitative and qualitative [94]. System architecting has boundaries to many other processes and is influenced by its context. Case studies are often used to investigate real industrial processes and therefore suitable for this research. Evidence can be collected from different sources in a case study. Documentation or archival records can be used if they are retrievable [99], but access to architectural information in industrial settings is often found to be limited. This is primarily because of commercial issues, but also due to the limitation of available documentation. Another source of case study evidence can be
Research methodology

retrieved through either direct observation or participant observation [99]. The strength of observation is that the process can be studied in its context and in real time, though this is often time-consuming. Industrial projects are not always on time. Observing real projects would add uncertainty to the research plan. Interviews are the main case study methodology chosen for this research and are discussed in detail in the following section.

3.1 Research design

Case studies have been used to answer all research questions, but the method has been applied somewhat differently for different questions. This section describes the methods used and how different threats to validity have been treated. “Case study is a strategy for doing research which involves an empirical investigation of a particular contemporary phenomenon within its real life context using multiple sources of evidence.” [99] Semi-structured interviews have been used to answer RQ 1, 2 and 3. A semi-structured interview has predetermined questions, but the order can be modified based upon the interviewer’s perception of what seems most appropriate. Question wording can be changed and explanations given [76].

To answer RQ 4, a case study was developed to investigate the usage of the developed method.

3.1.1 Method used for research questions 1 and 2

The data used to answer RQ 1 and 2 were obtained through analysis of semi-structured interviews at Volvo Cars and Scania. In total, 11 interviews were performed by the two authors who are native to Scania and Volvo Cars respectively (see [11] for the definition of “native” in this context).

1. The questions were developed and tested on people with similar roles at both companies, who were not included in the study.
2. All the architects who were available and willing to participate were interviewed, which resulted in more than half of the architects at each company participating, 4 at Scania and 5 at Volvo Cars. In addition to this, the managers for the architecture groups were interviewed at both companies, bringing the number of interviews to 11.
3. The interview was led by one person while the other took notes.
4. Each respondent had the possibility to read and comment on the notes from their interview in order to correct any misunderstandings, errors or other mistakes in the transcriptions.
5. The results were gathered in a database and analyzed.
6. The final results were reviewed by the manager at each participating company. The results of the study were presented for a broader audience at both companies.

3.1.2 Method used for research question 3

In order to answer RQ 3, the previous study was extended to include four additional companies. The same set of questions was used but a slightly different procedure was used:

1. The same set of questions was used.
2. The companies were selected through established contacts. All companies have significant development of software-intensive systems, but are different in size and production volume.
3. The architects were identified in collaboration with the contact person.
4. At least two interviews were held with architects at each company. With the respondents’ permission the interview was audio recorded.
5. The results of the study were presented to a broader audience at each company. During the presentation the situation at the visited company was also discussed.
6. Questions about the characteristics of each company were answered by the contact person.
7. The results were gathered in a database and analyzed.
8. The results were reviewed by the contact person at each participating company.

3.1.3 Method used for research question 4

A methodology has been developed to value the flexibility of design alternatives. To analyze the developed methodology and its industrial worth, the methodology has been applied to a real case in the automotive industry. Information about a current architecting case was acquired through discussion with the responsible architect. All available written information regarding the case was obtained and reviewed. The real case was then used to test the developed methodology and to evaluate the actual case. The result
from the case study was presented to the architect in order to be able to discuss its usefulness.

3.2 Validity

There are different types of threats to validity that need to be considered when conducting research. The different threats to validity are presented below, including the countermeasures made to improve validity.

3.2.1 Construct validity

Construct validity ensures that the studied artifacts can be applied to analyze this exact problem. To avoid bias in the respondents, a minimum of two people were interviewed at each company. All the written documentation of the interviews conducted in this research has been reviewed by the respondents. In the study used to answer RQ 3, the interviews were conducted by the author alone. Those interviews were recorded with the respondents’ permission and were therefore not reviewed. The final papers have also been reviewed by representatives of the companies in order to limit the risk of misunderstanding. The results of both studies have been presented for a broader audience at the participating companies, including a larger number of architects and managers. The presentations were another way of limiting the risk of misunderstanding. The working experience of the author will also help to ensure construct validity.

3.2.2 Internal validity

Internal validity ensures that the conclusions we draw from a study are the only ones possible and have not been affected by another possible cause. Internal validity is ensured by doing pilot interviews with informants similar to the ones questioned in the study. The questions can thereby be altered to ensure internal validity. The working experience of the author as an architect limits the risk of misunderstanding when talking to other architects. The analysis of the interview data used for RQ 1 and 2 was performed by the author, who is native to Scania, and another researcher, who is native to Volvo Cars. The analysis of the interview data used for RQ 3 was performed by the author, but using the method developed in the previous study.
3.2.3 External validity

External validity is the degree to which the conclusions in the study would hold for other organizations and at other times. Scania is used as a case-company in all parts of this research. The major threat to external validity is the degree to which the conclusions would hold for other companies developing software-intensive systems outside the automotive industry; therefore five other companies are included in the study of RQ 3. The companies studied are all developing long-lived software-intensive systems, but in different domains, and are of different size. All the companies have a very long history in Sweden, which is a disadvantage for external validity. This weakness is somewhat compensated by the fact that three of the companies have development outside of Sweden and all of them offer their products on a global market. Related work from other areas will also serve to support the validity of our studies.

3.2.4 Reliability

Increasing reliability is about minimizing faults and biases in a study and making the results repeatable. Reliability is ensured by well documented and planned case studies and interviews. Interview data has been stored in a database that has been used to analyze the data. Only a few questions used in the interviews have been published due to page limitations. The interview questions are to be made available online in the submitted journal paper. The study would be repeatable if it was repeated under the same circumstances. Unfortunately, circumstances do change; new people are employed, projects are dealing with different current issues, company strategies are changed and organizations are restructured. So for the part of the research that studies processes this might be difficult, because industrial processes are and should be continuously evolving. The results are therefore momentary views of a specific organization. The knowledge gained can still be transferred to another organization. The method and analysis would be repeatable and could therefore be used at another time for a longitudinal study.
Chapter 4.

Research results

This section summarizes the main contribution made by the thesis. The system development process at Scania was studied in prior work [39], to identify current practice within the automotive industry. The contribution of the thesis is based on the four publications. Figure 18 shows how the contributions relate to four dimensions of software engineering [90]. Paper A answers RQ 1, presenting how the architecting process can be improved by the use of Value Stream Mapping, and it thus mainly relates architecting to the process dimension. The different tasks found in an architecting process are presented in Paper B in order to answer RQ 2, and apart from the process dimension there is also a connection with organizational issues. To answer RQ 3, the successes of different methods related to the context in which they are used are discussed in Paper C, and this broad study relates to all four dimensions. RQ 4 is answered in Paper D, which presents how flexibility can be economically valued, thereby relating to the business dimension.
Chapter 4. Research results

This section summarizes the main contribution made by the thesis. The system development process at Scania was studied in prior work [39], to identify current practice within the automotive industry. The contribution of the thesis is based on the four publications. Figure 18 shows how the contributions relate to four dimensions of software engineering [90]. Paper A answers RQ 1, presenting how the architecting process can be improved by the use of Value Stream Mapping, and it thus mainly relates architecting to the process dimension. The different tasks found in an architecting process are presented in Paper B in order to answer RQ 2, and apart from the process dimension there is also a connection with organizational issues. To answer RQ 3, the successes of different methods related to the context in which they are used are discussed in Paper C, and this broad study relates to all four dimensions. RQ 4 is answered in Paper D, which presents how flexibility can be economically valued, thereby relating to the business dimension.

Figure 18 Relationships between the appended papers.
4.1 Paper A: Improving the system architecting process through the use of Lean tools

Value Stream Mapping (VSM) is a Lean tool used to improve value creation within a process. Using VSM on a manufacturing process has been promoted in the literature \[63\] and also evaluated \[48\]. Using VSM on a development process has also been promoted \[70\], but has to our knowledge not been evaluated within systems development.

**RQ 1: How can an architecting process be mapped in order to identify improvements?**

This paper is based on a case study at Scania and Volvo Cars and presents how Value Stream Mapping can be used to analyze the architecting process. Furthermore, it presents in general terms what types of wastes and improvements could be found. Waste could, for example, be due to handoffs, task switching, technology debt or delays. One result of this paper is to show how waste can be eliminated and maximize the value creation of the process through the use of Value Stream Mapping. An adapted version of Value Stream Mapping is found to be a suitable method for identifying improvement in the architecting process.

4.2 Paper B: Architecting automotive product lines: industrial practice

This paper aims at answering RQ 2 and presents an in-depth view of how architects work with maintaining product line architectures in the automotive industry. The study has been performed at Scania and Volvo Cars.

**RQ 2: What tasks are performed in the process of architecting automotive embedded systems?**

The striking conclusion and the answer to RQ 2 is the similarity between the two companies in the tasks performed when maintaining and changing architecture. The tasks mentioned by the architects at both companies are virtually identical; need ⇄ impact analysis ⇄ solution ⇄ decision ⇄ validation. The tasks do not seem to be different for architecture maintenance compared to developing a new architecture. Likewise, they seem to be the same whether it is updating a product line architecture or updating the architecture of a single-shot system. The study indicates what effect differences, such as a strong line organization or a strong project
organization, have on the architecting process. It also shows what consequence technical choices and business strategy have on the architecting process.

4.3 Paper C: A comparative case study of architecting practices in the embedded software industry

In Paper C, the methods used to solve the tasks within the architecting process are mapped to the context used in the industry. The results from Paper B are generalized by performing semi-structured interviews at four additional companies. One hypothesis was that in order to understand different architecting processes one must first understand the surrounding circumstances, the context. The attributes that should be gathered in order to understand the context were derived from the literature.

RQ 3: In what context are the methods used within the architecting process found successful?

To answer RQ 3, the paper studies the current state of architecting practices in three different industrial segments that are characterized by being software-intensive. An analysis of the case study indicates how different methods are more suitable to different environments. The context of the different companies, as well as the architecting practices, are compared and analyzed. Many of the successful practices found in the study can be explained by the context of the different companies. The use of global architects with their own budget in one company is a solution for initiating long-term architectural projects without having a customer order. The high degree of documented reasoning in the studied defense company is caused by the high degree of customer-specific demands and large orders of very similar products. This forces the architects to make branches of the architecture to fulfill customer demand, and the reasoning is then used to ensure quality. The defined architecting process found at one of the automotive companies and the use of visualization tools to track progress is explained by the strong influence of Lean Thinking. Other examples of practices, such as divided architectural teams and the lack of formal architects, are more difficult to explain.
4.4 Paper D: Evaluation of design options in embedded automotive product lines

Decision-making under uncertainty is influenced by a number of factors [52], and some of them lead to less rational decisions. The use of structured methods (design reviews, checklists, and expert support) is one way to improve decision-making.

**RQ 4: How can one value the flexibility needed to withstand an uncertain future in automotive embedded systems?**

A method and process is developed to answer this question. The developed method evaluates flexibility, using a concept called Real Options. The method is motivated and described by using an example from automotive embedded systems. To improve the usability of the method, a structured evaluation process is defined to aid practitioners such as developers and architects. The evaluation process provides a way of valuing system designs and enables the practitioner to think about the future in a systematic manner. Our literature survey has found three research contributions [6, 8, 13] that involve the usage of real options in system design involving software or hardware. None of them explicitly addresses embedded systems or the automotive domain.
Chapter 5. Discussion

In this chapter I take the opportunity to be less formal and give my view on some topics related to this research. I believe this chapter can be an input in the current discussion and perhaps provide ideas for future research. In the first section I try to explain what Lean architecting would be. The following section presents the difficulties of researching industrial processes. Finally the industrial impact of the research results is discussed.

5.1 Lean architecting

So what is Lean architecting? There is no right or wrong answer to that question, based on reasoning about the effect of context on how work is done. Based on the knowledge gained during this research I will still give my opinion on what Lean architecting of software-intensive systems would look like. A Lean product development process is frontloaded; much work is done early in the project in order to lower the risk later when the cost is much higher. Early design decisions often have a greater impact on the overall system than the developer alone can foresee. Architecting efforts should therefore be started early and in close collaboration with the development team. In the early phases, it is important to keep the design flexible without investing too much in architecture, bearing in mind the uncertainty.

Challenges in different development projects are very different, as is the necessary architecting. The process should therefore not be fixed, although each task can be described in detail to ensure nothing is forgotten. In this way, every architecting effort will be tailored to fit the needs of the specific case. Each task will be repeated regularly and checked for improvements after each use.

Reusable architectural knowledge such as heuristics, principles and patterns should be transferred through lightweight documentation, education and mentoring. By empowering people and giving them the necessary tools, but
without being obsessed by tools, quality will be designed into the system. The resulting artefacts should be challenged regularly to identify what documentation the internal customer needs. In an evolving complex system there will always be an increasing technical debt. Changes to the system will be more and more difficult to realize, therefore the technical debt must be monitored to plan for radical changes. Last but not least: in order to get a full return on the investment in Lean, the whole organization must apply Lean Thinking.

5.2 Identifying best practice

In an industrial setting, it is very difficult to produce empirical evidence to support that one method is better than another. In order to show anything, you need to measure the performance of the process before and after the change. If this is done you will get an indication of whether the change improved the process or not. If the tasks performed are comparable and made by the same group of people you will be more confident of the results. In order to study whether one improvement is better than another you would need to evaluate the alternatives. People or organizations that promote one practice often show how the performance of the development process has been radically improved by introducing a toolbox i.e. Six Sigma, Agile or Lean (Figure 19). The problem is that companies do not tend to fix something that is not broken. A development organization that is sinking in mud is very likely to accept a helping hand. The performance improvements will probably be very positive, but that does not mean that it was the best possible solution. Almost any toolbox handed to them would have helped them out of the mud and improved performance! Based on this reasoning, this research does not claim that the practices found are best practices. They are found suitable in a specific context and should be critically reviewed before being used in other contexts.
5.2 Identifying best practice

In an industrial setting, it is very difficult to produce empirical evidence to support that one method is better than another. In order to show anything, you need to measure the performance of the process before and after the change. If this is done you will get an indication of whether the change improved the process or not. If the tasks performed are comparable and made by the same group of people you will be more confident of the results. In order to study whether one improvement is better than another you would need to evaluate the alternatives. People or organizations that promote one practice often show how the performance of the development process has been radically improved by introducing a toolbox i.e. Six Sigma, Agile or Lean (Figure 19). The problem is that companies do not tend to fix something that is not broken. A development organization that is sinking in mud is very likely to accept a helping hand. The performance improvements will probably be very positive, but that does not mean that it was the best possible solution. Almost any toolbox handed to them would have helped them out of the mud and improved performance! Based on this reasoning, this research does not claim that the practices found are best practices. They are found suitable in a specific context and should be critically reviewed before being used in other contexts.

5.3 Industrial impact

Some time has elapsed since some of the studies included in the research project were finalized. Below is my understanding of the changes that have been made since the studies were presented.

In a case study included in the licentiate thesis [39], the system development process at Scania was studied to identify current practice. The result of the study was presented at Scania in different groups and forums. The following three improvements were suggested and prioritized:

1. Strengthen the role of the technical career path
2. Improve knowledge transfer by documenting design know-how
3. Educate engineers in the use of structured methods

Three years later, a great deal of work has been done to strengthen the role of the technical career path. There are now more specialists within system development and appointments are highlighted to the rest of the organization. Different activities have been carried out to improve knowledge transfer. Design guidelines are more frequently updated and Wiki solutions and A3 documents are becoming a more common way of sharing knowledge. Of course, this is a never-ending story, but there have been a few steps in the right direction. To my knowledge very little has been done to improve the usage of structured methods.

Paper A presented how Value Stream Mapping could be used to improve the architecting process. Since that time there have been various efforts to use Value Stream Mapping for different processes within system development.
A paper [83] co-authored by me shows how Value Stream Mapping can be implemented in practice at Scania.

One and a half years after the case study leading to Paper B was finalized and presented at the two companies, some interesting changes have taken place. The architecting group at Scania has grown, both in the number of architects and their experience. Tool support for the architects has been significantly improved. The two separated architecting groups at Volvo Cars have merged into one, resulting in only one single architectural description. The group responsible for testing and validation at Volvo Cars is now part of the same section as the architecting group.

I do not believe the changes were made because of the results presented from those two studies, even if I hope the ideas presented due to the studies inspired the change. It does however indicate the correctness of the results or at least indicate that the results correspond to industrial reasoning.

An evaluation process using Real Options was presented and tested in Paper D. The evaluation process provides a way of valuing system designs. I believe that the method is correct and will provide improved decisions support. The problem, as with many other methods, is that the information needed is rarely available in industry. When presenting the evaluation process, I have often been given a positive response to the thoughts behind the method. Architects and people responsible for parts of the system like the idea that the increased cost of a flexible design could be argued using financial measures. To be used in industry it would need to be even more lightweight and used to guide discussion, rather than the decision itself. The greatest contribution to industry is probably a structured way of reasoning about design alternatives as options that can be valued.
Chapter 6. Conclusions and future work

The overall goal of this research has been to investigate how system architecting is performed in the automotive industry and how it can be improved by the use of Lean Thinking. This chapter presents conclusions and future work.

6.1 Summary of results

An adapted Value Stream Mapping was tested on a case study at two different companies. A comparison between the two companies shows that there are a number of value-adding methods that could be borrowed from one company to the other. It also highlighted how no formal evaluation step of architectural alternatives were made; evaluation was only mentioned as occurring in rare cases. The results of the case study have been presented at the two companies, which found them interesting but most of all inspiring for their future process improvement. The indicator that best shows that the mapping was valuable to the companies is that the presentation was requested to be held twice.

One of the case studies reveals that the studied architects see themselves as interacting much more with other stakeholders than architects in general. The results indicate how the company’s different core values influence the architects when defining and maintaining the architectures over time. It also indicates the consequences that technical choices and business strategy entail for the architecting process.

This work provides a current view of the architecting process for software-intensive systems. Many of the architecting practices found in the study can be explained in the context of the different companies. A list of practices is provided for the industry reader and can be used as an inspiration or as a benchmark for improving current architecting practice.

A method has been developed to improve decision-making when making architectural changes in early phases within the automotive industry. The
developed method uses real options to provide guidance when making system design decisions and, more importantly, also shows that it can be used and accepted by system engineers.

It is important to stress that success is not achieved through the use of specific tools or methods. Using the right tools and methods will often simplify or enhance the process, but having the right people with the right mindset aligned toward a common goal is much more important; developing the employees and creating an organization that never stops improving is far more important.

### 6.2 Future work

During this research we have seen how the balance of power between line and project has a strong influence on how work is done. This relationship would be of interest for a future study. The connection between business strategy for Cost, Quality and Time-to-Market and architecting could also be further analyzed.

Value Stream Mapping is a frequently used tool for identifying improvements in a process, but there are few industrial examples of when it has been applied to parts of the development process.

The thesis has shown that communication is a large part of architecting activities and, in order to be Lean, the communication must be effective. Kruchten [58] suggests that the productive time spent by architects can be classified into three categories of communication: internal (architecture design), inwards (input from outside world) and outwards (providing information). He argues that they should roughly have the ratio 50% internal, 25% inwards, and 25% outwards. It is very hard to measure this in practice and we have not done so in this study, but communication patterns can still be observed. Even if no extreme variation can be seen, the understanding from this study is that there is a clear difference between the companies. The architects tend to be more satisfied when the inward and outward communication is distributed evenly and where the internal work is of significant size. Future research on how communication patterns vary depending on different contexts could improve the process, aid cross-cultural-teams and enable Lean architecting.

The use of Open Innovation is growing in many different domains. The software industry is moving more and more towards different types of open solutions. Open source software enables end users to add features to the
product and user involvement, such as Wikipedia, is very common, although few or no attempts are made within the automotive industry. One exception is the App-My-Ride contest arranged by Volkswagen [95]. A future research question is therefore how open innovation will enter the automotive domain and what new challenges the industry will face. The architecture would need to be adapted to accept new features being added in the aftermarket while keeping the same quality.

Working as an industrial Ph.D. student means that I have been employed as a researcher in industry and enrolled as a Ph.D. student in academia. This position means that I have experienced how industry and academia demand very different outputs from architecting research. The industry asks for best practice or success stories, while academia looks for practices proven in general. This difference could be further discussed and enable more effective knowledge transfers from academia to industry.
References


References

[1] "Applying the IEEE 1471-2000 Recommended Practice to a Software Integration Project."


<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
<th>Publisher/Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>[77]</td>
<td>M. Rother and J. Shook</td>
<td>Learning to see : value stream mapping to create value and eliminate muda</td>
<td>Brookline, MA: Lean Enterprise Institute, 2003.</td>
</tr>
</tbody>
</table>
References


