MASTER

Firefighting in an embedded systems context

van Hellemont, S.H.J.

Award date:
2014

Disclaimer
This document contains a student thesis (bachelor’s or master’s), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain.
Firefighting in an embedded systems context

By: S. van Hellemont
BEng. Mechanical Engineering (2010)
Student identity number 0758096

In partial fulfillment of the requirements for the degree of

Master of Science
in Innovation Management

Supervisors:
Dr. Ir. B. Walrave, TU/e, ITEM
Dr. S.A.M. Dolmans, TU/e, ITEM
Disclaimer

The views expressed in this thesis are those of the student and do not necessarily reflect the views of the Technical University Eindhoven and other stakeholders involved in this research project.
TU/e. School of Industrial Engineering.
Series Master Theses Innovation Management

Subject headings: Firefighting, Project management, Agile, Phase based, New product development, Embedded systems
Executive summary

Introduction

Innovation has become important for companies in order to maintain competitive. Consequently, project management has become the most important issue for most organizations to deliver new products from idea to the market. However, several methods on how to approach this project management exist. There is the traditional view, where projects are divided into several phases. Where these phases are executed sequentially. This traditional project management (TPM) is very suitable for repetitive project with a low amount of uncertainty. However when uncertainty increases, TPM become more difficult to use. On the other hand there is Agile software development. This method is created as a reaction to TPM, because many project failures were recorded. Especially in IT project where are more uncertain and complex. Therefore, Agile has an flexible approach to project management in order to adapt to the changing uncertain project environment.

Currently, in new product development products are often developed using multiple companies. Combining knowledge and resources to develop faster and better products, while sharing the risks. For example, in Embedded system, where previously the whole system was developed at one company, now the development of software and hardware are divided over multiple companies. Consequently, when these partners use different project management methods things become even more difficult in the already complex environment of embedded systems. Nevertheless, literature suggest these differences should not be a problem, however, using agile in embedded system is still not common. Why is this? Which problems complicate this process, and what are the main causes? Therefore the research question of this research project is:

How do the problems in a project, where ASD and TPM are used as a development method by multiple partners, emerge over time? What are the root causes and consequences of these problems, and how can they be prevented?

Method

This research question was answered using a process study. In this process study, three case studies were examined over time. In addition, the results from this process study were further used to develop a process theory. These case studies were done at SoftCo which is a large software development company. In the investigated projects, embedded system were developed by multiple partners. SoftCo was responsible for the software development and used ASD as a development
method, while the other partners developed the other components using a TPM approach. Therefore, these cases were a good fit to investigate the research question.

Results
The result of the process study show that many problems occurred. There were dependency problems. This was on an internal level with another department of SoftCo and on an external level between different suppliers. In addition, there were resource problems, due to interference of other projects, changes within the SoftCo organization, or due to unexpected work. Furthermore, there were the collaboration problems. This was around bad communication and transparency between the different stakeholders. These problems caused the project pressure to increase in all investigated projects. However, the way these projects handled this pressure was different. In two cases there was chosen to shift resources from rework discovery activities to development activities. Nevertheless, this short term fix backfired on the project, because in the end the system became very instable and caused the project team to put many effort into solving these issues right before the deadline. In one project, these issues were so severe that firefighting was triggered. This is a phenomenon where engineers are re-allocated in the project to fix unexpected problems, which were discovered late in the project. Firefighting can lead to a huge increase of costs, slipping deadlines, and decreasing performance; therefore, it has to prevented.

Meanwhile in the third case a scope decrease was achieved together with the involved partners. Consequently, the project was delivered successfully according the revised scope and on time.

From these problems a process theory was formed as an addition to the current firefighting literature. This theory identified bad collaboration, not managing dependencies effectively, and shifting resources from rework to development activities as causes of firefighting.

Discussion and conclusion
In investigating the product development process were ASD and TPM were used as an development process, several problems emerged over time. The problems that were linked back to the ASD and TPM context were the problems regarding the dependencies. ASD and TPM require information from each other at different times in the product development cycle. Therefore, in order not to delay each other, they have to adjust. ASD has to join in fixing requirements as fast as possible, so that the hardware part, developed by TPM partner can be made. TPM on the other hand has to deliver validation equipment in cadence with the ASD sprints so that the software can be validated against the hardware. This can be done using rapid prototyping techniques. Other problems which are not
specific to the ASD and TPM context, though equally important are collaboration and transparency. The development teams should have one goal of making the best product possible. Furthermore, the TPM partners should become more flexible. They should not approach the project planning rigid. A certain amount of flexibility should be built into the planning in order to react to unexpected problems. This should keep the project pressure at an acceptable level. When these recommendation are taken into account, ASD and TPM can be used together in the embedded systems context without firefighting.
Index

Executive summary ................................................................................................................................. iii

1. Introduction ..................................................................................................................................... 1

2. Literature review ............................................................................................................................. 3
   2.1 Traditional project management................................................................................................... 3
   2.2 Agile software development ......................................................................................................... 7
   2.3 Combining agile software development and traditional project management....................... 10
   2.4 Conclusion literature ................................................................................................................... 13

3. Method: ......................................................................................................................................... 13
   3.1 Research Design .......................................................................................................................... 13
   3.2 Research context ......................................................................................................................... 14
   3.3 Case selection .............................................................................................................................. 14
   3.4 Data analysis ................................................................................................................................ 16

4. Results ........................................................................................................................................... 18
   4.1 Project S Narrative ....................................................................................................................... 18
      4.1.1 Phase 1: False start ............................................................................................................... 18
      4.1.2 Phase 2: Gear-up phase ........................................................................................................ 24
      4.1.3 Phase 3: Bug-fixing phase ..................................................................................................... 28
      4.1.4 Phase 4: Stabilization phase ................................................................................................. 33
      4.1.5 Phase 5: All or nothing .......................................................................................................... 37
   4.2 Project D narrative ...................................................................................................................... 38
   4.3 Project M narrative ..................................................................................................................... 42

5. Analysis .......................................................................................................................................... 45
   5.1 Project S ....................................................................................................................................... 45
   5.2 Case comparison .......................................................................................................................... 48
   5.3 Process Theory ............................................................................................................................ 52

6. Discussion ...................................................................................................................................... 53

7. Managerial implications ................................................................................................................ 56

8. Limitations and future research .................................................................................................... 57

9. Conclusion ..................................................................................................................................... 58

References: ............................................................................................................................................ 59

Appendix I: Interview questions ............................................................................................................ 62
1. Introduction

Innovation is the process of bringing new products and services successful to the market. In order for companies to maintain competitive, they have to innovate (Hauser et al., 2006). By continuously innovate and invest appropriately into new products, long term success can be achieved (Kester et al., 2011). However, this is more difficult than it sounds and many companies fail in this process (Hauser et al., 2006). In the search for new successful products, companies only have limited resources available to do so and every new product development (NPD) project has to be successful on the NPD battlefield (Cooper and Kleinschmidt, 2001; Kester et al., 2011). Consequently, project management has become the most important issue for most organizations (Hauser et al., 2006; Maylor, 2001).

Before World War II product development was generally a trial and error method. However, due to huge losses and the increase of larger and more complex projects a more structured approach was needed. In the 1950s a phase-based approach was developed and was based upon the idea that system requirements can be fully specified up-front with a detailed sequential planning. This planning would then be executed accordingly (Kerzner, 2009). Even though this method is originated more than fifty years ago, only small changes have been made to it so far, and it is still widely used in construction and engineering (Maylor, 2001).

Nevertheless, there are also companies who have moved away from this phase based approach or also known as traditional project management (TPM) approach. This was caused because of the lack of fit with their current projects. The project environment had radically changed. Especially in IT project, which has become highly complex and due to that, many project failures were recorded (Maylor, 2001; The Standish Group International, 1995). As a reaction to the changed environment, Agile Software Development (ASD) was created by the software development community (Fowler and Highsmith, 2001). This method focuses mainly on adapting to change and delivering high quality products through simple processes (Dingsøyr et al., 2010). Although, no standard definition of ASD exists in the literature, the most comprehensive definition is “The continual readiness [...] to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment” (Conboy, 2009: P340).

Both methods have their advantages and disadvantages. Traditional project management (TPM) is very suitable for repetitive projects with low uncertainty due to its upfront planning and sequential
execution process (Stober and Hansmann, 2010; Wysocki, 2014). However, only a small part of the projects are repetitive and carry a low amount of uncertainty. When uncertainty is increased, the requirements often change too. Therefore, also the detailed planning has to change, resulting in many non-value added work and wasted time (Wysocki, 2014).

Dybå and Dingsøyr, (2008) reported about benefits about ASD that “project changes are incorporated more easily [and] business value is demonstrated more efficiently,” (p850), Though, because ASD is different, it is difficult to implement into a company, or to use for large projects (Boehm and Turner, 2005). Furthermore, ASD requires much more customer involvement than TPM, which is not always an option due to the lack of time at their side. This, reduces the benefits of ASD (Hoda et al., 2011).

Although these methods have different development paradigms, if used on their own this will not be a problem. Nonetheless, what happens if these methods are used together into one project? Projects where TPM and ASD can be used together is in the development of hardware/software products. During this process knowledge from multiple engineering perspectives and/or companies is integrated. Currently, software is often developed by using ASD and hardware by using TPM (Punkka, 2012). These hardware/software products, for instance embedded systems are becoming increasingly important in modern society and can be found in a wide variety of electronic devices such as; smartphones, digital cameras, coffee machines, etc. (Albuquerque et al., 2012; Vahid and Givargis, 2002). In addition, there is a high need for speed in the development process due to the globalization (George et al., 2005). Consequently, these hardware and software components need to be developed simultaneously which increases the complexity of the development process (Punkka, 2012). During this “co-developmental” process, software development is dependent on hardware development, because they specify constraints in which the software teams can operate. If this co-development process is not sufficiently aligned and these constraints are not specified early on, the software development team cannot continue properly. This can jeopardize the functionality and reliability of the product (Ronkainen and Abrahamsson, 2003; Wolf, 1994)

In order to overcome the problems in the development of products using ASD and TPM several recommendations have been made. Examples of these recommendations are the early identification of upfront requirements (Abrahamsson et al., 2003), alligned project cadence, early on prototyping (Punkka, 2012), appropriate test suites (Kaisti et al., 2013), (Abrahamsson et al., 2003), and functional responsibility (Boehm and Turner, 2005).
Literature suggests that ASD and TPM can easily be used together. Many recommendations are given to succeed in this process (Dybå and Dingsøyr, 2008; Kaisti et al., 2013). Though, combining these methods is still not common practice in e.g. embedded systems (Eklund and Bosch, 2012; Kaisti et al., 2013; Punkka, 2012). Why is this still not the case? Which problems complicate the process of product development using ASD and TPM together, and how do they influence this process? This leads to the overall research question:

How do the problems in a project, where ASD and TPM are used as a development method by multiple partners, emerge over time? What are the main causes and consequences of these problems, and how can they be prevented?

By identifying these problems, they can be prevented and the advantages from both methods can be used in this co-development process. The developmental process will benefit from this in several ways. First of all, by improving the new product development process, the quality of the product will be better in terms of better reliability, better performance and features, and lead to higher customer value. Second, the development time will decrease. This saves money and potentially creates a higher market share due to earlier market entrance. In addition, the developmental time can also be decreased. Finally, the company will gain competitive advantage over competitors on the long run. Consequently, it will create organic growth due to the fast and continuously delivery of new products on the market with higher margins and more customer value (George et al., 2005).

One aspect that is important in this process is that the involved partners should align their developmental processes. Even if these processes are different, clear appointments should be made when and what kind of information should be available and keep each other up to date on this process accordingly.

2. Literature review
By using state of the art academic literature, an overview will be provided on what is known on the current research topic.

2.1 Traditional project management
Project management has been gradually evolving over the centuries. In the early 1900’s project management was generally an unstructured flexible trial and error method with parallel trails. This method gave alternating results, with the Manhattan project as one of the major successes (Lenfle and Loch, 2010). After World War II the demand for a structured product development process grew,
especially by the US government. Many failures were registered in the Cold War space race development and cost overruns where up to 300%. Consequently, these failures it got blamed on the inappropriate use of project management (Kerzner, 2009). The first breakthrough in project management literature was in the 1950’s where Charles Gantt, developed his Gantt Chart, which is still a common used project management tool. After this “breakthrough”, project management left its flexible trial and error method, and the phase based project approach became a popular research area where project management techniques were developed such as PERT and CPM and led to the traditional project management (TPM) approach (Lenfle and Loch, 2010).

Although there are several versions of this TPM approach, in general they are composed out of the same five phases, which are executed sequentially (fig 1). In the first phase an potential new product idea will be evaluated and the feasibility of the project will be determined. The second phase, a detailed planning will be made on how the project will be executed. Also the impact on the required resources (e.g. time, costs, performance, and schedule) will be calculated in this phase. Third, In the definition and design phase, the product will be developed. Fourth, when the entire product is ready, it can be validated with the customer and introduced onto the market. This happens in the implementation phase. Finally, in the conversion phase, the project is monitored and when at the end of its lifecycle it can be closed (Kerzner, 2009; Wysocki, 2014).

**Evaluation of traditional project management**

If there is no uncertainty and complexity in a project, TPM is an efficient and logical way to conduct a project. A detailed project plan will be made upfront and executed accordingly. This approach is especially appropriate for repetitive projects with low uncertainty (Stober and Hansmann, 2010; Wysocki, 2014). However, only a small part of the projects are repetitive and carry a low amount of uncertainty. When uncertainty is increased, the requirements also often change, followed by a change in planning and resulting in many non-value added work and wasted time (Wysocki, 2014). This problem is caused by the difficulty to plan uncertainty. Each task has its own level of uncertainty, expressed in variability. Innovative tasks carry more uncertainty and thereby, have a higher amount
of variability. Therefore, the higher the variability of a task, the more difficult it is to predict and plan the duration of the task (Reinertsen, 2009). Tomke and Reinertsen (2012) recommend that that excessive planning is not recommended when dealing with innovative projects. In these projects requirements change often, because of their uncertain nature. The plan should consequently be treated as a hypothesis that is continuously revised when new information becomes available. Furthermore, because the validation of the product is done at the end of the project most of the problems show up when the design is almost finished. These unexpected tasks require an additional amount of time, which is not taken into account into the project planning. This can potentially delay the due date. In addition, customers do not always know what they really want, and requirements can change during the project causing delays (Beck, 1999). Lenfle and Loch (2010) pointed out, that TPM causes managers to spend too much time with bureaucratic work and documentations instead of focusing on strategy and innovation that brings value to their customers.

**Stage-gate:**
The stage-gate method is a phase based product development system created in the early 1980’s by R. Cooper who did research on how winning companies executed their new product development. He identified that although many companies used project management methods such as TPM, they were still failing to innovate. In the stage-gate method, the product development process moves through various steps to go from idea to launch as can be seen in fig 2 (Cooper, 2001). The stage gate process starts with an ideation stage, followed by the sequential stages of scoping, building business case, development, testing & validation, launch, and post-launch. Each stage ends with a hard decision gate, a so-called go or kill decision, to serve as a quality control mechanism (Cooper, 2008).

Over the years the method evolved, becoming less rigid, because the world became faster passed, global, competitive, and less predictable. Therefore, Stage-gate should be seen more as, “a blueprint for managing the new product development process to improve effectiveness and efficiency” (Cooper, 2008: p214) instead of an inflexible phase based approach from the 1960. This blueprint enables flexibility and adaptability to market changes and uncertainty. Since the first design several updates of this method have been developed. The newest update of the method led to Cooper’s latest
model; The triple A method (fig 3), where key values as Agile, Adaptive, and Accelerated form the basis to answer the need for a faster and more flexible development process (Cooper, 2014).

An advantage of the stage gate method is that it is a lightweight dynamic method. It is a map/blueprint used to go from A to B, not restricted by regulations, and documentations, to increase the speed of the total NPD process (Cooper, 2008). It gives managers their long desired control mechanism in developing new products in the organization (Sethi and Iqbal, 2008). Furthermore, by using cross-functional teams, the total process becomes business orientated instead of functional. This has the advantage of creating more commitment in the project, leading to higher performance. Therefore, tasks can be executed more easily in parallel. It avoids lengthy and over the wall processes, because the whole business is involved with the project (Cooper, 2008). In addition, Cooper further poses that stage-gate should not been seen as a linear system. Within these phases, iterations and loops are possible. Furthermore, by using rapid prototyping and spiral development in the development stage, customer feedback can used to optimize the design.

One of the disadvantages of stage-gate is that the method does not have opportunities to re-plan the project as a result of changes in the project. A project moves through the stage gate, stage after stage. Nevertheless, when corrections have to be made, and a project has to go back to a previous stage, is has to go through the gate review process again. This makes the stage gate a difficult and time consuming method (Becker, 2006). Van Oorschot et al. (2010) point out the difficulties regarding the first stage of the stage gate, the fuzzy front end. During this phase, ideas have to be developed and evaluated on potential business value. When uncertainties around these ideas are high, it is easy make a false negative (e.g. kill a potentially successful product), because only limited resources are available for this evaluation process. Sethi and Iqbal (2008) concluded that when there are strict gate restrictions, the NPD process will become inflexible and worsens learning process in a project. This is found to be crucial when working with novel products which requires a certain amount of flexibility.
2.2 Agile software development

During the 1980s the TPM approach caused more and more problems in software development. Due to the yearly improvement of microprocessors that had to be programmed, continuously larger and more complex projects had to be developed under tight schedules (Wolf, 1994). The TPM approach was not flexible enough to deal with this complexity and increasing time to market pressure. Consequently, many project failures were reported (Leffingwell, 2011; Maylor, 2001; The Standish Group International, 1995). As a reaction to this, the software development community developed new methods, to deal with the need for speed and innovation. These methods started with Spiral development, Rapid Application Development (RAD), and Rational Unified Processes (RUP) and can be placed on an continuum of becoming increasingly iterative (Leffingwell, 2011). In 2001 under the guidance of Fowler and Highsmith all these development ideas were combined into the Agile Manifesto. This manifesto provided the basis of where all current Agile methods are based upon.

The key values of the agile manifesto are:

1. **Working products** over comprehensive documentation.
2. **Responding to change** over following “the plan”.
3. **Customer collaboration** over contract negotiation.
4. **Individuals and interactions** over processes and tools.

Consequently, from the agile manifesto, *Agile software development (ASD)* was created. It focuses mainly on adapting to change and delivering high quality products through simple processes (Dingsøyr et al., 2010). ASD develops products in an iterative releases and translates customer requirements into prioritized product features (Wysocki, 2014). At each iteration features are added to the product. After one or more iteration(s) a potential product can be sent to the customer. This can be seen in figure 4 which represents the ASD project lifecycle. The customers can on their turn evaluate the product and provide feedback on a regular basis. Consequently, changes to the product due to complexity and uncertainty are discussed timely with the customers. This ensures the project team that at the end of the project, the customer is satisfied with the end-product (Wood et al., 2013).
Abrahamsson et al. (2003) identified that since the “birth” of ASD, several versions of the ASD method have been developed, such as Adaptive software development, Crystal, Dynamic system development, Extreme programming, Scrum, Feature driven development. Each method focused on different aspects of the Agile vision. In addition, the authors further compared these different methods with each other and noticed the need for a more uniform ASD vision. Because each method focused on different aspects, they were not suitable for every situation. As a result, it is not beneficial to describe every method in detail in this report. Therefore, the most common method retrieved from the 8th annual state of agile survey (2013), is chosen to further elaborate namely; Scrum, developed by Schwaber (1997).

The main focus of Scrum is flexibility, adaptability, and productivity to operate effectively in an complex environment (Schwaber, 1997; Dingsøyr et al., 2010). Together with the customer the required features are determined and organized on priority in the product backlog. Consequently, these features are broken down into tasks. For each sprint tasks are planned, based on the velocity (task execution rate) of the development team, for the sprint backlog. These features are then developed and tested by the Scrum teams. The goal is to work in short sprints, which are on average 2-4 weeks and after each sprint a potential launchable product can be delivered to the customer. (figure 5). Un their turn they can provide feedback on the product, in order to align the product with the customer needs. This all can be realized by keeping communication lines short by using day to day communication and co-located team members (Kniberg, 2007).

Evaluation of agile software development

One of the advantages of ASD is found to be the scope and quality management principles. As a project manager of an ASD team it is important to calculate the task execution speed or “velocity” of the team. By knowing this measure, iterations can be planned more precise. Thus, it is found to be a powerful management tool (Leffingwell, 2011). The quality management principles of testing, frequent reviews, and demonstrations is found to be a major advantage of ASD (Fitsilis, 2008). An engineer or programmer can only continue to a new task when the current task is completed, reviewed, tested, and demonstrated fully functional with no faults/bugs. This prevents the project
from unexpected rework at a later stage by those “almost finished” tasks, and potentially endangering the whole project (Kniberg, 2007). Another advantage of ASD is that it is built upon teamwork using self-empowered teams and co-located teams with daily meetings. These teamwork principles increase communication, which is a key factor of project success (Pinto and Slevin 1987). In addition, due to increased communication, changes in the project can be managed quickly and effectively (Fitsilis, 2008; Leffingwell, 2011).

As already stated in the Agile manifesto, customer involvement is one of the core values of ASD (Fowler and Highsmith, 2001). Together with these customers, the features that need to be developed can be discussed and prioritized, and consequently placed on the backlog (Dingsøyr et al., 2010; Kniberg et al., 2011; Leffingwell, 2011). Dingsøyr et al. (2010) further identified that close customer involvement prevents the phenomenon of overloading the developing teams with large obscure packages of requirements, because customers are more committed to the project.

ASD also has disadvantages. Fitsilis (2008) for example, mentions ASD does not explicit manage risk, because it does not perform risk analysis and extensive planning mechanisms. However, ASD advocates argue they manage risk differently. Instead of investing heavily into documentation and risk analyzing activities, risk is identified using estimation techniques. Consequently, it is managed accordingly by developing the high-risk, high value, low cost activities first in order to reduce the maximum amount of risk at the lowest costs. Secondly, after every sprint, the risk is reassessed again in order to maintain up-to-date project estimations (Boehm, 2002; Leffingwell, 2011). Finally, the ASD team derives their agility by relying on the teams knowledge and capabilities in order to respond to changes caused by risk and uncertainty (Fitsilis, 2008; Boehm, 2002).

Another disadvantage of ASD is around costs and procurement management. More specifically, they evolve around contracting (Fitsilis, 2008). Frequently, when a project is conducted in a buyer supplier context, companies use a fixed contract. This means that scope, timing, and costs are determined upfront and the suppliers are required to deliver accordingly. These buying companies prefer this type of contracting because it gives them a perceived sense of control in a world full of uncertainty (Hoda et al., 2009). However, this is in contrast with the ASD way of project management. Product development is an uncertain, flexible process and difficult to plan (Punkka, 2012). As a result it is very difficult to work in an uncertain environment with fixed contracts where resources, timing and scope are fixed upfront. Due to this uncertain nature of product development, it is not sure if the fixed contract can be realized, exposing the service provider with the complete project risks (Hoda et al., 2009).
Successful customer involvement is found to be one of the key features of ASD. However, this is not as easy as it sounds, especially when high levels of customer involvement are required. Hoda et al. (2011) revealed five problem areas and their consequences around customer involvement in ASD projects. The first problem is skepticism and hype resulting in customers under or overestimating the advantages of ASD. Consequently, using it inappropriately. The second problem is that customers are not always easily available for information requests, due to for example different time zones. This can lead to delays in the development process. The third problem is around ineffective customer representatives who do not have enough knowledge of software development, nor the customer requirements. This causes them to be not much of help to the ASD team. The fourth problem, is dealing with large customers who did not want to change to the ASD way of working. The current project has low priority and therefore they wanted to keep doing things their own way. The final problem was the lack of time commitment by their customers. No extra time was allocated to the customer representatives to engage in excessive collaboration. These problems had several consequences such as: difficulties with gathering, clarifying, and prioritizing requirements, loss of productivity due to delays in information, lack of customer feedback resulting in an inferior product, and in some cases conflicts which lead to a loss of business.

2.3 Combining agile software development and traditional project management
In the previous chapters ASD and TPM was discussed and evaluated. However, when these methods are interacting together, new difficulties show up. In the literature, two different interaction settings were found. In the first setting ASD is used in a TPM environment. In the second setting a project is executed using both ASD and TMP. These two settings were discussed below and recommendations were made on how to overcome the differences between these two methods.

Using ASD in a TPM environment
As already discussed above, when developing products using ASD for TPM suppliers, fixed contracts can cause many difficulties. Firstly, to solve this issue it is important to have sufficient customer involvement. The customers should understand and recognize the ASD way of working. In return they will receive flexibility in changing the requirements mid-ways. Features could be added, changed, or deleted to create a product with the highest customer value (Kniberg, 2007). Hoda et al. (2009), acknowledged the importance of customer involvement in fixed price contracts. In these contracts the timing, costs, scope and quality are predetermined. This works very good with predictable project. Tough, this is not the case with project with many uncertainty such as product development. Therefore, the authors argue that a change in the mindset of customers has to occur, because it is normal to them to use the fixed contract setting. Therefore, to resolve this issue, the
attitude that fixed contracts are the only option to do contracting should be changed. When this mindset is changed, different types of contract options can be provided. For example, it is possible to sell iterations. After each iteration a potential shippable product can be released. Next, together with the customer decisions can be made if more iterations are needed to improve the product. Using this option, trail basis iterations can be proposed to make the customer familiar with the advantages of ASD. Another option is to stress the ability to change for free. If a feature needs to be changed after a few iterations, this can be done without changing the contract or adding extra costs (Hoda et al., 2009). Nevertheless, while these options provide good alternatives, some customers simply do not want to change. This different way of working through customer involvement requires additional work from them which is something they are not always willing to do. Consequently, it can lead to a loss of profit or even business in some cases (Hoda et al., 2011, 2009).

As already discussed, customer involvement, is one of the core values of ASD. Using customer involvement requirements can be prioritized accordingly. Nevertheless, this process of prioritizing requirements and features, with for example, user mapping, and other ASD procedures, is much more difficult than is seems. Racheva et al. (2010) identified, for example that good customer collaboration is seldom achieved. Furthermore, the authors acknowledged that because the lack of collaboration, the perceived business value of certain features is not in line between the ASD team and the clients. These problems could lead to a product that does not bring the customer value that it was intended for. Hoda et al. (2011) focused on situations with a lack of customer involvement. In their research they interviewed numerous companies, and identified and proposed solutions six situations evolving regarding problems with customer commitment. He proposes ASD undercover strategies to solve these problems. These strategies evolve around creating awareness of the ASD methodology, ownership of ASD principles, and facilitating customer involvement. In general, the cooperation level should be changed form a buyer supplier relationship to a partnership. Nonetheless, it should be realized that customer involvement requires much more commitment from the customers than TPM, often resulting in working overtime. To help the customers, special teams and activities should be organized to help with this collaboration, making it not feel like a one side effort for the customers (Dingsøyr et al., 2010).

**Using ASD and TPM together in a development process.**

An area where ASD and TPM can develop products together is embedded systems. The hardware of the system can be developed using TPM and the software using ASD. However, as already mentioned, these methods are different from each other and co-development can cause problems. Conversely, even though guidelines are given how to organize this process, co-development in
embedded systems using ASD and TPM is not a common practice (Dybå and Dingsøyr, 2008; Eklund and Bosch, 2012). Ronkainen and Abrahamsson (2003) note that unless it is contrary to the original ASD principles, upfront requirement definition and documentation cannot be avoided in embedded systems projects, because of the involvement of several stakeholders. In addition, this documentation should be up-to-date on all times, and is effectively shared with the stakeholders. To help with this problem, specialized documentation systems can be used for this (Kaisti et al., 2013). Punkka (2012) emphasizes the importance of starting as soon as possible on prototyping. This upfront prototyping will serve as experimentation to reduce the uncertainty and prevents late changes in the design that could cause major problems. Rapid prototyping is also used and stressed as a key instrument in the next generation stage-gate (Cooper, 2014). Furthermore, if the project cadence of all stakeholders can be aligned, feature driven iteration can also be used in an embedded systems setting (Punkka, 2012). Boehm and Turner (2005) suggest that important project processes need to be identified in order to appoint the right assets. In addition, functional responsibility should be assigned and problem areas identified so appointments can be made around this issue. Finally, an appropriate test suite is required. The lack of testing is one of the main constrains in working on embedded systems, because software and hardware development are not yet aligned (Kaisti et al., 2013; Ronkainen and Abrahamsson, 2003).

Because ASD and TPM are also different outside the development process, modifications have to be made on business and people management in order to create a successful development process. Regarding the business process management aspects of combining ASD and TPM, it is important to set clear appointments and guidelines with the intention to inform all involved project members what is expected from them.

Karlström and Runeson, (2006) identified these recommendations above are also applicable at combining ASD and stage-gate. The authors suggested similar practices such as: early involvement of all developers, project planning suitable for both type of methods, short and fast feedback loops, early on prototyping, and create understanding of each other’s development roles to improve the cooperation. In addition, adjustments to these critiques were made by its “creator” R. Cooper. The latest version of the stage-gate focuses more on the importance of dealing with change and uncertainty, also identified by (Leffingwell, 2011), and is called is the triple A system. The core values of this new version stand for Adaptability, Agile, and Acceleration. Furthermore, in model the included customer involvement should create products, which are better aligned with the market. This new model could be a good basis in the future for developing embedded systems, because of its iterative nature and rapid prototyping principle. This is also found important by Punkka (2012).
2.4 Conclusion literature

Even though, many recommendations are given on the co-development of new products using ASD and TPM, it is still not a common practice (Eklund and Bosch, 2012; Kaisti et al., 2013; Punkka, 2012). This indicates that there are more problems influencing this co-development than presented in the literature. Therefore, the overall research remains:

*How do the problems in a project, where ASD and TPM are used as a development method by multiple partners, emerge over time? What are the main causes and consequences of these problems, and how can they be prevented?*

3. Method

3.1 Research Design

Because in this research project, we want to investigate how the problems in a project where ASD and TPM are used together as an development method by multiple partners emerge over time a longitudinal study is recommended. Therefore, the research question can be best investigated by performing a **process study** by Van de Ven (2007). This process study creates “a **narrative describing the sequence of events that unfolded while the change occurred**” (van de Ven, 2007: p197).

Consequently, a process theory is build using the **grounded theory** methodology. A grounded theory “is a theory that is inductively derived from the study of the phenomenon it represents” (Strauss and Corbin, 1990: p23). This grounded theory approach was chosen because we do not want to prove a theory, but to build a theory using a certain amount of openness and flexibility to create the best fit within the area of research.

The process study approach consists of the following steps proposed by Poole et al. (2000). First, all the available data was observed on the change over time. Documentation such as archival reports and personnel data were used as a data source. In addition, interviews were conducted to support this data and to create triangulation (Yin, 2009). Furthermore, event sequence data was collected which was organized chronologically in a database. These events were retrieved from the data as followed. During the data gathering process, each problem that was stated in the project documentation was evaluated and if found significant, copied, labeled, and placed as an event in a database. When all the data was collected it was coded into meaningful categories. For these categories, a coding template was used (King, 2004). This template is based upon the categories used in the Minnesota studies by Van de Ven et al. (2000) and Van Oorschot et al. (2013), and was
modified accordingly to fit the current research project. Further, these categories and collected event sequence data were analyzed using several strategies presented by Langley, (1999). From these analyses, patterns were derived to construct a narrative. Finally, based upon this narrative, the research question was answered and a process theory was created. The coding template and the data analysis’s that were used are further discussed in the data analysis section below.

3.2 Research context
SoftCo is a large supplier of software products. They currently have three product categories: consumer products, business to business content, and management systems. The focus of this research project was on the business to business product category. In this category SoftCo delivers software to manufacturing companies and/or to their suppliers integration in their embedded products. This requires close cooperation between SoftCo and the involved partners. In developing software systems for embedded products, each party takes care of their part of the system. SoftCo provides the software and services while the manufacturing companies and their suppliers provide the hardware and system software. SoftCo works with agile software development (ASD), while the manufacturing companies generally work with the so-called traditional project management approach (TPM). Therefore, this setting fits well with the research question and makes it appropriate for this research project.

3.3 Case selection
In analyzing this problem, cases from SoftCo were used where they developed new embedded products together with manufacturing partners. For this research project three cases were chosen to investigate, because multiple cases make the results more robust (Yin, 2009). Each case that was chosen is unique and has its own requirements. However, they are also similar on a basic level because all projects require at least the same basic software components to be integrated into a embedded system. Therefore, these cases have enough replication to be included into the research (Yin, 2009).

Project S Case
The main case that was examined is project S. This was a cooperation between SoftCo and ManuCo1. The aim of this project was to develop a new type of multimedia system for ManuCo1’s embedded products. In this project, SoftCo was responsible for the whole system development in terms of hardware, software, services, system integration. ManuCo1 would provide feedback on requirements and deliver validation equipment. This case was chosen as a main case, because it was
the largest and best documented project available. Furthermore, it met all the guidelines recommended by Pettigrew (1990). First, this project was large, because it had a development time of more than one year. Second, the project was unique. In the development project different suppliers were involved. Third, the roundup of the project got affected by numerous problems which made it an extreme case. Fourth, information was easily accessible though intranet and internal servers. Finally, there was already experience in undertaking a project in this setting due to previous successful projects involving SoftCo and ManuCo1.

Project D Case
After the project S case was examined, a comparative case study was used to compare the findings between project S and the other cases (Yin, 2009). One of these cases was project D. This was a project which was based on a cooperation between SoftCo, SupplyCo and ManuCo2. In this project ManuCo2 tried to create a new standard build environment for their multimedia systems. ManuCo2 developed the main system, SupplyCo delivered the hardware and did the system integration. SoftCo was responsible for the software and services for this system. The project D case also met the recommendations of Pettigrew (1990). First, it was a large project, having a development time of more than one year. Second, it was an unique case because they tried to develop a new standard for their multimedia systems. Third, it was an extreme case, because the project was balancing several times on the edge of termination. Fourth, the information around this project was easily available. Lastly, many experience was available due to previous projects.

Project M Case
The third case that was used to compare with project S was project M. This project is a cooperation between SoftCo, ManuCo3 and SupplyCo where a new multimedia system was developed. In this product, the hardware and system integration were the responsibility of SupplyCo2. The software, and services were the responsibility of SoftCo. This case almost met all the case recommendations of Pettigrew (1990). First, the developmental time was longer that one year. Second, it was unique because the project was the first multimedia system developed together with ManuCo3. Third, information regarding these cases was easily accessible. Lastly, there was a lot of experience in this type of projects because of previous project with type of systems. This case was not an extreme case; the project went reasonable smooth.
3.4 Data analysis
Quantification strategy

As mentioned in the research design chapter, in order to analyze the process of stability and change over time, the events retrieved from the data were sorted chronologically into several meaningful full categories to identify different process patterns that occurred over time (Van de Ven et al., 2000). These categories in which these events were sorted, were derived from a coding template. This template was based upon the categories used in the *Minnesota Studies* by van de Ven et al. (2000) and in a later study by Van Oorschot et al. (2013) in their longitudinal research project about product development. Because our research project also focuses on product development it has a good fit with the research done by Van Oorschot et al. (2013).

Therefore, the same main categories are used as a starting point and are modified accordingly to match this research project. The main categories were as followed: *Idea* relates to changes of the scope of the project due to innovation. *People* is about the perceptions and emotions people have about the project such as quality, atmosphere, and staff. *Transaction* relates to engagements people make with other stakeholders in order to achieve a certain outcome. *Outcome* relates to the actual results of the project plans (Van Oorschot et al., 2013). Sequentially, from these main categories project specific categories are made (table 1) and were used to analyze the sequence of events over time in the narratives. This was done by visualizing how frequently the sub-categories were used over time.

<table>
<thead>
<tr>
<th>Main category</th>
<th>Sub category</th>
<th>Explanation</th>
<th>Example of coded event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea</td>
<td>internal change</td>
<td>Changes on planning and scope by project team</td>
<td>Android switch requires architecture changes</td>
</tr>
<tr>
<td></td>
<td>external change</td>
<td>Changes on planning and scope by stakeholders</td>
<td>Interpretation of requirements, still incoming spec corrections</td>
</tr>
<tr>
<td>People</td>
<td>human Resources</td>
<td>Perception of human resources problems</td>
<td>Product Quality engineer will not start by the 1st November.</td>
</tr>
<tr>
<td></td>
<td>atmosphere</td>
<td>Perception of project atmosphere problems</td>
<td>Reorganization; bad atmosphere</td>
</tr>
<tr>
<td></td>
<td>progress</td>
<td>Perception of project progress problems</td>
<td>HMI interface took more time than expected</td>
</tr>
<tr>
<td></td>
<td>quality</td>
<td>Perception of product quality problems</td>
<td>Performance is bad</td>
</tr>
<tr>
<td>Transaction</td>
<td>dependency</td>
<td>Problems around dependency on</td>
<td>Software in test cars is not up to date</td>
</tr>
<tr>
<td></td>
<td>way of working</td>
<td>Problems around way of working and communicating in the project</td>
<td>Project organization not clear</td>
</tr>
<tr>
<td>Outcome</td>
<td>delay</td>
<td>Statements on Missing deadlines</td>
<td>Release V5 is delayed by 2 weeks</td>
</tr>
</tbody>
</table>

Table 1. categories used in this research project

In this research project a distinction was made between different levels in the organization. This was between the level of project management and the developmental level and was done to observe if there was a difference in the problems between the two levels.
Perceived project status analysis

Besides the quantification analysis, another analysis was used; the perceived project status analysis. This analysis shows the perceived status of the project over time. During the weekly core team meetings, each department of the project discussed their issues and reported a stoplight status as explained during one of the interviews “when you report a green status everything is under control. When a project that has difficulties the default is often orange and when there are serious problems the status is red.” (Project S interview 1, 3:00). This observed status is used to create an overview of the perceived project status. This was done by first quantifying the status. A green status received 0 points, an orange status received 1 point and the red received 3 point. Consequently, for each week the average score of all departments was calculated and represented in a graph over time.

Personnel records

From the personnel records an overview was created which represented the amount of personnel in FTE which were allocated over time to the projects. Furthermore, also the data regarding needed personnel was added to this overview to analyze the differences between them. In addition, also a distinction between software developers and testers was made.

Interviews

In order to find additional support for the process study, a total of 11 interviews were conducted with SoftCo stakeholders, spread over all cases. These interviews were conducted semi structured, asking the interviewees their opinion on the project in general, specific events, and problems occurring. In Appendix I, an example of a general interview questionnaire can be found. The results from the interviews, helped building the narratives. Furthermore, the interviews were all recorded and transcribed, thus they could be used as support for the process study by quotes. In addition, a stakeholder in project S from ManuCo1 side was interviewed. This allowed to give alternating views on the same events, improving the power of the narrative as recommended by Langley (1999).
4. Results
As mentioned in the previous section, in order to analyze the research question, a narrative was created for each case. In this chapter the first part of the research question was answered, namely; How do the problems in a project, where ASD and TPM are used as a development method by multiple partners, emerge over time? Meanwhile, these problems were also analyzed regarding the interaction between ASD and TPM.

4.1 Project S Narrative
Project S was a result of the launch of a new and improved multimedia system with ManuCo1. This product was a multimedia functional display, a system which included many features. This product was developed in cooperation between SoftCo which used ASD for the software components and ManuCo1 which used TPM to deliver the subcomponents to SoftCo. Furthermore, within SoftCo there was a separate department (SFD) which developed the software features for the SoftCo project team (fig 6.). In the following chapters the problems that occurred during this project will be described as a narrative.

4.1.1 Phase 1: False start
May 2010 – Feb 2011

Start
In May 2010, the project started, using a new set-up which was decided after an technology board meeting early 2010. The set-up of the project was that the SoftCo Feature Department (SFD), would be responsible for the development of main software and the associated features. This would create an of the shelve software product which could be easily used by SoftCo. Consequently, they would integrate this software together ManuCo1, into a complete multimedia system (SoftCo interview 3. 1:00).

Resources
Right from the start the project ran into resource problems, because of the interference of other projects which had difficulties and required additional attention. For SoftCo this was project B which was also done together with ManuCo1. For SFD this was another project. Therefore, resources were not fully available to start with project S and the project was already behind schedule right from the beginning. In addition, the resources that were dedicated to the project were not fully available and the project started to delay even further. This can also be seen in fig 7 which highlights the perceived resources demands.
Figure 7. Perceived resource demand. Project S phase 1

This graph shows that immediately from the start a high perceived demand of resources is needed for the project, both in the general as in the testing area. During this period, statements such as “Lack of resources” (CT meeting wk1028) or “Some impact to project milestones due to unreliable estimates and resource availability” (CT meeting wk1030) were found frequently at the core team meetings. This was also acknowledged during the interviews with SoftCo’s managers in the project with quotes for example, “Actually the whole project started too late, or... [To put it differently] started too early, against better judgment we started the project, even though everyone knew the resources were not available.” (Project S interview 1. 4:00) and “When you look at how resources are managed at SoftCo, they are allocated when you pass a milestone. So before we could start with [project S] there were the issues around [the project B] that needed to be solved. This project delayed; therefore, the resources couldn’t be released.” (Project S interview 2. 8:00)

All the same, this resource shortage cannot be found in the resource allocation overview of fig 8. This graph indicates that there were no problems regarding needed resources. One explanation for this is that the resource allocation overview is only from SoftCo, excluding SFD. SFD did not record their allocated resources, and consequently it can be that most of the resource shortage statements were from SFD. Another explanation can be that these resources were theoretically working on the project S, but in reality fixing problems for the project B. However, this was denied by one of the project S managers “Even though the project B project interfered with project S, we did not allocate many resources [from project S to B]” (project S interview 3. 16:40)
Communication and planning
As already mentioned above, the resource shortage caused the project to delay right from the start. In addition, because SFD had a new role, they were still in search of a good project organization. This resource shortage and lack of project organization at SFD had two consequences. The first consequence was that due to this, activities to decrease the amount of uncertainty were not done thoroughly. This caused the project plan to stay too high level. These problems around the planning can also be found by the idea internal line in fig 9. This line shows statements during the core team meetings around the amount of internal planning changes and uncertainty of the project. “Android integration plan needs to be more detailed” (CT meeting wk1043), “Backlog is unclear, unknown user stories, what is expected?” (Team retrospective sprint 28.1 team A and O)

The second consequence of these problems was that the project plans were not communicated sufficiently, leaving the development teams in the dark. This is displayed by the way of working line in fig 9. which represents issues such as “The project schedule needs to be communicated” (CT meeting wk1026), “Very bad communication in the team” (CT meeting wk1103), and “Project infrastructure not in place for Software [team]” (CT meeting wk1043), concerning not communicating information correctly and following the right procedures.
Android change
Two months after the start of the project management decided, due to technological driven circumstances, to change the development platform to Android. The software core component was already transferred to the development in Android; therefore, it would make future support much easier. In addition, Android had significant technological advantages over the old system such as app building. The decision to choose for Android was therefore quite logical as discussed during interviews with ManuCo1 and SoftCo; “The change to Android brought many advantages on app development and technology. The core software component was about to move Android. Therefore, if the old software component would be used only for ManuCo1 two different paths needed to be developed and maintained. This was not an option for SoftCo. Therefore, we convinced ManuCo1 to join us in this” (Project S interview 23:00) and “I’m sure [the change to Android] was a good way of thinking, it gave many advantage, [however] the timing was a little too late. This made it really risky for SoftCo. And also for ManuCo1.” (ManuCo1 interview 44:00)

Project progress
On top of this, the project turned out to be much larger and more complex than expected. This was identified during the interviews, “[The multimedia product] included many [features]. That was actually way too many and way to complex.” (Project S interview 1. 35:00), delaying the progress even further. This can be seen at the line in fig 10 which represents the perceived progress problems occurring each month. During phase 1 these problems increased every month. Indicating that many tasks took much more time than original estimated. The feature adherence graph represented in fig 11, also displayed the progress problems. The graph represents the amount of features which are finished and integrated into the software. At the end of phase 1 there was a huge gap between the actual features adherence from SFD and the by October 2010 planned adherence. This was also discussed by a ManuCo1 project manager, “The first project S release (1.0) had nothing inside, but we know why that was, because we had to launch the (project B system).” (ManuCo1 interview 45:00).
The complexity of the system was especially noticeable in the system architecture, also mentioned during the interview, “The product was actually way too complex, where there was not enough realization of the size of the project. It was just set up too complex. The system architecture was very complex.” (Project S interview 1. 34:00). Immediate from the start, numerous open issues showed up in the system architecture, as can be seen in fig 12 at the red bug line. However, many of them where difficult to fix because they were dependent on information from other teams such as hardware and software, which were also delaying due to the lack of resources. In addition, many issues required workshops with ManuCo1 that had to be planned as presented in a core team meeting “System architecture workshops [ManuCo1] requested, not planned” (CT meeting wk1043).

Also the switch to Android caused the progress to be hampered further, as identified by a ManuCo1 project manager, “I remember the [change to Android] presentation from SoftCo. Saying that, the next deliveries will be a little delayed, but after that we will regain some time and in june 2011 everything will be done. [However], this was not completely the case in the end” (ManuCo1 interview 44:00). This change to Android had especially a huge impact on SFD, because they had to switch their software and features development to Android. However, the project team did not felt much support from the organization in their change to Android as identified by one of the project managers, “About Android, nobody knew anything about it. It was just introduced and we just started
developing, while no trainings, gurus, or capabilities were bought. Everyone started to explore this by themselves, very good, but it took a lot extra of time” (Project S interview 1. 35:00). Consequently, Android knowledge was not spread enough throughout the development teams, increasing the complexity of the whole project. This was also identified during the core team meetings, “Lack of Android knowledge”, (CT meeting wk1037) and during the interview with a ManuCo1 project manager, “When we switched to Android, we kept the same backlog, but I’m pretty sure that is was a nightmare for SoftCo to integrate certain features on the Android platform, because of the lack of knowledge on that. Especially in the beginning, they were not specialist yet on developing in Android. That was quite a big risk to change to Android by SoftCo without enough knowledge (ManuCo interview 1:09:00).

Project pressure
These issues around resources shortage lasted for around 6 month. Therefore, it took six month before the project was fully geared up with all the dedicated resources. Meanwhile the project status getting worse month by month as can be seen in an analysis of the perceived project status shown in fig 13. When is looked at the output of this analysis, it shows a peak at November 2010 indicating that many components in the project had difficulties. After November 2010 all resources returned to project S and the progress problems decreased marginal, see fig 10. In addition, the project status increased slightly as can be seen at fig 13. However, this was not enough to get the project on track more actions were required from SoftCo’s management team. Until February 2011 no actions were taken and from March 2011 a gear-up action was finally announced and led to a new project phase; the gear-up phase.

Figure 13 perceived project pressure. Project S phase 1

Main problems phase 1:
- Resource availability
- Android change
- Bad project status
- Underestimation of complexity
- Slow progress

None of the problems were caused by ASD and TPM interaction in phase 1
4.1.2 Phase 2: Gear-up phase
Mar 2011 – Aug 2011

The delay of the project led to harsh discussions with the ManuCo1 management, because according to the original schedule the complete system had to be delivered within a few months. However, SoftCo replied this was not possible due to the size and complexity of the system. (Project S interview 3. 6:00) Therefore, to get the project back on track a gear-up action was announced in March 2011.

This resulted in first identifying all the work that needed be done. Secondly, additional resources were requested to execute all the additional work. Thirdly, a scope reduction was achieved to decrease the amount of work. Lastly, priority was given on progress in terms of developing features instead of bug-fixing activities. “Feature development till wk1130, so no bug-fixing” (CT meeting wk1112)

**Resources**

As already mentioned above, an action from the gear-up action was to first identify all the required work that needed to be done. This analysis identified a considerable amount of work so that extra resources were requested. However, the additional resources were not immediately available upon request and involved a delay of several months. It took around six months before these resources were fully available since the gear-up action was announced. This can be seen in resource allocation graph of fig 14 and was further mentioned by one of the interview with a project manager “[it took quite some time before] people were added to the project. However, there were also people taken away from the project and put on other projects” (Project S interview 1. 4:30)

![Figure 14. Resource allocation. Project S phase 2](image)

**Project pressure**

After the gear-up action the project got more on track as can be seen in the perceived project status graph at fig 15 in March 2011. However, because the requested resources were not immediately available the workload increased in order to stay committed to the new schedule. Consequently,
increasing the project pressure again. The high workload was also mentioned during several core team meetings by statements such as “high workload” (CT meeting wk1106-1113), and “Hardware schedule very tight” (CT meeting wk1130-1131).

Project progress
The complexity problem from the “false start phase” also had an impact on the second phase. Therefore, the project had difficulties keeping up with the schedules, and many tasks took much more time than expected (see fig 16). Subsequently, the project buffers decreased and caused that the main priority of the project was changed to developing features and neglecting bug fixing activities in order to commit to the project plan. Furthermore, the integration of features by SFD also fell behind schedule because of the complexity of the software. “Integration progress of features falls behind schedule” (CT meeting wk1118) and “Integrators cannot keep up with all activities” (CT meeting wk1120). This can also be seen in fig 17 which represents the feature adherence level. It shows the actual feature adherence lack behind the schedule, even though this was already adjusted several times. In addition, no feature commitment could be given by SFD on deliverables; “[SFD] could not deliver a feature breakdown, and a list of features, because they were not well organized and delayed month by month” (project S interview 3. 7:00) resulting that other departments were waiting on their deliveries. “Requests for SW and intermediate releases are in contradiction with expectation that SW is validated and integrated. How can expectation and reality become aligned?” (CT meeting wk1124) This caused that also problems showed up at different levels of the project such as hardware, and test problems, due to bad and unpredictable software deliveries.

Furthermore, it decreased the progress and the perceived project status got worse again as can be seen in fig 15.
Validation plan
The decision to focus on feature development only, was mainly caused by the pressure from ManuCo1. They requested that all features had to be delivered in order to validate them. Normally, for this complex procedure a validation plan is used. In this plan the integration and validation of all components are described in detail and in which order these components need to be built to validate the complete system. Furthermore, it prescribes which components have priority and which not. Having this plan, structure can be added to the complex integrating process. However, ManuCo1 did not have this plan as a SoftCo manager mentions during an interview. “In March 2011, together with ManuCo1 we organized a workshop to look at the validation plan and to decide which features are necessary to deliver first to validate the system. This workshop was a total failure because [ManuCo1] only focused on our plan and did not have a plan for their own at all. Therefore, they wanted to have all our features as soon as possible.” (Project S interview 3. 6:00). The lack of integration plan was also acknowledged during an interview with ManuCo1. The manager discussed that they completely underestimated the complexity of the systems. In addition, because there were several products which contained the SoftCo software, additional configurations were added. This complicated the integration plan even further. (ManuCo1 interview 13:00)
Quality
One of the actions management took to increase the progress was to focus on feature development. This meant that there would be little focus on the testing and validation of the software. These activities would be done after all features were finished. However, because there were already many problems with the software the feature development hampered, leaving many features incomplete.

“A part of the problem was that many features were not finished or incomplete. They didn’t work or only partly. Therefore we decided to finish these features first, resulting in full-feature full bug.” (Project S interview 1. 6:00). This lead to the decision to stop testing completely because the understaffed testing team, “Severe lack of resources in software test” (CT meeting wk1118), could not process all the issues anymore as each issue solved gave three new issues. “[At a certain point] I’ve said to the testing team, stop with testing because we know there are 1200 bugs. We first need to solve these back to 600 ... that will takes us four weeks, after that you are allowed to test again” (Project S interview 1. 5:00) This decision gave the testing team some air to fix some core problems; however, by not testing anymore, huge amounts of errors were incorporated into the software because they were not tested anymore and based on a basis which was already of low quality. These signs of many fault in the software were already early noticeable, see fig 18 for the amount of bugs and by statements in the core team meetings such as, “Software behaves strange in [product]” (CT meeting wk1124), “Performance is unacceptable” (CT meeting wk1125), However, the project still kept going on the same plan of developing as much features as possible, because there was no way back since they could not test properly before almost all features were finished.

Requirements
One issue during the “gear-up” phase was that not all requirements were fixed, also identified during the ManuCo1 meeting, “[The specification were not clear and [to clarify this we had] twice a week, sometimes three days a week, exchange on this. [Often] SoftCo was coming to have a workshop with the specialist from ManuCo1 to discuss the specification”. (ManuCo1 interview 1:06:00). A specific part were no decision was made on what the requirements were and what the involvement for SoftCo would be was around the software service, as can be seen on the blue line which indicates the scope and requirement changes by external partners in fig 19. This was also multiple times reported during the core team meeting with statements such as “Interpretation of requirements [problems]”
(CT meeting wk1129) and “Not all requirements fixed for services yet (discussions ManuCo1)” (CT meeting wk1115) this lead to extra work and uncertainty on the already pressured schedule.

![Figure 19 Scope and planning changes phase 2](image)

<table>
<thead>
<tr>
<th>Main problems phase 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Hiring delay</td>
</tr>
<tr>
<td>• High workload</td>
</tr>
<tr>
<td>• Feature priority</td>
</tr>
<tr>
<td>• Unreliable feature delivery SFD</td>
</tr>
<tr>
<td>• System instability</td>
</tr>
<tr>
<td>• Testing problems</td>
</tr>
<tr>
<td>• No decision on software services</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problems caused by ASD and TPM interaction phase 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High workload</td>
</tr>
<tr>
<td>• Feature priority</td>
</tr>
<tr>
<td>• Validation plan</td>
</tr>
</tbody>
</table>

4.1.3 Phase 3: Bug-fixing phase  
Sept 2011 – Dec 2011

After the features started to finish, the testing team got back on track and started testing again. However, because no large tests were executed for a while, the start of extensive testing resulted into a tremendous amount of inflow of issues as can be seen in fig 20. This was much more as the original estimated amount of issues and asked for a reaction by the management team who changed the focus of the project from feature development to bug-fixing. The priority switch initiated the start of phase three; Bug fixing phase. However, even though the priority was set to bug fixing, still feature commitment from SoftCo was expected by ManuCo1 as reported in one of the core team meetings “Feature commitment for [delivery] 1.0 is still expected” (CT meeting wk1145) and “Bringing as much functionality to the X10 product as feasible before [deadline]” (CT meeting wk1134)
Resources

Since the start of the project all requested resources were on the project (August 2011), see fig 22. However, this brought up a new problem. Many new project members had no or little experience with the project as indicated during the core team meetings, “Many new resources for project S, but they have little or no experience with SoftCo and/or the project.” (CT meeting wk1137). This caused the project teams to spend much effort to get these members familiar with the project. This issue was also discussed during the development team meetings, “Many changes in teams → slowdown” (Team T retrospectives sprint 33) and caused many disturbances in the way of working as can be seen in fig 21 which represents these problems at the development teams of the project “I don’t know how the project is doing; will we make it in January?” (Team T retrospective sprint 36).

Figure 20. Issue inflow. Project S phase 3

Figure 21. Way of working problems at development teams. Project S phase 3

Figure 22. Resource allocation overview. Project S phase 3
Quality
The amount of issues kept increasing week by week. Therefore, special priority was given to bug fixing, because the management underestimated this aspect, “More issues than expected” (CT meeting wk1141). Consequently, the issue prediction curve was adjusted several times, setting it higher and higher as can be seen in fig 23 going from 4500 from the start to 10,000 at the end of phase 3 “Issue prediction again modified” (CT meeting wk1150). These quality problems were noticeable on both team level and management level as can be seen in fig 24 and fig 25. As a result, many of these bugs caused for performance and stability issues, “The performance is too low” (CT meeting wk1146-50). The bugs endangered the project and if not solved on time leaving the possibility of an unsellable product.

![Issue prediction curve by management. Project S phase 3](image)

**Figure 23. Issue prediction curve by management. Project S phase 3**

![Quality problems at the development teams](image)

**Figure 24. Quality problem at development team. Project S phase 3**

![Perceived quality problems](image)

**Figure 25 perceived quality problems. Project S phase 3**

Validation equipment
In developing representative tests to test the software on bugs, the SoftCo testing team was dependent on validation equipment delivered by ManuCo1. This equipment allowed the SoftCo team to test the software on the latest available sub components. The project S was a multifunctional device with several sub-components. Therefore, multiple partners were involved delivering these different sub-components. SoftCo had the role as system integrator, thus having the latest updates
was very important for the testing of the system. Nevertheless, these validation materials were often poorly available or not managed properly by ManuCo1. This was also identified during one of the interviews with a SoftCo project manager, “There was no person responsible at ManuCo1 for organizing and bringing all the validation equipment up to date” (Project S interview 1. 16:00). In addition, ManuCo1 recognized the problems around these issues. During the interview they state that they realized too late that the validation of the system takes such huge amount of work, and consequently have problems providing feedback on time to SoftCo. (ManuCo1 interview 16:00). Therefore, potential bugs that were hidden deep into the software could not be identified. This made it more difficult to solve these bugs because parallel on this bug-fixing and testing still features were developed upon this faulty software as a result of the feature commitment. The graph in Fig 26 shows that during phase three there were large amount of dependency issues occurring such as “Test material not available” (CT meeting wk1138) and “Non-availability of up to date test equipment” (CT meeting wk1144)

![Figure 26 transaction and dependency problems phase 3](image)

**Requirements**

The progress during the bug fixing activities was low, due to the constant inflow of new bugs. On top of that, ManuCo1 computed many requirement changes, and therefore it became even more difficult to fix these bugs. Software had to be changed in order to meet the new requirements; consequently, the tests had to be adapted, slowing the progress even more down. In fig 27 can be seen that during phase 3, many scope and planning changes occurred, giving a sense of constant running one step behind. This changing of requirements was caused by two problems.

![Figure 27 External scope and planning changes. Project S phase 3](image)
The first problem was the absence of a validation plan by ManuCo1 as already described in the previous phase. Even though the importance of it was already stressed out early on by SoftCo, ManuCo1 still did not have one. Therefore, no feature priority could be given. The second problem around the requirement changes was the abstraction layer of the product. ManuCo1 has an universal abstraction layer in their products, hence when a computer is connected to this abstraction layer it can communicate with all the sensors and subsystems in the different models that have this abstraction layer. The problem was actually that there were some small bugs in this system which expressed them into the SoftCo software. Consequently, SoftCo was blamed by ManuCo1 that their software was not robust enough. “ManuCo1 did not want to talk about the problems [around the abstraction layer] and blamed our software of not being robust and uniform enough.” (Project S interview 1. 17:00). This caused many extra work for SoftCo to change their software in order to work around these problems in the abstraction layer. “[Because the abstraction layer was not uniform] workarounds had to be built for each different type of product” (Project S interview 1 17:00).

Reorganization
In the end of 2011 SoftCo announced a reorganization. This was caused due to a change of strategy. As a result, the different departments in the organization would be divided into several business unit with each their own task. There was also decided that hardware would not be done, and the main focus would be to only develop software components. Due to the reorganization, SFD was taken away from the project and received a new role in the organization. However, the retreat of SFD from the project resulted in a huge increase of extra work for the remaining team and caused many disturbances in project S “SFD developed many features for us. However, after [the reorganization] they pulled the plug out of SFD, and we were told to fix everything ourselves” (Project S interview 2. 10:00) In addition, due to the reorganization some project members had to leave. This also did not contributed to a positive team atmosphere. Especially the uncertainty around this reorganization, caused a bad atmosphere into the development teams. “Fuzzy reorganization – first people have to leave, the next day can stay” (Team B retrospective sprint 36) and “Reorganization; bad atmosphere” (Team T retrospective sprint 36).

512 MB
Another problem influencing the project was around the 512MB memory issue. In the original offer without Android, SoftCo offered a system which ran on 256 MB. However, during the project and due to the change of Android, the realization came that 256MB was probably not enough. However, SoftCo did not want to pay this upgrade, and neither did ManuCo1. Therefore, SoftCo put in many resources in improving the performance of the system to make it work on the 256MB hardware. In
the end, the amount of memory was not sufficient enough because additional apps had to be added to the system and consequently, the memory was upgraded to 512MB. If this had been decided earlier, it would have saved a lot of time. (Project S interview 1 2 & 3 and ManuCo1 interview)

<table>
<thead>
<tr>
<th>Main problems phase 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High amount of issues</td>
</tr>
<tr>
<td>• Inexperienced new project members</td>
</tr>
<tr>
<td>• Bad availability of up-to date validation equipment</td>
</tr>
<tr>
<td>• Integration plan</td>
</tr>
<tr>
<td>• Reorganization</td>
</tr>
<tr>
<td>• Withdraw of Client Technology group</td>
</tr>
<tr>
<td>• 512 MB issue</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problems caused by ASD and TPM interaction phase 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Bad availability of up-to date validation equipment</td>
</tr>
<tr>
<td>• Integration plan</td>
</tr>
<tr>
<td>• High amount of issues</td>
</tr>
</tbody>
</table>

4.1.4 Phase 4: Stabilization phase
Jan 2012 – Jun 2012

During the bug-fixing phase many bugs were solved, however, the stability and performance of the system was still very bad. Therefore the management team decided to give full priority on improving the stabilization and performance of the system in order to meet the deadline of week 24 of 2012. To assure this, a special stabilization team was create which would monitor the project. Furthermore, overtime was requested from the development teams to fix all the bugs.

Resources
After the reorganization, SFD moved out of the project. However, a large amount of work that was originally for the account of SFD shifted to the remaining development teams. Though, they were not prepared for this large intake of work. This increase of work can be seen in the resources overview of fig 28 by the gap between needed and acquired software engineers from January 2012. In addition, in the same period a few project members left SoftCo which created an even larger gap between needed and available resources. “I received [some] additional resources; however, this was not sufficient to [solve] everything” (Project S interview 2. 8:00). Almost no additional resources were added to the project because of several reasons. First, as seen at an earlier stage of the project, new hires are not immediate available upon request, and often involve a hiring delay. In addition, these new resources would disrupt the current workflow, because they are new to the project and needed support from other team members. Furthermore, after arriving these new hires were not productive immediate from the start. It takes a minimum of one month before they become productive. Thus, because the project was nearing its end and due to the hiring delay, new hires would be fully productive at the end of the project. This would only increase the costs without extra productivity. Therefore, no extra team members were hired and instead of adding manpower, the project
managers introduced at the beginning of 2012 a gear-up action of working structurally overtime in order to achieve the deadline.

![Figure 28. resource allocation. Project S phase 4](image)

**Project pressure**

While the deadline was nearing quickly and due to the withdraw of the SFD resources, a big gap was created as already mentioned above. In order the tackle this gap an overtime action was requested from the team to deal with the huge amount of work left. This would contribute to achieve the deadline at June 2012. Consequently, this persistent overtime caused the pressure on the project team to increase and therefore, the project atmosphere decreased. As can be seen in fig 29, which represents the amount of negative atmosphere events in the development teams such as “People are leaving” (team T retrospective sprint 38) and “Overtime” (Team T retrospective sprint 38). The increased pressure also caused priority setting problems in the project as ManuCo1 got more and more involved nearing the deadline. However, ManuCo1 had different priorities on certain tasks than SoftCo did, imposing certain problems to be fixed immediately “Priority list very volatile” (CT meeting wk1218). This resulted in problems around the way of working as can be seen in fig 30. These problems did not help the project achieving the deadline, because these changing priorities caused extra work and shifted the focus from solving the underlying causes of software instability to solving the issues as a result of software instability.

The increasing pressure was also identified by ManuCo1. During the stabilization phase, which they called Kaizen phase, it started by delivering in three weekly updates. However, when the pressure increased these updates became weekly, and moreover “extra Kaizen activities” were requested. This meant that on top of the weekly updates, extra daily updates had to be checked. Nevertheless, this disrupted the process in such way it gave a huge amount of additional work, such as extra documentation, meetings, leaving less time to focus on the real Kaizen process. “Receiving new software every day to validate is a nightmare for the team” (ManuCo1 interview 1:25:00) In addition, because project R, the successor of project S, was started in the end of 2011, they had to work in
parallel, validating new project R features while still fixing the old project S system and integrating these fixes regularly. This created a really difficult environment to work in.

![Atmosphere problems at the development teams](image)

**Figure 29.** Atmosphere issues at development teams. Project S phase 4

![Way of working problems](image)

**Figure 30.** Perceived way of working problems. Project S phase 4

**Quality**

While many issues already had been solved, the bug inflow still would not stabilize. Especially during the stabilization and performance enhancement activities, regression often occurred and many new issues showed up which decreased the progress. “Performance improvement is not visible in 11.0 due to regression” (CT meeting wk1214). Fig. 31 shows the amount of issue and the estimated issue curve. As can be seen the estimated curve was adjusted several times indicating still severe problems existed into the system. This felt very frustrating for the development team, and gave them a feeling of one step forward, two steps back. “We were not fast enough in solving issues. One of the reasons: regression” (Team T retrospective sprint 39).
Validation equipment
During the stabilization phase the problems around the availability of validation equipment remained, because ManuCo1 could not handle the amount of work (ManuCo1 interview 44:00). Often test-equipment were not available upon the requested date, or were equipped with the wrong validation software. Statements during the core team meetings such as “radios have wrong software for test” (CT meeting wk1206) and “Non availability of up-to-date test equipment” (CT meeting wk1214) were very common during and caused for delays in the testing team. In addition, because there was still no integration plan, ManuCo1 had problems prioritizing all activities as already described above. This caused for disturbance in the whole process.

Progress
In the end, the deadline in June 2012 was not achieved. SoftCo received a no-go, because the stability and the performance of the product was not good enough. This was a huge disappointment for all involved project teams due to the huge amount of effort they have put into the project as stated during one of the interviews “In week 24 we received a no go for the software part, which was a complete crisis for us.” (Project S interview 1. 30:00). This missed deadline caused SoftCo to miss a lot of revenue because the product that were supposed to have the SoftCo system now had a different system. “The first products [which should have had our system] were launched using a system” (Project S interview 1. 45:00). They received a second chance by receiving a new deadline at week 48.
4.1.5 Phase 5: All or nothing
Jul 2012 – Dec 2012

The last phase of the project called “all or nothing” is characterized by achieving the renewed deadline of 1248 that had to be achieved at all costs.

**Project pressure**
However, this all or nothing mentality created extreme workloads, as stated in several core team meetings “Extreme workload” (CT meeting wk1239), and the main focus was on solving issues and increasing the stability. In addition, many unexpected work was brought in because the system had to be compatible with several products, because as identified by the ManuCo manager “The team and the management did not know how much time it cost to finish the software for one product” (ManuCo1 interview 25:00). This resulted that schedules got tighter and tighter because the underestimation of the required work to make the system operational for each specific product.

During that phase the project atmosphere and way of working in the project to decreased enormous. This can be seen in fig 31 which shows an increase in way of working and atmosphere problems.

During the team retrospective statements such as “End is not in sight” (Team T retrospective sprint 42) and “ManuCo1 abuses power to force weekend overtime:” (Team T retrospective sprint 44), were not uncommon. This negativity was also mentioned during the interview by SoftCo project managers “We had to continue till the end ... [and] the poison cup had to be finished and we tasted every drop” (Project S interview 1. 40:00) and by ManuCo1 project manager “We did more than one year Kaizen mode, then it is really difficult to keep the team motivated and [ensure] that the team does not make extra errors.” (ManuCo1 interview 25:00)
Quality
Because there were still many architectural problems, the system had still some core issues. Nevertheless, SoftCo did not received time from ManuCo1 to solve these because the final deadline had to be achieved, leaving these problems for the successor project to be fixed. “Architectural defects cannot be fixed, no time allowed by customer” (CT meeting wk1247). In week 48 the deadline was achieved and week 52 the part submission warranty was signed ensuring “successful” ending of the project. “we [delivered the software to ManuCo1 in week 48 and] received in week 52 the part submission warranty for the first version” (Project S interview 1. 30:00).

End product
When reading through this narrative, the project did not went as it should be and blame can be put on all involved partners. However, the end result was actually an award winning product, ranking first on the SDB usability ranking of 2013. “[looking back] I think the result is very good and I’m glad we achieved high rankings with this product, even though we spend [really a lot of] money” (Project S interview 1. 37:00). In addition, the good result can be attributed to the team spirit engineering level. During all the interviews was mentioned that even though the project went really bad, the engineering teams kept spirits high and fought to finish the project in a good way.

<table>
<thead>
<tr>
<th>Main problems phase 5:</th>
<th>Problems caused by ASD and TPM interaction phase 5:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High workload</td>
<td>• High workload</td>
</tr>
<tr>
<td>• High project pressure</td>
<td>• High project pressure</td>
</tr>
<tr>
<td>• System stability</td>
<td>• System stability</td>
</tr>
</tbody>
</table>

4.2 Project D narrative
Project D, which started in September 2012, is based on a cooperation between SoftCo, SupplyCo and ManuCo2. In this project, ManuCo2 tried to create a new industry standard for multimedia systems. In this project ManuCo2 developed the main system, SupplyCo developed the hardware, the platform software, the system integration. SoftCo was responsible for the software and services for this system. In fig 32 an organization chart of the project is represented.

![Figure 32. Project D organization](image-url)
Problems:
The problems with project D were especially around shifting deadlines. This already started immediate from the start of the project. Because of the involvement of a third supplier, named SupplyCo, extra dependencies were created and positioned SoftCo at the end of the “development chain”. This meant that SoftCo was dependent on other suppliers in order to continue their development process. Consequently, if SupplyCo1 was late on a deadline, SoftCo needed to wait and had less time to develop their product.

Delays
As mentioned above immediate after the start of the project, SupplyCo had problems with delivering an important subcomponent for SoftCo. This was the HMI display and application framework, which started delaying week by week. In the end it took around six month before a workable version was delivered and caused many effort to be wasted, because the project team was waiting on the components. In addition, the quality of the delivered components was often that bad it required SoftCo to fix these issues before they could continue with their own progress, because it would be faster than sending it back (Project D interview 3. 38:00). As can be seen in the fig 33 below, from the start of the project there were many dependencies problems an resulted in to delays of the final project. These problems almost continued until the end of the project.

![Figure 33. delay and dependency problems](image)

This shifting of the deadlines, caused a snowball effect, delaying everything, week by week. In fig 34 an overview of the shifting delays is given. In the end the start of production also had to be delayed by 6 months in order to meet the set deadlines.

![Figure 34. Project D delays in project releases](image)
Dependency
According to the project team, the main concern on the delay of the different components was around transparency problems, uncertainty and fake promises that were done by SupplyCo. The project team was often not or little informed by the progress of their supplier. Therefore they could not plan accordingly. Frequently, subcomponents were promised on a certain date, but often they were delayed without notion on forehand or delivered with many issues which made it an unworkable version. Desperate to make progress, the team then tried to fix these issues for SupplyCo and therefore wasting many resources and increasing developing costs.

Atmosphere
Also the atmosphere on management level between SupplyCo, ManuCo2 and SoftCo was not good. All members were pointing fingers to each other who was to blame when deadlines were not met. This wasted a lot of effort that could have been spend more wisely into the project. “At a certain point the cooperation between SoftCo, SupplyCo and ManuCo2 was very bad. Constantly complaining about each other. I did not understand that, we are in this together. So, if SupplyCo is delayed, we should not have put our energy on blaming them, but simply align the project to the new situation and make the best of it.” (Project D interview 1. 8:00)

SoftCo Feature department
Furthermore, as discussed during one of the interviews, SFD, played a role in the delay of the progress. In the current structure of the company, the core software was now developed by SFD. This component has to be delivered to SoftCo, so they can integrate this into a product for their manufacturing customers. During the tender offerings, SoftCo sells their software based on this core system plus additional/customized features to the clients. Some features are generic features and are already on the roadmap and therefore they are the responsibility of SFD. The customized features are for the account of SoftCo. The problem around selling features that are on the roadmap of SFD is, because of the agile way of working that is used by SFD it is difficult to predict when these features will be ready. In addition, because SFD wants to respond to the requests from the market, priorities can change. This led to the problem that SFD had no time/priority left to develop the features that were sold for project D, creating even more pressure on the already behind schedule development teams. (Project D interview 1. 12:00)

Recovery plan
After one year into the project (September 2013), and shortly after the decision to delay of the start of production for the product by 6 months, the project team realized that even with the extended
time, the current scope was too large to finish before the deadline. This was not only a consequence of the SupplyCo delays. There was too many work that needed to be done, and on top of that no clear overview of the project progress was available as identified by one of the project D managers. “We were behind schedule, and everything was not on track. We did not really know what was finished and what not.” (Project D interview 1. 2:00). The project was going in the wrong direction and the termination of the project was a real threat. Therefore, SoftCo decided to take action and step up their game. De-scoping activities were proposed by SoftCo to reduce the amount of features that need to be delivered, and at the same time the start of a recovery plan was made by SoftCo in order to catch up their work and improve the velocity of the development teams. “The recovery plan was decided to catch up with all our work… You can say, we were dependent on SupplyCo, but that is too easy. We simply did not all the work that could have been done” (Project D interview 1. 22:00). This recovery plan is can also be seen in the resource allocation chart by the increase in needed and allocated resources (fig 35).

![Figure 35. Resource allocation. Project D.](image)

**Feature priority**

Even though the recovery plan was successful, the project maintained under high pressure. The V5 delivery (March 2014) was very important for ManuCo2. If this deadline was not achieved, the project could still be cancelled after almost two years of development. This would left all involved partners with a huge loss. Therefore, priority was given on feature development instead of quality, because on the progress of the software for V5 would be evaluated on the nr of features/requirements developed. “It is more efficient to fix all the bugs first, however, that is not the model how [ManuCo2] works. They want to see the features, and that’s how we get evaluated. Afterwards we receive time to fix these bugs” (Project D interview 2. 20:00)
Quality
After the V5 release, eventually even a V5.1 was released to deliver all the features. After that the project continued, nonetheless, it left the project with a new problem. Because of the focus on feature development, the quality of the software was decreasing and led to an increase in the bug inflow and cause system instability.

Future risk
Since the project is still under development, this is a great risk for the future. Now that the feature development is finished, the time of stabilization and performance improvement has started. Nevertheless, even though SupplyCo has a integration plan, it is still not on the desired level as SoftCo would like it to be and, therefore during the validating of the system that is planned for Q4 of 2014, many issues can show up.

4.3 Project M narrative
The final case that is used is project M, which started in September 2011. This project was a cooperation among SoftCo, ManuCo3, and SupplyCo where a multimedia device was developed. In this multimedia system, the hardware and system integration was the responsibility of SupplyCo. The software, services by SoftCo. This setup was similar to project D as represented in fig 36.

Hardware configuration
The problems around the development project M started in the design of the project. ManuCo3 wanted to use a very outdated hardware configuration as a basis for their multimedia system. Even though this was absolutely not recommended by SoftCo. Nevertheless, in the end SoftCo agreed to build their software on this
configuration. To cope with this outdated hardware configuration, SoftCo decided to use their old software core for this project, even though SDF, developer of this core software, recently transferred to the new and much improved version based on Android.

Quality
The tweaking of the old core software to make it applicable for the outdated hardware was a very complex progress. Furthermore, changes in the core software are not recommended, because of instability risks. However, due to the required performance improvements this had to be done. Consequently, many instability issues showed up, especially nearing the end see fig 37. In addition, little knowledge was available in the project team to solve these issues. Therefore, it took even more effort to get the system running.

![Quality problems](image)

**Figure 37. Quality problems. Project M.**

Atmosphere
Another problem from the start was the cooperation issues between SoftCo and SupplyCo. During the tender offerings, SupplyCo offered their solution to ManuCo3 using a different software supplier. This was their preferred supplier, which was not SoftCo. However, because of the good relation SoftCo had with ManuCo3, SoftCo won the bid and they were forced to work together in developing this product with SupplyCo. However, this was of discontent of SupplyCo and formed a bad basis to start the project with. “Beside the hardware issue, to make things even more difficult, SupplyCo actually did not want to work with us” (Project M interview 1. 13:00)

Dependency and progress
SoftCo was dependent on SupplyCo for the delivery of subcomponent to develop their software. When SupplyCo started to delay their deliverables, SoftCo could not continue either and the pressure increased. However, these delays were not the real problems. It was the way it was communicated as identified by a project M project manager. “We agreed upon a planning, however, every week the delivery got delayed by another week. This constant delaying is really unproductive and at a certain
point you really have enough of it [...] In addition, this does not create trust between the partners at all.” (Project M interview 1. 18:00). This behavior continued throughout the project and polluted SoftCo’s buffer for the project. In addition, due to the problems around the stability issues and the uncertainty around SupplyCo’s deliveries, the progress of the project was lower than expected as can be seen in fig 38. This graph represents the number of tasks that took more time than expected.

Figure 38. Progress problems project M.

Scope decrease
These two problems of dependencies and performance issues led to the realization that the deadline would not be achieved if no action would be taken. Therefore SoftCo strived for de-scoping activities in order to achieve the deadline. This was also mentioned during the management meetings, “The scope of the product needs to be changed to secure the end date” (CT meeting October 2012). After ManuCo3 and SupplyCo approved this, on the condition that these de-scoped features would then be integrated into the next release, more space was given on the schedule and the deadline was still within reach. In order to assure this deadline, overtime was required from the project team. However, it was only expected to be 20% extra of the normal work hours. This to make sure, the velocity and quality of the development did not suffer due to fatigue in the development teams.

Quality
A few months before the deadline (November 2012), a new and very harsh quality issue showed up. This issue was so harsh because, in their attempt to improve the performance of the software, which was necessary for the outdated hardware, the software had become instable. Small changes in the code had huge impacts on the behavior of the system. In addition, because SFD had already shifted to the new software core version, only little support was possible from their side to aid in this problem. This caused for high pressure on the development team to solve this problem.
Finish
In the end the performance issues were solved and the project received a GO in April for the SOP. In the follow up project, the postponed features were still delivered and due to additional effort the relationship with SupplyCo was improved.

Main problems project M:
- Dependency with SupplyCo
- Intercompany relationship
- Outdated hardware
- Stability issues

Problems caused by ASD and TPM interaction:
- Dependency with SupplyCo

## 5. Analysis
In this chapter the second part of the research question was answered. Namely: *What are the main causes and consequences of these problems?*

### 5.1 Project S
As already mentioned in the methods section, in order to answer the research question, on how the problems evolve over time when both ASD and TPM are used as a development method. Project S is used as the main case. This project ran into several problems, some more important than the other. The problems that occurred in project S can be reduced to three main causes. These will be discussed in the following paragraphs. Furthermore, consequences from these causes will also be examined.

**Resource Shortage**
The first main cause of the project was resource shortage and had three underlying events. First, the project started too late, because resources were still caught up in a delayed project in trouble. Another problem was the underestimation of the complexity of the product. Therefore, the required resources for the project were underestimated. Furthermore, due to the withdrawal of SFD right before the deadline, a huge amount of work was added to SoftCo’s backlog without much additional support.

The resource shortage had several consequences. First, not enough effort was put into the first phase of the project. Hence, uncertainty could not be reduced enough and the complexity of the system got underestimated. Moreover, the resource shortage created high project pressure, over-utilizing the workload of the development teams and incorporating additional faults into the system as a result of this. Lastly, the project got behind schedule, because of the resource shortage. This led the management team to retrieve resources from rework discovery activities and placed them on development activities in order to maintain progress.
Dependency
The second main cause was the dependency on other partners which were involved in the development process. In project S there were internal and external dependency problems. The internal dependency problems were around the deliveries made by SFD. Due to the fact that SFD developed the main part of the system, they had an important role in the project. Nonetheless, their collaboration with the SoftCo team, who had to integrate the system, was not good. First of all, there was no clear organization structure for SFD and this project based working was a new way of working form them. This caused that SFD had problems delivering according schedule. In addition, due to this new role and resource shortage, they underestimated the size and complexity of the system. This gave an additional challenge for them in terms of delivering according to schedule. Finally, the sudden change to Android made things worse, because not enough expertise was there to build on this platform. It delayed their deliveries to SoftCo even more, while the quality was decreasing.

The external dependency problem was around the integration plan. ManuCo1 did not have a detailed integration plan and caused requirements continuously to change. This resulted into many additional work for SoftCo and ManuCo1. Another external dependency problem was around the availability of validation equipment. This equipment was necessary for the testing teams to validate their software on. However, this equipment was not often available. Consequently, the testing team could not test their software on the latest updates.

Collaboration
The third main cause of the problems in project S was the lack of collaboration between the involved partners. In project S the collaboration was not good. Even though the collaboration on engineering level between the SoftCo and ManuCo1 was good, on a higher level there was a lesser sense of collaboration. For example, no clear strategy could be giving which deliveries had priority. In addition, the problems around the integration plan and the validation equipment caused that no transparency was giving in this process and consequently complicated the development process. Furthermore, collaboration was also an internal problem, because different business units were involved in project S. SFD was struggling with its new role in several ways, consequently delaying and minimizing their deliveries. They could not give transparency to the SoftCo project team. Subsequently, increasing the pressure on them to finishing the project on time.

Pressure
A consequence of the difficulties around project S was the pressure in which the project was working. Because ManuCo1 had their product releases already planned and promoted, including the
new multimedia system, the project had to be delivered. Any delay in this progress would lead to a loss of sales and reputation. However, even though the problems were caused by both SoftCo and ManuCo1, no attempts were made to decrease this pressure. Also not when the delivery date was in danger. Therefore, the schedule remained the same and asked tremendous efforts from the development teams to “catch up” and deliver this large and complex project on time.

Fixes that fail
As a reaction to the huge pressure, SoftCo decided to shift the resources form rework activities to feature development. This was because SoftCo was evaluated on the amount of features that were delivered as already explained in the narrative. This “feature priority” behavior shows many resemblance with the fixes that fail causal loop diagram archetype by Senge (1990) (fig 39). It evolves around the problem of focusing on fixing short term problems. However, by doing so the real underlying problems got neglected and making the problems bigger. When is looked at the project S, the problem is the schedule pressure. Management tried to fix this by shifting resources from rework discovery activities to feature development in order to reduce the schedule pressure. However, an unintended consequence of this was that after a few months the system became very instable, because many bugs remained incorporated into the software. This resulted that the schedule pressure became even bigger due to the extra work these quality problems brought with it.

Firefighting
When is looked back at the main causes of project S, the mechanism behind these problems show a strong pattern with the phenomenon called firefighting. It is described in the new product development context as “The unplanned allocation of engineers and other resources to fix problems discovered late in a products development cycle” (Repenning, 2001, P286). This is concluded because almost all causes of firefighting identified by the author occurred during the project. First, there were the resource problems at program level. Due to problems with other projects, resources were allocated to those downstream projects in trouble. However, this might improve the performance of the project in need. It decreases the performance of the current project in the long run because resources were taken away from this project. In project S, the project also got delayed by previous projects and subsequently it interfered with the follow-up ManuCo1 project. Second, the system got overloaded as a result of the resource shortage. This shortage caused the amount of work to build up and consequently, the workload of the project members got increased which tried to catch up with the progress. Nevertheless, this progress increase was only temporal, because productivity and the
quality of work will decrease at the long run due to fatigue in the development team (Tomke and Reinertsen, 2012). Third, is the underestimation of the project complexity. Project S turned out to be much more complex than expected, hence the required resource were underestimated. This partly explains the huge gear-up action in March 2011. Fourth, due to changes in the scope and requirements, more resources were necessary than anticipated. These scope and requirement changes were mainly caused by the integration plan and abstraction layer. These had external issues, caused to put “oil on the fire” because no room was left for unanticipated changes it lead to more firefighting activities. Finally, the decision to continue with the project in order to maintain progress even though signs that the system was not stable showed up let the already troubled project run into more problem. The firefighting behavior resulted that the project was late (missed deadline and therefore lost revenue), over budget (exact numbers not know), with less quality (large amount of issues) and less features (scope decreases) than expected.

5.2 Case comparison
In order to investigate if the problems around project S were project specific, the results were compared to project D and project M. The differences and similarities among the problems are represented in table2 and discussed below. In addition, the problems are compared if they can be linked to the differences between ASD and TPM.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Project S</th>
<th>Project D</th>
<th>Project M</th>
<th>ASD vs TPM related</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cause</td>
<td>Resource shortage</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cause</td>
<td>Internal dependency</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cause</td>
<td>External dependency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cause</td>
<td>Collaboration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Consequence</td>
<td>Schedule pressure</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Action</td>
<td>Feature priority</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Consequence of action</td>
<td>System performance</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Underlying mechanism</td>
<td>Fire fighting</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2. Comparative of problems between different cases

Resource shortage
Resource shortage was a serious problems in project S case. First, resources were caught up in other projects. Furthermore, there was a hiring delay when a gear-up was requested. In addition, the withdraw of SFD created a huge gap in the needed resources in project S. Project D, had a recovery plan to catch up with the schedule. However, there were no problems in the availability of the required resources, thus in project D and M there were no resource problems present. Resource problems were also not identified as issues around developing products using ASD and TPM in the literature review section.
Internal dependency
SFD played an important role in two of the three of the projects. It had the largest role in the project S. In this project they were expected to deliver almost all the features, while in project D they only had to deliver a small amount of features. Though, the problem was the same. There was no clear communication when the features were going to be delivered and what exactly was going to be delivered. This caused additional schedule pressure in both projects. In project M, SFD did not play a role, because the project was working with the old software core and no features were sold to ManuCo3 which were on the SFD roadmap. Furthermore, internal dependency did not play a role in the ASD and TPM context, because both SFD and SoftCo were using the ASD method.

External dependency
The external dependency problems in project S were different from the other projects, because SoftCo had a different role compared to project D and project M. In project S, SoftCo was not only responsible for the development of the software. It also had to develop the hardware and system software and made the process of designing the system more complex. Meanwhile ManuCo1 had difficulties delivering accurate requirements which caused for problems around interpretation of the requirements. Furthermore, SoftCo was dependent on ManuCo1 for the integration plan and validation equipment. Because ManuCo1 had problems delivering this, the system integration and validation of ManuCo1’s products was a very painful process.

In project D and M, SoftCo was located as a Tier 2 supplier. This meant they had to deliver to the Tier 1 supplier, namely SupplyCo. Nevertheless, as with developing embedded systems, the hardware defines the constraints in which the software can be developed. SupplyCo developed the Hardware and the system software. Therefore, SoftCo was very dependent on them in building the software suitable for this system. The problem around this setup was that SupplyCo had many difficulties with this role and thus frequently delaying deliveries that were important for SoftCo to continue their software development process.

The external dependency problems can be linked back to the ASD and TPM context. The problem was that validation equipment was not available on time. SoftCo used ASD and developed software in sprints of two to four weeks. Consequently, they wanted to validate their software after each sprint. However, because SupplyCo developed the system in TPM, they could only deliver their validation equipment at the end. Therefore, the software developed by SoftCo could only be tested internally on issues and had to wait until the end of the development to validate it on the complete system.
Collaboration

Among all projects there were collaboration problems. In project S there were collaboration problems with SFD, who could not give transparency around the feature delivery. In addition, there were collaboration problems with ManuCo1, who could not give information on their integration plan. Furthermore, the deliveries around the validation equipment from ManuCo1 was also not good, as it was not known when this equipment would arrive and if it had the right components and software. In project D and M, SupplyCo had problems with delivering according to schedule. However, the real problem was not the delays itself, but the uncertainty around these delays. Often there was no reliable information available when a component would come. In addition, new estimates by SupplyCo were frequently violated. This made the process very inefficient because SoftCo did not plan other work, because they were waiting on the expected deliveries from SupplyCo.

Even though the collaboration problems were important, they cannot be related to the ASD versus TPM context. In the literature review section, collaboration is presented as a solution to solve the difficulties around the problems when ASD and TPM develop together, not as a cause.

Feature priority and schedule pressure

As a result of the schedule problems, the project pressure increased. In all projects there was a schedule pressure, only there was a difference in how this pressure was handled. In project S and D there was “chosen” by SoftCo, to solve with this pressure by focusing on feature priority. This action was decided because according to the project contract, SoftCo had to deliver certain amount features at a certain deadline. Therefore, they were evaluated by the manufacturing partners on the amount of features they built and not the quality of the delivered features. Subsequently, when having schedule problems, the short term solution was chosen to focus only on feature development and deliver all the feature before the deadline. However, the risk of using this strategy was that less attention was paid to bug fixing activities, and many issues got incorporated into the software. Consequently, stability issues showed up later on as can be seen in project S and D. Meanwhile, in project M there was not chosen for this option. Here, a feature decrease was achieved, even though the schedule pressure was high, the focus on quality was maintained.

The feature priority was caused by the differences between ASD and TPM. In TPM they work often with fixed contracts. In these contracts a fixed amount of features have to be delivered against a certain price and a specific time. After that deadline, there will be time to solve issues. This is what they also expects from the ASD developers and that is how they evaluate them. When a project goes
according schedule and the pressure is “normal” ASD can continue their normal development process. Working in sprints and ensuring high quality coding by testing all delivered features. When the schedule pressure increases ASD normally chooses to decreases the amount of features so that a high quality of the software can be maintained. However, when TPM prohibits a feature decrease, because they want all features, the ASD has to increase the productivity. This can be done by adding resources and/or enforcing overtime on the development teams and stay to the ASD way of working. If that is not an option a short term fix is used by shifting the resources from bug fixing activities to feature development activities. Nevertheless, neglecting bug fixing activities can hide large a large amount of rework and will only be identified right before the deadline. This can cause a huge amount of problems and endangering the deadline of the project.

**System Performance**

In all three projects there were performance problems. In project M and partly in project S these problems were a result of keeping hardware costs low. Therefore, only a minimum amount of memory was available. This caused many additional work for the project teams to increase the performance and get the system operational on this “minimum” hardware setup. Especially in project M this was a big issue, because ManuCo3 refused to invest additional funds into a more up to date hardware configuration. The performance in project D and mostly in project S were caused by bad coding quality as a result of the feature priority.

System performance problems are not always caused due to the differences in development methods. If the performances issues are caused due to ignoring rework activities and developing with a feature priority, then it can be linked back as mentioned above. Nevertheless, if the cause of the system performance is because of outdated hardware configurations, then it is not. That is an issue of not willing to invest additional funds into a proper hardware configuration. Often this economical thinking backfires, because in the end additional resources need to be invested to get the software operational on this out-dated systems as this was the case in project M and S

**Firefighting**

In project S, as already discussed the firefighting phenomenon occurred due to the resource unavailability, underestimated complexity, and the high workload. After project S, SoftCo decided to focus on software development only. This decreased the size and complexity of the system for SoftCo, and made it more robust against firefighting. In project D some signs showed up, it was nearing the tipping point. Especially the amount of extra work that the dependency and collaboration problems between SupplyCo and SFD gave were dangerous. This caused the overview
of the project to decrease. To prevent the project from cancellation, a recovery plan was organized and additional resources were added to the project to create overview again and decrease the workload. Nevertheless, the feature priority decision could have gone wrong, undermining the long term performance of the software, as seen in project S.

Project M had problems, nevertheless these were not severe enough to force the project in firefighting mode. Also the dependency problems were not too large, because of the low complexity of the system. SupplyCo had les problems of delivering their components according schedule. In addition, SFD was not involved. Having a dependency less to care about. Furthermore, when the project got delayed and the deadline was not feasible anymore, a feature decrease was achieved and the future deadline could still be achieved.

Firefighting behavior is not directly linked back to the differences between ASD and TPM, it is a consequence of the problems. When a project has schedule pressure, and the ASD supplier shifts resources to feature development activities priority to decrease this schedule pressure, the amount of undiscovered rework increases. When the project is nearing the deadline, and the team starts working on the rework activities. A large amount of rework can show up. When the project team is not prepared for this amount of rework and does not have enough resources to fix this the project can go into firefighting mode. Furthermore, collaboration and dependency problems can add large amount of additional work to the project schedule causing the project to go into firefighting mode.

5.3 Process Theory
The findings derived from the previous section are used to create an process theory. After analyzing our cases, which involved several partners that used ASD or TPM as an development method to create a new product. The firefighting phenomenon was clearly present in project S and to a lesser extend in project D. Besides the original causes of the firefighting as identified by Repenning (2001), new causes are found in this research project. Therefore the process theory derived from this research project is an addition to the firefighting theory proposed by Repenning and is presented as general propositions:

a. If the collaboration among different partners in the development process is not good, a significant amount of unidentified work can be created and push a troubled project over the tipping point into firefighting mode.
b. If the dependencies among different partners in a complex development process are not managed well, productivity goes down, the schedule pressure increases, and the firefighting mode can be triggered.

c. If rework discovery activities are postponed in favor of development activities, undiscovered rework will likely show up at the end of the development process and firefighting mode will be activated.

When these propositions are linked back to the ASD and TPM development context, proposition two and three can be linked to the problems between the two methods as identified in the previous chapter.

6. Discussion

Firefighting

The process theory that is derived from the result of this research project contributes to the firefighting literature. In addition, it fills a gap in the firefighting theory proposed by Repenning (2001). The firefighting article by Repenning (2001) describes the problems of the allocation of scarce resources to solve unanticipated problems in the new product development process. However; one important issue has not been discussed in this article. The author assumes in the article that new product development in an internal process, and executed by one company only. Nonetheless, more frequently companies are developing new product together, forming a partnership to share expertise and speed up the process. (Chesbrough, 2003; Lee et al., 2012). This is also the situation with the current research projects at SoftCo. They were developing multimedia systems together with manufacturers and other suppliers. Consequently, these companies become more dependent on each other. This makes the system more vulnerable to the firefighting phenomenon. In this dependency, when companies are not collaborating effective enough, they could create a significant amount of work for each other and can push the system over the tipping point as Repenning (2001) illustrated. Even a small increase of five percent in work could tip the whole system over the tipping point into firefighting behavior.

Even though many managers know about the firefighting phenomenon, how come it still exists? Sterman (1994) states that people do not learn well, because many factors can influence an effective learning process. Furthermore, when the complexity of the system increases, more barriers to learn exists and need to be overcome. This complicates the process even further and is also identified by Repenning (2001). Therefore, when a project enters a firefighting mode managers often make it problems worse, because of the lack of learning around this phenomenon.
**Resource planning**

In order to prevent firefighting Repenning (2001) proposes dynamic resource planning in order to deal with future resource requirements. A recovery plan such as the one in project D can therefore prevent the system to go into firefighting mode. However, in software development it is not desirable or sometimes possible to constant decrease and increases the required resources, because these resources are not immediately productive upon request. Moreover, new team members disrupt the development process in a negative way, since other team members have to invest additional time to get the new team members familiar with the project. This is also one of the “main points” of Agile software development, developing software using and stable small teams which collaborate and communicate often and informal (Nerur et al., 2005). Changing these teams continuously, would therefore diminish these benefits. In addition, by constantly changing the development teams, important expertise might get lost or does not get penetrated to all the team members effectively, which is also not beneficial for the project. In order to cope with this, it is very important to estimate the required work that needs to be done, so the team size can remain the same. Furthermore, if project tend to get out of hand it might be a good solution to become more flexible by decreasing the amount of requirements or features, which has be done successfully in project M.

**Collaboration**

As already discussed in the literature review chapter, customer collaboration is very important using ASD (Hoda et al., 2011). Furthermore, when the customer is involved into the development process this becomes even more important because of the dependency on each other. This importance is also seen in the analysis of the case studies. For example, good collaboration in the development teams resulted that in the end project S achieved the deadline, even though the many problems emerged in the project. However, when collaboration is not good, such as in project D and M, by delaying components without proper communication and transparency by one of the involved partners, the project can be delayed. Subsequently, it can cause for many frustrations in the team and lead to wasted time. All partners involved should have the same goal, and by good collaboration and creating transparency, problems can be solved much more efficiently.

**Dependency**

The research further provides support for the importance managing dependencies. Because of the involvement of different development methods, guidelines should be created to align these two methods. These methods require information on different times in the product development cycle.
When developing software, testing is vital. Also in embedded systems where testing is largely the responsibility for the software development team. All the features need to be tested on bugs and if they work properly when integrated into the main system. (Ronkainen and Abrahamsson, 2003). Furthermore, the system needs to be tested against the hardware set-up, because the hardware prescribes the environment in which the software can work. (Wolf, 1994). When this validation equipment and/or hardware components are not available, the software cannot be tested properly. Consequently, bugs in the system software can only be identified at the end of the project and potentially add a large amount of unexpected rework onto the project backlog.

This can be prevented by aligning the developmental process in several ways. First of all, early prototyping is very important so that the software can test as soon as possible on the hardware device and potential problems can be found early on and reducing uncertainty. In addition, by aligning the delivery of software releases and the development of hardware prototypes, this process can be speed up, because both development teams do not have to wait on each other. Furthermore, requirement freezes are necessary early on. (Cooper, 2014; Punkka, 2012; Ronkainen and Abrahamsson, 2003). As seen in the results of project S, due to continuous requirements changes almost until the end of the project, many extra work was created due to changes the software to the new hardware configuration.

Quality of the research design:
To ensure the quality of the case study research, the validity and reliability must be guaranteed. This will be done using four tests which are described Yin (2009). Construct validity is achieved by using a chain of evidence. To maintain this chain of evidence, the use of an external observer is needed. Because the research is only done by one researcher, this is not an option. However, construct validity is therefore achieved by using data triangulation (Yin, 2009). Even though internal validity is not really an issue with exploratory studies such as this study, it is taken into account by using the analyzing technique explanation building, in which evidence is searched to find the “why” behind a relationship or phenomenon. (Yin, 2009) External validity is taken into account by investigating three cases. In addition, to increase this external validity further in this research, academic literature is used to support the results. Finally, reliability is the last test. In order to create reliability, a case study protocol is used in order to create structure in the research process. Consequently, a case study database with all raw data, reports by the investigator, and data analyses is stored separately, resulting that the followed steps in the research can be tracked.
7. Managerial implications

The final part of the research question, “how can [the problems] be prevented?” is answered using recommendation to fix the problems that emerged in the project. These recommendation are described as managerial implication in the following section.

This research has several important managerial implications. First of all, it is very important to prevent the project from going into firefighting mode and create a fire-resistant project environment. As in line with the recommendations by Repenning (2001), the resource allocation should be done carefully and managers should not overload the system by understaffing the project teams. Furthermore, switching resources between projects which are in trouble has to be avoided. This can solve the short term problems on the downstream project, but can cause large problems in the upstream project.

Second, before accepting to conduct large projects, such as project S. There has to be determined thoroughly if sufficient competences are available within the company to do such a project. To prevent conducting a too large project, it is better to divide larger project over multiple smaller projects because when the size of project increases, also probability increases that design problems become difficult to solve, and thus increases the complexity (Mihm et al., 2003). This problem around complexity is also identified by Repenning (2001) as a source of firefighting, and should be assessed carefully. Otherwise, halfway throughout the project, unexpected tasks or rework due to this complexity can push the project into firefighting mode.

Third, when products are developed using multiple partners, it is important to remain flexible. Due to uncertainty the project environment can change, new insight could be gained or customer preferences can shift which can lead to an increase of work (Tomke and Reinertsen, 2012). When a project runs into problems, managers rarely update their planning and assume “catching up” is a feasible option. However, If no additional resources are available to fix these unanticipated problems, it can lead to firefighting. One way to deal with the new situation is to update the schedules, reprioritize requirements, and/or deliver in multiple releases. This prevents firefighting to spread to future project (Repenning et al., 2001). Therefore, the project planning should be approached flexible and remain feasible throughout the project.

Fourth, when working with different partners on a project, the communication and transparency should be at the highest level possible. Dependencies should be managed, and a global project planning should be set up in order align deliveries and provide all involved stakeholder with the
information they need. This is especially important when different development methods are used because they require information at different time. Consequently, when due to uncertainty deliveries cannot be done, this should be communicated with transparency. All partners should have a shared goal of making the best product possible (Hoda et al., 2011).

Firth, feature priority decisions should be avoided at all costs. This behavior is in line with the theory of Ford and Sterman (2003) on concealing rework. They propose that in response of schedule pressure, resources are shifted away from rework discovery activities to the development process. This creates the impression that the project is perceiving as planned. Also known as the illusion of control (Van Oorschot et al., 2013). However, this decision backfires “by the time [problems] are discovered and acknowledged, project managers face a large backlog of hidden rework” (Ford and Sterman, 2003: p216). On turn this behavior can create firefighting behavior because of this unexpected increase in work right before the deadline.

8. Limitations and future research
A limitation of this research project is that due to collaboration and transparency problems which played a large role in the projects, the interaction between ASD and TPM could not fully be investigated, because these problems were not related to the ASD and TPM development context. Therefore, it would be recommended to conduct this type of research again over project which did not encounter these collaboration problems.

A good future research proposal would be to extend this research project by integrating additional researchers. This research was conducted as part of a graduation thesis, therefore, the research had specified time frame and was conducted by one researcher. By continuing this research with an additional researcher, more data can be collected, the cases can be analyzed even more into detail, and an inter-coder reliability check can be provided for the coding of the data. Consequently, the results will be further substantiated.

Another limitation of this research is that it is only conducted at one company. Even though multiple cases are used, there can be company specific factors that influence this process. A good future research topic would therefore be to extend this research with cases from different companies to compare and further strengthen these results.
Because the case study data that was collected from existing data sources and not collected real
time, the research can be influenced by hindsight bias. Therefore, the research can be biased
towards perceptions of involved stakeholders around the causes of the project problems. In order to
deal with this problem multiple data sources are used to create triangulation and rule out this type
bias as much as possible. However, it would be interesting for future research to investigate similar
projects in real-time environment to investigate if new factors who influence this process show up.

The final limitation of this research project is that the chosen cases are only one project of a whole
project portfolio. As identified by Repenning (2001), actions taken in one project can influence the
downstream project in many different ways, such as the firefighting phenomenon. Therefore, it
would be very interesting to analyze this problem over multiple sequential projects in a portfolio
program which uses the same resources.

9. Conclusion
ASD and TPM can be used together in developing new product. Nevertheless, this will not work if
they do not adjust to each other. Both methods have to adapt. ASD has to join in the upfront
requirements so that the hardware environment can be fixed. TPM has to change their development
process by incorporating rapid prototyping, thus that the ASD partner can test appropriately.
Furthermore, rapid prototyping will also bring advantages to the TPM partner, because it reduces
uncertainty and identifies problems early on. The TPM partner also has to become more flexible and
recognize that projects are uncertain and not always go as planned. Therefore, the project planning
should be approached flexible in order to cope with the changing environment and avoid a too high
workload. Another important issue is that collaboration and transparency always has to be good. The
involved partners should have the same mindset comparable to the slogan from the Dutch army
“One team. One task”. If these recommendations are kept in mind, there is a high probability that
firefighting can be avoided.
References:


59


Appendix I: Interview questions

During the semi-structured interview these general questions are used as a guidelines to discuss these topics. Furthermore, during the interviews specific case events are also discussed.

- What is your current role?
- How did the project got started?
- How does this product look like regarding complexity?
- How does this product look like regarding features?
- Does the project require new knowledge?
- How was this project organized, and what is the role of the involved partners?
- How did the project went?
  - What went good, and what went bad.
  - And why?
- How was the work pressure during the project?
- How was the cooperation between the involved partners?
- What did you do when the progress decreased?
- Did you experienced resource problems?
  - From the start?
  - Gear up actions?
- Did you achieve a feature decrease and how did this process went?
- To which extend did you do a full feature priority?
- Can you tell me something about the quality problems of the software?
  - What was the cause of that?
- How do you react when you notice these problems?
- What do you think is the core issue between the problems around ASD and TPM?
- How can this be solved?