Uppsala University
Department of Informatics and Media

Extending the Metaphor

– Technical Debt in General Product Development

Malin Hansson
Maria Hognesius

Course: Bachelor's Degree Project
Level: C
Term: Spring term 2015
Date: 150611

Acknowledgements

We would like to thank John Gilhooly at Elekta and Benny Helgesson and Jan Thunqvist at Technia, without whom this paper would have been very different. We would also like to thank them for their invaluable input and ideas, and Benny for mentoring us throughout our research.

Also, a thank you to the employees at GE Healthcare who let us have an insight into the way they work, and especially to Anna Hansson for her guidance during the initial stages of the project.

Finally, we would like to thank our mentor at Uppsala University, Stanislaw Zabramski, for his valuable feedback and each other for this time of collaboration, marking the end of our studies together.

Abstract

It is arguably an important matter to track and eliminate poor quality. One way of doing this within software development is by applying the technical debt metaphor and using it as a basis when estimating the quality level. There was interest expressed in investigating whether this metaphor could be extended to the development of non-software products and use it as a starting point for developing features in a PLM (product lifecycle management) system able to track and help monitor technical debt throughout the lifecycle of a product. Thus, a case study based on a literature review and interviews analysed qualitatively were conducted. The result consists of the knowledge that similar notions as found in the technical debt research could successfully be applied to other development processes and phenomena as well, a framework consisting of types of technical debt and a "sketch" for how to measure it, and that in a technical debt-tracking feature it could be beneficial with functionality for implementing models for debt quantification, for allowing the organisation to set up rules, and for crawling through the product taxonomy in the PLM system, detecting deviances from aforementioned rules.

Key words:

technical debt, product lifecycle management, quality, costs of poor quality, system

Table of Contents

1 Introduction

In this chapter, the general background and motif of the study is presented. First, the background to the problem is related, leading up to the problem discussion, research questions and aim of the project. Subsequently, the scope is discussed and our intended target audience is communicated.

1.1 Background

Good quality products and processes have become increasingly important in a growing, competitive market (Bergman & Klefsjö, 2002, p. 9; Tisell & AMU-gruppen, 1991, p. 4). By products, we mean both goods and services. In order to ascertain good quality products, it is arguably an important matter to track and eliminate poor quality.

There are several definitions for the concept of quality, a variation possible to attribute to different perspectives from which the product is scrutinised. Three definitions among many are Crosby's "conformance to requirements", Juran's "fitness for purpose" (Juran & De Feo, 2010, p. 5), and Deming's "'[q]uality should be aimed at the needs of the customer, present and future'" (Bergman & Klefsjö, 2001, p. 23), the perspective ranging from production-focused when the first defines it, usage-focused when Juran is concerned, and customer-focused when it comes to the latter. Placing this on a quality spectrum, as suggested by Seawright and Young (1996), it is highlighted how much is encompassed by the term. Consequently, a business has several areas to focus on when monitoring its costs of quality, something that can be a complex process.

Costs of poor quality (previously referred to as "cost of quality" in the field) refers to the costs of producing poor quality products (Bergman, & Klefsjö, 2001, pp. 64). These are costs that might not be immediately apparent to the business, but when products reach the customers, costs of quality can appear in the form of necessary redesign as a solution to customer complaints. Bergman and Klefsjö (2001, p. 65) mentions faulty goods or poorly performed services as examples of occurrences that are costly to the business.

Research has been done on the tracking and quantifying of costs of quality (Davis, Ledbetter & Burati Jr, 1989; Heravi & Jafari, 2014), but within software development, a concept with the purpose of encouraging the prevention of such costs has been developed since the 1990s, called technical debt. The debt metaphor was first suggested by Cunningham (1992) and he also coined the term. He argues that "[s]hipping first time code is like going into debt" (Cunningham, 1992). Some debt is acceptable as it might help speed up processes, but that consequences can arise when the debt is not paid back straight away and interest starts to accumulate. Cunningham appreciated the need for a model able to bring attention to what could be done about quality issues in the early stages of software development. In doing so, it would be possible to prevent having to make costly and time-consuming reconstructions of software products later on. The necessity to create a good foundation for high quality as early as possible in the life cycle of a product, preferably the development stage, has increased in the market (Bergman & Klefsjö,

2001, p. 57), and therefore a similar way to identify flaws such as with the technical debt metaphor has become interesting to practitioners.

The benefit of attempting to identify sources of quality deficiencies in a business in general occurred within Elekta, a human care company concerned with treatment of cancer and brain disorders. One point of interest was found in the Enovia product lifecycle management (PLM) workflows and a discussion of the topic was initiated with the supplier of this PLM system, Technia. This sparked an interest at Technia, being a company developing product lifecycle management solutions, for helping their customers with the ability to track sources of quality deficiencies and other issues with not immediately noticeable consequences.

The foundation for the system Technia develops is acquired from Dassault Systèmes and Technia develops a software product to go on top of it, Enovia, PLM software. The realisation that knowledge about technical debt used as a metaphor in general product lifecycle management would be beneficial motivated Technia to encourage further research to be conducted.

1.2 Problem discussion

Technia would be interested in being able to model and quantify issues similar to technical debt which will help the customers track issues in production which can have consequences in quality or revenue. Before this project can be initialised, however, it has to be investigated whether talking about technical debt is meaningful outside software development. If the technical debt metaphor is awkward and adds no value to the discussion, basing a system feature on it would be difficult. There is also an interest of Technia's how practitioners would view technical debt in their respective areas of expertise to gain a better understanding of what their monitoring feature could be able to find.

The main problem at this point is then the lack of a concept of metaphorical debt and a framework for discussing the sources of quality costs within non-software product development. A search in science databases produced no literature on the subject, probably because technical debt has not been applied to product lifecycle management before. However, there is a possibility that attempts to implement the debt concept in a similar way that we aspire to do have been performed, but that it then is referred to by a different term than technical debt.

In software development it is important to be able to talk about the drawbacks of different design decisions, assess their effects on the project as a whole and disperse such information about their existence and consequences throughout the organisation (between colleagues, to managers, et cetera). For this reason, Ward Cunningham (1992) introduced the metaphor of debt in software development in the early 90s, thus facilitating the discussion of this phenomenon within the field. Likewise, discussing the impact of engineering change requests, customer complaints, non-conformity issues et cetera has a value within product lifecycle management. These events can give rise to issues in the production, consequently creating potential for losses in time and/or money. The existence of this absence of a framework motivates research into how metaphorical debt can be applicable within product lifecycle management in a similar way it is within software

development, and how it can be tracked using software. These thoughts motivated our research questions, posed in the following section.

1.3 Research questions

- How can the concept of technical debt be applied to development of non-software products?
- Is there a perceived use for the technical debt concept among practitioners within product lifecycle management?
- What functionality would be beneficial for detecting technical debt in a PLM system?
- What is needed before one can even think about developing a technical debt-monitoring feature in a PLM system?

1.4 Aim

The practically oriented purpose of this paper is to lay the groundwork for the development of a program able to track technical debt in PLM systems. Therefore, the first piece of the final contribution will be the presentation of a framework for describing and measuring technical debt in product development. To do this we need to understand what defects of quality occur and what causes them. Thus, the cause of quality issues will be the main interest while gathering the data. There is already research on how to measure the quality cost for a company, the cost when something goes wrong due to something being not quite right. What we are interested in is tracking the reasons for quality costs and finding a way to notice them before they turn into actual quality costs for the organisation. We believe it will bring value to companies working with product lifecycle management to be able to be notified about quality issues before they create larger problems. This way they can be rectified at an early stage.

Today, no framework exists for talking about the losses in time and money that can occur in product lifecycle management due to events like customer complaints, engineering change requests and non-conformity issues, neither do standards for how to measure these losses in time or resources. Hence, the purpose of our research is to create a framework with which we will analyse the possibility to integrate the concept of technical debt into product lifecycle management. To enable doing this, the methods of measuring technical debt and how it is handled in software development will be investigated, a general orientation of the workings of product lifecycle management will be done, the importance of quality will be examined, and a summary of how all of these concepts fit together will be presented.

Groups for classifying technical debt exist for software development, and strategies for how to manage it. A framework with similar components for a discussion within product development would potentially be very useful and the aim of our research is to investigate whether such a transition is possible. We will investigate the elements of the established framework: prerequisites for measuring technical debt in software development and techniques to measure it. The new framework based on these elements would then fill the need that arises when employees

working with product lifecycle management and keeping track of all the products in product lifecycle management systems need to be able to communicate to the "higher-ups" about different issues.

When the existence of a meaningful framework has been established, our aim is to move on to the second and main piece of our contribution by outlining functionality that would be beneficial to include in a system for tracking and measuring technical debt in product lifecycle management.

The academic purpose of this essay is to initiate a new field of study in hopes that academia will realise the importance of proactive quality management and how technology can facilitate that. Li, Avgeriou and Liang (2015) argue that the coining of the technical debt metaphor helped spark an interest within academia and subsequently valuable research has been made. Likewise, the aspiration is to shine light on this unexplored area of product development and help setting the ball on motion.

In conclusion, the purpose of this research project is to suggest a new framework for dealing with metaphorical debt in the field of non-software product development, based on the knowledge obtained from analysing literature, previous research and our gathered data, and by doing so be able to reach our main goal: giving guidelines for what functionality a tracking software should have. The contribution to the academic sphere will be the results found in this paper that should be useful for researchers who wish to continue diving deeper into the subject matter of technical debt in general product development and systemic monitoring tools.

1.5 Scope

The scope of the essay is to lay the groundwork and present ideas that facilitate the future creation of a system implementation able to measure technical debt in non-software products. It has been investigated whether this new application of the technical debt metaphor is meaningful, and whether there is an interest among practitioners in the field of product lifecycle management. A draft of a framework for how to think when measuring technical debt in non-software products has been developed. The area of research is information systems, which is why we have been looking at this from a systemic viewpoint.

There is no previous research on the application of the technical debt metaphor outside software development, which restricts the scope of our study. Since no knowledge about whether it is meaningful can be drawn from non-existent data, it is not within the scope of the essay to develop a system or to present a concrete requirement specification. It is too early to do so, because it is necessary to do an initial study before the route for the future can be decided upon. Trying to develop functionality before the topic under scrutiny is better understood poses a risk of poor design decision being made.

1.6 Target audience

The presentation of our results has been done with practitioners and their managers in mind. We expect them to have insight into PLM and we have strived to present the uncovered knowledge in such a way that they can realise the benefits of discussing technical debt in their field of work.

Technia are stakeholders in this project as they initiated the research project and presented the basic idea. Technia is a Stockholm-based organisation with offices around the world and, among other things, develop and provide product lifecycle management solutions for other organisations. They supply their clients with, for example, PLM systems or CAD solutions accommodating the clients' business models. They would like to know whether technical debt is a phenomenon within product development in general, so that they can improve their PLM offering for tackling technical debt.

Our audience will also be researchers on a pre graduate level or above who might not have extensive knowledge on technical debt or quality management but are curious about the subject and perhaps would want to continue on our work. As no knowledge is expected prior to reading it, the theories and terms used throughout the paper will be explain in section 3.

2 Method

In this section the method, including research strategy, data generation method, sampling choices, method for data analysis and procedure, is described. Finally, there is a short evaluation of the weaknesses in our method.

2.1 Research strategy

As for the generating of new insight in relation to our research questions, the issues that can arise due to mistakes or errors during a product's lifecycle, what signifies such issues, and what consequences they carry had to be uncovered and explored thoroughly before the feasibility of applying the technical debt metaphor could be determined and a framework could be elaborated. A holistic view of these phenomena needed to be gained, since a framework per definition is not merely a collection of different aspects but on how these aspects interact and affect each other.

A case study is an appropriate strategy for generating data of such nature, since it, as described by Oates (2006, pp. 145), facilitates making abstractions such as concepts, theories and implications, and gaining rich insight based on the conducted research. It also has the advantage of dealing with situations where factors cannot be studied separately and the complexity is an important part of what is being investigated; the lifecycle of a product is complex, as can be seen in section 3.6.1. Furthermore, Oates points out how case study research can be linked to theory and used to develop a new theory in the shape of, for example, a conceptual framework. The aim of this project has been to expand the knowledge base in the field of product lifecycle management, an exploratory study, rather than descriptive or explanatory, has been considered most suitable. The study was carried out over a short period of time (approximately one and a half months) at the Technia headquarters and at GE Healthcare Life Sciences, because they agreed to aid us in our research and they pose typical cases within the product lifecycle management field as respectively a PLM systems developer and a user of the PLM approach.

2.2 Data generation method

First, the choice of data generation method, interviews, is justified and the upon that following subsection explains the design of the interview questions.

2.2.1 Interviews

One of the objectives of our project has been to find out what quality costs can arise within companies working with PLM, what the sources are, what consequences they give rise to, and ultimately whether it would be meaningful for them to have the system provide a method to keep track of "debt". In order to obtain people's views on these matters, we have conducted semi-structured interviews.

The reason for selecting interviews, and not for instance a questionnaire, is that more detailed information can be gathered from interviews (Oates, 2006, p. 187). We preferred to ask open-ended questions, which are easier to obtain exhaustive responses to in interviews than in questionnaires.

Oates further argues that complex questions are preferably asked in an interview, since one gets the opportunity to clarify if the respondent does not fully understand the concept. Then the interviewer is able to explain it in a way the respondent will understand, and the respondent in turn can ask questions until we feel confident we are on the same page and talking about the same thing. This fits the situation of having to explain the technical debt concept in software development before asking the respondent what they see it as being in their own field. Furthermore, we as interviewers can in turn ask follow-up questions if a response is unclear.

We did not need a series of very specific questions answered, but rather a general idea about the respondents' view on the subject matter, and neither was the intention to generalise the answers to an entire population. It was more important to get a flow in the conversation and to let the respondent explain their views, than making sure every question was phrased in the exact same way in each interview. These are reason for choosing a semi-structured interviewing technique, rather than a structured one (Oates, 2006, pp. 187-188).

We had, however, a specific topic and general questions we wanted answered, so unstructured interviews would have been an inferior option (Oates, 2006, p. 188). The aim has been to acquire knowledge of opinions and thoughts on our topic from people who handle the issues we have attempted to help solving.

2.2.2 Interview questions

When composing the interview questions (which can be found in appendix A), we had our research questions in mind and each question to be posed in the interviews were carefully analysed. That is, we made our best attempt at anticipating not the exact answer, but the type of answer such a question could elicit and how it could benefit our analysis. For example, the question "What consequences can the defect give rise to?" we concluded could potentially give a reply which would help us be able to identify an issue that causes what in the technical debt metaphor is described as interest. If, after formulating a question, speculating about its types of answers, it was concluded that such an answer would be of little use to answer the research questions, that interview question was discarded.

2.3 Sampling

The sampling frame in the case of our research is all practitioners involved in the field of product lifecycle management. From this frame we selected four from GE Healthcare Life Sciences, with work tasks as varied as we could find with our limited resources. The fifth interviewee was selected from Elekta as a complement because of their special insight in the matters (see section 2.3.1).

Equal measures purposive and convenience sampling were involved when the interviewees were selected for this research. We tried to get as varied a sample as possible, and tried to get in contact with people with different work tasks, which is why the sample is purposive. The convenience quality of sampling lies in the fact that we interviewed the first people at GE we could get in contact with who were willing to contribute, instead of putting effort into, for example, covering all phases of the lifecycle.

Elekta, which is the company within which the idea of technical debt outside software was brought up, are involved in the area of life sciences. We wanted to stay within that area, which is why GE Healthcare was selected as the company where we would conduct our interviews with practitioners.

2.3.1 Elekta

Elekta is a Swedish company, founded by a neurosurgeon, and create innovative solutions for the treatment of cancer and brain disorders. Their primary contribution is the creation of a gamma knife, with which gamma radiation is used to treat brain tumours. They are customers of Technia and employ a PLM solution provided by Technia. It was the employee at Elekta whom we interviewed who brought the idea of technical debt outside software to Technia's attention.

2.3.2 GE Healthcare Life sciences

GE Healthcare is a part of the GE group and is a large multinational company. GE Healthcare is engaged in providing laboratory equipment and medical equipment for diagnostics and treatment. Life Sciences is a branch of GE Healthcare who specialise in research concerning life sciences and production of equipment and medical chemical substances used in healthcare. They are customers of Technia and use Technia's PLM system software Enovia.

2.4 Method for data analysis

We have used a qualitative approach when analysing our gathered data. The empirical material we have gathered is in the form of transcripts of interviews. There was no intention to quantify this information in order to generalise our results for application in other areas; instead, the purpose of the analysis has been to gather understanding and a sense of recurring characteristics. Additionally, we have tried to get a picture of the issues found in product lifecycle management and how they are dealt with and because our data is in the form of text, a text analysis has been performed, where patterns and themes have been sought after (Oates, 2006, pp. 268-272).

2.5 Overall procedure of the study

To begin with, literature in the form of articles, books and websites on the relevant subjects was studied. This was in order to receive the necessary knowledge to proceed with the interviews

planned. The concepts of technical debt, product lifecycle management and quality were investigated and mapped out.

Then followed the interviews and to start with we worked out a plan for what we would like to know about what can be classified as technical debt for practitioners within the field of product lifecycle management. In order to receive the data necessary we devised a number of questions designed specifically to give us the best conditions to receive just that. The questions were reworked until they fit the purpose as closely as possible (see section 2.2.2). The interviews were conducted over the span of a week and a half and were transcribed continuously.

The interviews were then analysed based on the interview questions we wanted answered. The precise answers to them were picked out and comprised into a table (Appendix B). The problems and their causes were then analysed to determine if they could be considered as technical debt or not. The ones that could were further analysed and inserted into appropriate categories. The idea of categorising problems into groups was inspired by Li, Avgeriou and Liang (2015).

The gathered data was the base on which initial interpretations could be made with the starting point being studies of technical debt in software. We used technical debt as a lens when we studied the issues found in product lifecycle management in general and the reasons for them. The concept, or parts of it, was used to form a framework for measuring "debt" outside software.

2.6 Evaluation

Scopus, ProQuest and Google Scholar were used in the literature review. Scopus and ProQuest are databases for research papers in various fields and computer sciences respectively. Google Scholar is a search engine for science literature on any topic. What was not known about Google Scholar during the process was that "the returned papers in Google Scholar are sorted by Google's PageRank technique which may cause recently published relevant papers to rank very low." (Li, Avgeriou & Liang, 2015) Had this been known, more care had been taken to sort on more recent years, since older studies have had a longer period of time to increase their rank, thus appearing higher up in the search results than recent research.

Another source used is Wikipedia, which we realise is not one of the most trustworthy sources. However, it was never used to find theories or information on debatable subjects, but merely a definition of a technique. We already had a rather solid idea of what this technique amounted to as it had been shown to us by a user, and deemed the information on Wikipedia to be at an acceptable standard.

Due to the time frame, it was not possible to make an extensive amount of interviews in order to gather data for the study. Five interviews were conducted, and the data gathered should therefore be viewed as examples, and have not been generalised to their entire respective fields or considered the only issues to take into account in a possible system development later on.

As for the sampling technique, choosing the first people we could get in contact with who were willing to contribute, is not ideal. However, as we had no real insight into the various tasks of people working with product lifecycle management, we could not know beforehand if the chosen interviewees would be able to give us what we wanted or not. Had we had a longer time frame we could have started out with the interviewees we could reach on our own, and then used a snowballing technique to reach more suitable candidates, as the initial interviewees had better knowledge about who might know what they did not than we did.

During the interviews we began by asking general questions about general issues and did not explain the technical debt concept until towards the end. The idea was that we this way would avoid leading the interviewee into too narrow thinking and give us a better chance to analyse the problems ourselves. However, the answers given to us after presenting the technical debt notion tended to be more useful and if a similar study was to be conducted, we would suggest explaining the concept earlier on.

3 Theory

In this chapter the concepts essential to our work will be thoroughly explained. Initially *product* will be given a definition, followed by an investigation of the concept of *quality* and *costs of poor quality*. The concepts of *technical debt* and *PLM* conclude the chapter. Within the subchapter of technical debt, two theories in particular will be described for further discussion in the analysis and discussion chapter.

3.1 Product

The convention in the quality literature is to refer to both goods and services as products (Juran, 1998, 8.1). This means that both organisations providing services and organisations providing goods to customers are providing products. Furthermore, in this paper, "product" refers to not a physical object but the concept or idea of a product. To more readily grasp this notion, think of the difference between a car company that has different models of cars in their product portfolio and physical cars at a retailer. Perhaps no car of one model has yet been manufactured, but the product, the idea of the car, is still something that exists in the context of the company, and marketing or design decisions can be made regarding that product. In that case, those decisions are not intended for a specific physical instance of a product, but the product as an abstract object in the product portfolio of the company. The physical instances of a product are the identical cars found at the retailer.

The products are the most essential things for a company, without which they would have no revenue and no customers (Stark, 2011). A product is not necessarily a tangible thing. In fact, there are three different kinds of products: tangible, services and intangible (Sääksvuori & Immonen, 2005). Tangible products refer to physical goods, which can be anything from a cheap pen to a large multi-million pound piece of equipment. A service can be for instance a vacation package provided by a travel agency, including a seat on an aeroplane, a room at a hotel and possibly one or more guided tours. Every package is different, tailor made for the individual customer. Intangible non-physical products refer to computer programs and algorithms (Sääksvuori & Immonen, 2005). PLM helps keep track of all the information concerning these products, regardless of the type of product, which will be further discussed under section 3.6.

3.2 Quality

Quality has over time grown more and more important to businesses and become an essential part of a business' road to success. It is essential to businesses as a mean to acquire market edge (Seawright & Young, 1996) and Engdahl and Ottenvall points out that the discourse around quality is increasingly and rightly taking up more space, since quality is an important tool for distinguishing the business from its competitors (Tisell & AMU-gruppen, 1991, Förord [Preface]).

The term quality is an often used, but unfortunately also often misused, term (Bergman & Klefsjö, 2002; Tisell & AMU-gruppen, 1991). A probable reason for this is that "quality" means different things to different practitioners. They use it in their respective field the way it makes it the most relevant to them, resulting in varying definitions focusing on different matters. Therefore, it is important to be aware of these differences to be able to detect what someone refers to when casually talking about "quality".

A few noteworthy definitions are presented by quality gurus Crosby, Deming and Juran (Flynn, Schroeder & Sakakibara, 1994) and Bergman and Klefsjö (2002) gives a brief account of each of them. Crosby argues "quality" is "conformance to requirement", that is, when the product meets the requirements outlined by the organisation. Juran defines quality as "fitness for use", meaning how fit the product is for its being used by the customer. Quality can also be, according to Deming, a focus on the customer's needs as of today, but also tomorrow. He brings attention to the importance of a long term approach towards quality issues rather than a short term focus on what can be done at present. There is also a different attitude suggested by Genichi Taguchi. Rather than focusing on what quality is, Taguchi presents the absence of quality as the total loss caused to society by a product's delivery, including environmental and other issues arising in contexts outside the products direct use (Bergman & Klefsjö, 2002; Tisell & AMU-gruppen, 1991).

When reflecting on these definitions of quality, two points of view can be discerned. Crosby's idea of quality is centred around requirements drawn up by the organisation while Juran's and Deming's approaches emphasise the customer's role. Furthermore, according to Bergman and Klefsjö (2002, p. 35), the modern view of quality is to have the customer as the focal point. In other words, Crosby shifts attention to the internal workings of the organisation and the others shift it towards the customer. These attitudes are contrasting in their focus, but not mutually exclusive when applied within an actual business context. Arguing that a certain aspect of quality is more important than another might be true, but neglecting any one of them can give rise to problems. Disregarding the design specification or the customer requirements will result in a product deviating from what it should be according to the business.

Therefore, rather than regarding these various definitions as contradicting, there is a more including approach presented by Seawright and Young (1996). They acknowledge that there are diverse definitions of quality that overlap to different degrees, proposed by different actors, but connect these by describing them as parts of a continuum. On this continuum exists transcendent, manufacturing-based, product-based, user-based, value-based, multidimensional and strategic quality and they range from external to internal (to the organisation) and from subjective to objective. At one end is transcendent quality, judged by the customers' perception and "described as a condition of overall excellence" (Seawright & Young, 1996), making it quite subjective and external to the producer of the product. Towards the other end of the spectrum is the manufacturing-based quality, which is defined as centred on conformance to requirements drawn up by the firm. Subsequently, this kind of quality is simpler to measure by comparing the product to the specifications, making it more objective as well as something internal to the company. The aforementioned quality definitions settle at different points on the continuum, making it visible how different definitions can coexist without stripping the term "quality" of

useful meaning. It is important to remember the presence of the diversity in meaning and in any given context reflect on what is meant instead of jumping to conclusions.

3.3 Quality defects and quality costs

When a product deviates from the desired level of quality (of any kind, as described above) it is referred to as a quality defect. Based on a report from Centrum för verksamhetsutveckling [Centre for Business Development] (2004), two examples of quality defects are the fact that the finished product does not comply with the specification or processes associated with the product are not carried out in an adequate manner. Juran and Blanton Godfrey (1998, pp. 8.4) list, among other factors, failure to meet customer requirements and needs, inefficient processes, and lost opportunities for sales revenue as quality defects.

However, quality defects are rarely referred to simply as such, but as costs of poor quality (previously referred to as "cost of quality" in the field), and are synonymous with the costs they give rise to (Bergman, & Klefsjö, 2001, p. 64ff; Centrum för verksamhetsutveckling, 2004; Juran & Blanton Godfrey, 1998). Much similar to quality, "quality costs" means different things to different people. Depending on the context, practitioners commonly use the term in different ways and three examples of what it can mean are: costs imposed on the organisation by poor quality, costs of attaining good quality, and costs associated with operating the quality department as a whole (Juran & Blanton Godfrey, 1998, 8.1). Bergman and Klefsjö (2001, p. 65) mentions faulty goods or poorly performed services as examples of occurrences that are costly to the business. Juran and Blanton Godfrey (1998) states they mainly use costs of quality to mean costs amounting as a result of poor quality. These are costs that might not be immediately apparent to the business, but when products reach the customers, costs of quality can appear in the form of necessary redesign as a solution to customer complaints. Producing products of good quality is obviously also costly, but those costs are a part of a different issue.

Cost of (poor) quality is about finding what defects can be remedied to deliver a better product. Three reasons for identifying and measuring the costs arising from poor quality are: to motivate an effort to improve by quantifying the scope of the quality problem, to guide the direction of that effort, and to monitor the success of such efforts. (Juran & Blanton Godfrey, 1998, 8.1)

3.4 Technical debt

Technical debt is a concept used in software development and refers to the fact that use of suboptimal code practices leads to potential problems later on. Ward Cunningham introduced the metaphor of technical debt in 1992. He argues that:

> Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. Objects make the cost of this transaction tolerable. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise. (Cunningham, 1992)

The metaphor of debt is one taken from the financial sphere, wherein money is borrowed, and is to be returned later. In the meantime, interest on the loan has to be paid. In software development this could be expressed as when writing a program and trying to meet a deadline, a programmer can borrow time by taking a "programming shortcut", that is, choose to diverge from the generally accepted best practice. This time has to be paid back later, when amending the problem that was not fixed straight away. Sometimes resolving this issue takes longer if you do it later, than if you do it immediately; the extra time put into this is part of the interest you pay. Also, the use of a suboptimal program can lead to slower processes and lower production, and this also counts toward the interest you pay on that time you borrowed.

The causes of technical debt are many. Suboptimal architecture or design of the system, poorly written code, poor documentation, and unimplemented features constitute a few examples of this. Often they are shortcuts taken in order to save time. (Brown et al., 2010)

The problems result in for instance slower processes (Brown et al., 2010), which leads to lower productivity, and lower company morale (Tom, Aurum & Vidgen, 2013). This is also possible forms of aforementioned interest which has to be paid.

The longer you wait to pay back the debt, the more interest you have to pay. This is a real problem in the world of software development, and costs companies a fair amount of money every year (Tom, Aurum & Vidgen, 2013). As Cunningham suggests, a little debt can be acceptable, if it means the program is released before deadline, and the updates needed are done right after the release. The problem occurs when the short term solution is not fixed straight away, and the flawed system is kept in use as it is. Then the organisation suffers in the long term. Often, decisions are made to benefit the organisation in the short term, and the long term consequences are not taken into account. What is needed is a way of managing technical debt and for the financial impact of it to be made visible.

A great deal of research has been done on the topic of technical debt and Li, Avgeriou and Liang (2015) have made an extensive study of existing research and made a classification and thematic analysis of them in order to gain a comprehensive understanding of the technical debt metaphor. The descriptions of technical debt that they found they recognised could be classified into ten

main types and a number of subtypes within each main type. The main types Li, Avgeriou and Liang found are:

> *Requirements technical debt* refers to the discrepancy between the ideal requirement specification and the implementation of the system.
> *Architectural technical debt* refers to when architectural decisions have consequences compromising maintainability or other aspects of internal quality.
> *Design technical debt* refers to debt arising from decisions within more detailed design.
> *Code technical debt* refers to the fact that written code violates best coding practises or coding rules, such as code duplication and unnecessary complex code.
> *Test technical debt* stems from neglecting proper testing and skipping, for example unit, integration or acceptance testing.
> *Build technical debt* refers to problems in the build system or build process of a software system which make the build unnecessarily difficult or complex.
> *Documentation technical debt* refers to documentation that is incomplete or insufficient in some way. Any aspect of software development documentation is included, such as out-of-date architecture documentation or inadequate code commenting.
> *Infrastructure technical debt* refers to the setup of development-related processes, technologies or other supporting tools that are suboptimal.
> *Versioning technical debt* refers to problems in versioning of source code, such as problematic code forks.
> *Defect technical debt* refers to the empirical defects, bugs or failures observed in the software product.

Li, Avgeriou and Liang (2015) also collected notions found within the technical debt metaphor. They describe the notions they found in the studied literature and compiled them in a form of a dictionary. Definitions for concepts such as interest, principal, risk, type, cost, cause and others in the context of technical debt can be found.

As for quantifying technical debt, Nugroho, Visser and Kuipers (2011) have proposed an approach for quantifying debt and interest respectively. They start off by defining debt as "the cost of repairing quality issues in software systems to achieve an ideal quality level" and interest as "the extra maintenance cost spent for not achieving the ideal quality level" (Nugroho, Visser & Kuipers, 2011). They applied the new approach to a real system in a case study and found it facilitates obtaining a good idea of the benefits of mending system issues, the cost of those repairs and the anticipated payback time.

As a foundation for their quantification approach of technical debt they put maintainability as software quality, that is, the system's quality is calculated based on maintainability. Since their definition of debt hinges on "an ideal quality level", measuring debt requires a way to assess the quality of a system. The chosen method was SIG/TÜViT's software quality assessment method (a method for assessing quality, used by the organisation TÜViT to certify software), since it is developed for measuring and rating the quality of a software system from a technical standpoint based on the quality characteristics found in ISO/IEC 9126. The ratings of quality are then mapped to ratings for sub-characteristics of maintainability according to the ISO/IEC 9126.

Averaging these ratings will return the complete maintainability rating. The model is possible to adjust so that contemporary software system representativeness is kept. (Nugroho, Visser & Kuipers, 2011)

The constituents of the formula for calculating technical debt based on SIG/TÜViT's model is then *rework faction* and *rebuild value*, the latter being the product of the system size and the technology factor (Nugroho, Visser & Kuipers, 2011). These factors are estimated by looking at an estimate of how much of the code needs changing, and an estimate of how many man-months of effort would be necessary to improve it to the next level of quality. The rework fraction and rebuild value are multiplied to provide repair effort, which then is the cost of increasing the level of quality from a level lower than the ideal one. Ergo, this repair effort being the work needed to reach the ideal level, the definition of having no debt, is the amount of technical debt in the software system. (Nugroho, Visser & Kuipers, 2011)

3.5 Product Lifecycle Management

Product lifecycle management is concerned with managing products throughout their entire lifecycle. Before the concept of product lifecycle management is explored, the product lifecycle itself will be described. Finally, the advantages of product lifecycle management (PLM) will be accounted for.

When PLM is used, the management of the lifecycle of a product is referred to. This includes every step of the lifecycle of a product from the conception of the idea to the discontinuation. Steps such as the actual production of a product and the maintenance of a defect one are included. PLM is not to be confused with the term PLM system, described under section 3.7.

3.5.1 The product lifecycle

The lifecycle of a product is divided into different phases, from the initial idea to the retirement of the product. Different sources have different views on how many phases there are, and what to call them. For the purpose of this essay two books on PLM have been studied: *Product Lifecycle management* by Sääksvuori and Immonen (2005), and *Product Lifecycle Management: 21st Century Paradigm for Product Realisation* by Stark (2011). The phases of product lifecycle management are described differently in these two works. Sääksvuori and Immonen describe the cycle as ranging from *development*, through *planning*, *design*, and *production* to *use* (Sääksvuori & Immonen, 2005). Stark (2011) on the other hand is of the opinion that the product goes through the stages *imagine, define, realise, use/support* and *retire/dispose* in its lifecycle. We rely more on Stark's version since the retirement of a product is an important part of its lifecycle, and Sääksvuori and Immonen do not include that phase at all. All products are not used forever and some products or parts are discontinued for some reason. It is then important to have procedures to know how to, for instance, dispose of or recycle an instance of a product. If a product that is withdrawn contains hazardous materials it is important to know who will handle the disposal of this. On the manufacturing side there needs to be a record of the fact that the product or part has been taken out of use and may no longer be able to receive support on. If a

product contains dangerous materials or substances it is vital that it is known how to handle the disposal of this product as the consequences can be an environmental hazard if it ends up in a lake or somewhere else in nature.

3.5.2 What is PLM?

In his book *Product Lifecycle Management: 21st Century Paradigm for Product Realisation*, Stark writes that "PLM has a holistic approach to the management of a product" (Stark, 2011, p. 8). What he is referring to is that PLM not only helps manage the products in themselves, but focuses on all the components, such as products, data, processes, people, applications and equipment (Stark, 2011). He means that "PLM brings together what was previously separate, for example product development and product support." (Stark, 2011, p. 8) PLM ensures that everything that concerns the same product is kept in the same place.

Also Sääksvuori and Immonen talk about PLM being holistic. They hold that:

> PLM is a holistic business concept developed to manage a product and its lifecycle including not only items, documents, and BOM's [bill of material], but also analysis results, test specifications, environmental component information, quality standards, engineering requirements, change orders, manufacturing procedures, product performance information, component suppliers, and so forth. (Sääksvuori & Immonen, 2005, p. 2)

This then, the combining of everything concerning a product, and the fact that it is controlled throughout its entire lifecycle, is the crucial point of PLM.

When talking about PLM one does not necessarily refer to a computer solution, even though there often is one. PLM in itself is not a system or a program, but a concept - a set of systematic methods used to control information related to products (Sääksvuori & Immonen, 2005).

3.5.3 Why PLM?

Different teams within the company have different responsibilities within the product lifecycle, which makes it easier to lose control of a product. When the planning of a new product is finished it is up to production to produce that same product. This makes it very important to maintain control of all the information regarding the product, since other people, in the next stage of the product lifecycle, need to have all the information to be able to do their job right. For everything to go smoothly, it is important not to lose control. Sääksvuori and Immonen (2005) argue the importance of accessibility of everything that has been done with a product no matter where or when. In a world where globalisation is now a fact, companies become multinational and spread all over the world. This makes it more difficult to maintain control, which is why PLM is so important (Stark, 2011).

Globalisation also leads to increased competition, which pushes companies to produce products faster and at a lower cost (Stark, 2011). The ability to always have the necessary information at hand helps speed up the workflow and consequently lowers the cost of production. Furthermore,

the rules and regulations needed to be taken into account increase in number when a product is sold in more countries (Stark, 2011). It is therefore essential to know exactly what a product contains, which is kept track of with a Bill of material (BOM), a manufacturing list which specifies exactly what parts and materials a product consists of.

Stark argues that PLM is important because of all the problems which can occur throughout the product lifecycle (Stark, 2011). Each phase poses new possible problems and if one occurs, the presence of the PLM approach helps remedy the problem faster, and the logging of a particular problem reduces the risk of it happening again, and if it does, the solution is close at hand.

When we say "PLM system", we are referring to the digital solution created to aid and record the life and maintenance of products. These systems are often based on some sort of taxonomy so that it is possible to find different products and the parts, documents and other information associated with them.

3.6 Static code analysis

Static code analysis refers to the analysis of source code in its static state (i.e. not at runtime). The purpose of this analysis is to, by defining a set of rules, find deviations from the standards set up for the program. This is in order to either find security flaws in the code, or in general be able to track deviances from the optimal quality of the code. ("Static program analysis", 2015) These deviances are referred to as technical debt.

4 Results

In this section the results gathered during the study are presented. First there is an introduction to the interviewees and their work tasks, then the data from the interviews are shown in a diagram for a better overview and summarised in text, divided in subsections by topic. Lastly, the answers to the questions about issues management and practitioners' view on technical debt are summarised.

4.1 Introduction of interviewees

In order to gather the necessary data interviews have been conducted. They were carried out at two companies, GE Healthcare Life Sciences and Elekta. These companies are customers of Technia, using their PLM solution Enovia in their businesses. Within the GE Healthcare Life Sciences organisation, four people with varying roles and responsibilities were chosen. The fifth interviewee works at Elekta and was interviewed for their special insight into the matters discussed.

One interviewee works as staff quality engineer with product quality from a reliability perspective within product development, and with making robust products which are able to do what they are supposed to. This person is involved in the whole lifecycle, from specification to retirement.

Another is a systems architect. This person is involved both in system development and the development of chemical products. Their job is to work as a bridge between different areas of the product development and to make sure everything fits together in a good way and works well as one unit.

The third person is a design control leader in the quality assurance department. They make sure everything is documented properly and done correctly every step of the way. This person is involved with the development of chemicals in the early stages of the lifecycle, with design and development before the product or substance is produced and sold. This person is also involved in the phases from production until before retirement, which focuses on what happens if a change is made in a product and how that affects the customers who are in possession of the product if something goes wrong with it.

A representative from logistics has also been interviewed. This person is an inbound manager, responsible for incoming goods to the central warehouse. This is where they receive goods produced by GE around the world. This person's responsibility is to receive products and prepare them for further transport to its final destination, the customer. Another responsibility is warehouse management which consists of compression, optimisation of where to store products for easy access, making sure they are turned the right way around and stored in the right zones and that they are kept in correctly tempered areas (refrigerator, freezer, et cetera).

The fifth interviewee is a director of process improvement at Elekta and is responsible for managing general processes and ensuring they run smoothly.

The questions we asked concerned the problems they face in their everyday work, how they remedy them, what the timeframe for solving various issues may be, and possible extended consequences of issues. The questions can be found in full in appendix A.

4.2 Data synthesis

In order to get a comprehensive and clear view of the results from the interviews, the problems were entered into carefully comprised categories. A table with all the answers can be found in appendix B.

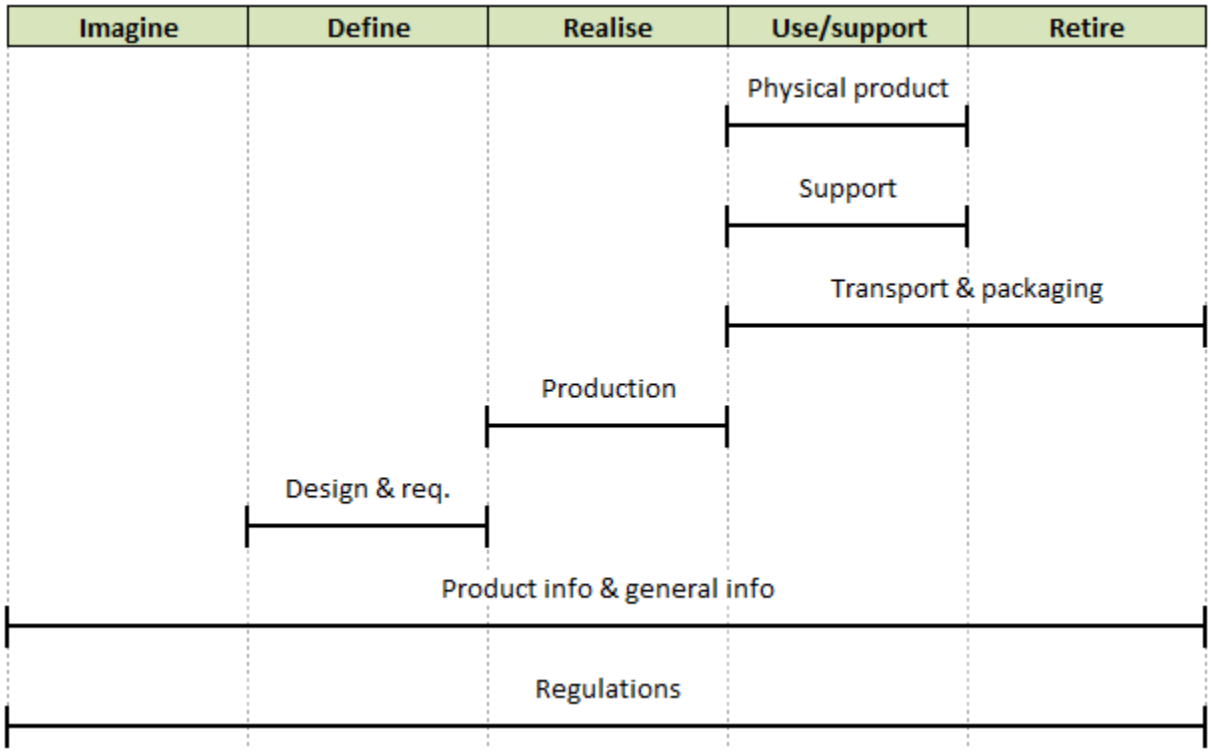

Figure 1. An overview of the results from the conducted interviews and during what part of the lifecycle different types of sources for poor quality show up.

The physical product refers to problems that occur with the actual finished product.
Support refers to problems arising from the disregarding of feedback from customers.
Transport & packaging problems are problems that occur during the transportation and packaging of products, usually within logistics.
Production (of the product) refers to errors within the production processes, when realising a design.
Design & requirements refers to non-optimal design, or not met requirements during the design process.

Product information (& other information) has to do with information around products not being correct or correctly entered into a system.
Regulations are external decisions made that affect products and processes.

Issues from these different groups are found in the following section.

4.3 Issues within the product lifecycle

Following below are tangible examples from each of the above mentioned categories, all gathered from the conducted interviews.

4.3.1 The physical product

Concerning hardware, problems that can arise are when a product malfunctions on the client site and a person from the service staff has to go there and investigate and, if possible, repair the product. Examples of problems are a leaking gasket due to over-polishing during production or the polishing process not being carried out with necessary carefulness, a valve that has been worn out prematurely, or a circuit board that ignites from overheating.

4.3.2 Design and requirements

Relating to design and requirements is an issue when a product fulfils all specification requirements but still does not work the way it is intended. For example, if a chemical substance is affected by a parameter during its production which hitherto has been unknown. Another source for problems cropping up during design can be an employee's unawareness of a component's function in the system while they are in charge of deciding how to conduct a redesign.

An example of an intentional design issue is when a suboptimal product is manufactured in order to speed up production and reach market faster, with the consequences being a necessary redesign to fulfil the specifications adequately.

4.3.3 Production (of the product)

Issues that are connected to the realisation process itself rather than a particular instance of a physical product are when a supplier for some reason no longer can deliver a wanted part, perhaps because the supplier has changed something in the production of that part which affects the product of which it is a part, has stopped producing the part altogether, or the supplier has gone out of business.

4.3.4 Regulations

An external source of the necessity to take action is regulatory demands on the organisation made by national, foreign or international authorities. There can be new regulations concerning what labels a product needs to have, what certificates need to be issued or the material concerning the product, such as the manual, is required in certain languages or with specific information.

4.3.5 Transport and packaging

Issues related to transport, logistics and packaging of products are when the warehouse is unable to receive inbound shipments, or the material is not placed properly on the shelves and thus creates problems when those goods are to be retrieved.

4.3.6 Product related information

Issues regarding the product information or the information in general are when a product received in a warehouse is missing necessary data, because someone earlier on in the chain has not entered the product related data sufficiently into the system. Other examples are when it is discovered that something is incorrect in a manual, which leads to having to rewrite and reprint that manual, or information about an incoming shipment is missing.

4.3.7 Support

Relating to support, an issue is when a small but severe problem affects very few customers and the reports of this problem is disregarded as a minuscule problem, when it actually is a large problem for a few customers. The issue is a lack of rigorous check on this feedback.

4.4 Issue management

The processes for managing issues that arise are slightly different depending on what area is concerned. In this section, there is an account of the issue management process from the interviewed practitioners and their suggestions for what can be improved.

4.4.1 What do the issue management processes look like?

The staff quality engineer working exclusively with hardware described the process of issue management as follows. If something goes wrong with a product at a client's site and the client calls to report it, a service case is created. This means a service repair person is sent out to remedy the issue and writes a report containing details about the service call, such as the parts replaced and the assumed cause of the problem. This report is inserted into a database stretching more than ten years back in time. Later, at a quality meeting, the report in the database is reviewed, it is identified what has happened, what the probable causes were and what can be

done about the issue. If the issue is serious and it is decided something has to be done about the product as such, a project team is put together to initiate redesign, make a prototype, test it and finally enter the updates on the product into the PLM system and later the updates are implemented in the concerned product.

The design control leader for quality assurance described that when an issue arises, the deviation is entered into a system, someone higher up is consulted, and if the deviance is accepted, or deemed to be a coincidence, nothing further is done about it. If the person in charge decides the issue needs to be resolved, a project team is put together who are responsible for solving the problem and making a preventive action so the issue will hopefully not arise again.

Following is the description of the issue management from the logistics operations leader's perspective. If something goes wrong, or a problem is noticed with a package, the problem is noted on a piece of paper, the box with the note on is put on a dedicated pallet for deviances, and the issue is registered in the workflow. The workflow process includes finding a receiver or "catcher" in a factory somewhere, whose responsibility it is to decide what to do about the problem. If all goes as planned the receiver responds. If not, they have to be reminded, which, if it goes on for a long enough time without a reply, results in the catching company or factory being black listed.

In the system architect's team they try to work on problems continuously. They employ agile methods with post-it notes on a board in order to have an overview of what all teams are working on at any given time. If a problem arises an investigation is set in motion in order to understand the cause and extent of the issue. A team is put together to prioritise the problems and decide which of them to act on during the following month. A task is given to a scrum team, where one person is probably the one with the most knowledge, but the whole team is backing that person up and contribute where they can. The team then work on a solution with necessary parties involved in order to solve this.

4.4.2 What can be done to improve them?

Only the systems architect had concrete suggestions for improvements that could be made regarding the issue management process.

Customer organisations have the opportunity to connect their system to a cloud service. This way they can get instant updates if something goes wrong with the system. This gives added value if they log relevant information about the system. What could be improved is if the company providing the cloud service can be alerted if a certain error message has cropped up several times. Perhaps some customers do not care about some error messages and just proceeds without noting it. What could improve is if the provider of the cloud service could collect all the logs in order to discover everything that might have gone wrong, even if the customer at the time did not bother to report it. This though is difficult to justify to the customer.

Minor issues with products may be fixed by a local instance of the company and never reach the head office. It would improve the issue management if reports on these minor issues were to reach the staff at the main office who are in control of the product lines.

Automated testing would be positive to have to a larger extent. These tests would have to be as effective as possible and at the same time taking up as few man hours as possible.

4.5 Meaning of technical debt to practitioners

After, if necessary, explaining the concept of technical debt to the practitioner, they were asked explicitly what technical debt means to them in their respective areas and below follow their replies.

Within staff quality leadership, the technical debt concept was associated with environmental requirements compliance that affects the design phase. Instances of a product already on the market can sometimes be allowed to violate new requirements but for the future the product needs to be redesigned.

To a project manager and a system architect with a background in both chemicals and software development, technical debt was described as being better applicable to chemicals than hardware. The parallel between spaghetti code (code that is "tangled" or has an unnecessary high level of complexity) and development in general was pointed out in the sense that adding modules without knowing how they work together is possible within both, and that later additions might thus be difficult to implement and the result become error-prone. Another matter suggested as possible to classify as technical debt is the fact that a particular employee or team has the necessary knowledge to complete a process, and the nervosity caused by the thought of transferring the process to another site or team indicates the unconscious understanding of that debt. A source of uneasiness apart from an employee or team could also be that all the parameters affecting production might not be known. This is applicable especially during the development of chemicals, as chemistry is more complex in that it can be affected by the weather and humidity, for example, to a much larger extent than production of hardware. Today, there is no formal way of discussing this nervosity and lack of potentially crucial knowledge about processes.

From a design control leader's point of view, there is in hardware development a perceived form of technical debt when a product deviates from the required level of producibility. During development, a decision might be made to make a construction solution such as putting a circuit board in a certain position; upon reaching production, the design team is alerted that the placement of the circuit board renders manufacturing cumbersome and this is then the technical debt being discovered. The project manager and system architect also mentioned designing for producibility and reliability and how not doing so shows up later during the lifecycle in ways possible to describe as technical debt. This is a focus and way of thinking which is absent in today's processes. An example of poor design for producibility is having an excessive amount of screws for assembling a product, resulting in a great deal of work for production and thus a low

level of producibility. Furthermore, it was expressed that quantifying the level of producibility and reliability of a product and present that as a value easy to communicate would be beneficial for classifying various modules in the system according to potential problems, ergo technical debt.

A source of issues potentially suitable to label as technical debt within hardware was also suggested by the project manager/system architect. A module from the organisation's product portfolio might have a high level of erratic malfunctioning but is still used in a minor line of products, because the necessary service or maintenance of those products is deemed manageable. There is also a choice when developing new products to use modules from the already existing product portfolio, and one aspect to be considered when making that choice is whether the level of erratic malfunctioning exceeds what is favourable in the new product.

In the response from the interview with the employee in the logistics department they emphasised that what they would describe as technical debt would largely be noticeable in their PLM system in the shape of incomplete information associated with articles. Ten percent of the articles are missing data such as weight and measurements and it is very time consuming to rectify. Moreover, if classification of items to be sent is not considered and entered into the system, goods that must be transported according to certain regulations because they are classified as dangerous become a mean to break regulations. Another source of non-software technical debt was suggested in the decision concerning how to package a product. For example, how items are organised in a box or how much of a substance can be put into one parcel without exceeding a limit making it classify as dangerous goods. In addition, there are quite physical aspects that are similar to the technical debt concept. When a member of staff receives a delivery and goes to put the packages on the shelves, they might not bother to align the boxes properly, neglect to cut the tape around it despite it being the proper procedure, or do not ensure that the labels are turned outwards for easy identification. This becomes a time-consuming task when trying to retrieve the packages or identify them, first having to rearrange the boxes.

A suggested example of technical debt from a process improvement perspective is closely related to engineering change requests. A design issue is when there exists a relatively mature product in a company's product portfolio and it has a backlog of non-conformances, engineering change requests and engineering change orders. This backlog might build up to unmanageable levels, with the possible consequence being that engineers in charge of a redesign have a difficult time knowing what impact various solutions have and if they will clash with other change orders in the backlog. It takes an increasing amount of time with each change to evaluate what impact a change has on other change orders in the backlog. The debt in this case is the backlog of engineering change requests.

5 Analysis and discussion

In this part we will first discuss the similarities between issues in software development and development of other products, and then suggest what support there is for a meaningful adoption of the technical debt metaphor outside software development. Secondly, a method and what tools would be necessary for quantifying technical debt will be presented. Thirdly, we will analyse our results and with support from current system development theories offer advice on the matter of developing a feature in a PLM system which tracks technical debt and presents it to the user.

5.1 Mapping of issues to technical debt types

If the categories derived by Li, Avgeriou and Liang (2015) for the various types of software technical debt are studied, parallels can be drawn to the categories used in the data synthesis for the technical debt in areas other than software. A broader definition of technical debt was chosen because narrowing it down to the initial definition within software development means we are left with technical debt only referring to complex or otherwise suboptimal code. This definition would not have been useful because all products in general do not consist of code. Therefore, employing a more general definition allows for a meaningful discussion of the collected data while also drawing parallels to the metaphor. Below, these parallels are discussed in more detail.

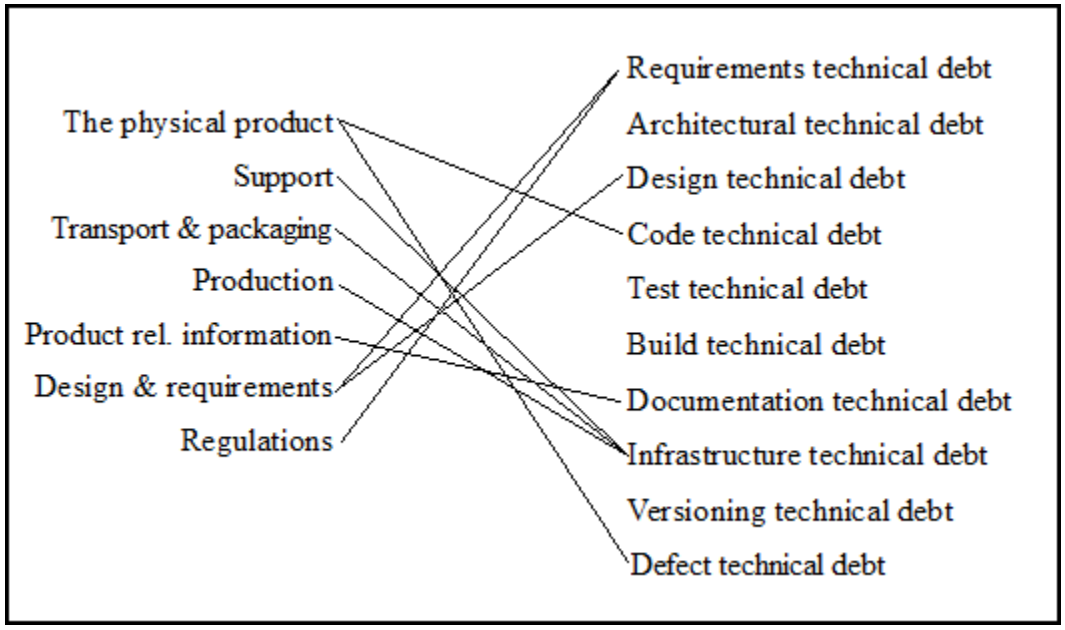

Figure 2. On the left: our categories, on the right: the types of technical debt as described by Li, Avgeriou and Liang (2015), and the connections between them.

Support, transport and packaging, and production are all infrastructural problems since they have to do with how development-related processes are set up, as described by Li, Avgeriou and Liang (2015). The physical product is comparable to the code technical debt as both categories concern the actual output, the product, of the work put in. It can also be viewed as a defect which

makes it a special case and will be discussed in more detail later on. Product related information is closely related to documentation technical debt since they both refer to peripheral information which is incomplete or suboptimal in some way. In our categories, we have merged design and requirements into one, so they relate to two categories in Li, Avgeriou and Liang's model: requirement technical debt and design technical debt. It was not considered meaningful to separate the two, since the empirical data was quite limited. When it comes to regulations, we find that this can be regarded as a requirement, as regulations are specifications of what is required, set up by external parties. Regulations are certain kinds of requirements, which is why we have made a separate category for it, but it can also be argued to be reflected in the requirement technical debt category Li, Avgeriou and Liang (2015) have specified.

The categories we sorted the found problems into are based on the interviews we have conducted, which are not enough to get an encompassing picture of all types of problems that can occur within the product lifecycle. If more interviews were conducted and a more holistic picture was created, more categories could possibly be added, perhaps categories that match the ones in Li, Avgeriou and Liang (2015) more closely.

The category *the physical product* is not technically a source of technical debt. In that case, the problems that fall within that category are results of technical debt that has occurred at an earlier stage in the product life cycle. This is in line with what Li, Avgeriou and Liang (2015) states about defects not being technical debt. The issues that falls within this category needs to be further examined and it needs to be investigated what the causes of them might be. The cause of the problem cropping up is most likely when the technical debt emerged. If it is a random error which has no discernible cause, it cannot be classified as technical debt or even a symptom of technical debt and can and should not be measured with the aid of system software.

5.2 Technical debt outside software development

By looking at the material from the interviews and comparing the different issues to the descriptions of the numerous notions identified by Li, Avgeriou and Liang (2015), it is possible to find many parallels.

Interest is “the extra effort needed to modify the part of the software system that contains [technical debt]” (Li, Avgeriou & Liang, 2015), in other words, when dealing with a software product, the interest is the extra effort necessary to put in because there is a debt in the product’s history that has not been paid back.

This kind of interest is also possible to find in issues uncovered in the interviews. One issue was that sometimes there is an intentional step away from what is known to be the better material or design for ensuring quality in a product, but the decision is made to use a suboptimal material or design because it will shorten the time to reach the market. This means that this information, that the quality of the product is less than it ought to be and that a redesign of some form is to take place, has to be kept available, or there might be consequences because this update is forgotten. These consequences could be having to send out many more members of the service staff to

change parts of a product than would have been necessary had the redesign happened soon after the first shortcut was decided upon.

In the realisation phase, during production, there are also decisions that can give rise to debt and interest. As described in one of the interviews, when receiving parts at the warehouse that are to be used in a product, the package can be put in a way which makes it difficult to retrieve from the shelf or to reach the items inside. This might be done because of negligent staff or because they have to hurry, resulting in more work every time the package is to be reached because it has to be repositioned. The debt is the time it takes to put the package in its proper position, and the interest is the extra time it takes to fulfil the task related to the package during the time it is incorrectly placed.

Filling in missing data concerning products in a system also matches the concept of interest. A member of staff receiving a package but due to time constraints does not submit all required data in the system would perhaps need less time to do so right away than would be necessary in the future. Unless that person manages to keep the data in their mind, going to check it again would require more time. Additional time could even be needed if another member of staff who does not know as much about the case is to rectify the omission.

The concept of principal is described as "the estimated cost of resolving a given type of [technical debt]" (Li, Avgeriou & Liang, 2015) and has, not surprisingly, many applications within the issues facing the interviewees. When product data is not entered into the system either during the define phase or later during realisation phase when packages are received at the warehouse, the time taken to later find the information that was not put into the system and enter it would be the principal.

In the interviews it was expressed how the issue of a supplier of a part going out of business, or being unable to continue to deliver the same product part to the organisation, triggers a rework process for the organisation before they are able to continue manufacturing the product. The principal, the cost of resolving this technical debt, is then the cost for the organisation to find a new supplier, conduct an audit if that is required by the internal policies, and carry out other activities associated with setting up a deal with a new supplier.

Technical debt is "a type of risk for a software project, since [technical debt] can eventually hurt the health of the project if the [debt] is left unresolved" and the risk of certain actions are found in the gathered data.

There is a risk of decisions being made on the wrong premises because the amount of a chemical substance which becomes classified as dangerous goods when shipped in certain amounts is exceeded. The risk of suffering repercussions for not following regulations arises when the amount of substance that is to be shipped is not entered into the system. Another example of a technical debt-induced risk is the aforementioned risk of having to dispatch service staff in larger numbers than expected to serve a product that was released to market with a suboptimal part or design.

Li, Avgeriou and Liang (2015) also found that a notion dealt with in technical debt-related research is interest probability, and the definition they give was originally formulated by Brown et al.: "the probability that a particular type of [technical debt] will in fact have visible consequences" (Brown et al., 2010). It is not entirely sure that missing information in a system will cause problems for the business. For example, there might be no need to use or retrieve the information later on, consequently there will be no need to pay back the debt, as in, there will never be necessary to find out the missing information and associate it with the concerned product in the system. The likelihood of the choice of material having consequences later is also not certain. There is a possibility that the chosen inferior material, which is more brittle, still does not break before a part made of the better material would have.

As these examples illustrate, the notions related to technical debt are comprehensible even when the product containing technical debt is not a software product. The concepts of interest, principal, risk, interest probability and others described in the literature (Li, Avgeriou & Liang, 2015) are present within issues such as not entering information into the system, the influx of engineering change orders stacking up, creating a backlog of increasing complexity, and taking short cuts within logistics. This supports the idea that the technical debt metaphor is applicable to development of products other than software.

5.3 Requirements for a model of quantification

As can be found in the literature, there are ways to measure technical debt in software systems, for example, by using the empirical model presented by Nugroho, Visser and Kuipers (2011). Some characteristics of this model, tested for accuracy in a case study, seem good to draw on when developing a model for non-software quality, for example, because it has been empirically tested and shown to work. There are many things to take into consideration when developing a model of quantification for technical debt and interest outside software. Similarly to the model suggested by Nugroho, Visser and Kuipers (2011), there needs to be a foundation for what is considered good quality. Either existing models for quality assessment can be used, and for types of quality where there at present are no such models, we suggest future research is done to facilitate developing models such as the SIG/TÜViT model (a method for assessing quality, used by the organisation TÜViT to certify software).

Furthermore, the empirical model has factors in its formulas allowing for the model to be adjusted to different kinds of programming languages, which is important (Nugroho, Visser & Kuipers, 2011). Similarly, the model for quantifying other kinds of debt ought to have factors that can be changed, making it adaptable and able to match experience of the team dealing with the debt, the type of product or work that is measured, interest (which will most likely look different because, as Nugroho et al. points out, it depends on historical maintenance data) and the costs of different actions for a particular business. It is also possible that different kinds of quality on the quality spectrum call for adjustments as well. All these other factors mentioned above are likely to come in to play in addition to other ones and they are likely to best be uncovered by future research.

This study has uncovered what further research is necessary to enable the creation of a model for quantifying technical debt in other contexts.

5.4 Types of technical debt

The types suggested by Li, Avgeriou and Liang (2015) is a good foundation for a framework to build the conversation on technical debt on, because having a term encompassing issues of widely different nature is not useful. A term that means too many things is arguably not helpful, as it does not clear up what the communication is about. To take it to somewhat of an extreme: too unspecific terms might lead to one side thinking the conversation is about one thing, and the other about something else. Also, separating the technical debt concept into types allows organisations to prioritise types depending on their business.

When the problem of technical debt is divided into sub-categories it is arguably easier to become aware of the size and the variousness of the problem of technical debt. The causes of technical debt range from requirements not being met to documentation on the finished product not being complete in software and customers being unsatisfied, and in a similar manner they range from externally decided regulations to packaging mistakes in non-software products and processes. If merely the concept of technical debt was applied to the whole range of problems, it would be difficult to grasp. If the problems are however divided into categories based on where and why they occur, the actual problem would become more specific and thus easier to grasp.

Depending on where the problem lies, and more specifically, where the problem arises, different people within the product lifecycle have the responsibility. This division of technical debt into sub-categories makes it easier to allocate responsibility for making sure the debt is correctly managed or even ensure it does not arise in the first place.

5.5 Evaluation of practitioners' ideas of technical debt

In the interview with a staff quality engineer's perspective, the technical debt concept was associated with environmental requirements compliance which affects the design phase. Instances of a product already on the market can sometimes be allowed to violate new requirements but for the future the product needs to be redesigned. The work related to new environmental requirements can be thought of as debt since the new regulation might trigger a redesign of a released product. In this case the released product no longer meets the requirements, and as stated earlier, and also by Li, Avgeriou and Liang, the discrepancy between the design of a product and the requirements for it can in both software and non-software be considered debt. However, if the new regulations do not trigger a reconstruction of released products, it does not turn into debt. What happens is that the requirements for future products change, which is only to be considered for what it is: new requirements, and not debt.

The parallel made in the interviews between spaghetti code and the practise of adding modules without proper evaluation, and the effect on the overall complexity and interaction between

different modules, is deemed appropriate. This phenomenon can be described as falling under the debt metaphor, because one subtype of technical debt is design being too rushed due to shortcuts (Li, Avgeriou and Liang, 2015), thus causing troubles when later additions become difficult to implement because the additions' effects on the overall design are difficult to scope.

Two points raised in the interviews as potential technical debt were when crucial information is known by only one particular person or a group of people, and when there are unknown factors in the development of chemicals. The information issue arguably falls under documentation technical debt, as one of its subtypes is insufficient documentation. The flaw in the process is that the knowledge is not presented in the documentation for other employees to find. It was mentioned how there is no way to put this nervousness of missing information into words; therefore, being able to classify it as documentation-related technical debt, and be able to draw attention to the risk and interest it composes, is probably useful.

The issues described in logistics and the ideas for what technical debt means in that context align. Boxes placed incorrectly on shelves and missing package information create problems that can have an interest if left unfixed because every time items from a box need to be retrieved or a misplaced box has to be found it takes extra time compared to if it was done correctly, creating an interest.

5.6 Development of debt-tracking feature in PLM-system

After our discussion of technical debt and, by argument, establishing that technical debt is an issue within areas outside software development, it is meaningful to give some pointers on the matter of developing a debt-tracking feature in a PLM system. Technical debt can arise during any phase of the product lifecycle, and since PLM systems are a common way to manage a product, and also being already a good method for managing problems (Stark, 2011), incorporating the new feature in such a system is regarded as feasible and favourable. The "organisation" referred to below is the potential user of a PLM solution with an implemented monitoring feature.

Firstly, the organisation should be able to set rules for what is allowed and what is not within the system, just like a static code analyser (see section 3.8). It would be useful if the program could crawl through the PLM system just like static code analysers and alert users when rules set up by the company are violated. For example, it should be able to track where there is missing information or files, or where engineering change requests have started to accumulate, as these are examples issues arising and functionality asked for by the interviewees, and likely to be things the organisation would like to track.

Secondly, as Nugroho, Visser and Kuipers (2011) say, understanding the impact of technical debt on the business, and being able to monitor it, aids software developing businesses in making better decisions about their investments. Consequently, functionality to facilitate calculation of amount of debt would be beneficial for other product development to estimate the cost of the debt and make business decisions based on that. Furthermore, similar to factors in the empirical

model used for software technical debt (Nugroho, Visser & Kuipers, 2011), there needs to be a way for the system to know what magnitude different factors have depending on what issues the program discovers. One way to enable such behaviour is functionality for the user or system administrator to enter these factors and map them to costs.

In the practitioners' respective field, problems are dealt with in different ways, using various systems and processes for administration. However, the quality leader and design control leader both described how a system is involved when some sort of issue record is made. In the new technical debt monitoring system, a feature to track the number of different kinds of issues associated with certain parts or products would be useful to uncover trends, which could be symptoms of technical debt and should therefore be included if possible. Without it, sources of technical debt could potentially go unnoticed. The system architect hoped to see a way to sync a cloud solution for reporting issues with a resource within their own company. The usefulness and feasibility of integrating such functionality into the new debt tracking feature needs further thought.

Since it is important to ensure a good foundation as early as possible in the product lifecycle to attain good quality (Bergman & Klefsjö, 2001, p. 57), functionality useful for the imagine phase and design phase of the lifecycle should be prioritised. It is arguably less work involved in changing a source of debt in an early design than in having to, for example, both redesign the product and dispatch service staff to fix the physical product on clients' sites, or deal with customer complaints. Therefore, it is beneficial to monitor sources of debt during those phases, so as to prevent the debt from having time to accrue.

6 Conclusion and future research

Through a joint analysis of the interviews, the practitioners' perception of the use for the metaphor and literature the conclusion was drawn that technical debt can usefully be applied to other kinds of product development outside the software development sphere. In the same way the technical debt concept has made research on how to quantify and manage quality issues within software development a point of interest in both the academic and practitioner sphere (Li, Avgeriou & Liang, 2015), it is possible that, by extending the metaphor, more eyes will be opened to debt issues in the production of other products. The more research that is done on the topic, the more information will be available to manage quality issues before they become quality costs. Our hope is that this initial study will prove useful for organisations delivering other sorts of products than software and that it will trigger further research, which will be necessary before a formal requirement specification for a technical debt tracking system can be formulated.

The knowledge contribution provided by this study is a framework for technical debt and a collection of functionality requirements based on empirical and theoretical data. The framework consists of two main constituents: categories for classifying types of debt, and a first "sketch" for a model to measure technical debt in products other than software. There is plenty of room to further look into how this model should look, what factors are necessary to take into consideration, and what practical ways to assess quality as a starting point for this model would be.

As for the functionality requirements, the analysis shows that a system that will work as a feature in a PLM system tracking technical debt should have functionality for implementing models for debt quantification, allow the organisation to set up rules, and be able to crawl through the product taxonomy in the PLM system, detecting deviances from those rules. Because no research on the topic has been done prior to this study, developing a concrete specification requires further research on the topics of how to quantify different types of debts, what factors affect the severity and solution of accrued debt, and the formulating and testing of quantifying models. This will help in attaining different types of quality found on the quality continuum and prevent debt from unknowingly accruing throughout the lifecycle of the product.

References

Bergman, B., & Klefsjö, B. (2001). Kvalitet från behov till användning. Lund: Studentlitteratur.

Bergman, B., & Klefsjö, B. (2002). Kvalitet i alla led. Lund: Studentlitteratur.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., ... & Zazworka, N. (2010). Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research* (pp. 47-52). ACM. doi:10.1145/1882362.1882373

Cunningham, W. (1992). The WyCash portfolio management system. *Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum) - OOPSLA '92* (pp. 29-30). New York, New York, USA. Doi: 10.1145/157709.157715

Davis, K., Ledbetter, W. B., & Burati Jr, J. L. (1989). *Measuring design and construction quality costs. Journal of Construction Engineering and Management, 115*(3), 385-400.

Flynn, B. B., Schroeder, R. G., & Sakakibara, S. (1994). A framework for quality management research and an associated measurement instrument. *Journal of Operations management, 11*(4), 339-366. doi:10.1016/S0272-6963(97)90004-8.

Heravi, G., & Jafari, A. (2014). Cost of Quality Evaluation in Mass-Housing Projects in Developing Countries. *Journal of Construction Engineering and Management, 140*(5).

Juran, J. M., & Blanton Godfrey, A. (1998). Juran's quality handbook (5th ed.). New York: McGraw Hill.

Juran, J. M., & De Feo, J. A. (2010). Juran's quality handbook: The complete guide to performance excellence (6th ed.). New York: McGraw Hill.

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, *101*, 193-220. doi:10.1016/j.jss.2014.12.027

Nugroho, A., Visser, J., & Kuipers, T. (2011). An empirical model of technical debt and interest. In *Proceedings of the 2nd Workshop on Managing Technical Debt* (pp. 1-8). ACM. 10.1145/1985362.1985364

Oates, B. J. (2006). Researching information systems and computing. London: SAGE

Seawright, K. W., & Young, S. T. (1996). A quality definition continuum. *Interfaces, 26*(3), 107-113. doi:10.1287/inte.26.3.107

Stark, J. (2011). Product lifecycle management: 21st century paradigm for product realisation. New York; London: Springer.

Static program analysis. (n.d.) In *Wikipedia.* Retrieved May 15, 2015, from http://en.wikipedia.org/wiki/Static_program_analysis

Sääksvuori, A., & Immonen, A. (2005). Product lifecycle management. Berlin; London: Springer. doi:10.1007/b138258

Tisell, J., & AMU-gruppen. (1991). Att arbeta med kvalitet. Ronneby; Stockholm: AMU-gruppen.

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. *Journal of Systems and Software*, *86*(6), 1498-1516. Doi:10.1016/j.jss.2012.12.052

Västra götalandsregionen: centrum för verksamhetsutveckling. (2004). *Kvalitetsbristkostnader* Göteborg: Centrum för verksamhetsutveckling. Hämtad 15 april, 2015, från http://www.vgregion.se/upload/CVU/Kvalitetsbristkostnader.pdf

Appendix A - Interview questions

Goal with the interviews:
Find out:

- What are the sources of quality costs?
- What are the sources of technical debt?
- How does the management of these issues work today and how would you like it to work?

Questions:

Please give us a (brief) description of your background as an employee and your role in the product development.

What part of the product lifecycle would you say you are involved with?

What kinds of quality defects or problems within the quality aspect can arise within your work?
When we know what problems can arise we can:
Focus on the various problems and find out what consequences they have and the remedies.

How do these problems arise?
Will probably already be answered.
When we know how problems arise we can:
Possibly start to come up with ways to track the origin of these issues.

Examples?
Will probably already be answered. This question can signify "How do you mean" if there is something we have not understood.

How do you remedy them?
Process concerning how they remedy problems. E.g. make a change in a system for all products containing a part which has been retired.
When we know how they remedy issues we can:
We will know what "currency" the debt will be paid back in.

How long does it take to remedy different types of problems or defects?
When we know how long it takes we can:
Start quantifying the debt in the unit of time. It becomes measurable.

What consequences can the defect give rise to? That is, in addition to the problem itself. Does it affect … other things?
When we know the consequences we can:
Quantify the debt further. What is the debt beyond the cost of remedying the cause. I.e. the interest.

For how long can the problem/issue/defect remain until it gives rise to any (larger) consequences?
When does the interest on the debt start accumulating? If there is a problem that does not affect anything, it constitutes no catastrophe payment-wise. Probably different for different problems.
When we know this we can:
Possibly grade different kinds of debt.

Describe the process from discovering a defect/problem/issue until it starts being remedied.
When we know this we can:
Consider if this is good or bad, if there are any flaws. If there is something that is handled in a particularly good manner and which can be used in a future possible prototype of a system implementation.

How does the managing of problems work today? (not just good or bad, describe)
When we know this we can:
Argue in favour of potential positive effects of changing it.

How would you like it to work? Anything needs changing? Anything that causes you headaches? Anything that works really well?
When we know this we can:
Start thinking about what a possible prototype could look like.

Introduce the concept of technical debt and how we are considering using it.
What do you think about that?

Appendix B - Interview response table

problem	cause	remedy	consequences	time
Product malfunctions on client site		Dispatch service people to repair		
Gasket leakage	Polish gasket surfaces so they become too smooth			
	Different finish on the gasket due to either not being careful with the polishing fluid when polishing surfaces, or using the same polishing fluid for a long time			
Circuit boards stop working, a component combusts	Component is overheated		Danger in case of proximity to e.g. flammable fumes	
Fluctuation in production processes or installation processes				
			Jeopardising staff or property	As soon as possible
??Geometry on an item needs changing	??Geometry on a machine made thing is not ideal	Minor reconstruction: geometry is changed		(The scope of the reconstruction determines the time it takes to remedy) Simpler reconstruction, could take a couple of weeks
Valve are worn out prematurely and leak	Chemical corrodes the material	Major reconstruction: change the material of the valve.	Harmful/poisonous/corrosive substance leakage	Takes longer since people who know about materials have to be contacted.

Problem	Cause	remedy	consequences	time
No longer able to buy from supplier, could be a circuit board	One of the components needed is discontinued, or the supplier has gone out of business	Have to change supplier, and if they no longer exist, have to reconstruct the e.g. circuit board		Simplest type if a component or supplier is no more. If it's a standard part at a supplier already approved by the company it can be quick. often more difficult. Have to contact new supplier, make sure they are ethically ok, which can take months.
No longer able to buy a circuit board from supplier	Supplier stopped selling that component	Change supplier		If other supplier already approved by the company can supply with the same thing, it can be quick
	The supplier has gone out of business			If not as above, have to contact new supplier, make sure they are ethically ok, which can take months
	One of the components is discontinued	reconstruction of the circuit board		
??Problems after reconstruction	The employee in charge of changing a component is unaware of its function in the system			
Something stops working, although the product still fulfils all of the specifications	E.g. the properties of a chemical substance is affected by a parameter hitherto unknown			

Intentional issue: exterior shell of a product, make a not so long-lived one very quickly for the first ten products just to get it to market /Slightly suboptimal quality product (due to intentional design decision to speed up production to market time)	shortcut to get it to market	Redesign it when it causes problems later on.		
There now needs to be a label on an item	Regulatory issues: foreign authority have decided on new rules for import/export			
A certificate needs to be issued				
Something more has to be written in the manual.				
The manual has to be translated into more languages.				
			Sales disruption for company and for customers	
			Company loses reputation/goodwill (due to e.g. the above mentioned)	
			A relatively rare but serious problem for a few looks like a teensy problem and is disregarded. Creates a large problem for a few.	

Something wrong with a certain batch of chemicals.		If a problem has occurred enough times, it has to be solved.		Can take several people several years, and they might not even find a solution.
		throw it away and make new batch		
		investigation first, prioritisation, what problems to act on, work on solution, formal process for adding into the system		
Problem	**Cause**	**Remedy**	**consequences**	**time**
Not able to receive something	Supplier goes out of business	Find new supplier	If not able to deliver to customer, their production stops,	
			Costs a lot of money,	
			It can affect patients	
Some medical substance does not work the same way as a previous one.	They have the same substance but the previous one was an isomer			
Changed particle size	Change of raw material	adjust the sifting process	Costs more money	
A microbiological problem, bacteria has entered into something, where it's not supposed to be		Find out whose "fault" it is.		
Must adjust product to fit new demands	New regulations or standards (from national or foreign organisations/customers/authorities)			
Something is incorrect in the		re-write and re-print		takes time

manual				
Question from customer		Write and send reply		one week
something went wrong in production		Make a Corrective and Preventive Action (CAPA) in order to find the deviance		
		start project		recurring problems must be remedied faster (trending)
problem	**Cause**	**remedy**	**consequences**	**time**
Cannot receive a product properly	something does not live up to the expectation			can take a couple of days
	A release on a purchase order reached zero but there are still goods left to store.			
missing data, measurements and weight	someone has not entered a product properly into magic(due to laziness possibly), not correct, missing measurements and weight.			
Expiration date passed				
	move the wrong product from the warehouse, wrong number of products or damaged product.			
	something is damaged, missing PO, wrong serial number, no SSO			
Send this 2.5 million SEK	Partially broken pallet	ask someone, they don't decide on		if a decision-maker has to approve

product or not?		logistics.		something it can take a long time
	Shock watch has been triggered			
	something is misplaced in the warehouse	Find solution inbound		
Pick the wrong number of items from a box	not ideally packed goods			
The people moving/packing/unpacking in the warehouse have a hard time doing their job	someone has not turned the label outwards, placed things carelessly in the warehouse, have not cut open the box			
impossible to retrieve items from out of a box	someone has put the box in the wrong place or not the right way around			
	Objects have been put in a box without proper separation or labelling			
				1day/week up to a year depending on...
something received is larger than expected			Preventing other products from arriving.	
			dangerous situation for workers	
	something not correct in the system		large orders have to be unpacked manually in the system	takes enormous amounts of time
		send the right thing		

		send new batch		
		return or dispose of the returned good, order/request new product if something wrong		
damaged box		depends on what country the company is shipping to. Some countries are very strict, have to repack in new box. Others are less strict, sufficient to tape over a minor damage.		