UNDERSTANDING TECHNICAL DEBT IN COMPLEX SYSTEM DEVELOPMENT PROJECTS: A VISUALIZATION MODEL

**A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY**

MURAT CAN GÜLER

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS**

JANUARY 2023

Approval of the thesis:

UNDERSTANDING TECHNICAL DEBT IN COMPLEX SYSTEM DEVELOPMENT PROJECTS: A VISUALIZATION MODEL

submitted by **Murat Can GÜLER** in partial fulfillment of the requirements for the degree of **Master of Science** in **Information Systems, Middle East Technical University** by,

24.01.2023

ABSTRACT

Understanding Technical Debt in Complex System Development Projects:
A Visualization Model

Murat Can GÜLER,
Master of Science, Department of Information Systems,
Supervisor: Asst. Prof. Dr. Özden ÖZCAN TOP

January 2023, 121 pages

A complex system combines multiple articles with specific requirements and essential functions, communicating with each other and the environment. Development stages of complex systems intertwine with each other due to high-level interactions and dependencies among different components of systems. Due to these factors, developing complex systems usually requires unique approaches to solving managerial and technical problems and adopting new technologies. In a complex systems development project, required budget can exponentially grow with the complexity level of a product and inappropriate/bad decisions can be made to deal with tight delivery schedules. Due to the complexity of the specifications, a compulsory module of the system often requires mass production on the hardware level. System engineers sometimes need to foresee all system requirements, even without a rapid prototype. Even a minor change in the system requirements may affect hardware design, developed software modules, and test cases. Mass production decisions could be made during the development stages. These issues may cause significant expenses in total product lifecycle or postponing delivery schedules. The high complexity of developing large systems makes the technical debt concept even more critical. This thesis aims to reveal the factors and decisions that cause technical debt in complex systems development (CSD). For this purpose, existing categories in the literature were determined. The thesis seeks to answer how sufficiently the available categories determine technical debt in complex system development projects. A qualitative research was performed on eight cases to identify the answers to this question. Afterwards, a model was created to visualize TD in complex system development projects (TDVM) XXX. . With the guidance of the TDVM model, new categories were proposed in the literature. These new categories were evaluated by experts in the field and subjected to qualitative analysis. As a result of the analysis of the cases on the model, six improvements regarding technical debt management in complex system projects were shared. In addition to the software developers, the impact of technical debt on other project stakeholders has been revealed.

Keywords: Technical Debt, Complex System Development, Technical Debt Visualization Model

ÖZ

KOMPLEKS SİSTEM GELİŞTİRME PROJELERİNDE TEKNİK BORCU ANLAMAK: GÖRSELLEŞTİRME MODELİ

Murat Can GÜLER,
Yüksek Lisans Tezi, Bilişim Sistemleri,
Tez Yöneticisi: Asst. Prof. Özden ÖZCAN TOP

Ocak 2023, 121 sayfa

Karmaşık bir sistem, birbiriyle ve çevreyle iletişim kuran, belirli gereksinimlere ve temel işlevlere sahip çok sayıda ürünü ve alt sistemi birleştirir. Karmaşık sistemlerin geliştirme aşamaları, sistemlerin farklı bileşenleri arasındaki etkileşimler ve bağımlılıklar nedeniyle birbiriyle iç içe geçer. Bu faktörler nedeniyle, kompleks sistemleri geliştirmek yönetimsel ve teknik sorunları çözmeyi ve yeni teknolojileri benimserken inovatif yaklaşımları gerektirir. Kompleks sistem geliştirme projelerinde; gerekli bütçe, sistemin karmaşıklığı ve teslimat takvimleriyle başa çıkmak için alınan uygunsuz/hatalı kararlarla katlanarak artar. Takvim baskısı ve tanımlı isterlerin karmaşıklığından dolayı, sistemin kritik bileşenleri dahi prototip aşamasındayken seri üretim paralelden başlatılabilir. Sistem mühendislerinin prototip bir ürün olmasa dahi sistem gereksinimlerini öngörmeleri beklenir. Sistem gereksinimlerindeki küçük bir değişiklik bile donanım tasarımını; geliştirilen yazılım modüllerini ve test senaryolarını etkileyebilir. Ürün yaşam döngüsünde bu hatalar; milyonlarca dolarlık kayba, teslimat programlarının ertelenmesine neden olabilir. Seri üretime yönelik kararlar bu tip kompleks sistemlerde, geliştirme aşamalarında alınmak zorunda kalınır. Bu tür kararlar genellikle teknik borç kapsamında ele alınabilir. Teknik borç, sistemin hızlı teslim gereksinimi nedeniyle sistemin genel kalitesini ve sürdürülebilirliğini etkileyen mühendislik kararlarıdır. Bu tür sistemleri geliştirmenin karmaşıklığı, teknik borç kavramını daha da kritik hale getirir. Bu tez, karmaşık sistem geliştirmede teknik borca neden olan faktörleri ve kararları ortaya çıkarmayı ve görünür kılmayı amaçlamaktadır. Bu bulgulara yönelik kompleks sistem geliştirme sürecinde teknik borcun üstesinden gelmek için görselleştirme modeli geliştirilmiştir. Bu amaçla öncelikle literatürde var olan teknik borç kategorileri belirlenmiştir. Araştırma sorularından ilki, Teknik borcun kompleks projelerde belirlenmesine mevut kategorilerin katkısını inceledi. Sonuca ulaşmak için sekiz vaka üzerinde niteliksel analiz yapılmıştır. Seçilen vakaların karmaşık yapıda projelerden olduğu literatür rehberliğinde doğrulanmıştır. Karmaşık sistem geliştirme projelerinde TD'leri görselleştirmek için oluşturulan model, ilgili vakalara uygulandı. TDVM modelinin çıktıları analiz edilerek literature yeni kategoriler önerildi. Ayrıca bu yeni kategoriler, alanında uzman kişiler tarafından değerlendirilerek nitel analize tabi tutulmuştur. Model üzerinde vakaların analizi sonucunda, karmaşık sistem projelerinde teknik borç kavramının yönetimine ilişkin altı iyileştirme paylaşılmıştır. Yazılım geliştiricilere ek olarak, teknik borcun diğer proje paydaşları üzerindeki etkisi bu araştırmayla ortaya çıkarılmayı hedeflendi.

Anahtar Kelimeler: Teknik borç, Karmaşık sistem geliştirme yöntemleri, Teknik borç görselleştirme modeli

To My Beloved Family

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A complex system combines multiple articles with specific requirements and essential functions, communicating with each other and the environment. The consequences and results of decisions concerning system architecture and technical design may be more complicated than predicted, depending on the system's complexity. A minor component modification could cause multiple changes between the associated modules (Austin Page et al., 2019). Complex systems require unique approaches to solve managerial and technical problems. A high level of interactions between modules and dependencies between disciplines make a program complex. Complex projects require huge budgets due to the number of components, modules, and subsystems they include. The development cycle includes phases such as: design, testing, integration, implementation, and maintenance.

The development cost of the prototype systems forces companies to make decisions with assumptions. Usually, system engineers had to foresee system requirements without a rapid prototype. Even a minor change in the system requirements affects the systems' software, hardware design, and testing phases (Austin Page et al., 2019). Due to scheduling constraints, companies make mass production decisions parallel to development phases. In that environment, a minor change in a critical component may cause millions of losses or postpone the delivery schedule because of long-lead times for production (Austin Page et al., 2019). High interaction between development stages increases the system's overall complexity and makes technical debt even more critical (Bar-Yam, 1997). Hiding a specification, skipping a vital component testing, or defining an incorrect requirement may cause the project to fail. As (Verdecchia et al.) indicate, as systems grow and become more complex, timely design decisions and the methods followed can become an obstacle to the development of the system. All these selected methods, approaches, and decisions are possible sources for technical debt occurrence during the product life cycle.

Current studies in the literature calculate and examine the effects of technical debt on software. However, the results of TD decisions also affect hardware, production, and the overall quality of the product. In 1993 Cunningham described the technical debt metaphor from the software development perspective (Cunningham, 1993). J. Kerievsky grew the analogy by adding decisions and processes related to design and architecture (Daughtry III & Kannampallil, 2005).

Architectural choices are especially critical during system development because the decisions for short-term benefits may affect the project team's productivity and the efficiency of the functional and non-functional requirements (MacCormack & Sturtevant, 2016). Since companies experience cutting-edge technologies, each development stage intertwines with the others. Providing early delivery for the customer requires managing TD in each development phase of the systems. The development of complex projects requires a significant investment and workforce. The prototype can be delivered to the customer as a final product as the management's decision because prototyping can be costly. The milestones in the development timeline of a single system may require years to complete. Technical debt monitoring and management activities are challenging in such systems for project teams because the details and histories of each scenario may easily be misconducted or lost (Guo et al., 2016). According to Li, one vital technical debt management activity is the concept's visualization. An increased number of determinations requires a support mechanism to understand the possible result of the decisions and actions (Li et al., 2015a). The chosen solution is usually the most vital stakeholder's opinion without proper guidance, management standard, or a supporting tool. Since complex systems have increased connections and interfaces, they may need to foresee the effects of the given decision on the systems' architectural health (Fernández-Sánchez et al., 2017).

Today, as the user interfaces of the tools improve and their contribution to TD management becomes visible, their market demand continues to increase among companies (Vassallo et al., 2020)(Zampetti et al., 2017). (Christian B. Almazan) examined Bandera, Esc/java2, FindBugs, Jlind, and PMD tools and revealed that the results of these tools were not consistent with each other and contained false negatives and false positives. TD tools were examined by (Tomas et al., 2013) and it was shared which technical debts and which metrics it used. Moreover, soon, (Avgeriou et al., 2021) discussed the technical debt management tools in the market in their research and stated that TD management still needs to have a standard approach.

Completion of complex projects is critical for companies to achieve their goals. Because it either introduces a new technology or promises high profit. The expectations from the delivery of the systems force upper management to involve management processes of the development cycles. While developers try to adopt TD management methods, senior management has not yet mastered the metaphor. The upper management makes technical debt decisions that negatively affect project management activities (Ernst et al., 2015).

1.1. Problem Statement

The effects of technical debt should be addressed in a broader framework, not just software-based; other areas can be taken into account for the result and the reasons that create the technical debt (Rosser & Norton, 2021). Many areas such as production, purchasing, quality, systems, and project management can benefit from technical debt management. In addition, the identification, measurement, and management of technical debt in complex system development still remain an open question for complex system projects.

In today's world, it is accepted that technical debt is a factor that affects the management of the systems and their quality throughout their life. However, methodologies and tools for managing, detecting, preventing, or turning it into an advantage cannot serve system developers yet (The Future of Managing Technical Debt). Rosser & Ouzzif, (2021b) analyzed metaphor in terms of hardware. However, the management of technical debt in complex systems has yet to be studied and standardized in the literature. The experience gained in the technical debt literature can also be used in hardware-based decisions and processes. And this use can help manage inefficiencies in production, use resources more efficiently, and speed up calendars (Rosser & Ouzzif, 2021). Technical debt appears at every stage of the system development process and places a heavy burden on project management, including system recalls, even after the delivery of the strategies to the customer (Harvard Business School, 2003; Henderson & Venkatraman, 1993). Despite this, technical debt management still needs to be covered in the literature in areas other than software. TD monitoring offers cost-effectiveness and project success. Still, the industry needs software or a model that makes the traceability of technical debt effective throughout the system development process.

Avgeriou suggested that these tools still need to be improved on the issues below:

- TD management tools fail to include all technical debt categories,
- Literature does not have a standard method for technical debt detection and mon.itoring,
- Tools fail to consider factors other than software metrics.

The study aims to understand the effects of technical debt in complex system development. To better understand the metaphor in complex system development, a visualization tool is developed that represents the complex lifecycle of each TDs. With that model, the study team aims to present the detailed connections between decisions and causes of effects related to the technical debt in complex system.

1.2. Research Questions

We defined three research questions aligned with the issues discussed above. The first question seeks to answer how the adverse effects of technical debt can be pretended or managed during the complex system development cycle. The question also addressed the effects of TD stakeholders other than software developers.

1. How to identify and track technical debt in complex system (hardware-software) development projects?

Different technical debt categories are provided in the literature. Since the current studies mainly focus on software-related technical debt, other stakeholders like hardware development, supply chain, production, quality, and management are open for research. The third question seeks how efficiently the current categories in the literature help companies or individuals to answer their questions on the detection and management of TD.

2. How sufficient are the categories available for detecting technical debt in complex system projects?
3. How can negative effects on product quality, cost, and program schedule be avoided when making technical debt decisions in complex system development?

The literature shows that the management of projects becomes challenging as the complexity of the systems increases. TD identification and management has a significant role in achieving the project milestones in that complex environment.

1.3. Research Strategy

The research study was completed using qualitative research methods. The qualitative research decision collected data from documents, observations and interviews. The collected data continued with the development of the technical debt visualization model and the application of the developed model in eight different cases.

Before starting the model's development process, the literature was reviewed in detail. As a result of this study, existing technical debt types were determined, and general information was obtained about the importance of technical debt management in complex system development.

After a detailed review of the literature, interviews were held with field experts. Complex system development projects were selected for research, and the selection method was validated based on the standards in the literature.

The technical debt visualization model was applied to the first case, and feedback was collected from the experts. This information was used to improve the model's structure and components. Iteratively, the findings were examined over 8 cases, and the model continued to be developed. This approach aims to develop the model in a way that will meet the needs of the users.

After the model's development, interview data were analyzed with the help of a qualitative analysis tool, and the sufficiency of existing technical debt categories was examined. New categories were shared as a result of this study.

For the validation of the study, interviews were done with field experts, and the results were examined. Open-ended and scale-based questions were used in these interviews. Research questions were answered with the developed model and qualitative analysis, verified with validation interviews.

1.4. Structure of the Thesis

The contents of the remaining chapters of the thesis are explained below:

Chapter 2 includes a detailed literature review. First, this research investigated the definitions of complex and technical debt concepts. As a result of this literature research, the reasons, causes, and types of technical debts were shared with the readers. Then, with the knowledge in the literature, the question of what factors make a project complex and how the complexity level of the project is determined has been answered.

Chapter 3 describes the methods used throughout the research. The research proceeded through developing a visualization model and qualitative analysis of the case data.

Chapter 4 explains why corporations require the TDVM model. The structure of the model and its components are provided in subsequent sections.

In Chapter 5, methods followed during the implementation of the cases, and the purpose of the case study was shared.

The questions that will reveal the suitability of the cases to answer the research questions are explained in this chapter.

In the 6th chapter, the visualization of eight cases and detailed analyses of these cases are shared. In addition, the collected interview data were subjected to qualitative analysis in this chapter, and the results were reported.

The case study findings and qualitative analysis were collected in Chapter 7 and evaluated from a macro perspective. This chapter includes new technical debt categories proposed to participate in the literature and future works for the study.

In Chapter 8, the findings and results of the study were evaluated together with the information gathered in the literature. The study's findings propose six improvement articles that will affect technical debt management in complex system development. Comprehensive findings and accomplishments of the thesis are described in chapter 8.

CHAPTER 2

BACKGROUND & LITERATURE REVIEW

Since its introduction, the technical debt metaphor has been discussed in many areas of the literature. This section examines current studies on the detection, effects, and causes of technical debt. The literature research included corresponding publications, analysis of the universities, and studies of the institutions related to technical debt and complex system development. For the sake of the study, the definition of complexity and complex projects are also reviewed. In addition, factors in deciding whether projects should be considered complex are presented in this chapter.

2.1. Technical Debt

This chapter provides a comprehensive technical debt definition in complex system development projects from the conducted literature review. The following sections represented the most common effects and causes of technical debt and guided the study throughout the analysis and the conducting the cases.

2.1.1. What is technical Debt

Technical debt is a theory proposed by Cunningham (Cunningham, 1993). According to Cunningham,

> *"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, 1993).

Cunningham exhibits that as long as technical debt is repaid and the source code's quality is managed, software development activities will see the effects and benefits of the theory.

In addition to Cunningham TD's definition: Kruchten, Ozkaya, and Nord extended the theory, including the importance of architecture and other software development activities (Kruchten et al., 2012)

(MacCormack & Sturtevant 2016) cited that challenging project delivery targets and limited budgets are the main factors in the emergence of technical debt and increase of future maintainability costs. These decisions are made in order to benefit in the short term.

This research will utilize Ipek Özkaya's (Ipek Özkaya, 2019) technical debt definition, especially it examines and reflects existing findings of the literature in a clear and comprehensive way (MacCormack & Sturtevant, 2016).

> *In software-intensive systems, technical debt consists of design or implementation constructs that are expedient in the short term but that set up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal systems qualities-primarily, but not only, maintainability and evolvability (Ipek Ozkaya, 2019) .*

This scenario also happens in critical system development. With each iteration, the system has more connections, bringing a more complex system with new questions. These questions are increasing with the size of the software and the system.

Martin Fowler grouped the reasons behind the emergence of technical debts under four main categories. These groups are of great importance in identifying and solving TD-related problems. Because the decision mechanisms or approaches behind the decisions is the first step toward identifying the problem. Since technical debt is directly affected by the decisions made by project stakeholders, this quadrant was taken into account in the analysis throughout the study (Martin Fowler).

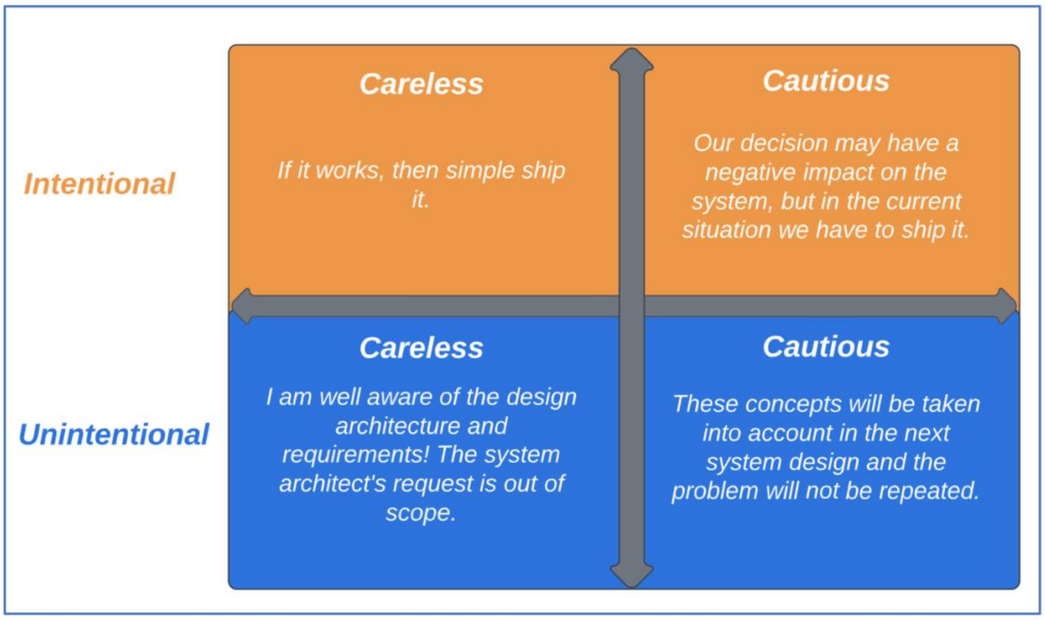

Intentional
Careless
If it works, then simple ship it.
Cautious
Our decision may have a negative impact on the system, but in the current situation we have to ship it.
Unintentional
Careless
I am well aware of the design architecture and requirements! The system architect's request is out of scope.
Cautious
These concepts will be taken into account in the next system design and the problem will not be repeated.

Figure 1 Martin Fowler's technical debt quadrant *(Martin Fowler)*

Carnegie Mellon University Software Engineering Institute examines the technical debt lifecycle in four parts. According to (Carnegie Mellon University, 2016) technical debt may be divided into four main categories:

- Awareness
- Occurrence
- Tipping Point
- Payoff

are the four classes of technical debt. In order to experience these steps, the project has to evolve.

Occurrence is the first place when a project team faces TD. Awareness is the part when the whole organization realizes the debt. When the actual cost of technical debt starts to get higher than the original benefit, it is considered a tipping point. The last part of the lifecycle is the final decision to manage technical debt from the system.

Technical debt arises from faulty decisions made especially in the early stages of system architecture. More than finding technical debt, it also needs to be tracked and managed. (Ernst et al., 2015) states that the lack of these tools and the fact that they are not standardized cause the top management to stay away from the subject. The article draws attention to the fact that the concept of technical debt is largely unknown to employees, especially senior management. (Ernst et al., 2015) points out companies does not have a standard management and solution process in general, and the use of tools is not common.

The key problem is that technical debt cannot be made visible and cannot be measured only with metrics. However, all the previously mentioned tools suffer from some limitations. Existing metrics do not draw complete conclusions for architectural issues. TD metrics measure the static quality of the code and can bring it closer to a decent standard. Difficulties arise, however, when an attempt is made to implement the process at the system level especially when the structure is complex. The most serious disadvantage of the current studies is that they don't consider hardware and system management when they deal with technical debt.

In addition to the TD metrics, it is only possible to manage and determine the effects of technical debts if the system's overall structure is analyzed and monitored at a macro level with a standart or a tool. Analyses of those metrics do not reveal or make visible the problems that changes or errors may cause in connected modules. It does not produce solutions and conclusions that support designers. In support of this study; according to (Zazworka et al., 2013), the decisions or changes affecting the design and system architecture cannot be detected effectively by the TD management tools.

The results of decisions about system architecture and technical design can create more significant effects and results than expected, depending on the complexity of the structure. Even a simple component change can lead to many changes, directly or indirectly, between the connected modules. The source of technical debt may arises from the problems and wrong decisions experienced in the architecture and design of the system.

However, all the studies reviewed so far suffer from the fact that there are no standard methodologies and tools for detecting, monitoring and managing technical debt in complex systems (Stephany Bellomo).

2.1.1.1. Types of technical debt

The most common types of technical debt in the literature are presented in Table 1**.** These types have been used throughout the study to analyze and answer the research questions.

Table 1 Common types of technical debt

Technical Debt Types	Alves et al., 2016)	(Rosser & Norton, 2021)	(Li et al., 2015a)	(Rosser & Ouzzif, 2021)	(Verdecchia)
Architecture	√	√	√	√	√
Build	√	√		√	√
Documentation	√	√		√	√
Requirement	√	√	√	√	√
Test	√	√	√	√	√
Defect	√				√
Design	√		√		√
Implementation		√		√	
Infrastructure	√				√
Code	√		√		√
Process	√				
Quality		√		√	
Service	√				√
Versioning	√		√		√
Configuration		√		√	
Integration		√		√	
Modelling		√		√	
Depreciation		√		√	
People	√				√
Automated Test	√				
Database		√			
Usability	√				

Architecture Debt:

(Alves et al., 2016) (Rosser & Norton, 2021) (Li et al., 2015a) (Rosser & Ouzzif, 2021a) (Verdecchia)(Alves et al., 2016)

The key causes of Architectural Technical Debt may be listed as follows; rework, re-design, loss of performance in non-functional requirements, and decreased productivity in the system architecture. ATD has been an important factor in controlling the level of complexity of the system structure. This type of debt becomes especially important as the project becomes more complex because the scope of the decisions grows, and their effects cannot be easily predicted. In addition, the decisions are taken by considering many criteria and constraints, which prepares the atmosphere for technical debt. For instance, decisions that move system structure to a more complex architecture than the system should be considered an architectural debt. Architectural debt may consist of the following indicators:

- Dependency debt
    - Needles dependencies
    - Cyclic dependencies
    - Underutilized dependencies
- Layering debt
    - Flexibility
    - Single point of failure
- System aging
- Violation of modularity

Build Debt:

(Ramač et al., 2022) (Alves et al., 2016) (Rosser & Norton, 2021) (Rosser & Ouzzif, 2021) (Verdecchia) (Alves et al., 2016)

Build debt is the technical debt type experienced during the "build" or decisions/activities that affect the build process. Additional features the customer does not need or want to extend the build time can be considered build debt. Build debt may consist of the following indicators:

- Dependency debt
    - Needles dependencies
    - Cyclic dependencies
    - Underutilized dependencies
- Additional features
- Slow algorithm
- Bad design
- Lack of quality
- Non-adoption of good practices

Documentation Debt:

(Nicolli, 2020) (Alves et al., 2016) (Rosser & Norton, 2021) (Rosser & Ouzzif, 2021) (Verdecchia) (O'REILLY, 2022)(Alves et al., 2016)

Document plays a vital role throughout the development and production process of the project. The design, production quality, and other phases require explicit expressions. The document defines the work to be done in all processes and the method used through the given process. At this point, the fact that the document needs to be clarified, understandable, and transparent can cause documentation debt. A consequence of document debt is inefficiency and error-prone processes in both the R&D and production stages. Documentation debt may consist of the following indicators:

- Non-adaption of good practices
- Deadline
- Inaccurate time estimate
- Inappropriate planning
- Comments
- Incremental documentation
- Outdated/Incomplete documentation
- Nonexistent documentation
- Team Overload
- Postponing documentation activities
- Noneffective project management
- Poor allocation of resources
- The company does not give importance to documentation

Requirement Debt:

(Lenarduzzi & Fucci, 2019)((Li et al., 2015a) (Alves et al., 2016) (Rosser & Norton, 2021) (Rosser & Ouzzif, 2021) (Verdecchia)

There are three reasons why the requirement debt language has become so important for the management of technical debt. These are structure, format, and content. Stakeholders may interpret the same content in different ways depending on their proficiency. Even experience level plays a role in understanding the content behind requirements. Requirement debt may consist of the following indicators:

- Open-ended, non-verifiable terms
- Ambiguous Adverbs and Adjectives
- Subjective Language
- Non-adaption of good practices
- Requirement smell
- Management Debt
    - Bad development productivity

    - Not enough time for development
    - Inappropriate planning
    - Change Management Debt
- Requirement changes

Test Debt:

(Alves et al., 2016) (Rosser & Norton, 2021) (Li et al., 2015a) (Rosser & Ouzzif, 2021a) (Verdecchia)

Complex systems have many components and subsystems interfacing with each other through different iteration levels. The on-site and timely verification of each stage is, therefore, critical. Steps that are skipped, misconfigured, or postponed may return as debt later. Incorrect planning of this process can lead to hidden problems, especially in the large structure of complex systems. These issues make it challenging to get to the real cause of the issues when faced with other variables. Test debt may consist of the following indicators:

- Lack of KPI
- Lack of feedback
- Lack of standards
- Focus on short-term needs
- Non-adaption of good practices
- Lack of transparency between clients and developers
- Lack of team communication
- Bad motivation
- Non-adaption of good practices
    - Don't interfere if it works
- The different scopes and functioning of the development and test environment

Defect Debt:

(Alves et al., 2016) (Verdecchia)(Akbarinasaji et al., 2016)(Alves et al., 2016)
Defects are frequently encountered in system architecture. In this case, system administrators need to predict the current and near-future effects of the defect. In particular, the possibility of the defect affecting many complex interfaces creates a problem. There is a possibility that the entire system will be affected while the problem is being resolved. Defect debt may consist of the following indicators:

- Non-adaption of good practices
    - Don't interfere if it works
- Not enough time for development
- Lack of team communication
- Budget and time pressure
    - System-level fix/rework time

- People Debt
    - Bad motivation
    - Lack of professionals
- Focus on short-term needs
- Uncorrected known defects

Design Debt:

(Ramač et al., 2022) (Alves et al., 2016) (Li et al., 2015a) (Verdecchia) (Alves et al., 2016)

According to a definition provided by Ward Cunningham:

> *"Design debt is all the good design concepts of solutions that you skipped in order to reach short-term goals. It's all the corners you cut during or after the design stage, the moments when somebody said: "Forget it, let's do it the simpler way, the users will make do."*

As discussed above, Design debt is the sum of the costs that we will have to pay in the long term to solve the problem in the short term, throughout the entire design cycle. As the systems grow and the architecture becomes complex, choosing to pay off design debt may make more sense than solving the problem. Because interfering with complex systems can have unexpected and unpredictable results. Design debt may consist of the following indicators:

- Non-adaption of good practices
    - Don't interfere if it works
- Budget and time pressure
- Lack of standards
- Grime
- Micro view to design/requirements
- Lack of transparency between clients and developers
- UX Debt:
    - Not having design standards
- Lack of team communication
- Bad motivation

Implementation Debt:

(Rosser & Norton, 2021) (Rosser & Ouzzif, 2021a)

To prevent technical debt, companies have to manage software and hardware development cycles without separating one phase from another. Especially systems that require the implementation of different sub-systems, modules, and components need complex decisions frequently. Each decision can affect different teams, corresponding parts, and procedures. Implementation debt arises from incorrect or missing decisions during the integration of the system. The result of implementation debt usually leads to unpredictable behavior in the system. Implementation debt may consist of the following indicators:

- Lack of monitoring activities
- Lack of system-level analysis
- Lack of standards
- Test Debt
    - Non-adaption of good practices
        - Not updating test modules
- Process Debt
    - Not doing the tests in each increment
- Lack of team communication
- Bad motivation

Infrastructure Debt:

(Alves et al., 2016) (Verdecchia)

Infrastructure plays an active role in system development because if it is not managed correctly, it becomes a significant constraint in the product's or system's lifecycle. After the completion of the design phases, testing and production begins. Most of the infrastructure needs to be completed before producing the first product or prototype. That means infrastructure decisions usually have tight schedule constraints. This situation may lead to some parts of the systems partially being tested, produced, or integrated due to a lack of infrastructure. This type of technical debt usually arises from the constraints related to project management activities, schedule pressure, and cutting-edge technology. System architecture does not have a management plan for scenarios that partially implement development lifecycle activities. Especially for complex projects creating this type of plan requires a massive amount of time and experienced system personnel. Technical debts arising from a lack of infrastructure may therefore reveal many debts and risks that cause each other. Infrastructure debt may consist of the following indicators:

- Outdated Infrastructure
- Not enough time for development
- Lack of team communication
- Inappropriate planning
- Design Debt
- Technology constraints

Code Debt:

(Alves et al., 2016)(Li et al., 2015a) (Verdecchia)

Software engineers or designers choose alternatives that provide partial or missing code solutions depending on the system's current need. This solution solves the problem in the short term but may possess numerous adverse effects in the long term. Such alternatives usually require rework or even redesign processes to protect system functions or performance soon. Code debt may consist of the following indicators:

- Non-adaption of good practices
    - Don't interfere if it works
- Budget and time pressure
- Lack of standards
- Micro view to design/requirements
- Lack of transparency between clients and developers
- Lack of team communication
- Bad motivation

Process Debt:

(Alves et al., 2016)

It can be called all the works that contribute to the emergence of the system. Process improvements need to give concrete direct results. Improvements are usually reflected in the system in the long run. However, incomplete definition, non-implementation, or faulty processes often occur during system development. Problems arising from the need for more processes may cause additional requirements for designers in complex systems. Skipping a test step or quality control for a faulty assembly can present a problem that would not typically occur. Process debt can cause the designer to try to solve a problem that would not usually exist or, worse, adds an unnecessary improvement to the system. From time to time, this problem cannot be recreated in the development environment and the time spent can even be wasted entirely. Process debt may consist of the following indicators:

- Lack of standards
- Non-adaption of good practices
    - Old methods and maps
- Lack of well-defined process

Quality Debt:

(Rosser & Norton, 2021) (Rosser & Ouzzif, 2021a)(Lenarduzzi et al., 2021)(Alves et al., 2016)

Both hardware and software development activities depend on effective quality control processes. Quality has a vital role in the detection and prevention of both known and unknown practices and decisions in complex system management. Controlling/monitoring standards creates an effective decision-making process. A change in software configuration may require an update in the related hardware part. A misapplication of configuration management may result in a decision without the hardware team's opinion, creating a system malfunction. Quality prevents the misapplication of these processes. Also, KPIs related to code or products cannot be measured efficiently without effective quality management. A system with different components requires quality control standards and measurement of each part to manage problems even before they appear proactively. Measurement of system health during software development with coding metrics has a similar role in development. Misapplication of these standards, management procedures or short-term solution-oriented decisions may result in problems in the system architecture in the long term. Quality debt becomes more dangerous as the system's complexity increases. Quality debt may consist of the following indicators:

- Lack of standards
- Lack of KPI’s
- # of issues or their co-occurrence
- Process Debt
  - Lack of well-defined process
    - Lack of quality control process
- Tolerance of bad practices
- Low external / internal quality
- Loss of confidence in the quality
- Failure to follow non-functional requirements
  - Security
  - Robustness
  - Scalability
  - Maintainability
  - Performance

Service Debt:

(Alves et al., 2016) (Verdecchia)(Alves et al., 2016)
The term service will be used solely when referring to external or web services to provide and manage some functions and requirements. Interfaces with these services must be verified, managed, and maintained as long as the system remains functional. For example, the deterioration of a predetermined message format over time or incomplete transmission of data from time to time may cause problems in the system. Service debt may consist of the following indicators:

- Lack of transparency between clients and developers
- Lack of team communication
- The number of interfaces affected
- The number of users affected
- Replacement of web service

- Lack of standards

Versioning Debt:

(Alves et al., 2016) (Li et al., 2015a) (Verdecchia)

Processes and documents, especially software and hardware, are stored under different versions during system development. These versions are critical for easier tracking and management of complex systems. Therefore, version changes play a vital role in configuration management in detecting whether the change is minor or significant. Redundant versions make the process difficult to manage in the long run, while unfragmented unbuilt versions increase the system's complexity. Versioning debt may consist of the following indicators:

- Unnecessary version update
- Lack of standards
- Failure to update versions

Configuration Debt:

(O'REILLY, 2022) (Rosser & Norton, 2021)(Rosser & Ouzzif, 2021)

Configuration management is essential to the orderly building of complex systems. In order to understand the cause of the system's problems, analyses are often made with configuration. Configuration management allows it to reach the system's history and provides a broader look at sub-systems and components. For example, even a malfunction in the chipset level can cause a defect at the system level. That problem may arise from the specific batch of that chipset. Configuration management shows the batch and test reports to find the solution. This type of analysis is only possible with a detailed structured report. The development team could obtain valuable information to determine the defects' sources with a detailed configuration. Configuration debt may consist of the following indicators:

- Versioning Debt
    - Failure to plan and manage
- Collapsing branches
- Not having a branching strategy
- Documentation Debt
- Building from scratch
- Non-adaption of good practices

Integration Debt:

(Alves et al., 2016) (Rosser & Norton, 2021) (Li et al., 2015a) (Rosser & Ouzzif, 2021a) (Verdecchia)(Lenarduzzi et al., 2021)

Products that have been verified and working independently from each other may experience unexpected problems when interfaces are conducted. In software management, it may appear as wrong-structured messages or unmanageable buffers. Practices that violate the laws of physics can be an example of hardware-related integration problems, which have a considerable potential to create technical debt in the system. The system's complexity generally requires a lighter, more ergonomic design suitable for narrower spaces. Because in today's technology, we need to optimize all our resources, even in micro sizes, to increase efficiency and ensure innovation. These constraints show the criticalness of the integration phase and how it creates an atmosphere for the appearance of technical debt. Integration is also a complex process with the potential for all other troubles. Because, at this stage, all systems are expected to work in full communication with each other. Integration debt may consist of the following indicators:

- Lack of monitoring activities
- Lack of system-level analysis
- Lack of standards
- Impact on other features
- Wasted time/effort
- Number of people working
- Test Debt
    - Non-adaption of good practices
        - Not updating test modules
- Process Debt
    - Not doing the tests in each increment
- Lack of team communication
- Bad motivation

Modelling & Simulation Debt:

(Rosser & Norton, 2021) (Rosser & Ouzzif, 2021a) (Alves et al., 2016; Lenarduzzi et al., 2021)

Modeling and simulation are critical tool for complex system management because it provides better alternatives for companies in many ways. An important example of modeling and simulation advantages is cheaper solutions and shorter development time for critical activities. Due to infrastructure or the nature of development requirements, some tests and actions can only be performed via simulation or modeling. Modelling & Simulation debt may consist of the following indicators:

- Project size and complexity
- Lack of system-level analysis
- Implementation debt
- Violation of modularity
- Requirement debt
    - Requirement backlog list

- Test debt
    - Incomplete tests

Depreciation Debt:

(Rosser & Norton, 2021) (Rosser & Ouzzif, 2021a) (Alves et al., 2016)

Complex systems designed for long-term needs require a stable and updated design. Even as technology advances, design should stay caught up. Some parts may become obsolete, or the design may be costly and inefficient as production methods require out-of-date technologies. System managers could choose a rework or reuse. Production may become impossible with current infrastructures since they could be unsuitable for modern interfaces. This type of environment makes the management of TD difficult and unpredictable. Depreciation debt may consist of the following indicators:

- Infrastructure debt
    - Outdated infrastructure
- Service Debt
    - Outdated service
- Failure to follow non-functional requirements
    - Security
    - Robustness
    - Scalability
    - Maintainability
    - Performance
- Feature over product
- Defect debt
    - Uncorrected known defects

People Debt:

(Alves et al., 2016)

Complex systems require complex collaboration. These complicated teams have numerous people with various backgrounds and expertise. Emotional intelligence and IQs play a vital role in successfully forming the system. If complex communication does not manage between teams, this can create serious consequences. Correspondingly, employees start acting independently or not complying with the processes. People's debt may consist of the following indicators:

- Lack of training and development
- Failure to adopt mission and vision
- Lack of feedback
- Lack of transparency between team members
- Lack of team communication

- Unsolved individual problems

Automated Test Debt:

(Alves et al., 2016) (Li et al., 2015a)

Automation is of great importance in terms of efficiency, especially when it comes to the production phase. Thanks to automation, resources can be shifted to other projects. Problems arising from human error can be prevented when automation is performed correctly. Of course, the opposite is possible. Incorrectly defined steps or components that lose their function over time can cause severe problems if overlooked. Automated Test debt may consist of the following indicators:

- Lack of KPI
- Reuse of modules and scripts
- Lack of feedback
- Lack of standards
- Focus on short-term needs
- Non-adaption of good practices
- Lack of transparency between clients and developers
- Lack of team communication
    - Bad motivation
- Non-adaption of good practices
- Don't interfere if it works
- The different scopes and functioning of the development and test environment

Usability Debt:

(Shedd, 2015) (Alves et al., 2016) (Li et al., 2015a)

Usability issues need to comprehensively demonstrate their impact. Even in simple systems, usability issues occur after going to the customer or the end user. Complex systems go through many stages until the product reaches the end user, and some functions may encounter problems after passing many layers. Alternatively, the problem may grow and emerge with a more significant impact at the last point. Usability debt may consist of the following indicators:

- Lack of standards
    - Usability standards
- Not having a shared language
- Failure to monitor issues

2.1.1.2. Causes of technical debt

The most common effects of technical debt in the literature are presented in **Table 2**. These causes have been used throughout the study to analyze and answer the research questions.

Table 2 Common causes of TD

Causes of Technical Debt	(Ramac et al., 2020)	(Ramač et al., 2022b)	(Rios et al., 2018)	(Rios et al., 2020)	(Ernst et al., 2015)	(Martini et al., 2014)
Deadline	√	√	√	√		√
Ineffective Project Management	√	√	√	√		√
Lack of Experience	√	√	√	√		
Test not performed	√			√	√	
Misconduct	√					
Focus on producing more at the expense of quality	√	√				
Lack of qualified professionals	√	√	√	√		
Non-adoption of good practicies	√	√	√	√		
Lack of refactoring	√			√	√	
Poor allocation of resources	√					
Inappropriate planning		√	√	√		
Preassure		√				
Inaccurate time estimate		√				
Lack of a well-defined process		√	√	√	√	

Table 2 (cont'd)

Lack of knowledge			√	√		
Incomplete documentation			√	√	√	√
Lack of commitment			√	√		
Bad architectural choices					√	√
Overly complex structure					√	
Obsolete technology					√	
Insufficient test automation					√	
Inter module dependencies					√	
Poor deployment process					√	
Business evolution						√
Feature over product						√
Reuse						√
Paralel development						√
Technology evolution						√
Human factor						√

2.1.1.3. Effects of technical debt

The most common effects of technical debt in the literature are presented in the **Table 3**. These effects have been used throughout the study to analyze and answer the research questions.

Table 3 Common effects of TD

Effects of Technical Debt	(Ramac et al., 2020)	(Ramač et al., 2022b)	(Rios et al., 2018)	(Rios et al., 2020)	(Rios et al., 2019)
Low maintainability	√	√	√	√	√
Increased effort	√				
Rework	√	√	√	√	√
Low external quality	√	√			
Increased cost	√	√			
Delivery delay	√	√	√	√	√
Developer dissatisfaction	√				
Poor code readability	√				
Need of recaftoring	√	√			
Poor allocation of resources	√				
Increased effort		√			
Bad code		√	√	√	√
Low performance		√	√	√	√
Financial loss		√	√	√	√
Low quality			√	√	√
Team demotivation			√	√	√
Stakeholder dissatisfaction			√	√	√
Inadequate documantation			√	√	√

2.1.2. What makes a project complex?

This research also utilizes two definitions of the word "complex," which, based on the Cambridge Dictionary, is "*difficult to understand or find an answer to because of having many different parts" and "involving a lot of different but related parts."* Managing the high number of interconnected elements and their relationships with each other to provide final behaviour determines the system's complexity. (Bar-Yam, 1997)

In addition, as the size of the system increases, management of the system architecture becomes difficult (ROBERT NORD, 2016).For the sake of this research, an increased level of interconnection between elements will be considered a significant factor in indicating a high level of complexity.

> *"Given today's dynamic security environment, it is impossible to formulate a complete set of software requirements ahead of time. Without a robust underlying architecture, someone working on a low-level function will be unable to understand all the end applications in which a function might be used. Therefore, the architect must try to define modules in a way that avoids cross-couplings, whereby changes in one module impact and require changes to other modules* (Defense Science Board, 2018).

As the project's complexity increases, the number of systems and components that are affected by TD decisions increases. That finding reveals the need to evaluate the impacts in all potential project management topics, from schedule management to budget plans, from risk management to design and development (Baccarini, 1996). The decisions taken during the planning and management of project management areas create an *environment for the occurrence of technical debt.*

The need for low coupling and high cohesion is highlighted in the study because of the long-term benefit of effective system management. Precisely planned and designed system architecture is essential to eliminate technical debts before they occur. This proactive approach is vital in preventing unexpected scenarios, problems, and errors that potentially cause TD occurrence.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter presents the model development method created to facilitate the analysis of the data collected in the research and to visualize technical debt management. In addition, the method used to analyze interview data to determine the technical debt categories is shared.

3.1. Design science methodology

Today's industry forces companies to invest in IT systems. With this approach, companies can achieve an advantage in system development and quickly manage risks with less cost (Harvard Business School, 2015) (Henderson & Venkatraman, 1993) . Managing technical debt with IT systems is still new on companies' radars. Companies that follow up technical debt with IT systems in system management will make an innovative development, even in software, with a broader experience and knowledge about the requirement of the direction of the topic.

Design science research aims to find solutions by addressing fundamental problems in academics and industry. These solutions produce innovative artifacts and contribute to the literature (Brocke et al., 2020). The steps of the research strategy with the guidance of design science research have been presented below in Figure 2.

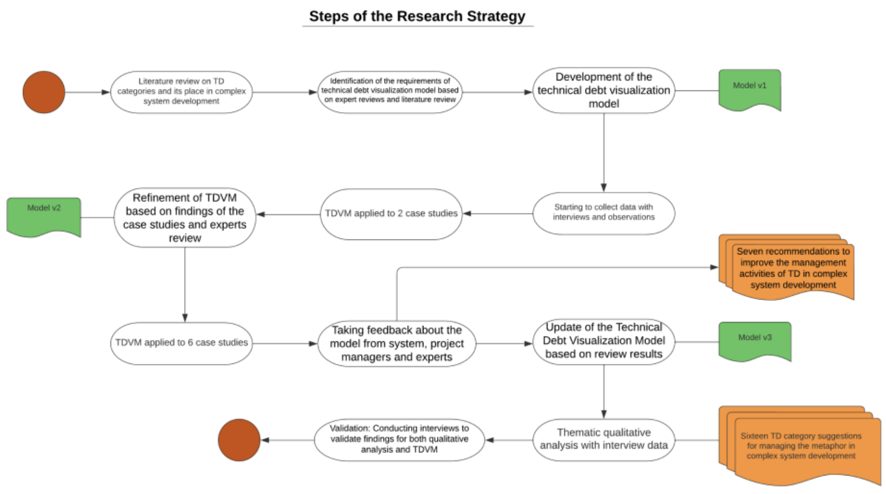

Steps of the Research Strategy
Literature review on TD categories and its place in complex system development
Identification of the requirements of technical debt visualization model based on expert reviews and literature review
Development of the technical debt visualization model
Model v1
Starting to collect data with interviews and observations
TDVM applied to 2 case studies
Refinement of TDVM based on findings of the case studies and experts review
Model v2
Seven recommendations to improve the management activities of TD in complex system development
TDVM applied to 6 case studies
Taking feedback about the model from system, project managers and experts
Update of the Technical Debt Visualization Model based on review results
Model v3
Validation: Conducting interviews to validate findings for both qualitative analysis and TDVM
Thematic qualitative analysis with interview data
Sixteen TD category suggestions for managing the metaphor in complex system development

Figure 2 Steps of the research strategy

Below we provide the steps we follow to develop a technical debt visualization model. Resources selected for the development of the TDVM had to represent the experimental, descriptive, and enhancing aspects of complex systems, which have been provided with the selected case studies.

Appendix E illustrates the stages followed while developing the technical debt visualization model. The following steps are:

- Definition of the problem with literature review and feedback from experts,
- Specification of the requirements for the model,
- Development of the selected solution alternative,
- Iterations for the improvement of the model,
- Verification and validation of the model.

With the analysis of the interview data and expert feedback, the technical debt visualization model experienced three different iterations. The model reflected each iteration of various departments and development environments' needs. The Technical Debt Visualization Model (TDVM) aims to present the timeline of technical debts and their relations with each other with an easy-to-understand interface. The details of the iterations between models are shown in Table 4.

Table 4 Model iteration details

Iterations:	**Model V.1.**	**Model V.2.**	**Model V.3.**
Timeline	-	Integrated.	Iteration was not available.
TDVM Lanes	-	Integrated.	Iteration was not available.
Development Environment	-	Integrated.	Iteration was not available.
Symbols and notations	Effects are integrated under a single category.	The effects of technical debts on different departments were integrated.	The effect rates have been re-adjusted according to the expert's feedback.

3.2. TD categorization in complex projects

To provide data from different departments variety of subjects has been chosen. Interviews have been conducted with the questions provided in appendix-A. Open-ended questions directed to subjects. After the interviews with the subjects, the text was uploaded to the Dedoose software.

As described in the case study chapter, information was gathered from the production, quality, system, design, and project management professionals. These domain experts come from different educational backgrounds, which can be categorized as a doctorate, master, and bachelors. The research team aimed to reveal the effects of TD and its examples outside the perspective of software with different discipline experts. Complex system projects were talked about at different development stages and different periods. Some of these were projects in production, and some were under development. During the interview, it was understood that these processes did not progress linearly. There were many feedbacks from one development stage to another on the timeline.

The data was analyzed, and TDs were determined using the existing categories. The research team created new categories and sub-categories where the categories in the literature were missing or insufficient to describe the TDs. After the analysis, an interview was conducted to evaluate the categories found and to receive feedback from the subjects. The subjects determined for final interviews were selected from the executive staff. The aim was to evaluate the views and perspectives of people involved in complex system development projects who have experienced different TD examples.

CHAPTER 4

VISUALIZATION

The visualization chapter introduces the reasons why companies and individuals need the model. Subsequent sections provide the details and the components for the use of the model.

4.1. What is the technical debt visualization model?

The technical debt visualization model analyzes and manages real-time or historical technical debt scenarios. It shows the decisions and processes that lead to technical debt on the timeline. It reveals their effects and allows us to analyze the points where they were noticed. It demonstrates the effects not only on the software but also on other relevant stakeholders.

Technical debt not follows a linear timeline. It may be necessary to go back from time to time to understand and learn from the process. For example, the exact time when technical debt occurs may be much later than when it was created. Extending this range creates risks. In addition, it is of great importance to follow these stages to improve the processes.

The complex system development process dealing with many decisions and factors is intertwined with technical debts. Moreover, understanding the effects of decisions is quite tricky as it depends on many interfaces. This model aims to express the process more clearly. At the same time, it provides an opportunity to analyze the gaps in the process, erroneous decisions, and points that can be improved. In addition, with this model, decisions that may lead to technical debt can be determined in advance, and awareness of the existing technical obligations in the system can be achieved.

4.2. Description of TDVM Components

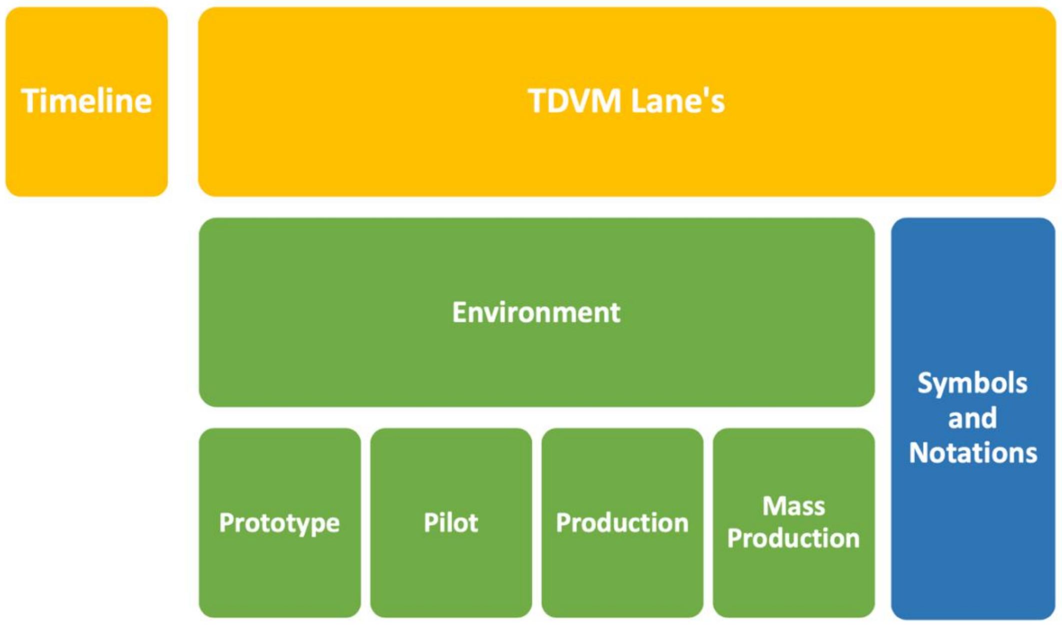


Figure 3 Hierarchical relationship between TDVM components

Figure 3 presents the hierarchical relationship between the use of TDVM components.

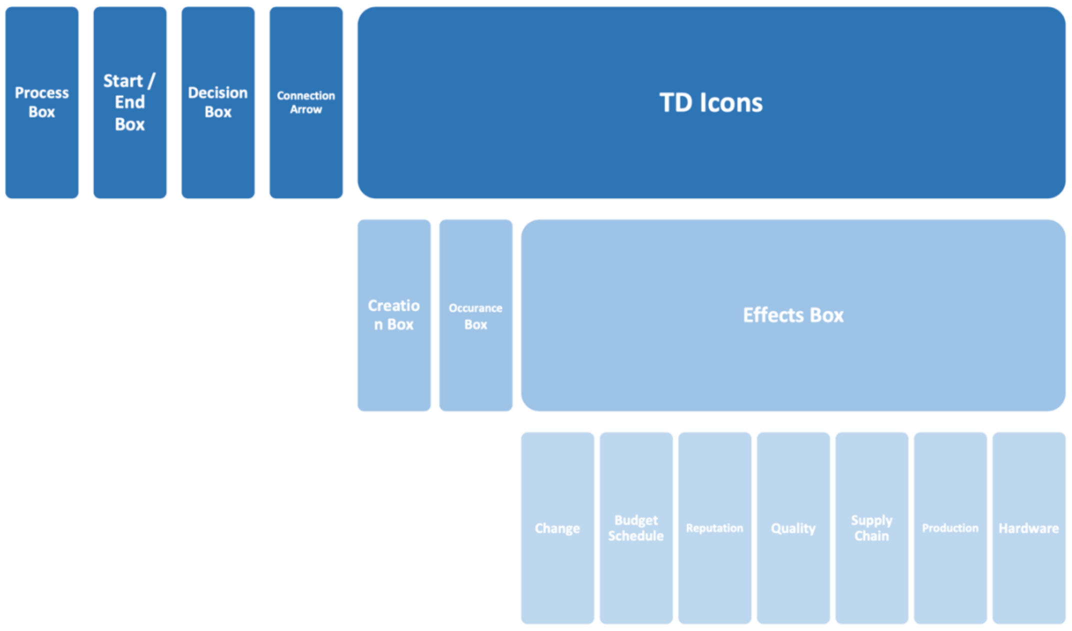


Figure 4 Hierarchical relationship between the use of Symbols and Notations

Figure 4 presents the hierarchical relationship between the use of Symbols and Notations.

The description, details, and usage of each component are given below.

4.2.1. TDVM Timeline

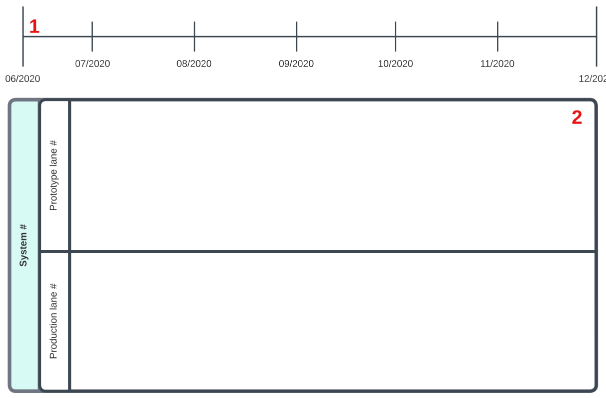


Figure 5 TDVM Timeline

The timeline is represented in icon #1 in Figure 5. It creates a timeline to indicate the relevant technical debt scenario's start, end, and critical milestones.

4.2.2. TDVM Lanes

Lanes of the TDVM represented at box #2 in Figure 5. Swimlanes are used in 2 stages. In the outer lane, selected systems are expressed. As many systems as desired can be created. Lane can be created for multiple productions of the same system. Alternatively, different systems can be added to manage TDs simultaneously.

An extra lane can be created using TDVM components to provide extra information and explain the process. It is critical to describe the process as short and straight as possible. Presenting the details in these additional lanes helps make TDVM clear and understandable.

4.2.3. TDVM Environment

Inner lanes are defined to monitor the progress according to the development environment.

- Production,
- Pilot,
- Job shop,
- Prototype

This list above is non-limiting but can be used as a starting point. The person or organization can use the model to shape their business need.

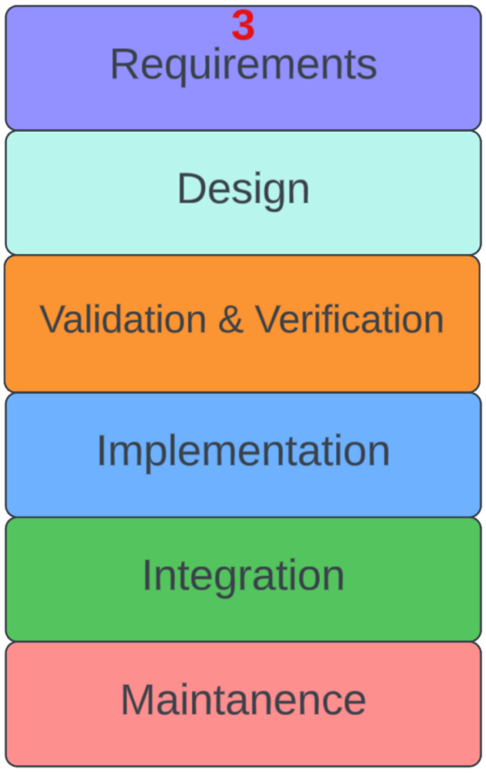


Figure 6 Phases of Development in TDVM

The following phases in Figure 6 may alternate or repeat throughout the system development cycle. It depends on the status of the project and the system development method. Depending on the system's requirements, the project team may choose not to implement or partially implement some of the system development phases. These phases are:

- Requirements:

This stage of system development is where the requirements of all modules, units, subsystems, and systems of the architecture are determined.

- Design:

It is the phase where the design that meets the system requirements is made according to the determined needs. It includes not only the software but also the hardware design.

- Validation and Verification:

This phase expresses the verification and validation stages of designs. It can be repeated, depending on the way companies do business, and can be completed using different methods.

- Implementation:

The implementation phase is when the complex system components are created
Or implemented. Facilities and systems are also tested, inspected, adjusted, modified, and validated during the implementation phase to ensure that the project performs to specifications.

- Integration

Integration is the phase where the software and hardware components of the project are integrated at the system level. Modules and sub-systems that have been produced and verified to meet different requirements in an integrated manner fulfill the system's task at the integration phase.

- Maintenance

Complex systems usually have a very long life. After the completion of acceptance tests with the customer, the maintenance period begins. During this process, the system's maintenance, rework, and redesign processes are expressed at this stage. Companies or individuals can add or remove system development phases from the list according to their development cycles.

4.2.4. Symbols and notations used in TDVM

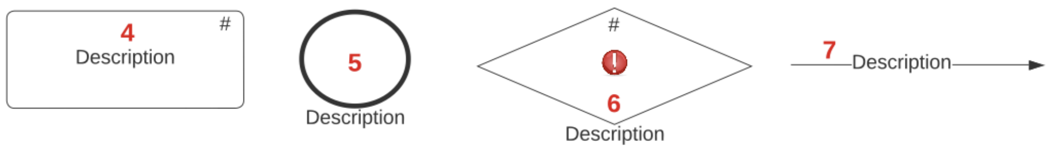


Figure 7 Symbols and notations used in TDVM

- Process is represented in box #4 in Figure 7.

The process box, represented as four, expresses any step in the technical debt scenario contributing to the system development. The details of the operation performed is expressed in the description section. A unique id is given via # for easy tracking of TD.

- Start – End represented in box #5 in Figure 7.

Five is used to indicate that the process has started or ended. In the Description part, the beginning or the end is expressed. A unique id is given via # for easy tracking of TD.

- The decision box is represented in box #6 in Figure 7.

Decisions are of great importance in technical debt management. For this reason, the "red exclamation point" is used to express the importance of the relevant decision. The importance of the TD effect is directly proportional to the number of exclamation points #7.

- The connection arrow is represented at box #7 in Figure 7.

#7 is used to connect two entities. The description section includes details to be given in the process transition. The dashed arrow can be used to give additional information between two entities.

- The components expressed in Figure 7. represent essential details about technical debt. In particular, these components are expressed in different colors to convey that these points on the TDVM are critical.

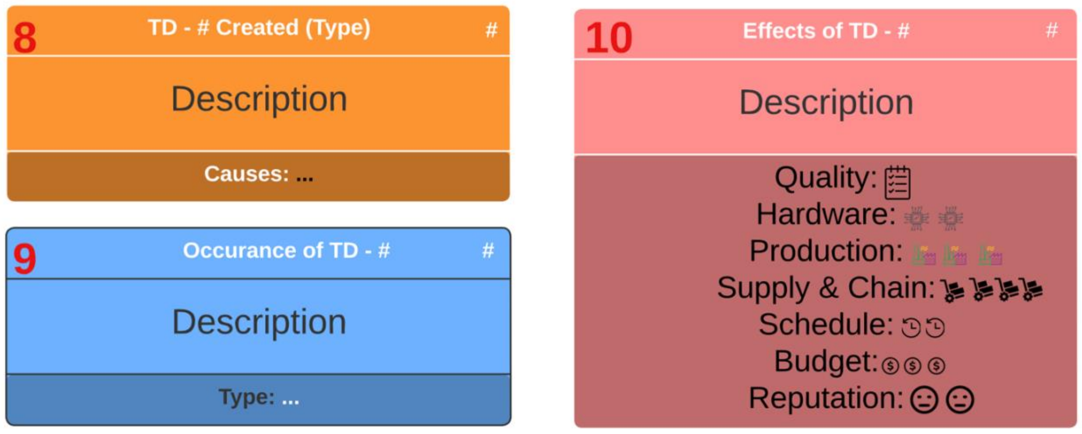


Figure 8 TD specific components

- Creation of TD represented at box #8 in Figure 8.

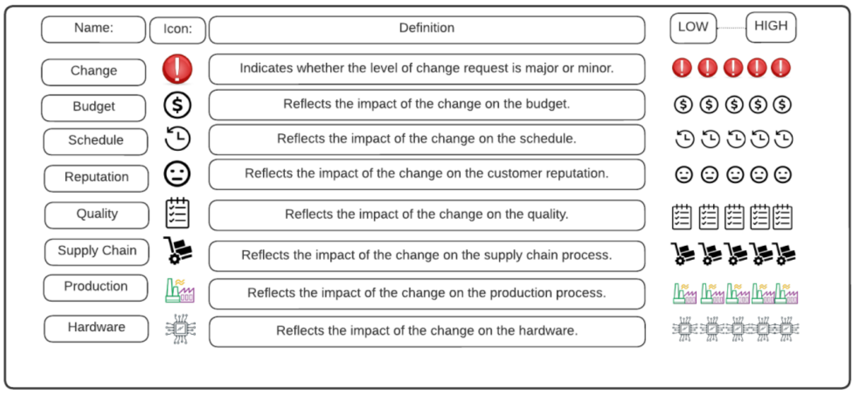
8 indicates the moment when technical debt created on the system. Technical debt often needs to be noticed when it is created in the system. Many teams may only be aware that they are struggling with technical debt once the effects become apparent. TD type can be intentional or unintentional, as in the literature. A unique id is given via # for easy tracking of TD. In the Description section, the details that caused the creation of technical debt are given. In the Causes section, the reasons for creating technical debt are given.

- Occurance of TD represented at box #9 in Figure 8.

Nine indicates the first moment when technical debt is noticed in the system. A unique id is given via # for easy tracking of TD and. The details that cause the technical debt occurrence are shown in the description section. In the type section, the kind of technical debt is issued.

- Effect of TD represented at box #10 in Figure 8.

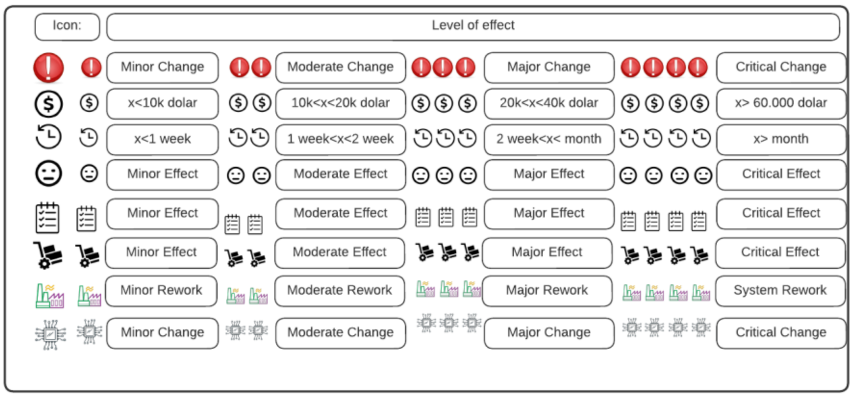
The effects of technical debt are expressed in Figure 8. Details of the results are given in the description section. Icons show the distribution of the impact across different departments. A unique id is given via # for easy tracking of TD. The Figure 8. provides a classification of the effects of the icons against the number of uses.



Name:	Icon:	Definition	LOW — HIGH
Change	❗	Indicates whether the level of change request is major or minor.	❗❗❗❗❗
Budget	💲	Reflects the impact of the change on the budget.	💲💲💲💲💲
Schedule	🕒	Reflects the impact of the change on the schedule.	🕒🕒🕒🕒🕒
Reputation	😐	Reflects the impact of the change on the customer reputation.	😐😐😐😐😐
Quality	📋	Reflects the impact of the change on the quality.	📋📋📋📋📋
Supply Chain	🛒	Reflects the impact of the change on the supply chain process.	🛒🛒🛒🛒🛒
Production	🏭	Reflects the impact of the change on the production process.	🏭🏭🏭🏭🏭
Hardware	🔲	Reflects the impact of the change on the hardware.	🔲🔲🔲🔲🔲

Figure 9 Classification of the effects of the icons

Technical debts can create various levels of impact on different fields in the complex system development cycle. The level of the effects in different categories is given in the Figure 9.



Icon:	Level of effect							
❗	❗	Minor Change	❗❗	Moderate Change	❗❗❗	Major Change	❗❗❗❗	Critical Change
💲	💲	x<10k dolar	💲💲	10k<x<20k dolar	💲💲💲	20k<x<40k dolar	💲💲💲💲	x> 60.000 dolar
🕒	🕒	x<1 week	🕒🕒	1 week<x<2 week	🕒🕒🕒	2 week<x< month	🕒🕒🕒🕒	x> month
😐	😐	Minor Effect	😐😐	Moderate Effect	😐😐😐	Major Effect	😐😐😐😐	Critical Effect
📋	📋	Minor Effect	📋📋	Moderate Effect	📋📋📋	Major Effect	📋📋📋📋	Critical Effect
🛒	🛒	Minor Effect	🛒🛒	Moderate Effect	🛒🛒🛒	Major Effect	🛒🛒🛒🛒	Critical Effect
🏭	🏭	Minor Rework	🏭🏭	Moderate Rework	🏭🏭🏭	Major Rework	🏭🏭🏭🏭	System Rework
🔲	🔲	Minor Change	🔲🔲	Moderate Change	🔲🔲🔲	Major Change	🔲🔲🔲🔲	Critical Change

Figure 10 Level of the effects in different categories

Institutions, organizations, and individuals can reconstruct or modify these tables according to monitor and present their needs. Details of the icons are given in the table below:

The categorization of technical debt effects for different stakeholders is given in the Table 5. The thesis adhered to the scope in this table while determining the effects.

Table 5 The categorization of technical debt effects for different stakeholder

Category	**Effects:**
Quality	The effect of technical debt on quality processes: Program Quality: Failure to operate project management processes from quality perspective, Production Quality: Failure to operate the quality standards in the production, Process Quality: Failure to follow quality standards and methods System quality: Failure to meet the functional and non-functional requirements
Hardware	Rework, redesign and reuse actions for hardware.
Production	Failure to operate production lines efficiently Preventing the efficient use of production resources
Supply Chain	Failure to manage supply chain operations because of risky purchasing decisions, Repetitive purchasing and logistics operations with no added value Loss of price advantage and bargaining opportunity
Schedule	Shifts in the system delivery schedule due to technical debt
Budget	Additional expenses in the project budget due to technical debt
Reputation	Loss of customer confidence, The customer devises a pessimistic perspective against the system.

4.3. How to use the technical debt visualization model on your complex projects?

In previous chapters, the purpose and components of TDVM were shared in detail. This section will explain how the model can be used in any case.

1. **Create TDVM swim lanes:**

As a starting point, swim lanes should be drawn for the system which will be developed or produced. For example,

- System-1 Production,
- System-1 Prototype, and
- System-2 Production

Can be drawn for the basis of the model.

2. **Create a timeline and update through the case:**

Technical debt only sometimes follows a straight line on the timeline. For this reason, the timeline should be updated as the process progresses and new developments are noticed in the system development cycle. The timeline must represent the critical milestones and the case's start and end date.

3. **Name the TDVM diagram:**

After the swimlane and timeline are created, companies should assign a name to TDVM to track and store the model in the company's knowledge database.

4. **Start the process by adding the starting event:**

The open circle is added to the associated swimlane to initiate operations. If a description wants to be added, it can be written inside the circle.

5. **Add activities:**

After the start node is placed, activities that create value for the case begin to add to the model. Decision boxes must be used at decision points. Exclamation marks can be placed inside the decision boxes that indicate the importance of the decision. The number of exclamation marks indicates whether the change is minor or significant. Decisions are important because they are the possible points at which technical debt can be born.

For this reason, it is essential for the model. Technical debt is sometimes determined at the time of its creation. In this case, the creation box is used. The reasons causing the technical debt should include in the creation box to be used in later process improvement analysis. Technical debt occurrence is quite critical. Because the factors that can affect the system's performance begin to be managed at the occurrence of TD. Type of technical debt should be included in the occurrence box.

6. **Customize TD effect categories:**

Technical debt can have multiple effects. Effects are represent in the effect box. The effect box is essential to show the effects of technical debt on the system and stakeholders.

As a basis, various categories of technical debt effects have been shared above for the use of the model. Companies and individuals can interpret the model according to their usage industries. They can create new categories for TD effects or update the degrees of effects. When the effects of a technical debt arise, the technical debt effect box is filled using this table.

7. **Add development phases:**

After the activities are added, the relevant phase is drawn below. Thus, the model users are warned when a new phase is passed. This step must be repeated at each phase transition.

8. **End the process by adding the ending event:**

The open circle is added to the associated swimlane to initiate operations. If a description wants to be added, it can be written inside the circle.

9. **Draw an extra lane:**

Scenarios in complex structures contain many activities. For this reason, they can be challenging to understand and follow. For activities that require additional details in TDVM, a lane is drawn under the relevant component, and the information flow is given there. The complete picture remains more transparent and more understandable with using extra lanes.

CHAPTER 5

IMPLEMENTATION

The purpose of case studies is to collect data to analyze finding answers to the study's research questions. In order to complete the study, complex project data from different development strategies were collected from the participants. This information was analyzed with the developed model provided with our study. For the analysis to conclude, all processes impacting technical debt were visualized to answer RQ2.

The sufficiency of the existing categories and new category suggestions were shared. We used the available categories in the literature to constitute the final result. A third-party tool was used for the qualitative analysis. The detailed analysis of outputs used in answering RQ3.

By evaluating the case studies and qualitative analysis findings, suggestions are presented to assist companies and individuals in managing technical debt in complex system development projects. The research team aimed to answer RQ1 with recommendations.

5.1. Multiple Case Study

Literature and studies show no standard definition or method to describe a case study (Baškarada, 2014). A *case study* is a research approach that assists researchers, individuals, and companies in comprehending phenomena in various fields in real-life settings (Karlsson). The case study's main goal and the most complex challenge is creating an understanding of the given subject for the readers (Issue-Based Observation Form for Case Studies in Science Education)(Gustafsson & Gustafsson). Due to the complex nature of the systems, the analysis of the cases was quite challenging in this study. Creating a common conclusion from the perspectives and scenarios of different experts required a great deal of effort, as the literature indicates.

5.2. Purpose of the case study

The complex system development cycle often encounters technical debt. The purpose of that multiple case study is to reveal the effects of technical debt on hardware and other stakeholders involved in the development cycle. With these exploratory project studies, the research team investigated and aimed to indicate improvement possibilities for visualizing TD in complex system development.

These goals evaluate the following research questions:

RQ1: How sufficient is TDVM for monitoring and visualizing technical debt in complex systems?

RQ2: Can hardware, production, quality, and stakeholders other than software teams make decisions that create technical debt during the system development process? In complex systems environment, how accurate to make decisions without considering the conditions of other stakeholders?

RQ3: Should other stakeholders of the system development lifecycle be included in the management of technical debt?

Subsequent chapters describe the findings and the background of each project.

5.3. Case Study Design

This chapter gives the mechanism behind the research details for the cases selected and analyzed throughout the research. In the following sub-sections, with the guidance of the knowledge of the literature;

- The considered factors for the case selection,
- The system architecture and the structure of the project on which observations and interviews are made,
- Tools and methods used when collecting and analyzing data

are described in detail and documented in that manner.

5.4. Case selection criteria

While determining the case selection criteria, the experts' opinions were gathered in the first phase. As a result, sub-systems and systems have many interfaces, and the increased size of project teams considered as projects require management of TD. Selected project scenarios include the following categories:

- Technical debt has been created in the past, and its effects have ended,
- Technical debt that has occurred in the past and whose effects continue,
- Technical debt has been created, but the effects still need to be discovered.

It is observed that these categories occurred both individually or together, depending on the scenarios.

Current studies in the literature examine technical debt in terms of software. This research also focuses on the effects of TD on different areas like; hardware, production, quality, supply chain, and software development processes. Selected cases also include decisions affecting the systems' hardware and software development cycles. The standard prepared by IPMA was used to determine the project's complexity level where the interview and case data will be collected. 2 people did this analysis, and the result showed that the project was complex. The corresponding result was shared in Appendix B and C.

5.5. Case Conduct

Case product consists of 7 sub-systems that have interfaces with each other. The details of the connection between different sub-systems are detailed in Figure 11. The total system bill of the material consists of more than 13.000 products and raw materials. Although the complexity of the architecture, system managers succeeded in determining the low coupled design. Despite the proper system structure design, this project is significant in examining the effects of technical debt on complex system development.

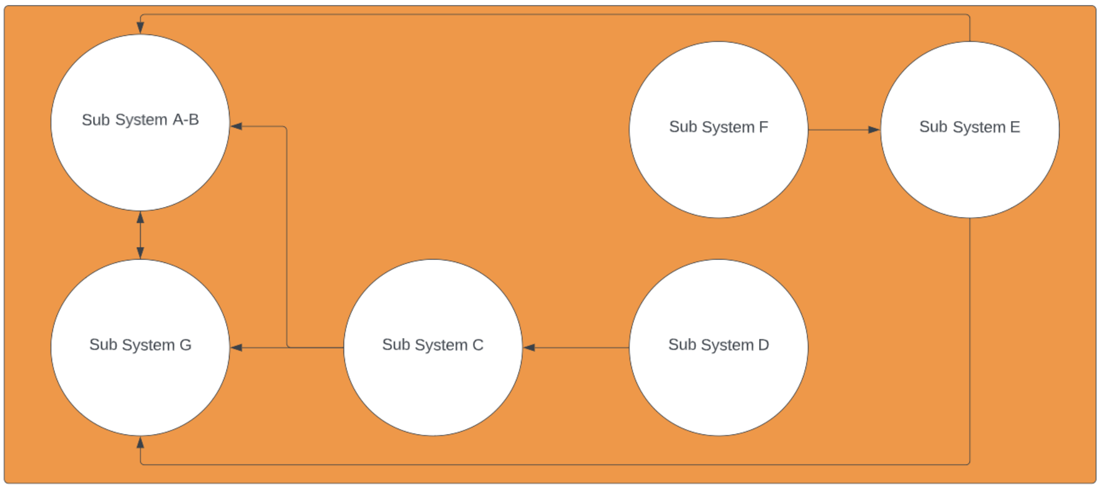


Figure 11 The system structure

Although the complexity of the architecture, system managers succeeded in determining the low coupled design. Despite the proper system structure design, this project is significant in examining the effects of technical debt on complex system development.

The primary subsystem that manages the functions of others is called Sub-System A-B. In Figure 11. Figure 11, Subsystem A-B is detailed in subcomponents. Interfaces have been lowered and need to be partially reflected due to confidentiality. Each subsystem, A and B, provides the technical requirements of most of the equivalent products in the market. Moreover, those two complex architectures operate together with complex interfaces and networks. For this reason, most of the cases were selected from subsystem A-B.

The development and production period of the most straightforward component of the system was recorded as 14 months. It took more than 7+ years to completion of the entire system, from design to production. Although more than 200 engineers and technicians have worked in developing the system, it is currently considered one of the company's most innovative products.

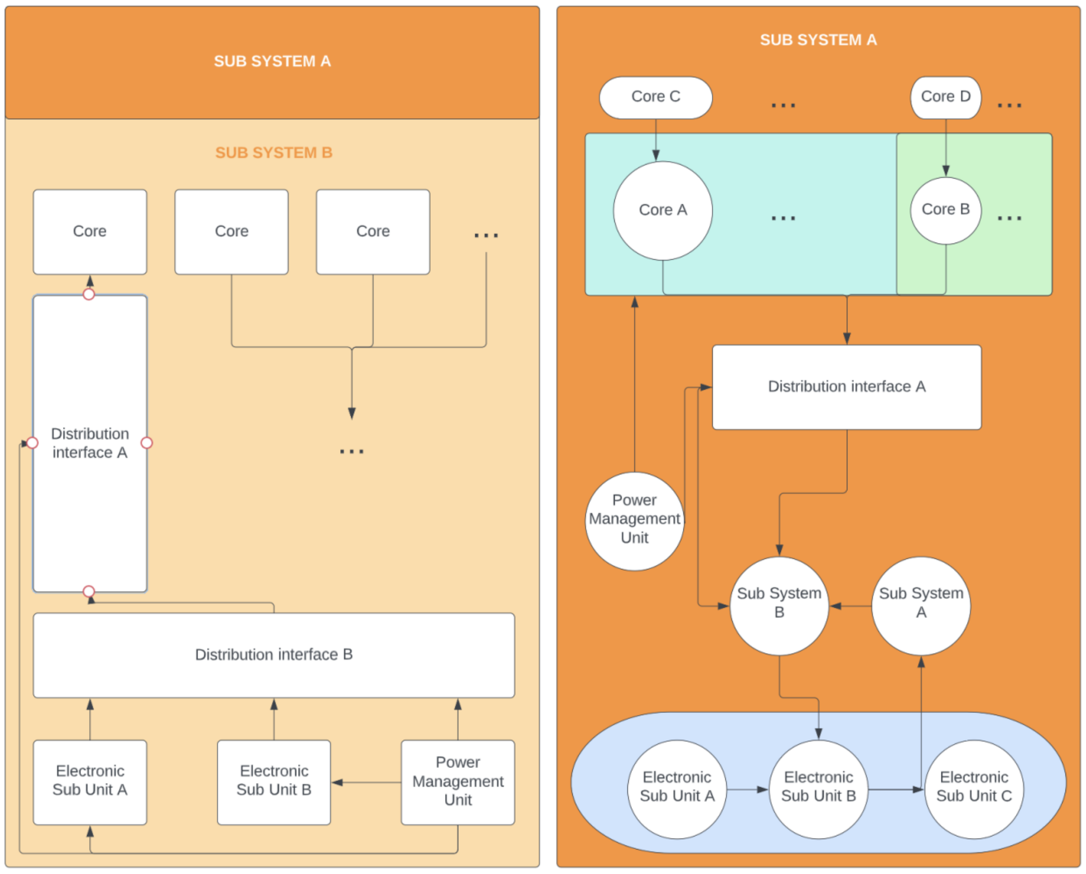


Figure 12 The Sub-System structures

5.6. Data Collection Protocol:

The researcher is an active participant of the project, and the study was conducted with direct observations. The findings are recorded continuously in an unstructured manner. In Table 6, the details of the interviews with the participants during the data collection process are given.

Table 6 Data collection protocol

Phases of data collection	Number of Participants	Hours of Data	Number of Sessions	Roles Involved
Preliminary interviews	8	7,2	1	Developers, System Architects, Project Managers, Testers, Team Leaders, Production Manager, Supply and Chain Manager, Supplier General Manager
Validation interviews	3	3	1	Program Manager, System Manager, Design Team Leader
Informal interactions	10	2	1	Developers, System Architects, Project Managers, Testers, Team Leaders, Production Manager, Supply and Chain Manager, Supplier General Manager

5.6.1. Interviews

The questions directed to the participants during the Interviews are presented in Appendix A.

Table 7 Participants' roles and responsibilities

Role:	Definition:
Software developers and team leaders	- Responsible from the design of software development and management of sub systems in terms of software - Verification and deployment of the designs - Technical documentation - Improvements of processes and designs
Program managers	- Monitoring the project, - Making strategic decisions, - Management of the teams, - Budget management - Stakeholder management - Creating WBS and milestones for project teams
Hardware developers and team leaders	- Responsible from the design of hardware development and management of sub systems in terms of hardware - Verification and deployment of the designs - Technical documentation - Improvements of processes and designs

Table 7 (cont'd)

System managers	- Responsible from the requirement elicitation, - Management of the system architecture, - Management of the technical interaction between software and hardware teams -Validation and verification of system level requirements -Responsible from the integration of the system
Program sponsors	- Making decisions for strategic problems, - Monitoring long term objectives, - Program management
Project planning engineers	- Responsible from the master schedule, - Monitoring the budget and time constraints, - Reporting the status of the project
Production managers	- Planning and organizing master schedule, - Monitoring the budget and time constraints of production, - Reporting the status of the projects, - Assigning resources - Coordinating production teams
Supply chain managers	- Monitoring and planning supply chain needs, - Inventory planning and management, - Warehouse management, - Assigning resources, - Supply chain and data management

The company's employees' responsibilities and hierarchy, represented **in Table 7**, are essential when analyzing the decisions taken regarding managing technical debt in complex system development.

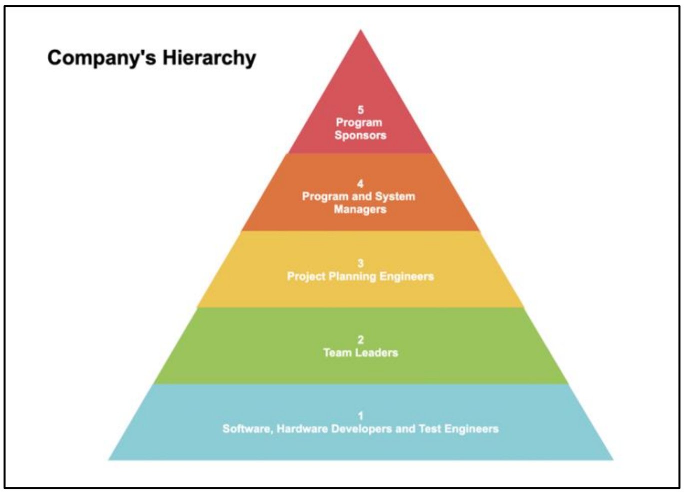

Company's Hierarchy
5 Program Sponsors
4 Program and System Managers
3 Project Planning Engineers
2 Team Leaders
1 Software, Hardware Developers and Test Engineers

Figure 13 The Companies hierarchy

5.6.2. Observation Forms

Each stage of the system development lifecycle was observed with the observation forms, which are represented in Appendix D. These findings are from the scrum, system, and project meetings.

Feedback from system architects and project managers was also essential to understanding technical debt's effect on the system architecture. Furthermore, these feedbacks become guidelines to focus on the troubling parts of the system development activities and decision mechanisms.

Observations were conducted with two participants to provide validation to the findings. If the results match 80 percent, the statement was deemed validated.

5.6.3. The Dedoose Tool

Dedoos software was used to analyze the qualitative data collected from interviews and observations.

CHAPTER 6

CASE ANALYSIS

The data collected for the case study were visualized using the technical debt visualization model. Afterward, the research results for improvements and developments were shared on the emerging model.

6.1. Using TDVM to identify and monitor TD

Chapter 7 provides a detailed analysis of 8 projects by implementing a technical debt visualization model. Interviews with domain experts are used for the study. The model offers a management and monitoring mechanism for decisions and processes affecting TD throughout the development and production lifecycle of the system. At the end of the chapter, the findings are presented.

6.1.1. Background

Table 8 Background information of Project#1

Case Subject:	20+ years of engineering domain experience (Abbreviated as Alpha
Company:	Subcontractor Named as Theta
Role:	Ceo of the company Theta
Contractor:	Company named as CA
Interview Duration:	47 minutes
Product / Sub-System:	An essential part of sub-system A,B (Named as product Beta)

Since 1999, Theta operates as a subcontractor in the aviation industry. The company has over 150 employees, most of whom are R&D personnel. Theta's domain of the subject is the aviation industry, and the subcontractor has more than 20 experience with the Theta.

Theta is responsible for developing an essential component of Sub-System A-B as a subcontractor (referred to as product Beta). During the contracting stage, it was discussed and agreed upon that any delay in the delivery schedule of the product would affect the sub-system A-B's delivery schedule.

Recently, in 2019, the prototype of the product Beta was developed by Theta. The basis of this interview covers the period during the transition of the product from the prototype development stage to the mass production process. Theta has worked to develop and produce Beta during this transition period with its design, testing, manufacturing, purchasing, quality, and product development teams. Different stakeholder management was crucial for a project's success. At the same time, this process overlapped with a period when Theta grew uncontrollably.

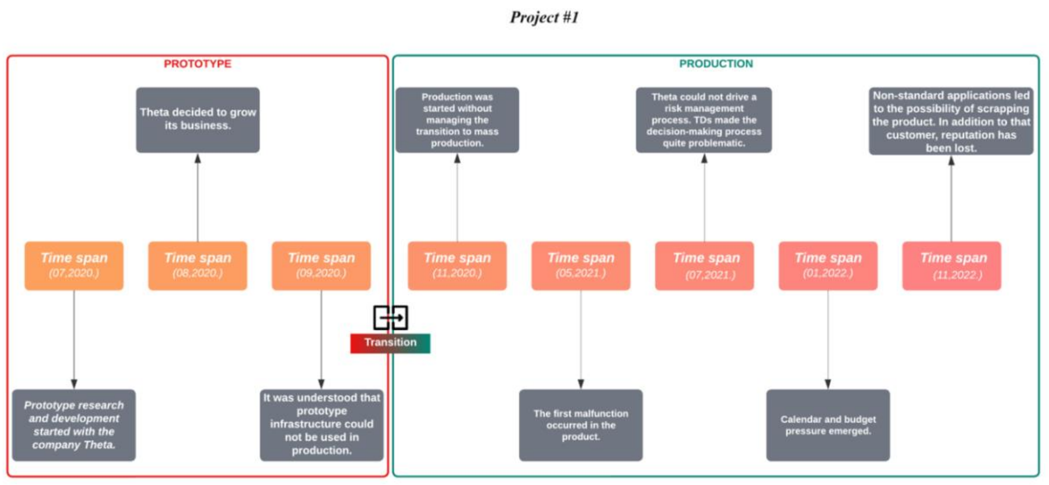

Project #1
PROTOTYPE
PRODUCTION
Theta decided to grow its business.
Production was started without managing the transition to mass production.
Theta could not drive a risk management process. TDs made the decision-making process quite problematic.
Non-standard applications led to the possibility of scrapping the product. In addition to that customer, reputation has been lost.
Time span (07.2020.)
Time span (08.2020.)
Time span (09.2020.)
Time span (11.2020.)
Time span (05.2021.)
Time span (07.2021.)
Time span (01.2022.)
Time span (11.2022.)
Transition
Prototype research and development started with the company Theta.
It was understood that prototype infrastructure could not be used in production.
The first malfunction occurred in the product.
Calendar and budget pressure emerged.

Figure 14 Timeline of Project #1

As a result, Theta needed help managing quality control, configuration, design, and system development processes despite its ERP and quality management systems. The project schedule was well-planned for both the production and the R&D processes. After consecutive wrong decisions, the product Beta faced the risk of being scrapped. In addition, decisions to be made now had to consider calendar pressure and the risk of delayed delivery of systems A-B.

6.1.1.1. Findings

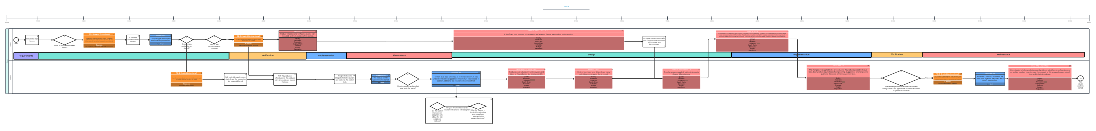


Figure 15 TDVM of Project #1

The larger version of the

Figure 15 is given below in three parts.

The development and production timeline of Beta is represented in above figure. Theta worked and gained domain-specific experience with the customer from the early stages of the project.

As a summary, the table indicates that the problems listed below occurred due to not reflecting the experiences gained in the prototyping stage to mass production. Moreover, the risks noticed at the prototype stage are not managed.

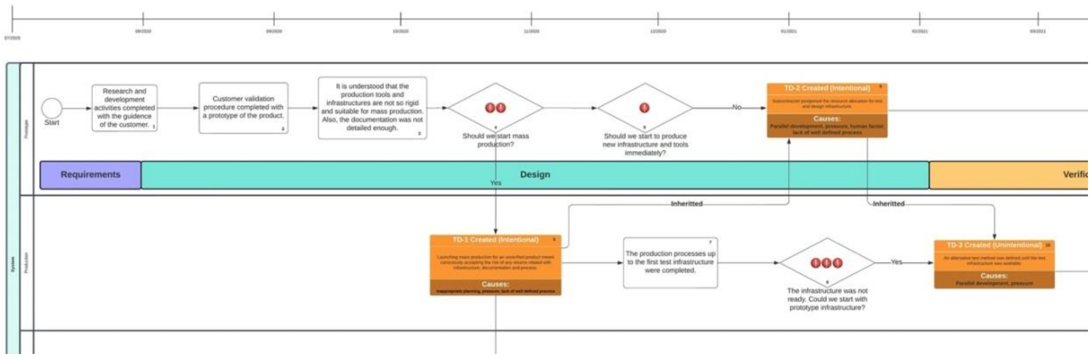

Start
Research and development activities completed with the guidence of the customer.
Customer validation procedure completed with a prototype of the product.
It is understood that the production tools and infrastructures are not so rigid and suitable for mass production. Also, the documentation was not detailed enough.
Should we start mass production?
Should we start to produce new infrastructure and tools immediately?
No
Yes
TD-2 Created (Intentional)
Causes:
Requirements
Design
Inheritted
TD-1 Created (Intentional)
The production processes up to the first test infrastructure were completed.
The infrastructure was not ready. Could we start with prototype infrastructure?
TD-3 Created (Unintentional)

Figure 16 TDVM of Project #1.1

Figure 16 shows that there are five decisions why technical debts occurred. These are:

- Starting the production of Beta before the development of infrastructures (TD-1),
- Failure to initiate the design and manufacture of the infrastructure required for the production of Beta (TD-2),
- Using prototype production methods and infrastructures for mass production rather than standard production methods and infrastructures (TD-3),
- Placing and operating an alternative test method without validation of the techniques(TD- 4),
- Implementing non-standard rework method without customer approval (TD-5).

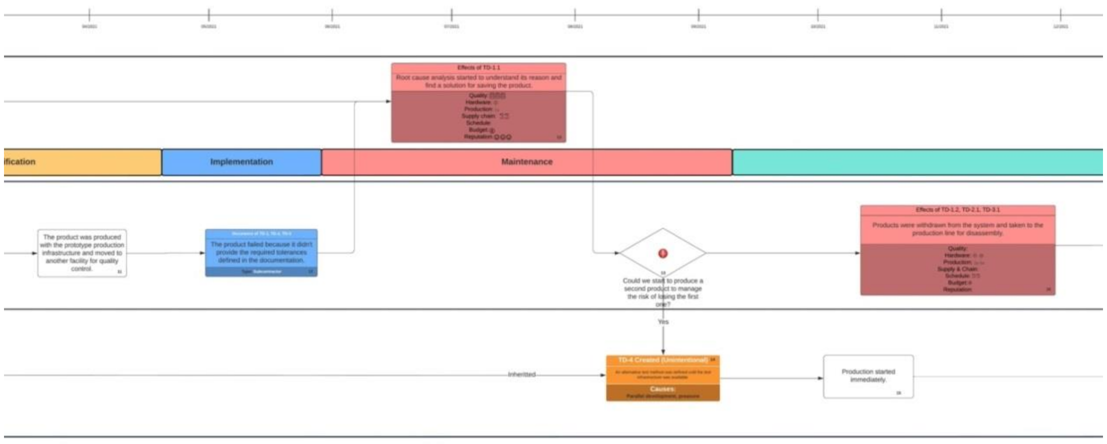

Effects of TD-1.1
Root cause analysis started to understand its reason and find a solution for saving the product.
Implementation
Maintenance
The product was produced with the prototype production infrastructure and moved to another facility for quality control.
The product failed because it didn't provide the required tolerances defined in the documentation.
Could we start to produce a second product to manage the risk of losing the first one?
Yes
Effects of TD-1.2, TD-2.1, TD-3.1
Products were withdrawn from the system and taken to the production line for disassembly.
Inheritted
TD-4 Created (Unintentional)
Causes:
Production started immediately.

Figure 17 TDVM of Project #1.2

After the implementation was completed, the required measurements were made to look at the mechanic tolerances according to the document. It turned out that the Beta did not meet the required tolerances and was unsuitable for the next stage. This decision is expressed in box 12 when TD 1, 2, and 3 occur. As a solution, the PM of Theta alternatively decided to produce a second product that was identical to the first one. Although there was a problem with the first product, the decision to start a second production with the same test infrastructure caused TD-4. The effects of TD-1.1 and TD 1.2, 2.1, 3.1 in different categories are given below:

Table 9 Effects of TD-1.1

Category	Effects of TD-1.1:
Quality	The quality personnel identified and improved the problems in the prototyping to production, and production processes. Additional quality analysis and measurement effort are required to reveal the current status of the Beta, including high-budget testing methods like ultrasonic measurement.
Hardware	The hardware team performed re-work and re-design to recover the incorrectly processed product.
Reputation	Theta's reputation was poorly affected. Additional features are offered to the customer without any charge.
Cost	Cost of additional R&D activities to fix the problem such as - Development life cyle was repeated to fix the problem, - Labor cost of senior personnel attending recurring meetings and activities for finding a solution to the possibility of scrap. Cost of stopping the production line of Beta for two months during the rework process.
Production	The production line was stopped for 2 months. The overall production schedule of the Theta was shifted because Beta's production line was used to produce different kinds of products.
Category	**Effects of TD 1.2, 2.1, 3.1:**
Quality	Efford required to design a procedure to be followed during the disassembly of the product Beta, FAI(First article inspectation) process has been executed, A document describing the quality management plan for the disassembly process was prepared.
Hardware	Requirement specification process for Beta repeated. Preperation of a detailed rework document and process.
Schedule	The time to disassemble the product Beta was scheduled to be twice the production time.
Cost	The disassembly process was an extra step and cost more than the product assembly of Beta as it required delicate artistry.
Production	Production teams had to separate all raw materials and units of Beta to start the rework process. Rework had a severe negative impact on the morale of the team.

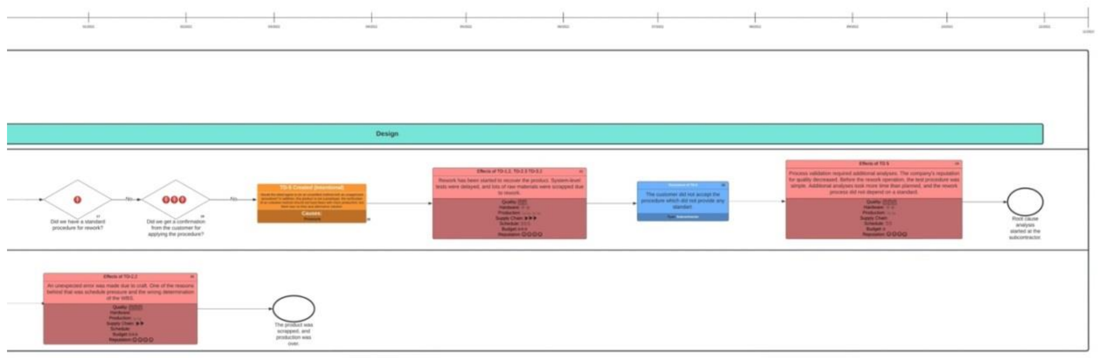


Figure 18 TDVM of Project #1.3

Theta tried to rework the product using a non-standard method. However, the customer disapproved of the method without validation. The decision represented in Figure 18 in box 19 made technical debt 5. Many additional tests and analyzes were required after the client requested verification of the method. The rejection of this non-standard method triggered Theta to notice that they were experiencing technical debt 5.

Table 10 Effects of TD-2.2, 1.2, 2.3, 3.2, 5:

Category	**Effects of TD-2.2:**
Quality	The effort to find the defect in the production of the Beta process that scrapes the product,
Customer	Customer loss confidence in Beta's overall product quality, The customer brought up a penalty for scrapping the product,
Cost	Total production effort including the cost of holding the production line for non-value-added operation, Raw materials required fort the production of Beta Raw materials had to be procured from stock with increased price,
Supply Chain	The effort required for bidding and material procurement for raw materials again,
Production	The production line was used for 4 months, Planned resources for beta production annihilated.
Category	**Effects of TD 1.2, 2.3, 3.2:**
Reputation	The reputation was harshly affected when the customer, who had already lost months, learned of another rework operation that would take two months,
Hardware	Requirement specification process for Beta repeated, Preparation of a detailed rework document and process,
Schedule	Delivery of product Beta postponed for two months, A-B system level tests delayed for three months

Table 10 (cont'd)

Supply Chain	During the rework operation, many materials were scrapped, which must be resupplied, The new method required a rapid supply of new materials, which increased both the workload and the prices,
Cost	The total cost of rework operations, including quality, design, and production process, and an hourly personnel wage, The total cost of rework operations,
Production	Production line reserved additional two months for the rework operations of Beta, Due to both the use of infrastructure and the use of qualified personnel, different production operations could not be scheduled.
Category	**Effects of TD 5:**
Quality	Visual inspection was sufficient before the operation was applied to the product. However, after its implementation, a detailed computer model was required. That result was the result of the quality needing approval from the customer, Efforts were made to re-evaluate all quality processes.
Hardware	Hardware designers' effort were required for the design of the verification and validation model of Beta,
Schedule	Model design, validation test, and analyses with customer take extra 4 weeks,
Cost	Cost of extra analysis and validation test for nonstandard operation, Design cost of reworked hardware computer model,
Reputation	There was chaos for the customer. While the quality approach is critical, the company has yet to make a non-standard application. The removal of the Theta from the status of the subcontractor was evaluated from customer, Customer lost time and money on regular meetings.

6.1.2. Background of Project #2

Table 11 Background information of Project#2

Case Subject:	Design team leader with 15+ of experience (Abbreviated as Gamma)
Company:	Company named as CA
Role:	Hardware and software team leader
Interview Duration:	63 minutes
Product / Sub-System:	Electronic sub unit B (Abbreviated as product Nu)

CA has been a technology leader in the defense industry sector for many years. Most of the company comprises R&D personnel with master's and doctorate degrees. Most of the projects focus on developing and acquiring new technologies. CA manages everything from raw material procurement to the operational execution of the system. The systems produced by the company require integrating the designs of professionals from many fields of expertise. Implementing these different interfaces and their successful operation drives many technical debts in the development and production processes. Moreover, the market requires increased delivery time and completion of challenging specifications.

Gamma has been working as a designer since 2004. Gamma started his career as a hardware designer. Due to the complex system structure and working domain, it was also necessary to manage and coordinate the teams in the software field. Gamma was one of the operational staffs involved in the company's technological revolution. This experience gave him the experience and knowledge required to develop complex system structures. During the 63- minute Interview, CE reveals the acute effects of not managing technical debt from the perspective of the supply chain department.

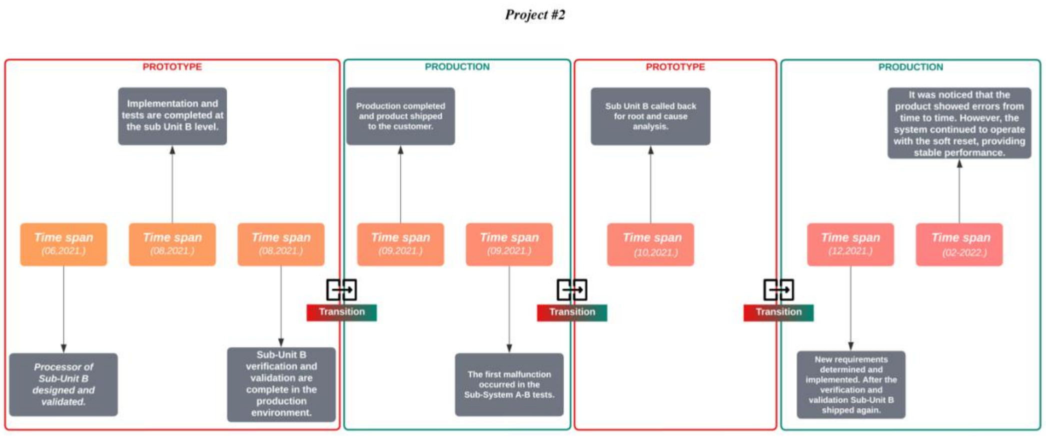

Project #2
PROTOTYPE
PRODUCTION
PROTOTYPE
PRODUCTION
Implementation and tests are completed at the sub Unit B level.
Production completed and product shipped to the customer.
Sub Unit B called back for root and cause analysis.
It was noticed that the product showed errors from time to time. However, the system continued to operate with the soft reset, providing stable performance.
Time span (06.2021.)
Time span (08.2021.)
Time span (08.2021.)
Time span (09.2021.)
Time span (09.2021.)
Time span (10.2021.)
Time span (12.2021.)
Time span (02-2022.)
Transition
Transition
Transition
Processor of Sub-Unit B designed and validated.
Sub-Unit B verification and validation are complete in the production environment.
The first malfunction occurred in the Sub-System A-B tests.
New requirements determined and implemented. After the verification and validation Sub-Unit B shipped again.

Figure 19 Timeline of Project #2

Despite years of experience and acquired technology, innovation management still requires managing technical debt. The interview covers the period during the R&D and production of Nu. Gamma draws attention to the point that Nu's product development and mass production process are intertwined. Furthermore, the product had to be withdrawn from the customer to analyze the reason behind the malfunctions realized after the successful compilation of verification and validation. The fact that the production and development environments of the system are different shows that the technical debt has turned into an active factor that can cause the product to be recalled.

6.1.2.1. Findings of Project #2

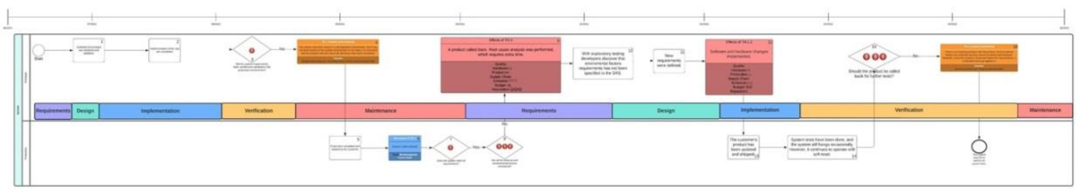


Figure 20 TDVM of Project #2

The larger version of

Figure 20 is given below in two parts.

The prototype life cycle, which started with the processor design of Nu, continued with the implementation of the product to the Nu level. Thus, the product was produced as a prototype and met all requirements.

As stated in box four in Figure 20, it was noticed that all product tests were carried out in the production environment. Validation tests were done, assuming all other interfaces were simulated in the test infrastructure and working correctly. That decision was due to the client's failure to plan the resources for system-level testing.

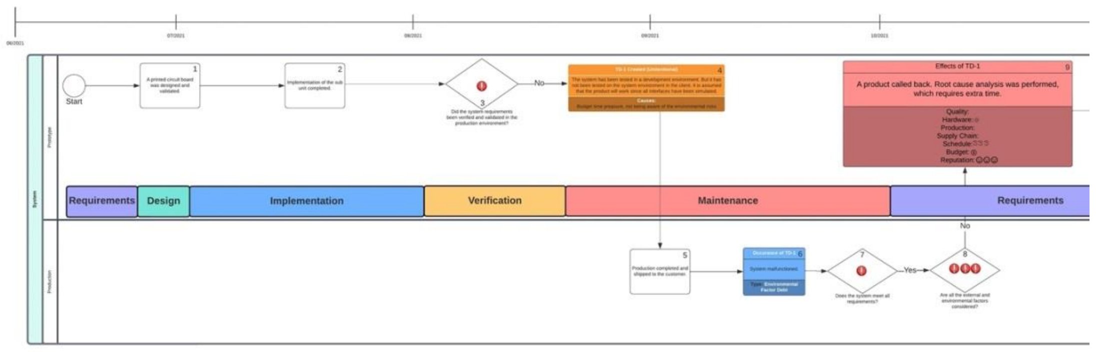

06/2021
07/2021
08/2021
09/2021
10/2021
System
Prototype
Production
Start
1
A printed circuit board was designed and validated
2
Implementation of the sub unit completed
3
Did the system requirements been verified and validated in the production environment?
No
4
TD-1 Created (Unintentional)
Causes:
Effects of TD-1
9
A product called back. Root cause analysis was performed, which requires extra time.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
Requirements
Design
Implementation
Verification
Maintenance
Requirements
No
5
Production completed and shipped to the customer.
Occurrence of TD-1
6
System malfunctioned
Type: Environmental Factor Debt
7
Does the system meet all requirements?
Yes
8
Are all the external and environmental factors considered?

Figure 21 TDVM of Project #2.1

Since Nu's test required Sub-System A-B, it had to be scheduled before. Nevertheless, since system-level verification was not done in a production environment, this resulted in the first technical debt. The assumption that the product would work efficiently in real life posed a severe risk that had to be managed. As expected, Nu failed during the system integration test. Box 6 in Figure 21 shows the first moment CA realized it was living with technical debt. This caused the product to be recalled. TD1 effects represented at
Figure 21:

Table 12 Effects of TD-1.1

Category	Effects of TD-1.1:
Hardware	• The design team had to consider all possible effects, which required considerable effort. If testing could be done on the Sub-System A-B level, assumptions would not be necessary. However, as this is no longer possible in the current situation, they had to consider every scenario and take precautions.
Reputation	• The product's failure affected the planning of upper-level system tests. This position damaged the customer's view of the company.
Cost	• R&D labor expenses, • Additional testing and development activities did not plan at the beginning of the project, • Logistics budget spent on product recall
Schedule	• Three weeks Nu delivery delay • Sub System A-B system delivery schedule was delayed by two weeks.

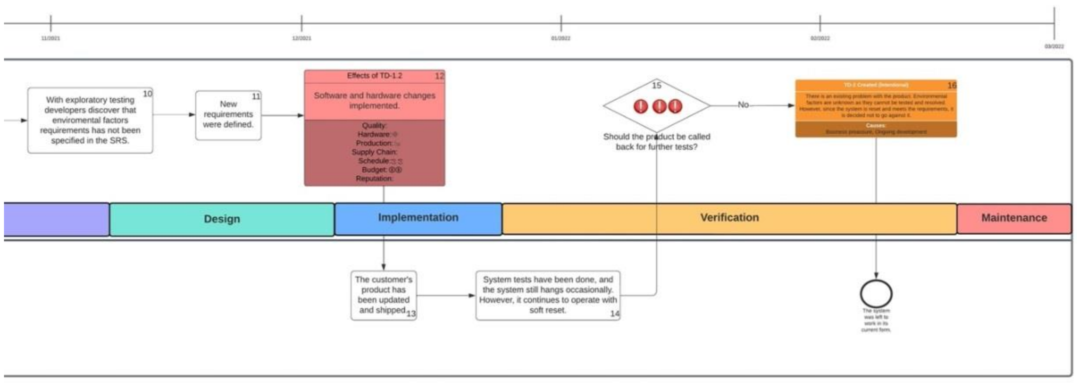

11/2021
12/2021
01/2022
02/2022
03/2022
With exploratory testing developers discover that enviromental factors requirements has not been specified in the SRS. 10
New requirements were defined. 11
Effects of TD-1.2 12
Software and hardware changes implemented.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
15
Should the product be called back for further tests?
No
16
Causes:
Design
Implementation
Verification
Maintenance
The customer's product has been updated and shipped 13
System tests have been done, and the system still hangs occasionally. However, it continues to operate with soft reset. 14
The system was left to work in its current form.

Figure 22 TDVM of Project #2.2

After weeks of R&D work, the design teams took protection against any possible scenarios. This approach generated many unnecessary functions to be integrated into Nu. The product was first verified as a prototype, then transferred to production and reworked. Passing all tests, the Nu was retested at the Sub System A-B level. The effort expended at this stage was a continuation of the effects of TD-1, as shown in Figure 22. The effects of TD-1.2 are represented in

.

Table 13 Effects of TD-1.2

Category	Effects of TD-1.2:
Hardware	• The effort required for rework and verification activities on both the prototype and the final product
Production	• Production plans were disrupted because an unexpected rework came. • Products still in production have been stopped, and their delivery schedule has been affected.
Cost	• R&D developer and designer expenses, • Production labor expenses, • Cost of delayed Project because of production lane stoppage
Schedule	• System delivery schedule delayed by three weeks • Sub System A-B system delivery schedule was delayed by two weeks.

Although the product occasionally malfunctioned, this time, it did its job with a soft reset without turning itself off. Overall system performance and requirements have been achieved this time. However, this solution only partially solved the problem. Because the answer to the question of if the product does not work one day despite a soft reset could not be given clearly, that question counted as TD 2 to the system. Technical debt two is a version currently living in the design and has yet to appear.

6.1.3. Background of Project #3

Table 14 Background information of Project#3

Case Subject:	10+ years of Project management experience (Abbreviated as Tau)
Company:	**Company named as CA**
Role:	Project Manager
Interview Duration:	**33 minutes**
Product / Sub-System:	Core A of Sub-System A-B (Abbreviated as product Omicron)

The 3rd case subject describes the process of outsourcing one of the main components of subsystem A-B under CA’s management. Although the company in question has served in the market for many years, it has just entered the sector that Omicron needs. It was desired both to contribute to developing a new company and benefit from the price advantage.

Subcontracting choice created a risk for Omicron, which began from the development stage and continued during mass production. Omicron, where more than 20 experienced engineers are actively working, was designed and mass-produced with the joint work of two companies.

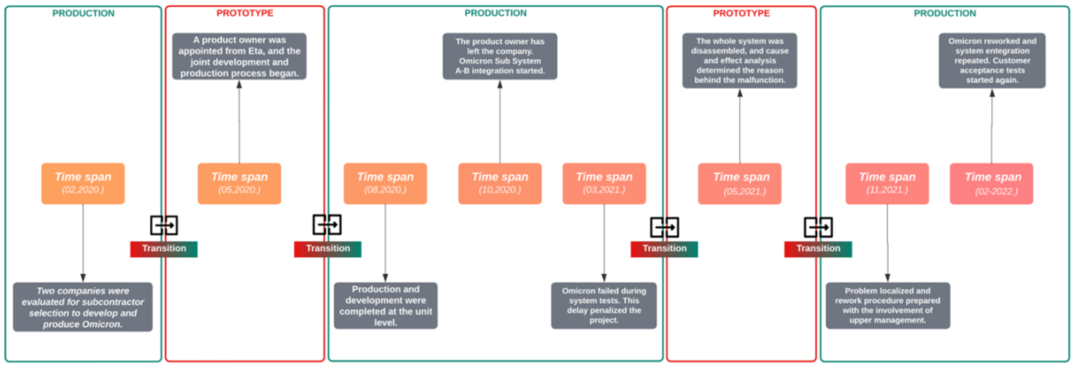

PRODUCTION
PROTOTYPE
PRODUCTION
PROTOTYPE
PRODUCTION
A product owner was appointed from Eta, and the joint development and production process began.
The product owner has left the company. Omicron Sub System A-B integration started.
The whole system was disassembled, and cause and effect analysis determined the reason behind the malfunction.
Omicron reworked and system entegration repeated. Customer acceptance tests started again.
Time span (02.2020.)
Time span (05.2020.)
Time span (08.2020.)
Time span (10.2020.)
Time span (03.2021.)
Time span (05.2021.)
Time span (11.2021.)
Time span (02-2022.)
Transition
Transition
Transition
Transition
Two companies were evaluated for subcontractor selection to develop and produce Omicron.
Production and development were completed at the unit level.
Omicron failed during system tests. This delay penalized the project.
Problem localized and rework procedure prepared with the involvement of upper management.

Figure 23 Timeline of Project #3

Tau's interview provided a perfect example of how significant the impact of technical debt can be if not managed in complex projects. Because the technical debt taken in one of the sub-system sub-components caused all system integration activities to be repeated at the highest level.

6.1.3.1. Findings of Project #3

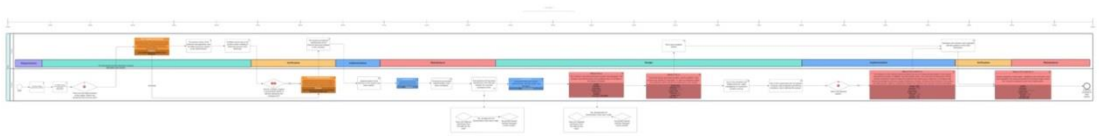


Figure 24 TDVM of Project #3

The larger version of the Figure 24 is given below in three parts.

As seen in the Figure 24, the process started with the decision of which subcontractor would launch the design and production of Omicron.

Project management chose the new company alternative because of the factory's mission and the price advantage. This position caused technical debt 1 to occur, which would cause CA to lose the advantage of domain-specific experience. At the same time, the fact that the alternative company will experience the operation and process for the first time was another factor in creating technical debt 1.

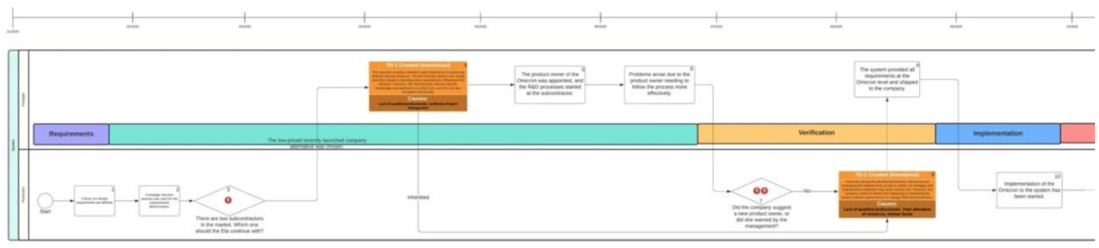


Figure 25 TDVM of Project #3.1

An experienced staff had to be appointed to manage the process to minimize the risk. However, the company has appointed a product owner who caused many problems in product lifecycle management. Moreover, despite the increased conflicts between the two companies, CA did not replace the product owner, which caused technical debt 2. An experienced individual had to be appointed to manage the Omicron process before more problems arose. The subcontractor, who already has no field experience, had a complicated design process with noneffective guidance from CA. Still, unit-level tests were completed, and the Omicron was shipped to the factory for sub-system A-B integration.

Because of the structural complexity of Sub-System A-B, installation and transportation were very difficult. Considering that Omicron was working, the system integration was completed, and the system was transferred to the A-B test area. Nevertheless, in the first test scenario, Omicron failed. Moreover, this error caused damage to the entire system. Transportation was re-arranged, and the system was disassembled for troubleshooting. CA came under extreme pressure when the customer learned that the system had been damaged. The effects of TD-1.1, TD-1.2 represented in the Table 15:

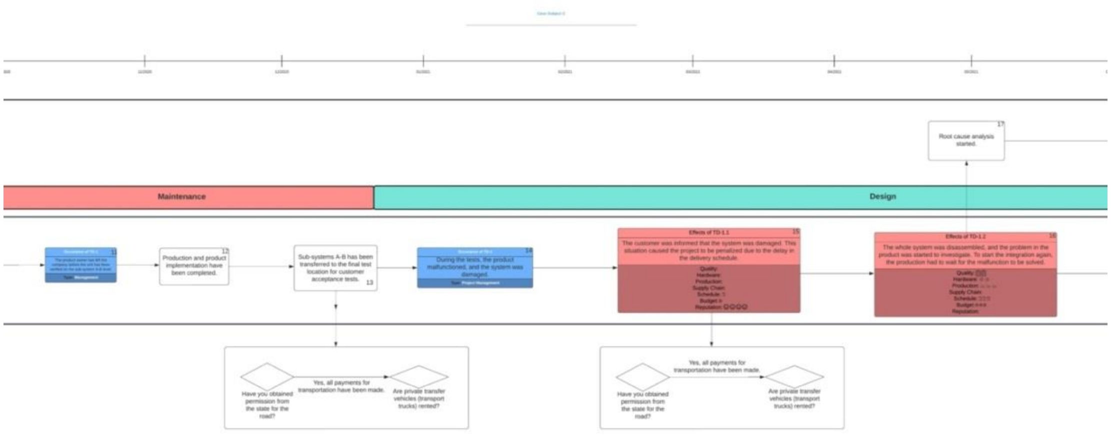

Root cause analysis started. 17
Maintenance
Design
11
Production and product implementation have been completed. 12
Sub-systems A-B has been transferred to the final test location for customer acceptance tests. 13
During the tests, the product malfunctioned, and the system was damaged. 14
Effects of TD-1.1 15
The customer was informed that the system was damaged. This situation caused the project to be penalized due to the delay in the delivery schedule.
Quality
Hardware
Production
Supply Chain
Schedule
Budget
Reputation
Effects of TD-1.2 16
The whole system was disassembled, and the problem in the product was started to investigate. To start the integration again, the production had to wait for the malfunction to be solved.
Have you obtained permission from the state for the road?
Yes, all payments for transportation have been made.
Are private transfer vehicles (transport trucks) rented?

Figure 26 TDVM of Project #3.2

Table 15 Effects of TD-1.1,1.2

Category	Effects of TD-1.1:
Reputation	The customer learned that the system was malfunctioning and would be delayed. This postponement caused a loss of confidence in the customer.
Cost	The penalty was applied with the conditions mentioned in the contract. The integration and transportation cost of the system is repaid.
Schedule	The system delivery schedule has been postponed by two weeks

Table 15 (cont'd)

Category	Effects of TD-1.2:
Hardware	Efford of a design team is required to understand the reason behind the malfunction at the system level.
Production	Production line stopped by three weeks. Resources have been allocated for system-level rework in production.
Quality	System-level quality assurance failed. An audit of quality processes was initiated for the company's processes, related to system verification and validation.
Schedule	Rework shifted the system schedule by three weeks.
Cost	Cost of quality, hardware, production, and upper management personnel person-hour expenses. The project penalty continued to be applied until the system was delivered.

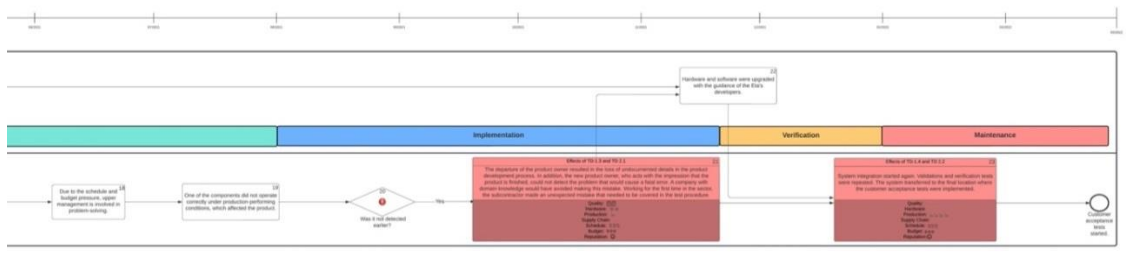


Figure 27 TDVM of Project #3.3

After studying the cause and effect behind the problem, corrective actions were initiated by Omicron. Problem-solving period caused the CA to waste many valuable resources, including the top management personnel. Problem-solving period wasted valuable resources, including the effort of top management personnel.

- Inaccurate management of R&D to the production process,
- The inability to establish a bridge between the manufacturer and the design team,
- The selection of a subcontractor who is not an expert in the field caused the system delivery schedule to be postponed and wasted resources.

caused the system delivery schedule to be postponed and wasted resources.

The effects of TD-1.3, TD-2.1 and TD-1.4, TD-2.2 represented in the Table 16:

Table 16 Effects of TD-1.3, 2.1, 1.4, 2.2

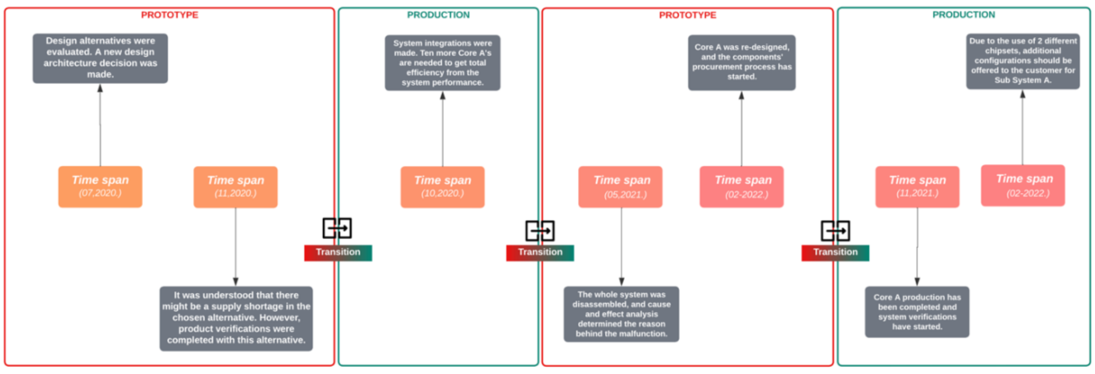
Category	Effects of TD-1.3, TD-2.1:
Hardware	The experience that the subcontractor should have was lacking in the manufacturer in this scenario. Closing this gap required additional person-hours from the hardware team.
Production	The production line stopped for another three weeks
Quality	The quality control process of the Subcontractor turned out to need to be improved.
Schedule	Brainstorming to find the cause of the problem shifted the system schedule by three weeks.
Cost	Cost of quality, hardware, production, and upper management personnel person-hour expenses. The project penalty continued to be applied until the system was delivered.
Reputation	The customer was still awaiting a solution to the system's functionality problems.
Category	**Effects of TD-1.4, TD-2.2:**
Schedule	The production line was reserved for four weeks, which affected the production schedule of other project teams.
Production	System-level disassembly and integration with the newly requested changes have been completed.
Cost	Cost of quality, production personnel person-hour expenses. The project penalty continued to be applied until the system was delivered. Cost of transportation and the integration of the system to the field.
Reputation	This time until the system was reworked was a serious negative factor for the customer.

6.1.4. Background of Project #4

Table 17 Background information of Project#4

Case Subject:	10+ years of engineering domain experience (Named as subject CD)
Company:	Company named as CA
Role:	Design Team Leader
Interview Duration:	52minutes
Product / Sub-System:	Core A (Named as product PD)

Case subject CD has been a senior engineer for CA since 2010. Recently, he experiences actively involved in the development of critical system projects. CD started as a designer and successfully took part in many different subsystems and system deliveries of the company. Lately, he had the opportunity to work in different hardware design fields as a team manager. CD managed and coordinated over 50 people from 6 different departments throughout Core product A's development and production process. This 52-minute Interview presents the TD's and their effects during the development and production lifecycleof product PD.



PROTOTYPE
Design alternatives were evaluated. A new design architecture decision was made.
Time span (07.2020.)
Time span (11.2020.)
It was understood that there might be a supply shortage in the chosen alternative. However, product verifications were completed with this alternative.
Transition
PRODUCTION
System integrations were made. Ten more Core A's are needed to get total efficiency from the system performance.
Time span (10.2020.)
Transition
PROTOTYPE
Core A was re-designed, and the components' procurement process has started.
Time span (05.2021.)
Time span (02-2022.)
The whole system was disassembled, and cause and effect analysis determined the reason behind the malfunction.
Transition
PRODUCTION
Due to the use of 2 different chipsets, additional configurations should be offered to the customer for Sub System A.
Time span (11.2021.)
Time span (02-2022.)
Core A production has been completed and system verifications have started.

Figure 28 Timeline of Project #4

Figure 28 shows that the Core A interview starts with a critical design decision that would affect the system architecture. Selected design alternative directly influences the operations of the Supply Chain, Production, Quality, Design, and Software departments. CA's business domain, history, mission, and vision were shared in the background of Case project #1.

The team leader's failure to take action despite being warned about the supply of products led to TD being paid with a redesign effort. These decisions also caused the prototype and production stages to be intertwined repeatedly. Project #4 proves how important technical debt management is in hardware development.

6.1.4.1. Findings of Project #4

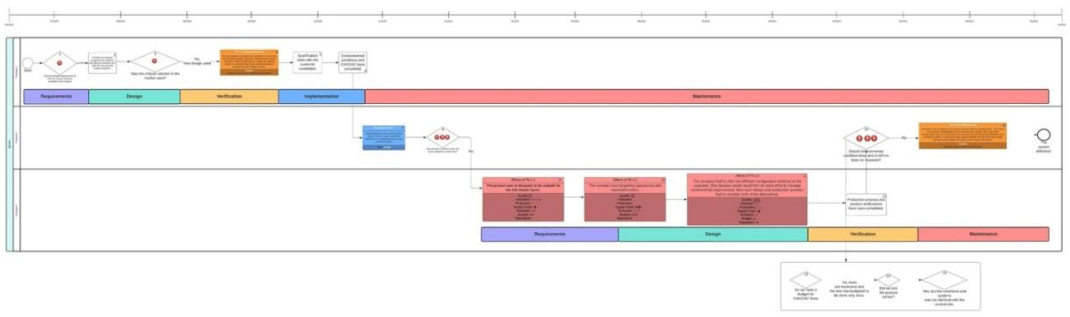
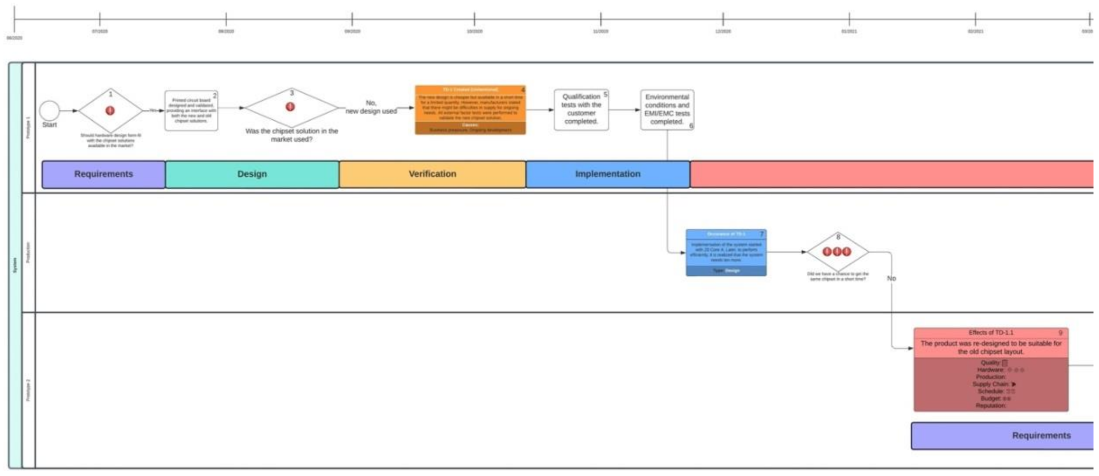


Figure 29 TDVM of Project #4

The larger version of the Figure 29 is given below in two parts.

Project #4 starts with a critical design strategy decision. The design team decided on the architectural solution strategy described in decision box one. The new design requires extra effort but creates an opportunity to use both chipset solutions. If the designer had not chosen a form-fit architecture, this would have created a technical debt. However, CD chose to eliminate this technical debt without creating it. After the backplane verification and validation process represented in box two designer chose a new chipset solution.

The supply chain department informed the designer that the current chip is difficult to obtain and cannot be repurchased in the short term. However, the designer continued with the existing chipset, which led to the creation of technical debt 1. As explained in box 4 in Figure 29, the design strategy also had to consider verification and validation tests. CD realized that they only planned these test scenarios for one option because of the budget and schedule constraints.



Start
1
Should hardware design form-fit with the chipset solutions available in the market?
Yes
2
Printed circuit board designed and validated, providing an interface with both the new and old chipset solutions.
3
Was the chipset solution in the market used?
No, new design used
4
Qualification tests with the customer completed.
5
Environmental conditions and EMI/EMC tests completed.
6
7
Type: Design
8
Did we have a chance to get the same chipset in a short time?
No
Effects of TD-1.1
9
The product was re-designed to be suitable for the old chipset layout.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
Requirements
Design
Verification
Implementation
Requirements
System
Production

Figure 30 TDVM of Project #4.1

Procurement of the old chipsets meant reexperiencing the product development cycle. Most importantly, CA had to operate the system with two different configurations during the system's lifespan. Effects of TD 1.1(represented in box 9), 1.2(represented in box 10), and 1.3(described in box 11) are shown in the.

Table 18 Effects of TD-1.3,2.1

Category	Effects of TD-1.3, TD-2.1:
Quality	Quality personnel effort is required for sub-unit level validation and verification studies Preparation of documentation and procedure for a quality control process of the new design
Hardware	Hardware design team effort is required for old chipset design. Preparation of documentation and production wbs for new design
Cost	R&D, quality, and supply chain personnel expenses
Supply Chain	The effort required for the supply of changing products
Schedule	System delivery schedule shifted three weeks for hardware design and verification

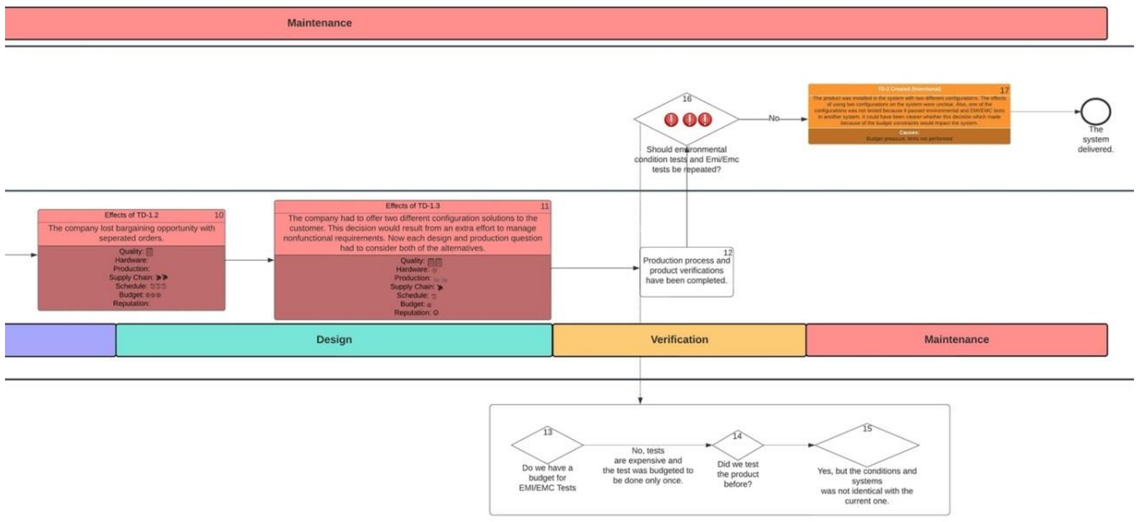

Maintenance
16
Should environmental condition tests and Emi/Emc tests be repeated?
No
17
The system delivered.
Effects of TD-1.2
10
The company lost bargaining opportunity with seperated orders.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
Effects of TD-1.3
11
The company had to offer two different configuration solutions to the customer. This decision would result from an extra effort to manage nonfunctional requirements. Now each design and production question had to consider both of the alternatives.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
12
Production process and product verifications have been completed.
Design
Verification
Maintenance
13
Do we have a budget for EMI/EMC Tests
No, tests are expensive and the test was budgeted to be done only once.
14
Did we test the product before?
15
Yes, but the conditions and systems was not identical with the current one.

Figure 31 TDVM of Project #4.2

As stated in boxes 13, 14, and 15, highly costly and lengthy tests were planned for only one option at the planning stage of project #4. The relevant chipset was verified on another system. It was decided to submit this report instead of repeating the test. As stated in box 16, environmental condition tests were not product-specific, and two different configurations in the system architecture created technical debt 2. Problems arising from Core A in the system would have to be managed separately for two different configurations. The effects of TD-1.3,2.1 represented in Table 19 and the effects of TD-1.2 represented in Table 20.

Table 19 Effects of TD-1.3,2.1

Category	Effects of TD-1.2
Quality	Efford is required to conduct an audit of the company.
Cost	Supply chain and quality personnel expenses Travel expenses of quality personnel The chipset was purchased at a high price for the supply of the product in a short time
Supply Chain	The effort required for the supply of changing products
Schedule	System delivery schedule shifted four weeks for chipset supply

Table 20 Effects of TD-1.2

Category	Effects of TD-1.2
Quality	Configuration management now had to be done for two products. All quality checks and qualifications had to be done with both products.
Hardware	Hardware analyzes and simulations were conducted to determine whether the two designs of the products work form-fit in the system efficiently.
Production	The production line stopped for the production of the new configuration for three weeks
Supply Chain	The supply chain must find suppliers for two alternative products during the system's lifecycle
Schedule	The system delivery schedule shifted two weeks for acceptance meetings of the new configuration
Cost	R&D, quality, and supply chain, production personnel expenses Product verification and development costs refunded
Reputation	The customer initially disapproved of a configuration change not presented in the project's initiation phase.

6.1.5. Background of Project #5

Table 21 Background information of Project#5

Case Subject:	13 years of engineering domain experience (Named as subject CE)
Company:	Company named as CA
Role:	Supply chain manager
Interview Duration:	47 minutes
Product / Sub-System:	Core A (Named as product PE)

CD's career started as a production planner who had found a chance to manage and coordinate system-level production plans. He has had the opportunity to work in the supply chain department as a manager. Working in CA since 2009, CD currently manages a team of 11 people. However, since this team is responsible for the supply of the entire factory, it interfaces with many different teams and departments. During the 52-minute Interview, CE reveals the acute effects of not managing technical debt from the perspective of the supply chain department.

The participants of our research study prioritized planning and procurement as the most problematic issue in the last five years in critical system development. This interview is essential as it approaches the TD topic from the perspective of a supply chain and production planning professional.

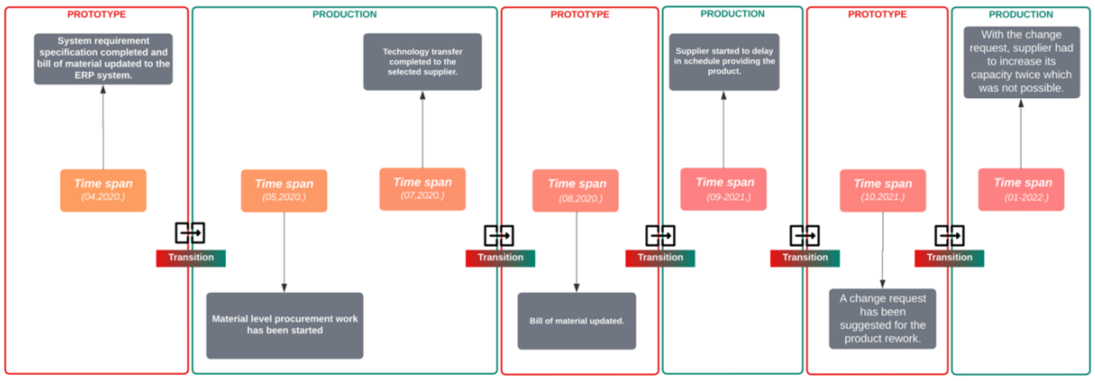

PROTOTYPE
System requirement specification completed and bill of material updated to the ERP system.
Time span (04.2020.)
Transition
PRODUCTION
Time span (05.2020.)
Material level procurement work has been started
Technology transfer completed to the selected supplier.
Time span (07.2020.)
Transition
PROTOTYPE
Time span (08.2020.)
Bill of material updated.
Transition
PRODUCTION
Supplier started to delay in schedule providing the product.
Time span (09-2021.)
Transition
PROTOTYPE
Time span (10.2021.)
A change request has been suggested for the product rework.
Transition
PRODUCTION
With the change request, supplier had to increase its capacity twice which was not possible.
Time span (01-2022.)

Figure 32 Timeline of Project #5

The project team experienced frequent oscillations between prototype and production processes starting from the system design phase. These transitions, which are expected in the system development lifecycle, become very difficult to manage with the start of mass production. While mass production expects clear documents and procedures, system development requires experiments, prototypes, and updates from the outputs of that studies.

As CE stated, the effective management of the project schedule becomes risky when faced with problems in subcontractor management and change decisions from the system architecture simultaneously. Moreover, in this scenario, it is decided to move forward with a supplier that cannot handle the job. System delivery is at risk when this limited-capacity supplier cannot respond to additional system-level requests.

6.1.5.1. Findings of Project #5

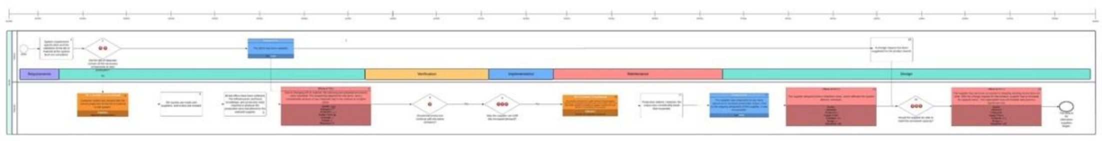


Figure 33 TDVM of Project #5

The larger version of the Figure 33 is given below in two parts.

Bill of material is critical in accurately detecting and monitoring milestones during the development and production phases of the system. As stated in the first three boxes, Project 5 started before transferring the bill of material to the ERP due to the tight delivery schedule and the intertwining of production and development processes. The decision to initiate the production assumed that the BOM would not change. However, the fact that the system structure was complex kept the possibility of change quite significant.

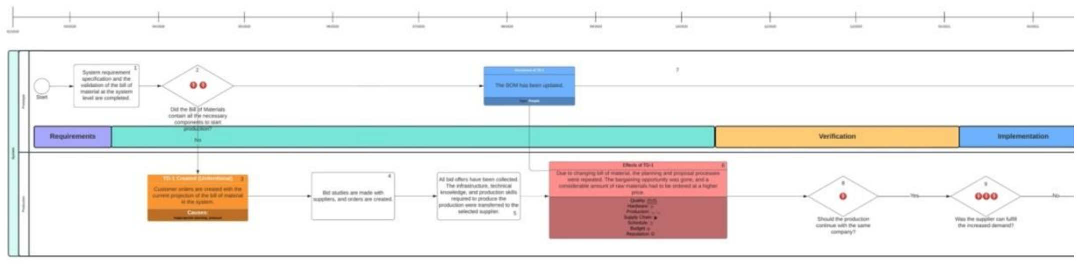

Start
System requirement specification and the validation of the bill of material at the system level are completed.
Did the Bill of Materials contain all the necessary components to start production?
No
The BOM has been updated.
Requirements
Verification
Implementation
TD-1 Created (Unintentional)
Customer orders are created with the current projection of the bill of material in the system.
Causes:
Bid studies are made with suppliers, and orders are created.
All bid offers have been collected. The infrastructure, technical knowledge, and production skills required to produce the production were transferred to the selected supplier.
Effects of TD-1
Due to changing bill of material, the planning and proposal processes were repeated. The bargaining opportunity was gone, and a considerable amount of raw materials had to be ordered at a higher price.
Quality
Hardware
Production
Supply Chain
Schedule
Budget
Reputation
Should the production continue with the same company?
Yes
Was the supplier can fulfil the increased demand?
No

Figure 34 TDVM of Project #5.1

The proposal for the change caused the first technical debt to be created. The risk could have caused a loss in the effort to transfer the methods and technology to the selected supplier. BOM change also required an increased number of raw materials and subunits. The expected production capacity had nearly doubled with this change. The effects of this change are represented in Figure 34:

Table 22 Effects of TD-1

Category	Effects of TD-1
Quality	. Efford required for all quality checks and qualifications
Hardware	Hardware analyzes and simulations were conducted to determine an alternative for the sytem due to materials with long lead times.
Production	The production line stopped for the rework process
Supply Chain	Efford required for the bidding process.
Schedule	The system delivery schedule shifted two weeks fort he supply of new materials
Cost	R&D, quality, and supply chain, production, hardware personnel expenses Cost of scrapped raw materials Cost of production stoppage
Reputation	The customer realized that system delivery schedule delayed due to the rework.

Increasing needs in system BOM required answering two questions. Would the CA continue with the same supplier, and would this supplier be able to meet its production capacity? The decision to continue with the same supplier created technical debt 2. Because the supplier could not produce the expected amount, it was expected to increase its capacity by two times. However, the CA continued with the same supplier to save time during technology and method transfer again. However, this decision depended on the supplier's ability to achieve the required increased capacity.

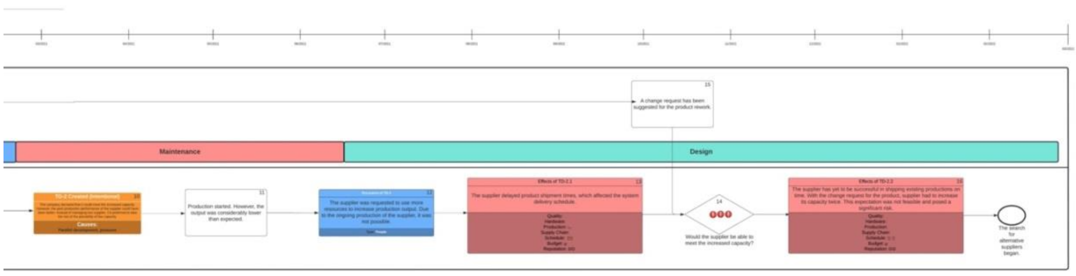

15
A change request has been suggested for the product rework.
Maintenance
Design
TD-2 Created (Intentional)
10
Causes
11
Production started. However, the output was considerably lower than expected.
12
The supplier was requested to use more resources to increase production output. Due to the ongoing production of the supplier, it was not possible.
Effects of TD-2.1
13
The supplier delayed product shipment times, which affected the system delivery schedule.
Quality
Hardware
Production
Supply Chain
Schedule
Budget
Reputation
14
Would the supplier be able to meet the increased capacity?
Effects of TD-2.2
16
The supplier has yet to be successful in shipping existing productions on time. With the change request for the product, supplier had to increase its capacity twice. This expectation was not feasible and posed a significant risk.
Quality
Hardware
Production
Supply Chain
Schedule
Budget
Reputation
The search for alternative suppliers began.

Figure 35 TDVM of Project #5.2

The expected risk was realized, and the supplier needed help to deliver the requested capacity increase. Late-delivered products postponed the systems delivery schedule. The effects of technical debt 2.1 and 2.2 are represented in the and Table 24.

Table 23 Effects of TD-2.1

Category	Effects of TD-2.1
Production	Production resources could have been planned more effectively because the allocated resources could be used for different production lines.
Schedule	System delivery schedule was delayed two weeks.
Cost	The production line stopped for two additional weeks.
Reputation	The customer had to wait extra time for the rework.

Table 24 Effects of TD-2.2

Category	Effects of TD-2.2
Schedule	With the current capacity, three-week delay in the system delivery is expected.
Cost	The project was penalized for delay.
Reputation	It was understood that there was no chance that the production goals could not be achieved with the current supplier.

6.1.6. Background of Project #6

Table 25 Background information of Project#6

Case Subject:	17 years of project manager experience (Named as subject CF)
Company:	Company named as CA
Role:	Project Manager
Interview Duration:	84 minutes
Product / Sub-System:	Electronic Sub Unit B (Named as product PE)

CF had a broad range of experience in different domains, which include design, system, and project management. That career path provided an opportunity to experience and manage each cycle of product and system development. Today CF manages a project team of 80+ people. He is a critical actor in the project management activities of CA. During the 84 minutes interview, CF explained that indicating mass production for a high-tech product requires managing risks and TD.

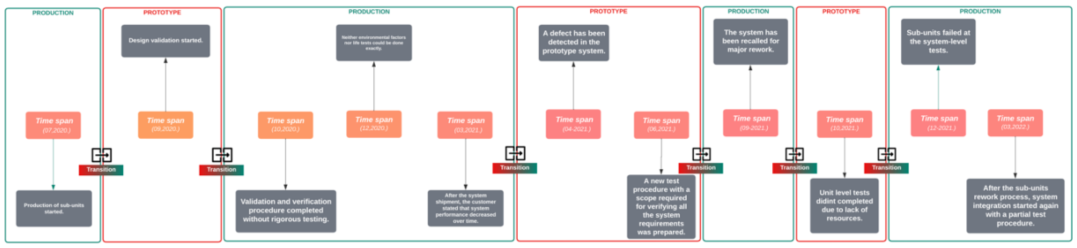

PRODUCTION
Time span (07.2020.)
Production of sub-units started.
Transition
PROTOTYPE
Design validation started.
Time span (09.2020.)
Transition
PRODUCTION
Neither environmental factors nor life tests could be done exactly.
Time span (10.2020.)
Validation and verification procedure completed without rigorous testing.
Time span (12.2020.)
Time span (03.2021.)
After the system shipment, the customer stated that system performance decreased over time.
Transition
PROTOTYPE
A defect has been detected in the prototype system.
Time span (04-2021.)
Time span (06.2021.)
A new test procedure with a scope required for verifying all the system requirements was prepared.
Transition
PRODUCTION
The system has been recalled for major rework.
Time span (09-2021.)
Transition
PROTOTYPE
Time span (10.2021.)
Unit level tests didint completed due to lack of resources.
Transition
PRODUCTION
Sub-units failed at the system-level tests.
Time span (12-2021.)
Time span (03.2022.)
After the sub-units rework process, system integration started again with a partial test procedure.

Figure 36 Timeline of Project #6

Project #6 differs from other projects with the repetitive transitions between the system's production and the prototype's development. System development processes are represented in Figure 36 Timeline of Project #6. During Project #6, TD decisions result in the recall of the systems from the customer.

6.1.6.1. Findings of Project #6

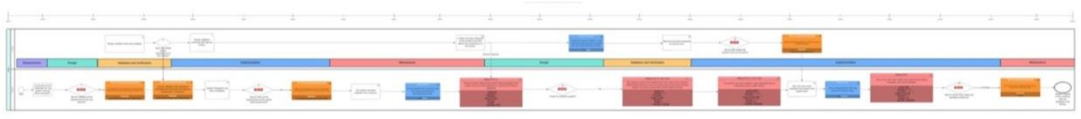


Figure 37 TDVM of Project #6

The larger version of the Figure 37 is given below in three parts.

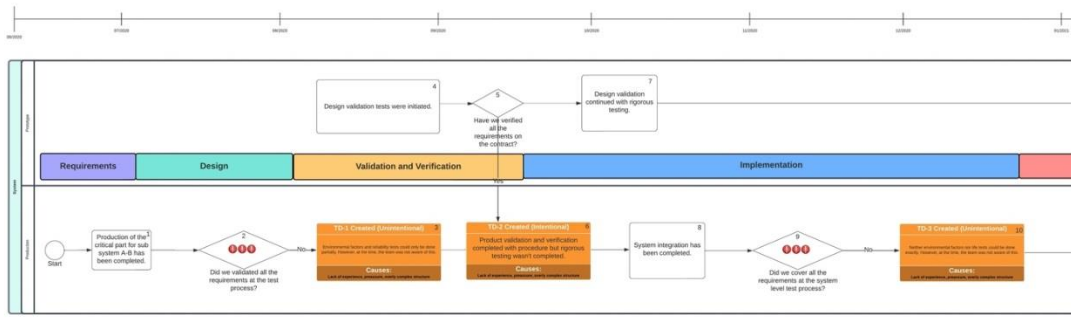
Project 68 timeline, represented in Figure 37, started with the production of PE. Complex system development requires the integration of sub-units with various interfaces with other sub-systems. Sub-System B is one of the system's central parts, and the architects and managers had to control and consider the effect of other modules and the factors. That requires excellent communication and sharing of knowledge between the project stakeholders. In order to provide a system and unit-level tests, designers had to consider constraints coming from the other interfaces and the sub-systems.

The question in box 2 in Figure 37. tries to find an answer if the test procedure covers all the requirements and factors. The nature of complexity comes with unexpected and unpredictable risks for the system developers. However, the project team chose to complete a partial test procedure for the PE, although they were aware of the possibility of additional risks, which created TD1, represented in Figure 37, box 3. The real reason behind that decision was that the validation studies still needed to be completed when the system integration phase proceeded to the test level. The production team also decided to move to the next iteration without the result of rigorous testing studies because of the project's tight schedule. That was the reason behind TD2, shown in Figure 37, box 6. The design team initiated rigorous testing with the prototype simultaneously with the system integration. The system integration was completed, and the system-level test started with the modules and sub-units. Rigorous studies continued while the system tests began. As a result:

TD1 born, accepting the system test procedure without being aware of the environmental factor effects,

TD2 was created with the advancing integration at the sub-unit level without waiting for the rigorous tests to complete,

TD3 was born from advancing system-level integration without waiting for rigorous tests to complete.



Design validation tests were initiated.
4
5
Have we verified all the requirements on the contract?
7
Design validation continued with rigorous testing.
Requirements
Design
Validation and Verification
Implementation
Yes
Start
Production of the critical part for sub system A-B has been completed.
1
2
Did we validated all the requirements at the test process?
No
TD-1 Created (Unintentional)
3
Causes:
TD-2 Created (Intentional)
6
Product validation and verification completed with procedure but rigorous testing wasn't completed.
Causes:
System integration has been completed.
8
9
Did we cover all the requirements at the system level test process?
No
TD-3 Created (Unintentional)
10
Causes:

Figure 38 TDVM of Project #6.1

Due to unknown factors system was shipped to the customer. The problems can be summarized under two significant categories: partially implemented tests and the probability of occurrence of environmental factors. That risk may even result in the system's recall from the customer.

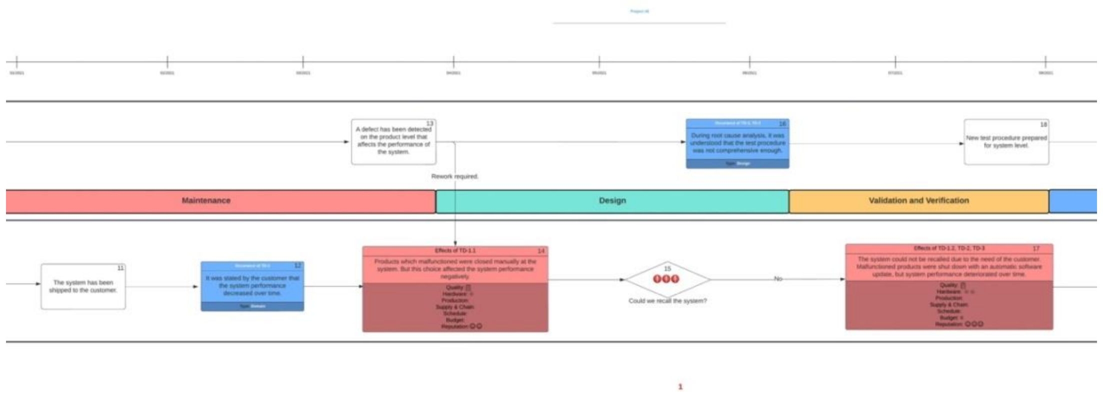

A defect has been detected on the product level that affects the performance of the system.
13
During root cause analysis, it was understood that the test procedure was not comprehensive enough.
16
New test procedure prepared for system level.
18
Rework required.
Maintenance
Design
Validation and Verification
The system has been shipped to the customer.
11
It was stated by the customer that the system performance decreased over time.
12
Effects of TD-1.1
14
Products which malfunctioned were closed manually at the system. But this choice affected the system performance negatively.
Quality:
Hardware:
Production:
Supply & Chain:
Schedule:
Budget:
Reputation:
15
Could we recall the system?
No
Effects of TD-1.2, TD-2, TD-3
17
The system could not be recalled due to the need of the customer. Malfunctioned products were shut down with an automatic software update, but system performance deteriorated over time.
1

Figure 39 TDVM of Project #6.2

After the system's integration with the customer, PE meets the expectations and requirements. While the system was operating, the customer realized that the system's overall performance started to decrease over time. That was the occurrence of TD1. The system team created a temporary solution by manually turning off the malfunctioned PEs. The effects of TD1 are represented in the Table 26:

Table 26 Effects of TD-1.1

Category	Effects of TD-1.1
Quality	Customers' reputation in the overall quality of the product was damaged. It was examined why the problems that cause reliability and performance loss were not examined in the tests before the shipment of the system.
Hardware	Efford was required to design a controller for malfunctioning units without requiring manual operations.
Cost	Cost of quality, hardware and production personnel expenses. Cost of implementing the change request at the operation side.
Reputation	The system could not perform its requirements effectively. The customer expects a solution for a system immediately required from the operators.

In the meantime, the rigorous tests showed a need for a significant change at the PE level. Also, it was understood that the health of the system architecture would be affected negatively if the requested change was implemented after some time. Cause and effect analysis shows the test procedure had to be prepared in more detail, and the requirements' coverage has to be broader. The occurrence of TD2 and TD3 was revealed with the request for a new test procedure. It was also understood that the system had to be recalled for modifications and re-tests.

The system was mission-critical for the customer. The status of the project made the recall decision hard to operate. Also, that would result in the enormous risk of losing the customer's reputation. To solve the issue and create a short time, the system team initiated a software update that controls the effects of the PEs at the system level.

With that update, the system would continue its operations, but the efficiency would decrease stochastically over time. That was the first effect of TD1.2, TD2, and TD3, shown in Figure 39, box 17. As expected, the project management decided that the system had to be recalled due to the loss of performance, represented in Figure 39, box 20. Table 27 shows the effects of TD1.2, 2, and 3:

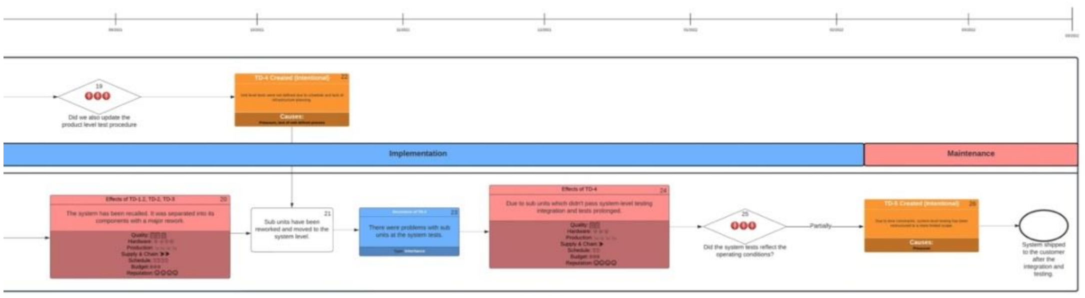

19
Did we also update the product level test procedure
TD-4 Created (Intentional)
22
Causes:
Implementation
Maintenance
Effects of TD-1.2, TD-2, TD-3
20
The system has been recalled. It was separated into its components with a major rework.
Quality
Hardware
Production
Supply & Chain
Schedule
Budget
Reputation
21
Sub units have been reworked and moved to the system level.
23
There were problems with sub units at the system tests.
Effects of TD-4
24
Due to sub units which didn't pass system-level testing integration and tests prolonged.
25
Did the system tests reflect the operating conditions?
Partially
TD-5 Created (Intentional)
26
Causes:
System shipped to the customer after the integration and testing.

Figure 40 TDVM of Project #6.3

Table 27 Effects of TD-1.2, 2, 3

Category	Effects of TD-1.2, 2.3
Quality	The quality personnel investigated each operation to be sure to perform everything successfully. Due to the schedule and budget pressure, there were no options for another rework.
Hardware	The hardware design team prepared a detailed documentation for the disassembly of the system.
Cost	R&D, quality, and supply chain, production, hardware personnel expenses Cost of scrapped raw materials Cost of production stoppage Cost of penalty
Supply Chain	Bidding studies were made to supply materials added to the change request and scrapped units.
Schedule	System delivery schedule delayed 2 months for rework.
Reputation	The project was penalized for delay.
Production	Production capacity plans were affected negatively by the new rework expectation. The fact that a completed job would be back to the starting point damaged the morale of the production personnel.

The primary source of the problem had to be identified and solved at the unit level, but the design and test team still needed to update the unit-level test procedure. That failure in planning the test at the unit level created TD-4. The rework was initiated, and the change request was completed at the product PE level. All units passed the tests, which did not include the operations required to verify the change. System integration started again, and at the system tests, some of the performance of the sub-units could have provided the expected results. Faulty products were disassembled and reworked again. Since there was no infrastructure at the unit level, verifications of the PEs had to continue at the system level. Each product must be transferred to a new location and integrated into the system to initiate unit tests. Effects of TD-4 represented in the table:

Table 28 Effects of TD-4

Category	Effects of TD - 4
Quality	Each quality control operation is repeated for every product reworked.
Hardware	The design team had to be involved in the testing process because the defined scope at the unit level does not present the system's needs. They had to modify tests case by case because an automatic software update required massive time and effort, which the project did not have.
Cost	R&D, quality, and supply chain, production, hardware personnel expenses Cost of scrapped raw materials Cost of production stoppage Cost of penalty
Supply Chain	Bidding studies continued for the supply of the needs of reworked units.
Production	Production capacity plans were affected negatively by the re-test and rework operations. Due to faulty design decisions, challenging targets were imposed on the production personnel.
Schedule	The system delivery schedule was delayed two months for re-test processes.
Reputation	The project was penalized for delay. The customer's reputation against the product was negatively affected because of the 4-month reworked process.

After completing the new test procedure and change request, the system was ready. Still, the project team needed to provide operation and environmental conditions for the system test procedure. These requirements come from the company's past experiences rather than the customer's agreement. The project team recalled the system, and the delivery was delayed. In light of these conditions, the company decided to design a prototype and conduct these tests in parallel. The system was shipped to the customer. With that decision, the TD5 shown in Figure 40. in box four was born. TD-5 was critical as an indication that technical debts still need to be managed and monitored even if the system is delivered.

6.1.7. Background of Project #7

Table 29 Background information of Project#7

Case Subject:	21 years of system manager experience (Named as subject CG)
Company:	Company named as CA
Role:	System Manager
Interview Duration:	56 minutes
Product / Sub-System:	Electronic Sub Unit A (Named as product PF)

Project #7 reveals the importance of the management of TDs when more than one system is integrated and produced simultaneously. According to CG, the effects of TD decisions at any phase of the development cycle increased proportionally with the number of integrated systems. CG has tremendous system management background, which has been experienced in CA for more than 20 years. The challenge of project #7 comes from the increased size of the teams and the system itself. A change request discovered in the project's later phases may require massive cost and time to complete. Since CA's two most extensive product lines are reserved for developing project #7's systems, any problem resulting in the production line stoppage would not be accepted.

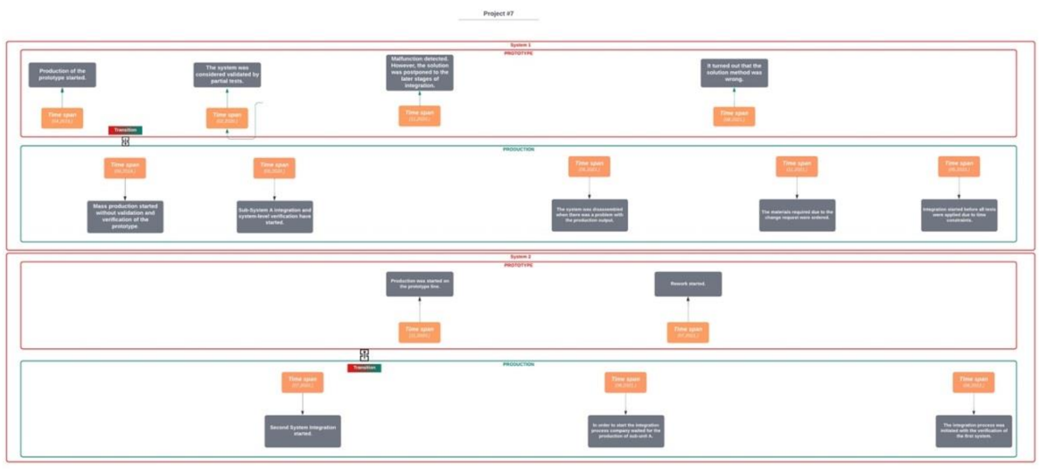

Project #7
System 1
PROTOTYPE
Production of the prototype started.
The system was considered validated by partial tests.
Malfunction detected. However, the solution was postponed to the later stages of integration.
It turned out that the solution method was wrong.
Time span
Transition
PRODUCTION
Mass production started without validation and verification of the prototype
Sub-System A integration and system-level verification have started.
The system was disassembled when there was a problem with the production output.
The materials required due to the change request were ordered.
Integration started before all tests were applied due to time constraints.
System 2
PROTOTYPE
Production was started on the prototype line.
Rework started.
PRODUCTION
Second System Integration started.
In order to start the integration process company waited for the production of sub-unit A.
The integration process was initiated with the verification of the first system.

Figure 41 Timeline of project #7

The first swimlane in Figure 41. represents the timeline of system one, while the second lane represents the other. During the R&D and production phases, there were decisions taken for the management of TD. The result of that decisions affects both systems' resource and project plans. The company has to resolve the issues and initiate the rework process for both, which brings increased effort for the project team. During the 56 minutes interview, it was stated that decisions taken to eliminate risks became the actual threats behind the delay of the system's schedule.

6.1.7.1. Findings of Project #7

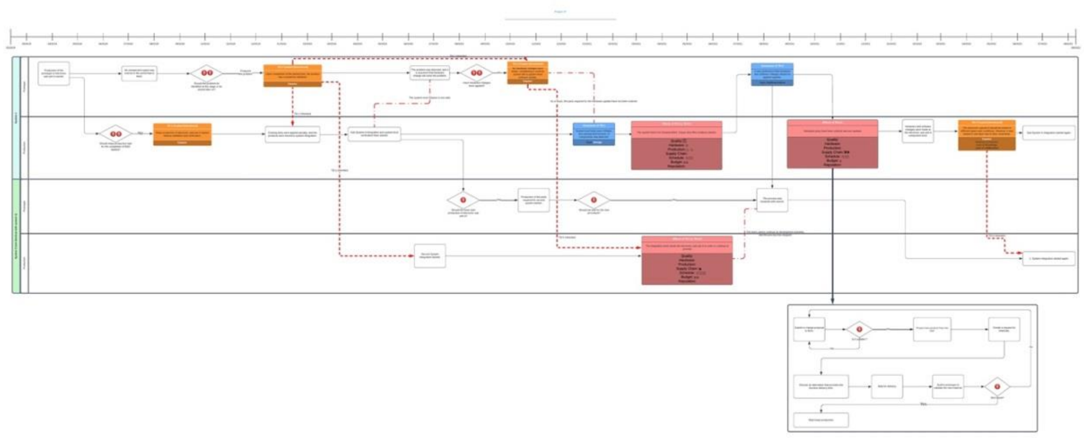
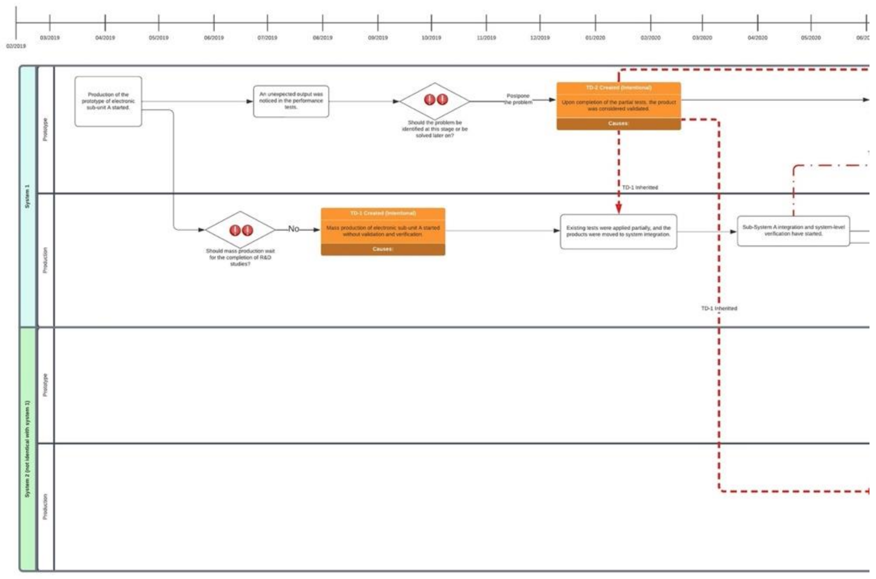


Figure 42 TDVM of Project #7

The timeline represented in Figure 42. started in early 2019. After the production of the prototype electronic sub unit A, raw material supply and mass production have been created without waiting for the completion of the verification and validation phases. That was the reason behind the creation of TD-1.

It was stated that while the start of the production decision was taken, the design team discovered a problem with the prototype, which did not share with the project team. Communication of that information may even prevent technical debt, but the design team chooses to postpone the studies to find the solution.



02/2019
03/2019
04/2019
05/2019
06/2019
07/2019
08/2019
09/2019
10/2019
11/2019
12/2019
01/2020
02/2020
03/2020
04/2020
05/2020
System 1
Prototype
Production
System 2 (not identical with system 1)
Production of the prototype of electronic sub-unit A started.
An unexpected output was noticed in the performance tests.
Should the problem be identified at this stage or be solved later on?
Postpone the problem
TD-2 Created (Intentional)
Upon completion of the partial tests, the product was considered validated.
Causes:
TD-1 Inherited
Should mass production wait for the completion of R&D studies?
No
TD-1 Created (Intentional)
Mass production of electronic sub-unit A started without validation and verification.
Causes:
Existing tests were applied partially, and the products were moved to system integration.
Sub-System A integration and system-level verification have started.
TD-1 Inherited

Figure 43 TDVM of Project #7.1

They should have realized the importance of the malfunction and the postponement of the solution decision that created TD-2, with the alternative partial testing method developed by the design team system verified and moved to integration.

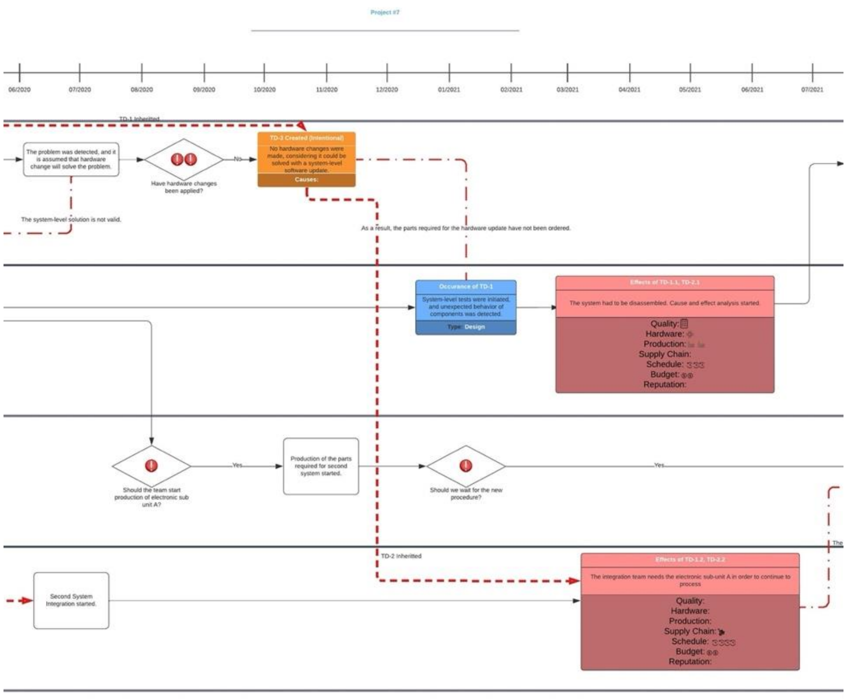

Project #7
06/2020
07/2020
08/2020
09/2020
10/2020
11/2020
12/2020
01/2021
02/2021
03/2021
04/2021
05/2021
06/2021
07/2021
TD-1 Inherited
The problem was detected, and it is assumed that hardware change will solve the problem.
Have hardware changes been applied?
No
TD-2 Created (Intentional)
No hardware changes were made, considering it could be solved with a system-level software update.
Causes:
The system-level solution is not valid.
As a result, the parts required for the hardware update have not been ordered.
Occurance of TD-1
System-level tests were initiated, and unexpected behavior of components was detected.
Type: Design
Effects of TD-1.1, TD-2.1
The system had to be disassembled. Cause and effect analysis started.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
Should the team start production of electronic sub unit A?
Yes
Production of the parts required for second system started.
Should we wait for the new procedure?
Yes
The
TD-2 Inherited
Second System Integration started.
Effects of TD-1.2, TD-2.2
The integration team needs the electronic sub-unit A in order to continue to process
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:

Figure 44 TDVM of Project #7.2

Verification of the first system initiated the production of second systems units. In the meantime, the development team discovered the reason behind the malfunction and developed a solution that did not require hardware changes.

Software updates are usually a better alternative for system developers because hardware changes may require a supply of the materials with production personnel effort. The choice alternative, which did not require a hardware change, was created TD3 because the software solution was needed to solve the problem.

System-level tests were initiated for the first system, and unexpected behaviour of components was detected. That was the occurrence of TD-1. The system was disassembled to analyze the behaviour and find the reason behind the malfunction. Second systems integration required sub-product A to continue, resulting from a stoppage in production. The design team stated that hardware and software changes should be applied together. That discovery was the occurrence of TD-2. The effects of TD-1.1,1.2, 2.1 and 2.2 are represented in the Table 30:

Table 30 Effects of TD-1.1,1.2, 2.1,2.2

Category	**TD-1.1, 1.2**
Schedule	First system delivery delayed for 3 months.
Cost	The first system production line stopped for three months. R&D, quality, supply chain, production and hardware personnel expenses
Production	The stoppage of 2 production lines consumed the capacity plans of the company. Efford is required for the disassembly of the system.
Hardware	The hardware design team prepared a detailed documentation for the disassembly of the system. Efford is required to conduct cause and effect analysis.
Quality	Efford is required for the preparation of the quality control procedures.
Category	**TD-1.2, 2.2**
Schedule	The project's delivery plan depends on the second production line was delayed for three months.
Cost	The second system production line stopped for three months. Cost of supply chain and production personnel expenses.
Supply Chain	Bidding studies were made to supply materials (software licenses) added to the change request.

To initiate a change request, required materials had to supply immediately. Supply of the materials for change requests brings considerable effort to the supply chain team. The details about the process represented in the extra lane are presented in Figure 44. Sourcing raw materials initiated the rework process for the change request linked to electronic sub-unit A's malfunction. Technical Debt 4 was created because of the partial test implementation decision. Since the schedule of the system integration was delayed with the change request, the project team has to find a way to provide a solution for catching up with the project delivery plan. Test coverage changed with a limited procedure to achieve the system delivery goal.

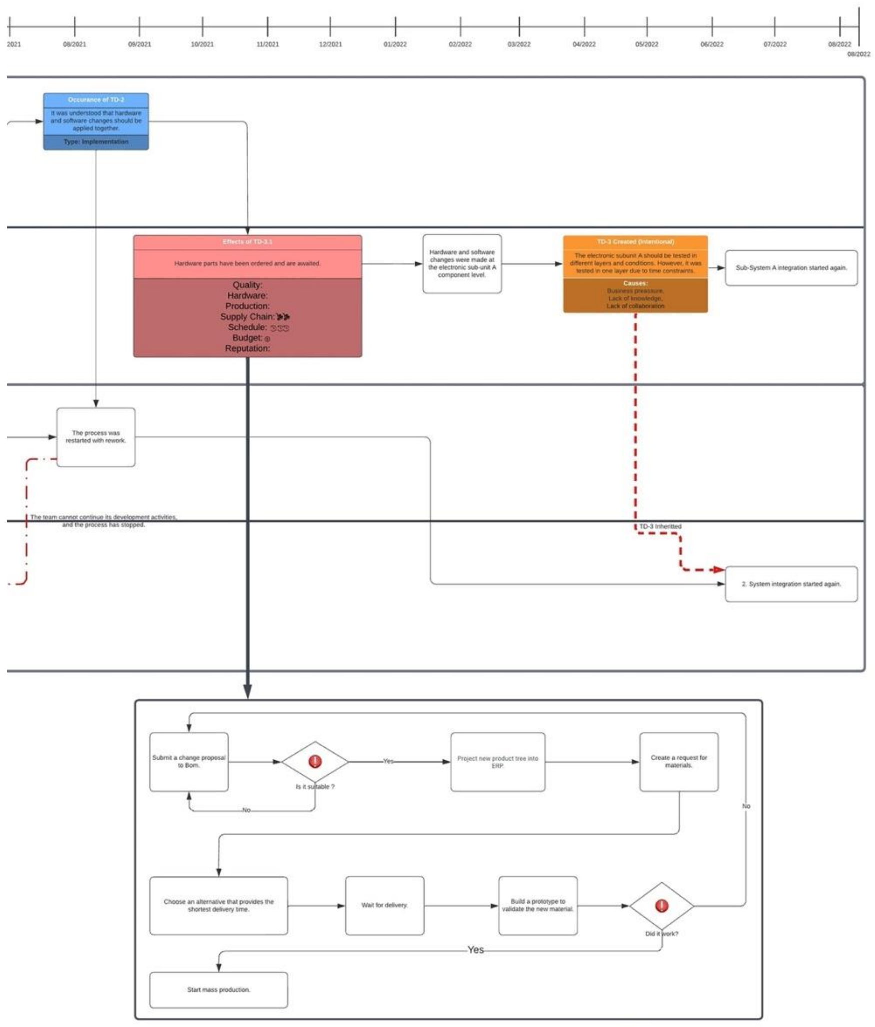

2021
08/2021
09/2021
10/2021
11/2021
12/2021
01/2022
02/2022
03/2022
04/2022
05/2022
06/2022
07/2022
08/2022
08/2022
Occurance of TD-2
It was understood that hardware and software changes should be applied together.
Type: Implementation
Effects of TD-3.1
Hardware parts have been ordered and are awaited.
Quality:
Hardware:
Production:
Supply Chain:
Schedule:
Budget:
Reputation:
Hardware and software changes were made at the electronic sub-unit A component level.
TD-3 Created (Intentional)
The electronic subunit A should be tested in different layers and conditions. However, it was tested in one layer due to time constraints.
Causes:
Business preassure,
Lack of knowledge,
Lack of collaboration
Sub-System A integration started again.
The process was restarted with rework.
The team cannot continue its development activities, and the process has stopped.
TD-3 Inheritted
2. System integration started again.
Submit a change proposal to Bom.
Is it suitable ?
Yes
No
Project new product tree into ERP.
Create a request for materials.
No
Choose an alternative that provides the shortest delivery time.
Wait for delivery.
Build a prototype to validate the new material.
Did it work?
Yes
Start mass production.

Figure 45 TDVM of Project #7.3

The effects of TD-3 are represented in Table 31 Effects of TD-3.1:

Table 31 Effects of TD-3.1

Category	TD-1.1, 1.2
Schedule	The project's delivery plan depends on the second production line was delayed for three months. First system delivery schedule was delayed for three months.
Cost	The second system production line stopped for three months. The first system production line stopped for three months. Cost of supply chain personnel expenses. Cost of scrapped materials.
Supply Chain	Bidding studies were made to supply materials (software licenses) added to the change request.

6.1.8. Background of Project #8

Table 32 Background information of Project #8

Case Subject:	7 years of project manager experience (Named as subject CH)
Company:	Company named as CA
Role:	System Manager
Interview Duration:	93 minutes
Product / Sub-System:	Sub System A Core (Named as product PI)

Unlike other interviewers, PI's professional life started in a different company. After three years of supply and chain experience in the automotive industry, he transferred to the CA as a project engineer. The experience of coming from an industry that does not serve space and defense was essential because a broad perspective could create effective results for answering the thesis research questions. PI worked as a project engineer for two years on the project he talked about in the interview. His successful decisions and actions made him attented as project manager.

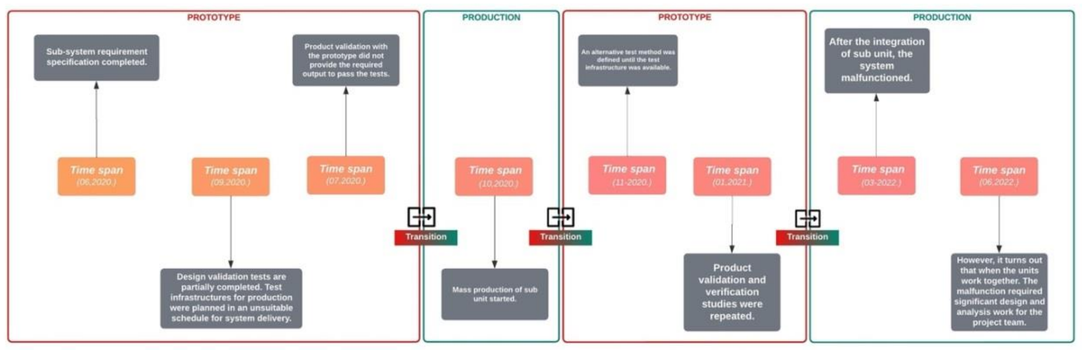

PROTOTYPE
Sub-system requirement specification completed.
Time span (06.2020.)
Time span (09.2020.)
Design validation tests are partially completed. Test infrastructures for production were planned in an unsuitable schedule for system delivery.
Time span (07.2020.)
Product validation with the prototype did not provide the required output to pass the tests.
Transition
PRODUCTION
Time span (10.2020.)
Mass production of sub unit started.
Transition
PROTOTYPE
An alternative test method was defined until the test infrastructure was available.
Time span (11-2020.)
Time span (01.2021.)
Product validation and verification studies were repeated.
Transition
PRODUCTION
After the integration of sub unit, the system malfunctioned.
Time span (03-2022.)
Time span (06.2022.)
However, it turns out that when the units work together. The malfunction required significant design and analysis work for the project team.

Figure 46 Timeline of project #8

Project #8 in Figure 46. reveals how big an impact hardware-related technical debt can have if not managed. Because of the unplanned hardware infrastructure, a team prepared an alternative test method that requires both hardware and software design. Since the development baseline did not include verification and validation for extra infrastructure, some vital steps failed to complete. Inheritance of technical debt created new technical debts, resulting in a chaotic system management environment for development decisions.

6.1.8.1. Findings of Project #8

Figure 47 TDVM of project #8

Project #8 timeline started with the requirement specification at the system level. With the detailed WBS structure, electronic sub-unit B requirements are determined. The complex architecture of the system required extra tests and analysis, which the customer could not specify because this experience comes from domain-specific knowledge. First technical debt was created with the decision to skip some of the design verification tests. Especially high tech developing designs required rigorous tests, which had to be planned before because of the requirement of a high budget and long process times.

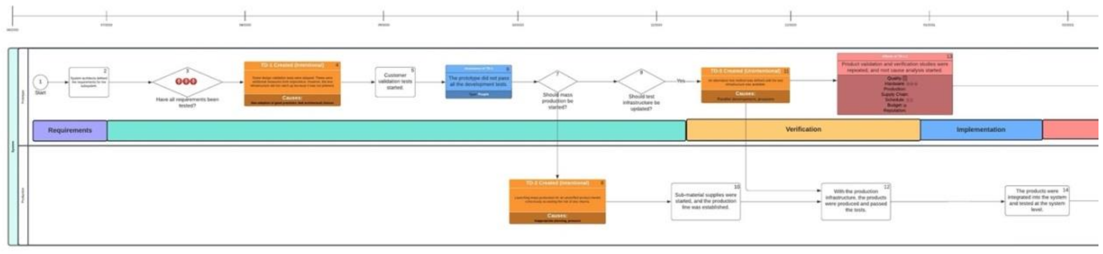

Start
Have all requirements been tested?
TD-1 Created (Intentional)
Causes:
Customer validation tests started.
The prototype did not pass all the development tests.
Should mass production be started?
Should test infrastructure be updated?
Yes
TD-3 Created (Intentional)
Causes:
Product validation and verification studies were repeated, and root cause analysis started.
Quality
Hardware
Production
Supply Chain
Schedule
Budget
Reputation
Requirements
Verification
Implementation
TD-2 Created (Intentional)
Causes:
Sub-material supplies were started, and the production line was established.
With the production infrastructure, the products were produced and passed the tests.
The products were integrated into the system and tested at the system level.

Figure 48 TDVM of project #8.1

Information at Box 6 in Figure 48 shows that the prototype failed at customer tests. This is the first time when company A realizes TD-1. Even so, the project management team initiated mass production. That decision revealed TD-2 because postponing the problem-solving means accepting any returns from the possible change request that may come in the future. Mass production also initiated the supply of all required materials in the bill of materials. Also, one of the reasons behind the failure of the customer verification test was the unplanned infrastructure. With the current infrastructure, the development team only has the option to complete some tests and analyses. The unplanned infrastructure required extra time in the system schedule, forcing the design team to create an alternative test method and infrastructure. The reason behind that option was the inheritance of the TD-2. TD-3 was created because of the non-standard alternative test method and infrastructure.

The system-level production tests were completed with the alternative infrastructure and test methods. While the mass production continued, the development team tried to verify the new method and infrastructure. At the beginning of the project, using an alternative was the best option from the team leader's perspective, but now it was evident that the partial test method required massive verification and validation efford and studies. Table 33 represents the effects of TD-1.1:

Table 33 Effects of TD-1.1

Category	Effects of TD-1.1
Quality	. The effort required for quality checks and qualifications of the new procedure
Hardware	Hardware analyzes and simulations are needed to design the alternative.
Schedule	The design team lost three weeks for the root cause analysis
Cost	Extra budget required fort he design of new model R&D, quality, and hardware personnel expenses

The designers realized that product passed all unit-level tests but failed at system integration. Analysis revealed that the alternative test method covered only some of the required verification and validation steps. The company CA faced two Technical Debts. TD-2 occurred when the system failed at the integration, and TD-3 occurred with the realization of the unplanned test infrastructures effect.

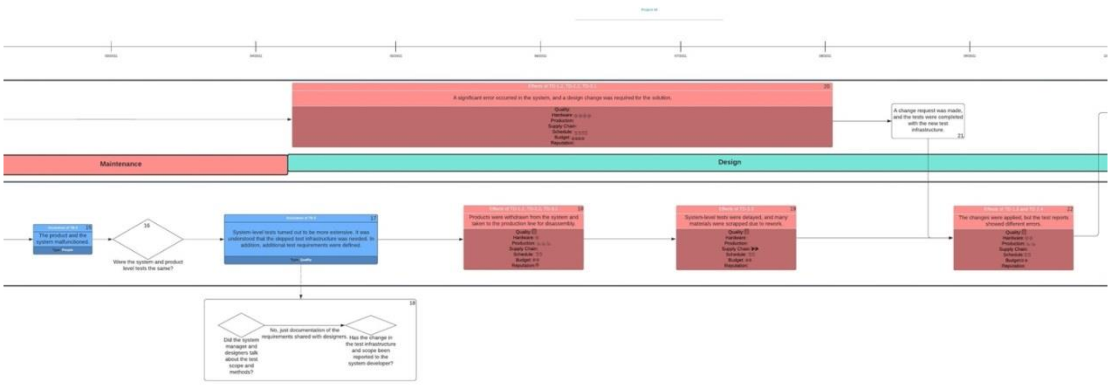


Figure 49 TDVM of project #8.2

The designers realized that product passed all unit-level tests but failed at system integration. Analysis revealed that the alternative test method covered only some of the required verification and validation steps. The company CA faced two Technical Debts. TD-2 occurred when the system failed at the integration, and TD-3 occurred with the realization of the unplanned test infrastructures effect.

The project team had to manage a domino effect of TDs, with the nonefficient management of the TD system architects faced with complex decisions and constraints which require a tremendous amount of engineering effort to understand. In the meantime, the system's delivery schedule becomes tight, affecting system and project managers' decisions about the integration. Produced Sub Products behave individually and require job-shop manufacturing. They passed all the unit tests but never worked as a system, and their interfaces were never tested with the high-level components. However, it turns out that when the units work together, they affect each other's performance, negatively affecting the system's nonfunctional requirements. Box 26 and 27 show the decision and the occurrence of TD-4, which can be seen in Figure 49

The effects of the TD that was detailed before are represented in the Table 34.

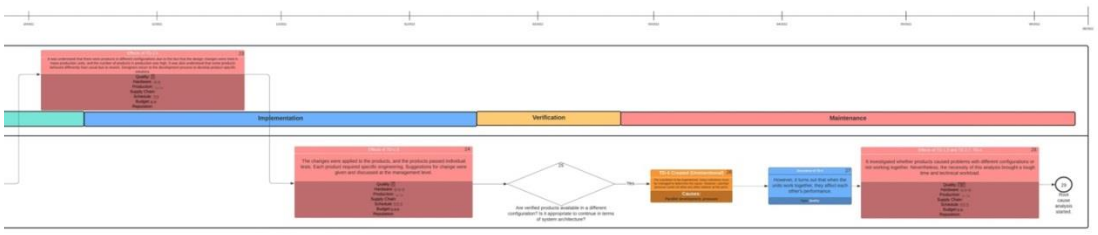


Figure 50 TDVM of project #8.3

Table 34 Effects of TD-1.2,2.1,3.1,2.3,2.4, 2.5,2.6,1.3,4,2.7

Category	Effects of TD-1.2,2.1,3.1,2.3,2.4
Quality	. Efford required for material level quality control, The effort required for quality checks and qualifications of the new procedure
Hardware	An engineering study has been performed to understand the behavior of the system. Data must be collected at the unit level that was not stored because of the unplanned infrastructure. To identify the exact cause behind the malfunction, hardware teams focus on creating the problem in the development environment. After the implementation of the change there were unexpected outputs and behaviors in the system. The design team conduct a root and cause study for finding the issue behind it.
Schedule	The system delivery delayed for 4 months.
Cost	Costs of the scrapped materials. R&D, quality, supply chain, production and hardware personnel expenses Production line stopped for 3 months Integration processes repaid for two times.
Supply Chain	Efford required for the bidding process.
Production	Production resources cannot be planned efficiently due to unpredictable system integration needs.
Reputation	Customer informed about the delay of the project and the major rework at the system level result in the loss of confidence against the system.

Table 34 (cont'd)

Category	Effects of TD-2.5,2.6,1.3,4,2.7
Quality	Sub-products are produced in different configurations and with different methods due to the complexity of the change request. The quality control process had to be studied and implemented case by case for each unit.
Hardware	The designers had to analyze the problem in each product and find a solution. TDs in project #8 led them to spend all their time in production to solve product-specific malfunctions. Moreover, each product has a unique configuration, which requires modeling and analyzing for validation and verification tests.
Schedule	System delivery delayed for 3 more months.
Cost	R&D, quality, supply chain, production and hardware personnel expenses Cost of penalty due to systems delay Cost of developing simulation and design models Projects were penalized because of the delay of Project #8
Reputation	The end-user was irritated with the job shop manufacturing method instead of mass production. Furthermore, the system delivery was delayed.
Production	CA's production lines were adversely affected. Capacity plans had to be revised.

6.2. Qualitative analysis of the cases using Dedoose

The purpose of the analysis is to determine how practical is the current categories in literature in identifying technical debt. The data were analyzed by considering the age of the participants, work experience, education level, how many companies they worked in before, how many different departments they worked in, and their areas of expertise. The above categories were considered critical for the reliability of the collected data.

A study was conducted with eight people. Their age range is represented in Figure 50. Years of experience in complex system development among these people can be seen in. The analysis of the two graphs shows that most of the interviewers belong to middle age professionals with a significant domain and work knowledge.

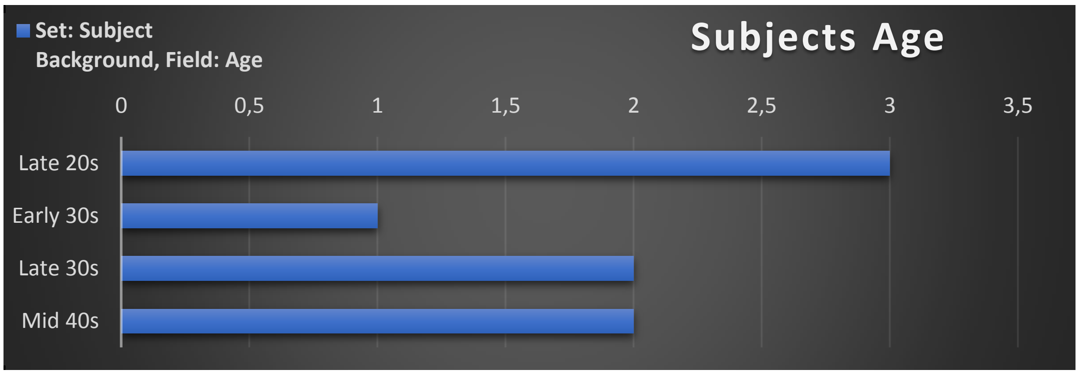


Figure 51 Participants age distribution

People with 0-5 years of experience could not be found for the interview. This result can be presented as an example of the fact that a certain level of expertise is considered when selecting the resources working in complex system development.

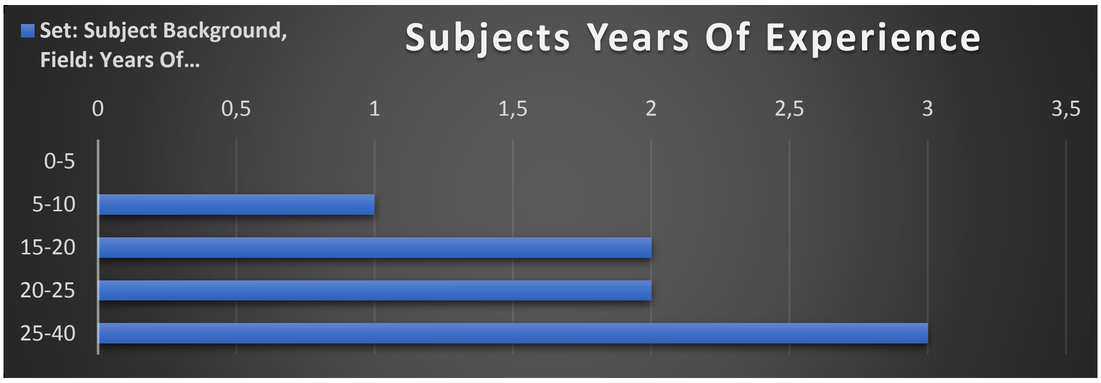


Figure 52 Distribution of participants years of experience

Companies prefer to evaluate inexperienced personnel in low-budget projects, which usually require something other than high-tech designs and innovation. Reused modules, software, and hardware constitute the components of the systems. This environment is suitable for new engineers to learn and adopt the practices for future use. The given distribution of subjects' ages and experiences is the required domain knowledge of the participants for complex system development.

The complex systems development team consists of various people from different backgrounds. Even teams of teams are created depending on the need and expectations of the department. Technical debt occurs during the development and production cycle of the system. We include supply chain, production, and other development support departments because of their significant role throughout the process. Figure 52. represents the roles and proficiencies of the interviewers.

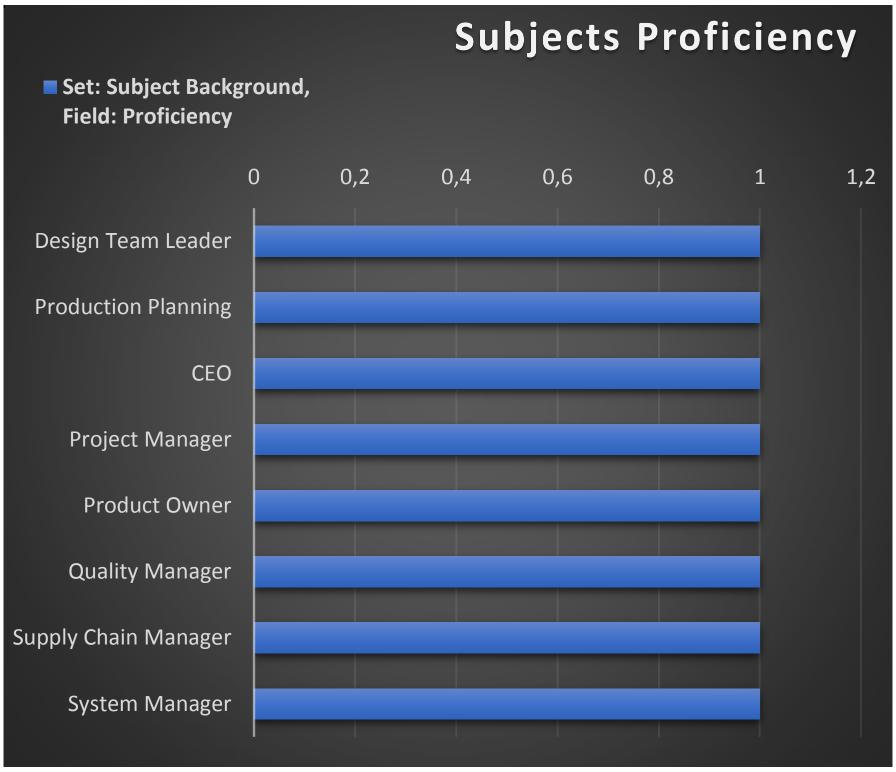


Figure 53 The distribution of participants' proficiency

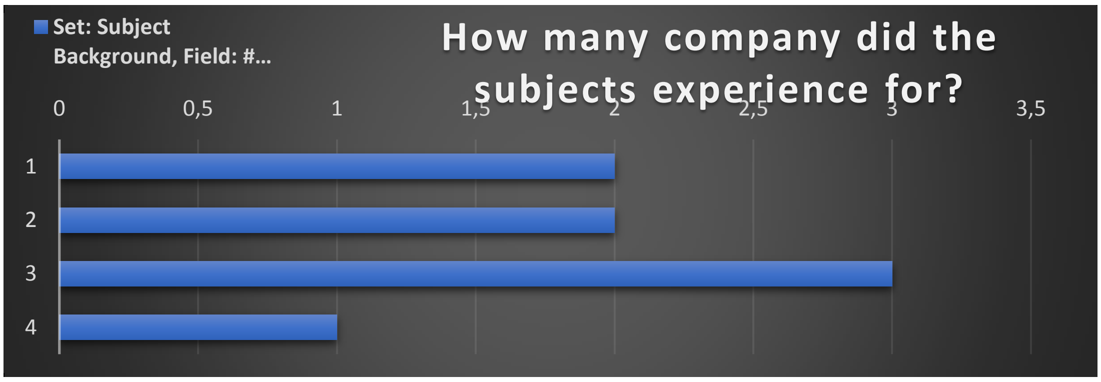


Figure 54 The distribution of participants' experiences - 1

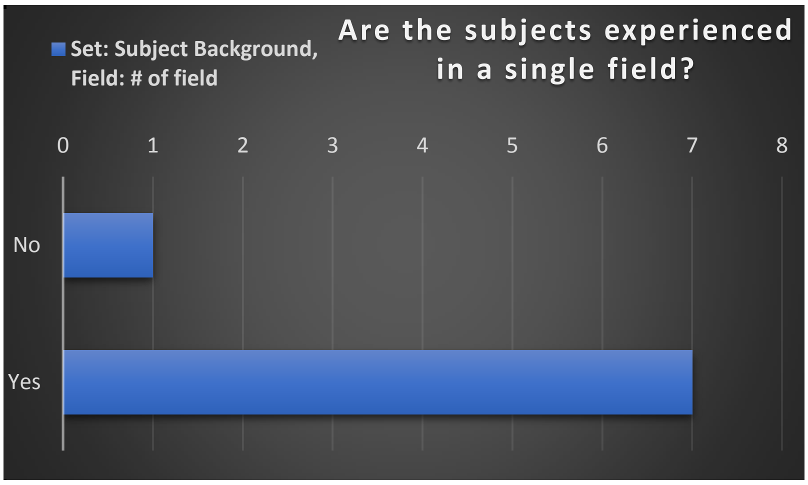


Figure 55 Distribution of participants' experiences -2

Awareness of technical debt requires a business background. There needs to be more than the experience from problematic system development cases to manage TD. The project or system team participant had to understand the management principles of TD. Since there is no single standard for that operation, companies follow their routines and approaches. The culture of the company also affected the decision process of project teams. For this reason, Figure 55 presents the companies that the participants have worked for.

The more experience gained in different companies, the more likely it is to encounter a culture of technical debt and gain experience related to management experience. Since complex system management requires broader management and integration of different professions, experience in various fields makes managing technical debt more effortless. Figure 55. shows that most of the interviewers had worked in more than one field, which means that they also considered the effects of their decisions on other departments when making technical debt determinations.

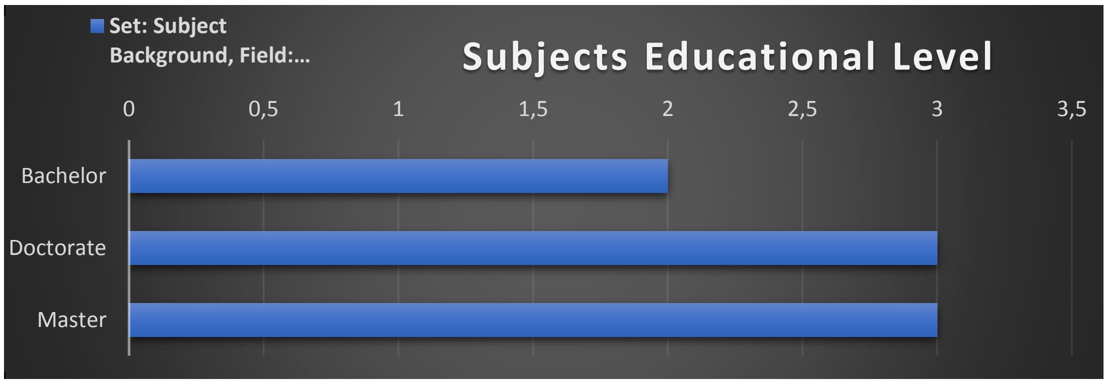


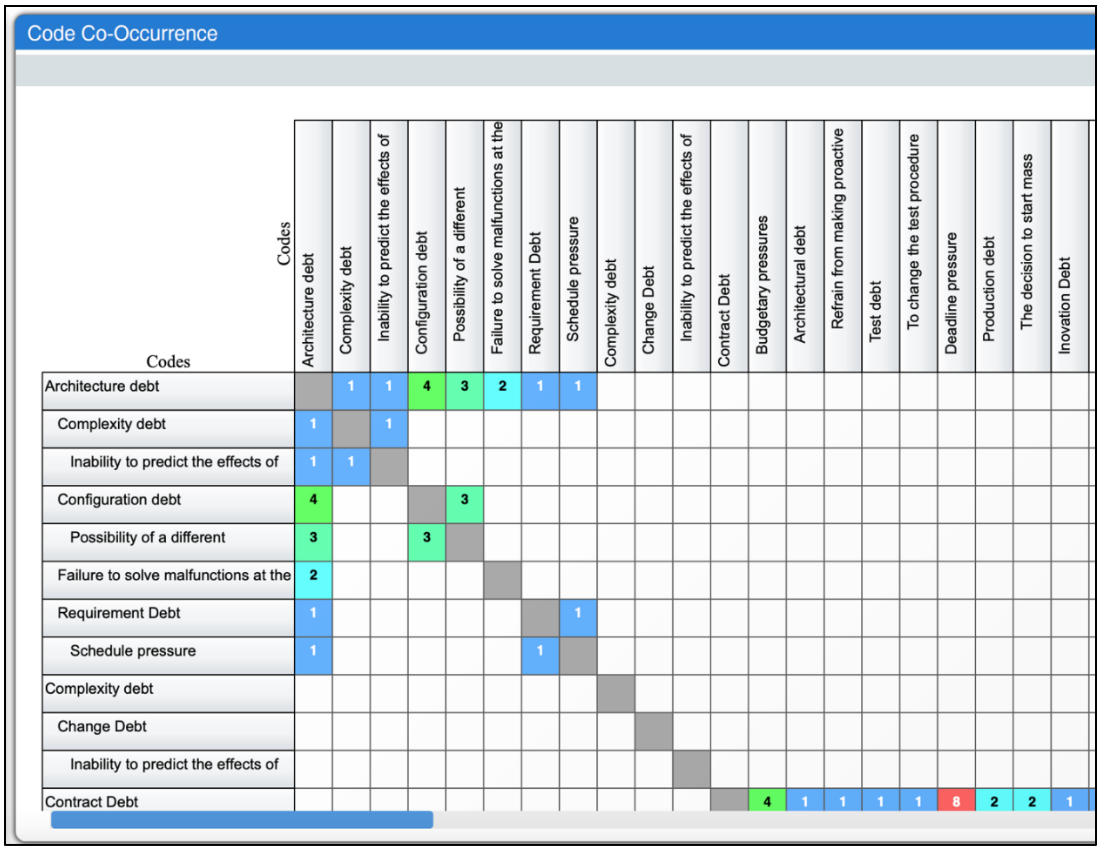
Figure 56 Distribution of participants' educational level

Last, understanding the participants' behaviour is essential to know their educational level which represented in Figure 56. Since there is no single standard for managing technical debt, domain experience, which usually comes from the field, is necessary.

A methodology to efficiently use that knowledge is more efficient with academic guidance. The management also includes the effects, causes, monitoring, and analysis of the TDs. Business professionals use their educational background and experiences to shape their complex system projects TD administration. Academic levels of participants; show an equal distribution in terms of bachelor, master, and doctorate.

6.2.1. Findings of the qualitative study

There exist 161 technical debts in 8 participants' 10 cases which are represented in Figure 58. That 161 TD was grouped into 17 categories that included 26 subcategories. These levels had technical debts that led to or inherited each other. An example reports of the Dedoose interface used to determine these codes is shown in Figure 56.

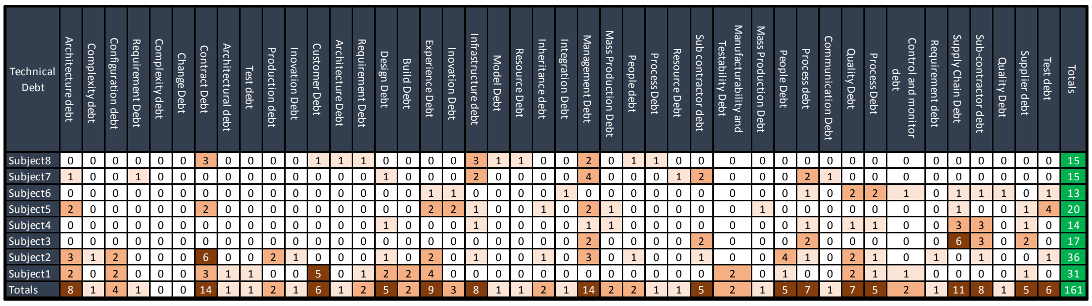


Code Co-Occurrence

Codes	Architecture debt	Complexity debt	Inability to predict the effects of	Configuration debt	Possibility of a different	Failure to solve malfunctions at the	Requirement Debt	Schedule pressure	Complexity debt	Change Debt	Inability to predict the effects of	Contract Debt	Budgetary pressures	Architectural debt	Refrain from making proactive	Test debt	To change the test procedure	Deadline pressure	Production debt	The decision to start mass	Inovation Debt
Architecture debt	▨	1	1	4	3	2	1	1													
Complexity debt	1	▨	1																		
Inability to predict the effects of	1	1	▨																		
Configuration debt	4			▨	3																
Possibility of a different	3			3	▨																
Failure to solve malfunctions at the	2					▨															
Requirement Debt	1						▨	1													
Schedule pressure	1						1	▨													
Complexity debt									▨												
Change Debt										▨											
Inability to predict the effects of											▨										
Contract Debt												▨	4	1	1	1	1	8	2	2	1

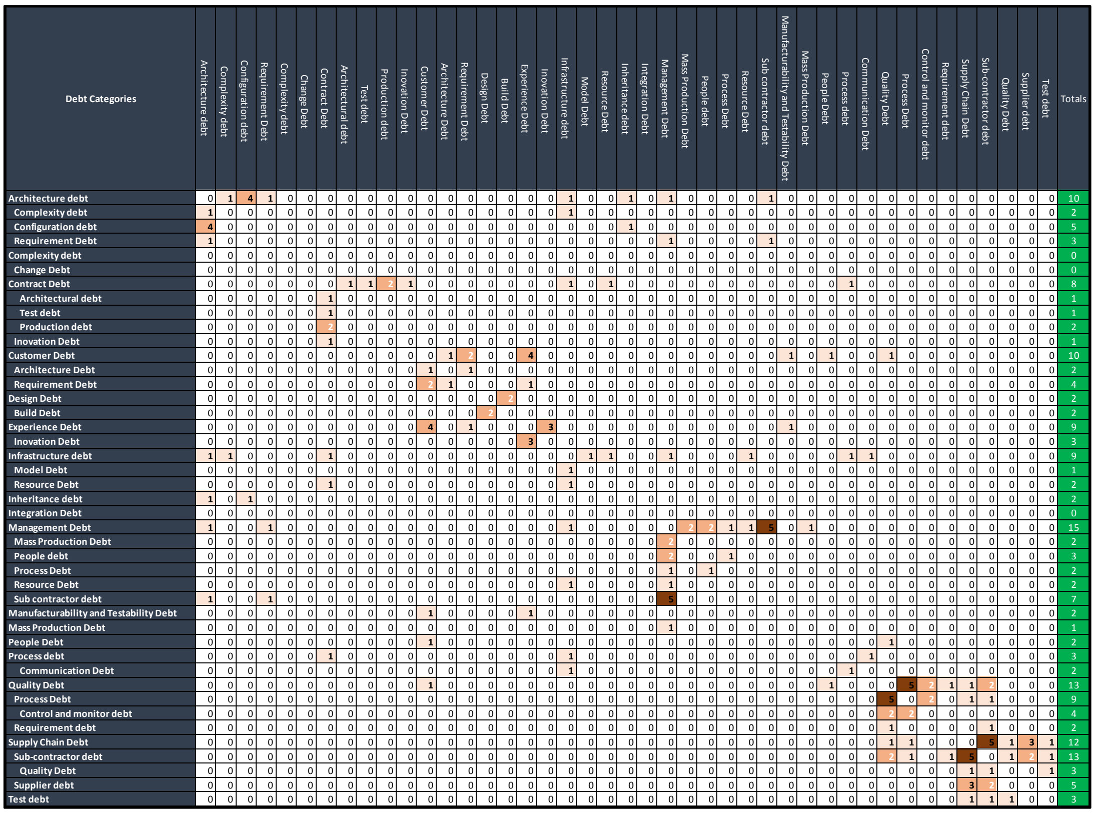
Figure 57 A sample Dedoose qualitative analysis screenshot

Our findings are shown in the matrix structure in Figure 57 below. The notation in Figure 57 is preferred for a straightforward expression of which technical debts cause each other and to show the related categories. As a result of this analysis, it was revealed that technical debt categories are effective in the formation of other technical debts and that there may be relationships at different levels between the categories. Moreover, these relationships between categories can change positions in other scenarios and situations. This case reveals that the causes and effects that cause the occurrence of technical debt should be analyzed, monitored, and managed for each topic individually.



Technical Debt	Architecture debt	Complexity debt	Configuration debt	Requirement Debt	Complexity debt	Change Debt	Contract Debt	Architectural debt	Test debt	Production debt	Inovation Debt	Customer Debt	Architecture Debt	Requirement Debt	Design Debt	Build Debt	Experience Debt	Inovation Debt	Infrastructure debt	Model Debt	Resource Debt	Inheritance debt	Integration Debt	Management Debt	Mass Production Debt	People debt	Process Debt	Resource Debt	Sub contractor debt	Manufacturability and Testability Debt	Mass Production Debt	People Debt	Process debt	Communication Debt	Quality Debt	Process Debt	Control and monitor debt	Requirement debt	Supply Chain Debt	Sub-contractor debt	Quality Debt	Supplier debt	Test debt	Totals
Subject8	0	0	0	0	0	0	3	0	0	0	0	1	1	1	0	0	0	0	3	1	1	0	0	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
Subject7	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	4	0	0	0	1	2	0	0	0	2	1	0	0	0	0	0	0	0	0	0	15
Subject6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	2	2	1	0	1	1	1	0	1	13
Subject5	2	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	2	1	0	0	1	0	2	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	4	20
Subject4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	1	1	0	0	3	3	0	1	0	14
Subject3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	2	0	0	0	2	0	0	0	0	0	6	3	0	2	0	17
Subject2	3	1	2	0	0	0	6	0	0	2	1	0	0	0	1	0	2	0	1	0	0	1	0	3	0	1	0	0	1	0	0	4	1	0	2	1	0	1	0	1	0	0	1	36
Subject1	2	0	2	0	0	0	3	1	1	0	0	5	0	1	2	2	4	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	2	1	1	0	0	0	0	1	0	31
Totals	8	1	4	1	0	0	14	1	1	2	1	6	1	2	5	2	9	3	8	1	1	2	1	14	2	2	1	1	5	2	1	5	7	1	7	5	2	1	11	8	1	5	6	161

Figure 58 Technical debts in categories and subcategories



Debt Categories	Architecture debt	Complexity debt	Configuration debt	Requirement Debt	Complexity debt	Change Debt	Contract Debt	Architectural debt	Test debt	Production debt	Inovation Debt	Customer Debt	Architecture Debt	Requirement Debt	Design Debt	Build Debt	Experience Debt	Inovation Debt	Infrastructure debt	Model Debt	Resource Debt	Inheritance debt	Integration Debt	Management Debt	Mass Production Debt	People debt	Process Debt	Resource Debt	Sub contractor debt	Manufacturability and Testability Debt	Mass Production Debt	People Debt	Process debt	Communication Debt	Quality Debt	Process Debt	Control and monitor debt	Requirement debt	Supply Chain Debt	Sub-contractor debt	Quality Debt	Supplier debt	Test debt	Totals
Architecture debt	0	1	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
Complexity debt	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Configuration debt	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
Requirement Debt	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Complexity debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Change Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Contract Debt	0	0	0	0	0	0	0	1	1	2	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8
Architectural debt	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Test debt	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Production debt	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Inovation Debt	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Customer Debt	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	10
Architecture Debt	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Requirement Debt	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Design Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Build Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Experience Debt	0	0	0	0	0	0	0	0	0	0	0	4	0	1	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	9
Inovation Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Infrastructure debt	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	9
Model Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Resource Debt	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Inheritance debt	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Integration Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Management Debt	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	2	1	1	5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	15
Mass Production Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
People debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Process Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Resource Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Sub contractor debt	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Manufacturability and Testability Debt	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Mass Production Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
People Debt	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2
Process debt	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3
Communication Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2
Quality Debt	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	5	2	1	1	2	0	0	0	13
Process Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	2	0	1	1	0	0	0	9
Control and monitor debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	4
Requirement debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	2
Supply Chain Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	5	1	3	1	12
Sub-contractor debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	1	5	0	1	2	1	13
Quality Debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	3
Supplier debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	0	0	0	5
Test debt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	3

Figure 59 Matrix structure of qualitative analysis

Technical debts in categories and subcategories specific to individuals are given in Figure 59. Findings were reflected using light-to-dark colouring, depending on their increased frequency of occurrence.

The technical debts that emerged at the end of eight interviews revealed that the existing literature provides a basis for determining technical debt in complex system projects. However, it also became clear that the categories needed to be studied and reproduced for different departments and areas of expertise. In addition, it was understood that technical debts could be inherited and managed with subcategories depending on the circumstances. The new categories found due to the multiple case study and their explanations are detailed in Chapter 8.

6.3 Validation of the Study

Since there is no similar model in the literature to show the effects of the emergence of technical debt, to validate the study, the results of the case study were shared with the subject experts. These experts' feedback and evaluations were used to validate the analysis. Validation interviews were completed with the participants whose case studies were conducted.

First, text base scenarios, including technical debt, were shown to the participants. Then, examples of strategies that create technical debt in projects were verbally explained. Finally, the same scenarios were presented with the technical debt visualization model. With this method, the participants could evaluate the model's contribution to the literature and technical debt management.

All the data collected from the first Interview were processed on the model, and the technical debts were visualized. The results of these case studies were included in the research by creating technical reports. During the valid ation Interview, the participants were asked open-ended questions, except for the 4th and 5th question. These questions aimed to identify the obstacles in operation and verify the model's effectiveness in visualizing technical debt. The questions asked to the participants during the validation process are given in the

Table 35 Validation questions

ID:	Question:
1.	What can be improved in the technical debt visualization model?
2.	What can be done for its wide use in the company?
3.	What is the most significant limitation of using the model?
4.	To what extent will the stakeholders in the project benefit if the model is used for complex system development?
5.	How adequate are the proposed categories to detect technical debt in complex projects?
6.	Do you have a technical debt category suggestion for determining the concept in complex system development?

For the validation of specifying and understanding the effects of TD in complex system development research team used the first four questions. The scale for the 4th question was presented to the participants. Scale categorized as:

- Strongly disagree,
- Disagree,
- Undecided,
- Agree

- Strongly agree,

The scale for the 5th question was presented to the participants. Scale categorized as: *Table 37 Ratings of the* 4th question answers

Table 36 Ratings of the anwers to the 4th question

Interviewers ID:	**Scale**
1.	Strongly agree
2.	Strongly agree
3.	Strongly agree
4.	Agree
5.	Strongly agree
6.	Strongly agree
7.	Strongly agree
8.	Strongly agree

Participants shared that the model will be a critical tool to support them in managing complex systems. %90 of the participants stated that they were aware of the technical debt metaphor with the model.They also shared that technical debt is critical in complex system development projects. They shared that as the structures get complex, the problems become intertwined, making their management a critical topic.

For the validation of the qualitative study research team used the last 2 interview question. The scale for the 5th question was presented to the participants. Scale categorized as:
Table 37 Ratings of the 5th question answers

Interviewers ID:	**Scale**
1.	Strongly agree
2.	Strongly agree
3.	Strongly agree
4.	Strongly agree
5.	Strongly agree
6.	Strongly agree
7.	Strongly agree
8.	Strongly agree

To validate the result obtained in the second research question, the research team used the result of the interviews. Answers to the 5th question in table 38 shows that the suggested categories are promising for detecting technical debt in complex systems. However, this study needs to be evaluated on more cases, and categories should be developed.

In addition, it has been demonstrated by this study that risky and erroneous decisions that are not followed over time damage the system over a long period. According to the participants, With TDVM, such problems can be overcome, and complex components, modules, or parts in the system can be tracked and managed effectively. Only 1 participant stated that the model would be challenging to use in the workload in the development cycle. For this reason, he presented his opinion partially. However, this participant stated that TDVM would play an essential role in complex system development if this problem is overcome.

For the improvements of the model:

- All participants stated that the model should be moved to an interface via software,
- It was suggested to include the result and solution alternatives for TDs in the model,
- The need for a module capable of reporting the results of TDs was specified.

The most critical issue for using the model within the company was the need for a user-friendly interface. In addition, while technical debt was considered new even in the software world, it was revealed that all project stakeholders should be informed about this issue. Otherwise, the model cannot be used and operated correctly. In addition, the employees should accept the importance of managing technical debt. Because if the project team does not consider it necessary, valuable data will be lost.

The widespread use of the model for the company was considered a subject of imitation. It was stated that the PMO office might support the operational activities of the model. It has been evaluated that employees can be appointed to operate the model. Furthermore, most interviewers agree that project engineers who follow the technical debt process during the complex system development can be assigned. These individuals may ensure that technical debt is assessed and considered when making decisions.

At the end of the validation interviews, the participants agreed on the positive impact of the visualization of technical debt on the system development cycle. In addition, since the participants are not only software developers but also different area experts from project stakeholders, the importance of the effects of the concept of technical debt for different system units has also been revealed.

CHAPTER 7

DISCUSSION AND SYNTHESIS

Austin M. Page states that technical debt is the cost of doing nothing (Austin Page et al., 2019). The metaphor may appear at any stage of the system development cycle and places a heavy burden on project management, including system recalls, even after the delivery of the systems to the customer. The findings obtained as a result of the research revealed that the management of technical debt is significant not only for software developers but also for all system developers, stakeholders, and manufacturers**(Baccarini, 1996; ROBERT NORD, 2016; Rosser & Ouzzif, 2021a)** In addition, it has been demonstrated by the findings of the case studies we made and the literature review; as systems become complex, the decision-making process is affected by many factors (ROBERT NORD, 2016). The developers and managers need to estimate the possible effects of decisions on the system's overall performance. As a result of researching the deficiencies in the tools and methods, we have developed and improved the technical debt visualization model and offered a solution alternative for both the industry and the research (Avgeriou et al., 2021) (Ernst et al., 2015; Zazworka et al., 2013) . The following sections describe the research findings, implications, and recommendations.

7.1. Summary and results

The study was conducted through ten interviews with eight people. As a result, 25 technical debts were identified.
Table ***38*** and Figure 59 presents the effects of TDs among different stakeholders.

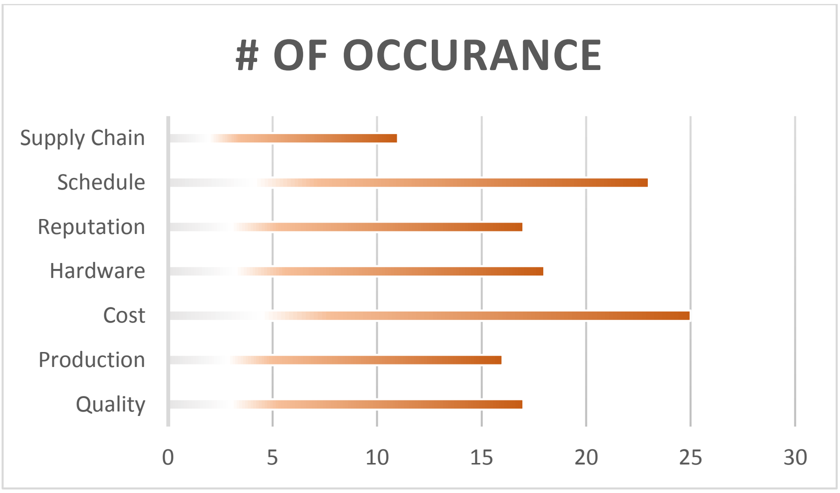


Figure 60 Td # of occurrence

Table 38 Td # of occurrence

TD Effect:	**# of occurance:**
Quality	17
Production	16
Cost	25
Hardware	18
Reputation	17
Schedule	23
Supply Chain	11

Cost and Schedule are the most affected categories depending on the result of the analysis of the Projects. Although not mentioned in the literature, it has been revealed that the effects of technical debt on other departments and stakeholders are too significant to be underestimated. One of the study's findings was that the categories given are wider than Table 35 and open for improvements. Companies or individuals may study the metaphor and find new categories depending on their development environment and business domain.
Main effects of TD observed in each project, excluding the software category:

- Re-supplying the products that came with the change request,
- Reproduction of scraped products and supply of raw materials,
- Loss of bargaining advantage in product prices due to re-supply works,

- The unplanned additional effort of all stakeholders, such as quality, hardware, software, system, project, production, supply chain,
- Involving top management in the process,
- The customer loses confidence in the system,
- Negative impact on future sales due to declining customer reputation,
- The inability to apply standards in prototype and production processes and configurations becomes unmanageable,
- The need for alternative design methods,
- The emergence of erroneous decisions that trigger each other,
- Having to make decisions that will reduce the total quality of the system after irreversible steps,
- Shifting of the system delivery schedule,
- Inability to make resource plans for teams and infrastructures,
- Influencing the calendar and management plans of projects using the same resources,
- The decreased motivation of the teams and damage to their faith in the work done.
- Inefficient use of production lines
- As a result of TD decisions, new constraints and requirements may be added to the system requirement specifications, which the standard system development lifecycle would not include.

7.2. Technical debt categories for complex system development

The following technical debt categories were found by analyzing the data collected from complex system development projects. These categories have been created based on information and experience in the literature for software management and system development methodologies.

Complexity Debt:

Complexity plays an essential role in the development cycle. In terms of communication efficiency between development teams, complexity is vital. Complexity debt arises from the difficulty of managing the impact of decisions taken in the complex structure of the system. A single line of code change can have unexpected effects on the entire system due to the interfaces and inter and outer connections between couples of components. The complex system architecture provides a structure that will help the emergence of technical debts If the phenomena are not managed and tracked correctly.

Change Management Debt:

Change management; requires managing the change's effects, method, and necessity. Employees who resist change need to be convinced, and the need for change needs to be accepted by the stakeholders of the development activities. In order to implement changes in complex system structures, all stakeholders need to be included in the management process of the change requests. Possible impacts and unforeseen risks can be reduced with the correct management of the change; otherwise, the unpredictability of the results of the requests and the resistance against the change triggers technical debt decisions.

Contract Debt:

Especially with the growth of structures and the increase of integrated systems, contract clauses can be imposed on development teams by agreement. Some specifications were deemed unnecessary during the production phase, which assumed a need during the prototyping. Although these requirements do not contribute to the system's performance, their development due to the contract requires additional resources, affecting both calendar and budget plans. These updates to resource planning decisions may cause decisions to lead to technical debt. The client may force the development team with out-of-date requests, citing the clauses in the contract due to the need for domain-specific knowledge.

Experience and Training Debt

All processes, from development to production, require individuals and institutions to be trained and gain experience from the experts. This requirement continues even after the delivery of the system to the customer. This type of debt is examined under two categories. The first category is not transferring the current technological know-how to the employees and not encouraging the developers to gain experience in the field. This category also arises from needing to meet the requirements of resources such as education and research for self-development. The second category arises from the system's need for trained final users. These deficiencies may lead to the search for solutions to problems that would not exist in the system, which can be summarized as inaccurate planning of resources and inefficient performance of designs.

Culture Debt

The culture of the located country and the company's way of doing business influence the development activities of systems. Moreover, cultural influences arise from traditional activities, which need to become standardized procedures. Failure to embrace, manage and adapt to this understanding can create misunderstandings between people and departments. Complex systems are affected by such communication problems, and technical debts may arise due to a lack of understanding of culture management.

Resource Debt:

The culture of the located country and the company's way of doing business influence the development activities of systems. Moreover, cultural influences arise from traditional activities, which need to become standardized procedures. Failure to embrace, manage and adapt to this understanding can create misunderstandings between people and departments. Complex systems are affected by such communication problems, and technical debts may arise due to a lack of understanding of culture management.

Mass Production Debt:

Complex systems project delivery schedules require development cycles to pass into the production phase as soon as possible. However, the systems' production lifecycle, which contradicts prototype development with the standards and methods to be followed, requires following process management procedures while making these decisions. Suppose the components moved to mass production and are faced with significant change requests in the following phases of the project. In that case, the project managers encounter decisions that may create technical debt because of the schedule and cost constraints. In some cases, changes may even be impractical because of the project budget requirements. Such scenarios require the creation of workarounds for system performance with the existing product.

Management Debt:

Organizational structures directly affect the decisions taken by the project and system teams. Sometimes subject matter experts and employees must act according to senior management-dictated methods and decisions. These decisions depend on the companies' profit and must be implemented even if they are incorrect for the system's performance. Restriction of resources, dictation of design method, calendar pressure, and faulty planning can be examples of the above issues. Choosing an iterative development cycle in a company where the communication between the customer and the designers is not deemed appropriate may result in the faulty or incomplete transmission of a significant amount of requirements. Management debt is one of the most challenging and influential categories regarding technical debt.

Supply Chain Debt:

The supplier and subcontractor debt must be examined under the Supply Chain Debt category. Complex systems consist of thousands of materials and products. Most of these components are outsourced because of the company's resource planning activities to increase profits. The supply of these products requires the management of the selection of firms, and the issuance of contracts requires that all existing debt categories be evaluated. This category is particularly critical as it had to consider the factors of other debt categories when making these decisions. Suppliers or subcontractors may unconsciously or consciously make decisions affecting many stakeholders, from design teams to system architectures. Since the designers have yet to learn the existence of these decisions at the integration stage, their effects can be interpreted differently, leading to decisions for the occurrence of technical debts.

Motivation Debt:

Project team members' approach to their responsibilities and how they do business while performing their job definitions may cause consequences that affect the system architecture's health. Employees with low motivation may intentionally or unintentionally cause problems to be hidden or postponed because they do not take the movements to act proactively. Companies that ignore this problem or do not look for a solution may face failures in their projects because of the effects of TDs.

Customer Debt:

The success of the project in complex structures depends on the management of the sub-system and the redundancy of the interfaces that need to be communicated and their relations with each other. The communication between the teams and departments involved in the development of these units and with each other is as effective as the system architecture and design decisions. When the sensitivity shown in the interfaces is not shown in the management and monitoring of this interaction and even in the detailed planning, it leads to the emergence of many problems that will not be on the agenda and the solution alternatives of the errors and problems that arise, resulting in technical debts.

Communication Debt:

The project's success in complex structures depends on managing the communication network of sub-systems. These systems should be designed to interfere with each other as little as possible. The fewer interfaces are easier to manage in every way. As the need for communication increases, difficulties may arise between development teams. The impact of these teams' communication on the development cycle plays as much of a role as system design. If the teams do not communicate effectively, they may make decisions that affect each other's development outputs, affecting the system's behavior. A system under development waits to be updated based on feedback from all relevant stakeholders. Technical debts are inevitable if the communication network collapses or consists of teams that need to transmit valuable information to each other.

Strategy Debt

System architectures are directly affected by development methods. Complex projects require a strategic plan for project management activities, including the method of development lifecycles. This plan's lack of a strategy program may result in technical debts. Inaccurately determined strategies are also the origins of significant project risks. During the system's development process, it acted around the determined strategy. System development is continued depending on the strategic plan. Predetermined or planned actions are followed when companies face problems and risks. These plans affect development cycles with requirements to adhere to methodologies such as agile or waterfall. When trying to implement strategies that do not meet the project's needs, system engineers or designers may be forced to follow methods that will result in technical debt.

Control and Monitor Debt:

Failure to monitor the defined processes or the findings that potentially pose a risk may lead to problems that can be easily avoided. All stakeholders of the system development cycle should adopt this control and monitor loop. The proactive approach requires identifying, evaluating, and monitoring scenarios where problems are likely to arise. Standards may fail to be a precaution against every issue encountered in complex structures. Companies or individuals that do not adopt this approach may face technical debt.

Manufacturability and Testability Debt:

Designs are completed when they are manufacturable. Manufacturability also requires testability and verifiability. These steps can be skipped when designing the prototype. Non-standard methods can be used. However, the customer product requires the completion of all processes. If production methods or requirements for infrastructure are forgotten at the design stage, temporary solutions and alternatives are sought in the following phases. This development strategy may lead to unpredictable results in complex systems architecture structures and may create technical debts.

Prototype Debt

Prototype technical debt is not just about non-transferable decisions and methods from development to production. Necessary details that are overlooked during the development phase and the critical processes that need to be documented cause prototype debt to occur. The findings obtained in the system's prototype or the performance outputs that cannot be achieved with the production standards can increase customer expectations. Management approaches to the process and scope of the project may lead to technical debts. It is challenging to estimate the impact of these decisions on the complex system's architecture. Designers may damage the system architecture when trying to achieve results that are not possible with current technology or production standards. Moreover, when the effects of the fault development approaches start to appear, detection and resolution of the malfunctions may take time and require resources for the cause and effect analysis in the project's later phases, which may lead to the emergence of technical debts.

7.3. Future research and limitations of the study

The literature has not examined technical debt metaphors in detail except for software management. Although the current analysis provides an idea, the technical debt metaphor needs to be studied for stakeholders other than software developers in detail and included in the literature. In addition, new technical debt categories may emerge with the analysis of these cases. The results and findings must be evaluated by conducting cases for different projects and companies. In particular, data from companies serving different markets will help to reveal new findings.

The model still needs to be developed into a tool. Technical debt visualization model could be manageable via a web interface or a third-party program that facilitates technical debt management. This tool can be integrated into companies' systems or manage the process independently.

CHAPTER 8

CONCLUSION

The research seeks to comprehend the impacts of technical debt in the evolution of complex systems. The literature does not provide a standard method for the visualization and the detection of the metaphor. The technical debt visualization model of TD in complex system development has been developed. Using that model, the study team provides specific links between actions and causes of impacts associated with technical debts in the system. With the assistance of the developed model, it aims to prove that despite the effects of technical debt were only studied for software management in the literature, its effects on other stakeholders also have essential effects on complex system development.

Eight case studies were conducted to answer the research questions. These studies were analyzed using the technical debt visualization model. The scenarios studied were selected from complex system development projects at different stages of their development cycle. Field experts and system and project managers evaluated TDVM outputs. The result was considered technical debt management vital in complex system development. In addition, the model is regarded as a critical resource for detecting, managing, and preventing technical debt.

Qualitative analysis was conducted on the Interview and observation data. The findings of this study were used to answer how sufficient the technical debt categories in the literature are for detecting technical debt in complex systems. It has been revealed that the existing literature offers a basis for technical debt detection but needs improvement and study.

Sixteen unique technical debt types were discovered in the complex system development activities literature. These 16 new categories were used to answer the third research question, revealing that the existing literature provides a foundation but needs improvement. The details and examples of these technical debts and their detailed explanations are given in Chapter 7. This study revealed that many technical debts are still waiting to be discovered and added to the literature. Although the current categories in the literature occur through studies about software management, creating a framework for the other stakeholders of the system development projects is very important. Categorizing, monitoring, and managing technical debts specific to additional fields through this framework is significant for companies to succeed in projects to prevent high costs or schedule delays due to TDs. In complex structures, it becomes difficult to follow the history of the scenarios that led to problems and create a cause-effect relationship for their management. TDVM successfully transformed the picture into a simple, understandable form for detecting and managing technical debt in this environment. In addition, the developed model offers successful and promising results in detecting reasons leading to technical debts. The final result shows that it is also successful at revealing the effects of TD on various partners and stakeholders, according to the feedback received from the experts.

As a result of using TDVM to analyze cases, the research shared seven improvement suggestions for companies and individuals who want to manage the technical debt metaphor in complex systems development projects. The details of these improvements are detailed in the following sections.

As a result, it has been revealed that technical debt management offers companies significant advantages in many different areas, especially in calendars and budgets. In addition, TDVM has proven itself as a promising model both in revealing the factors that cause the emergence of technical debt and managing it throughout the system development cycle.

Implications for industry

Recommendations for managing the TD process for the complex system development activities were significant in objectifying the result into explicit outputs. Implications for the industry can be summarized as follows:

- Decisions taken in complex systems can lead to many unexpected results. With TDVM, the effects of these decisions will be more predictable.
- Thanks to TDVM, possible malfunctions that will cause problems in the future can be prevented before they occur in the system.
- The model allows the evaluation of TD decisions' effects from the perspective of many stakeholders. Monitoring the metaphor in that way more clearly reveals the impact of the consequences of decisions on project plans such as calendars and budgets.

Three questions that will promote the use of the model by individuals and companies in the complex system development process are answered below so that the study can quickly adapt to their operations.

Q-1: How will companies or individuals visualize their systems when there is not enough data to display?

TDVM is used for more than just real-time monitoring. It can also be used to analyze retrospective cases. In the development process, steps and standards that need to be improved can be revealed using the model. The effects of the decisions and the reasons behind the problems can be managed effectively with the implementation of the TDVM. Identification of the decisions and examination of internal and external factors affecting the system's overall performance can be managed by analyzing the model's output.

There needs to be more data and problems to analyze to mean that the processes are operated correctly and perfectly. On the contrary, it may mean that the existing problems still need to be detected. The research team recommends that companies' processes should have a certain level of maturity to integrate TDVM. For example, the effectiveness of decision support systems and the guidance of standards and methods in managerial and operational decisions are the basis for implementing the model. Otherwise, the data used to create the model may need to be reviewed and interpreted. It may misrepresent processes or cause project and system management decisions to misdirect.

Q-2: Can the technical debt visualization model be applied to systems rather than complex projects?

The only constraint to implementing the technical debt visualization model is not the project's complexity. The effects of decisions in complex structures cannot be easily predicted. However, complexity increases with the project size and the number of system components' interfaces. Changes in the development method, lack of standard methodologies, data that cannot be traced, and non-repeatable processes require real-time and retrospective project decision mechanisms. The necessity of managing technical debt arises independently of the project's complexity.

Complexity is only one factor that affects the need to manage technical debt. Different areas of expertise and decision mechanisms involving teams create the necessary environment for the emergence and management of technical debt. As seen in the cases handled during the research process, technical debt can occur in project life cycles regardless of the system's complex structure and the difficulty level of the decisions made.

Q-3: What is the right level for the depth of the decision?

According to (Chugh et al., 2008), wrong decisions are costly and become even costlier as they must be managed and monitored. Focusing on the exact question is essential to make the right decisions. As the interfaces between the system components increase, systems operations may be affected by the decisions raised. Both case studies and literature have demonstrated throughout the study that a minor decision has the potential to cause technical debt in such scenarios. For this reason, different decision types have been created below to raise awareness among project stakeholders when making decisions in complex structures. Depending on the type of decision, technical debt can be managed more effectively with awareness of the case or procedure to be followed, and risks that may arise can be prevented before they occur. Even if the decision type does not provide a solution to the problem, with the help of institutional memory, it can enable one to benefit from the experiences gained in the past.

These decisions are discussed below under six headings. But the decision types are not limited to the list; they can be improved and reviewed by the companies, individuals, and the academy.

Predictable or Unpredictable

System stakeholders made many decisions during the system development cycle to obtain project goals. Some of that decisions can be categorized as routine and regular. The possible effects of those everyday decisions can be easily predicted. Imagine a dye defect in the system's mechanical surface. A standard painting operation will be enough for the solution to the problem. Possible outcomes can be easily predicted.

On the other hand, complex systems usually require difficult decisions. Imagine that the dye has radioactivity and heat-blocking properties. Using a standard product will harm the system's performance. Decision-makers must consider the system's functional and non-functional requirements, especially in an uncertain environment. In addition, monitoring the decisions taken on the design and operating the approval process facilitates the management of possible effects.

Make or Buy

Resource allocation and management have a significant role in developing and producing complex systems. Companies must outsource parts, modules, and sub-systems to achieve project schedules and budget goals. A partial outsourcing option is also combined with using companies' developers and resources. Making or buying decisions requires budget and resource management. But still, subcontract and supplier management is challenging when companies prefer to outsource critical modules or parts of the system.

Especially mass production with sub-contractor may even need to plan their capacity and output to achieve systems goals. Also, choosing a company that doesn’t have enough technological background or knowledge may even fail to meet the critical requirement of the system. That decision type is vital in TD management in complex system development.

Phase Transition

Phase transitions are discussed under two headings in the thesis. The first type is the decision of the product development method to use prototype or mass production of the systems; the second is expressed for transitions between different stages in the product development cycle. These development cycles can be categorized as; requirement, design, verification and validation, implementation, integration, and maintenance.

Many companies must follow standards or procedures in the prototype development environment and plan their resources for collecting detailed configuration data. The project team aims to have the working product as soon as possible. However, complex systems often require time and budget constraints to develop prototypes. This causes the development and production cycles to become intertwined. These transitions are critical in technical debt management. Decisions should be carefully examined, and their effects on the product and system should be closely monitored and predicted. Assume that a subcomponent is tested with non-standard applications or procedures while developing the prototype. The expected result will not be obtained when trying to transfer prototype methods, which need to be validated or even accepted by the industry standards for mass production.

Program – Portfolio (Strategic)

Completion of complex projects is critical for companies' goals. For this reason, they are frequently monitored by the top management and the project stakeholders. If the system and project teams fail to develop or produce these systems, companies face many negative consequences regarding their business conduct. This is why companies transfer their resources to complex projects, and upper management is involved in the decision process. Upper management support for the decision process only sometimes makes things easier.

Management may dictate decisions that do not conform to the design and system architecture. Such decisions leave adverse traces and effects on the life cycle of projects. In addition to system needs, companies have goals arising from their vision and mission. These strategic plans may create additional requirements for projects. Top management can prioritize these requirements. These prioritizations may result in not allocating resources for critical improvements or negatively impacting the system architecture. System and project managers should carefully manage these requirements and their effects and results.

Behavioural

No matter how much hardware and software the systems contain, the decisions require human interaction. Although the developing technology offers many tools and auxiliary tools that serve the decision mechanisms, the human factor maintains its importance. Companies with specific standards and control mechanisms may be less affected by behavioural decisions. Because possible behavioural decisions are controlled and monitored with standards and procedures, the possible risk is distributed to different stakeholders to be managed effectively.

However, managing all decisions actively in mass production takes time and effort. Since the decision mechanism is not easily manageable, judgments are often made instantly and do not go through control. For example, as stated in one of the case studies in the research, details that needed to be documented caused the project to fail in the later stages. The fact that the standard did not guide the designer in document preparation and that he needed to detail the document to evaluate his design created significant problems for the system.

Situational

Due to budget and calendar pressure, companies faced difficult decisions during the development phases of complex system development. Managers or decision-makers should make unpredictable, non-repetitive decisions. These types of decisions are non-standard routines and must be followed. Results are not easily predictable and have yet to be previously validated. It can also create unexpected situations when interacting with other system components and modules.

Assume a critical component(A) design has changed due to an unavailable raw material. The system needs three A to continue its operation without losing its efficiency. Due to the unavailable raw material, two different designs were made (Component A - Component B). These two designs form-fit to each other. This situation requires the operation of the system with two different configurations. For example, the new system configuration has to operate with two products, A, and 1 product, B. Due to schedule and budget pressure, product B needed help to complete validation tests. Just because, in the project plan, these tests were only planned for product A rather than product B. However, since its effects and results require an extended analysis and modelling, the problems that may be faced in the future become critical risks for the system. As in this scenario, the decisions taken in the project because of the instant developments and changes can be evaluated under this heading.

Six recommendations to improve the management activities of TD in complex system development are shared below.

1. Production processes must be managed separately from the prototype development as much as possible.

Mass production decisions before the completion of the system development cycle have complicated consequences. When a change request is necessary for the mass production cycle, it creates an unplanned workload, cost, and calendar disadvantage for the project management. Depending on the progress in the production, change requests may even be inapplicable to the systems. The effects even result in system modules being scrapped and remanufactured. Technical debt management is complicated when transitioning from prototype development to mass production. Especially when there is feedback between the two processes, project management gets involved. Also, environmental factors cannot be detected in that scenarios, making the system development extremely difficult to manage.

2. Additional technical debt probabilities must be considered when making decisions and managing risks when technical debt is detected.

Technical debts put extra constraints on the systems' development or production cycle. As management activities of technical debts are postponed, these factors can become the main factors in decision-making in system management. For this reason, technical debt should be monitored and evaluated, and solution decisions should be made without delay. Awareness should be raised of technical debts, and risks should be monitored and assessed regularly. Technical debts can cause other technical debts to arise. Especially in complex structures, additional variables bring difficult decisions to manage. A single line of code can change the main components in the system. This situation may cause new interfaces to be built, test infrastructures to be updated, and integration routes to be updated. Technical debts can also affect other projects' calendars and resource management, especially in projects which require the same resources. Alternative solutions are created when these changes cannot be made due to calendar pressure or project budget. These alternatives lead to the emergence of new technical debts. Parallel to the system's complexity, the development structure becomes more complicated to manage when making decisions.

3. Technical debts should be managed from the perspective of all stakeholders, not just software developers.

Technical debt has severe effects on other stakeholders of the project, not just the software developers. These stakeholders take an active role in the emergence of technical debts that affect the system development process. The findings of the study show that quality, production, supply chain, and hardware development teams are examples of the stakeholders of TD management. This list is not limited to the given proficiencies.

4. Subcontractor/supplier selection and management for critical products are essential in complex system management.

Especially in mass production, outsourcing is preferred for efficient resource planning. From documentation to manufacturability, transferring product-specific domain experience to outsourcing companies plays a vital role in TD management. Actions taken by these companies to reduce costs in mass production to increase their profit create risks that can lead to TD. Ultimately well-intentioned methods or critical information that may affect the system's performance may be kept from the contractor due to a lack of knowledge and field-specific experience. This information is necessary for system administrators to search for the solution in the system's relevant parts; otherwise, it will cause a delay in the schedule or a major malfunction at the system level after the integration with the system.

5. New technologies and innovation in system development often come with technical debts that require efficient management of the metaphor.

First, since the system structure is enormous and the cost is high, the system's prototype may become the final product in such complex systems. These prototypes may be the critical parts of the system or the system itself. Companies are forced to take that unplanned action because, after many iterations building the system again may not become the most efficient option. While trying to meet demanding requirements, non-standard applications can be made, which may not be kept under configuration. Also, prototype development with designer interventions instead of production standards creates a chaotic environment for the management of TD. Technical debt management becomes challenging with non-standard methods and untraceable, unrepeatable work processes and documentation. Companies have to separate their customer products from prototype development, or they should have a plan for its management in scenarios where it is mandatory.

6. Technical debt management continues after system delivery. It needs to be followed and monitored.

Complex systems involve sub-systems and modules. Integration of different parts and an increased number of interfaces makes it harder to specify the reasons behind troubles. The effects of technical debt may even occur after the delivery of the systems that require the monitoring of TD throughout the system lifecycle. Especially unintentional TDs are critical since they usually hide in the design and cannot be discovered before their effects occur. After the transportation of the system, any rework or change request related to the solution of TD requires customer management. The probability of reputation loss creates new constraints for the project team. Developers are forced to try to find alternatives that can be applied without recalling the system from the customer.

7. Standardizing the transition from prototype development to mass production and creating a team that manages this process facilitates TD management.

The development environment usually does not require production standards. Companies do not follow procedures or methods during the development lifecycle. During prototyping, critical information for the systems may lose. The system design also depends on the operating conditions. These conditions may change assembly methods or architectural plans. Specific requirements and qualified personnel with domain experience must conduct the systems' integration process. It is essential to prevent the loss of information in this process and to ensure that the necessary methods and tools are used to avoid the system from being exposed to TDs. Creating a department and a team that manages and monitors the transition process from prototype to mass production is vital for managing complex systems at different stages and levels.

Implications for research

The concept of technical debt still deserves detailed study in the literature from a complex system development perspective. Conducting research from the perspective of all stakeholders affected by metaphor will allow more profitable and efficient management of the complex system development process. Moreover, managing technical debt in the system development cycle will be one of the guiding factors for companies to succeed. The following two articles present the study's implications for future research:

- Findings of the multiple case studies with TDVM suggest that TD management has a significant role in system and hardware development. This concept needs to be studied in more detail in the literature, and our research provides a basis.

- Technical debt has significant effects on stakeholders other than software developers. This result suggests that technical debt effects should be evaluated for software developers and all project stakeholders.

REFERENCES

Akbarinasaji, S., Bener, A. B., & Erdem, A. (2016). Measuring the principal of defect debt. *Proceedings - 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, RAISE 2016*, 1–7. https://doi.org/10.1145/2896995.2896999

Alves, N. S. R., Mendes, T. S., de Mendonça, M. G., Spinola, R. O., Shull, F., & Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology, 70*, 100–121. https://doi.org/10.1016/j.infsof.2015.10.008

Austin Page, by M., Eppinger, S. D., & THN iSsign Joan Rubin OFTECHNLOGY--UJ, O. O. (2019). *Technical Debt: The Cost of Doing Nothing Signature redacted Signature redacted Signature redacted Signature redacted*.

Avgeriou, P. C., Taibi, D., Ampatzoglou, A., Arcelli Fontana, F., Besker, T., Chatzigeorgiou, A., Lenarduzzi, V., Martini, A., Moschou, A., Pigazzini, I., Saarimaki, N., Sas, D. D., de Toledo, S. S., & Tsintzira, A. A. (2021). An Overview and Comparison of Technical Debt Measurement Tools. *IEEE Software*, *38*(3), 61–71. https://doi.org/10.1109/MS.2020.3024958

Baccarini, D. (1996). The concept of project complexity - A review. *International Journal of Project Management*, *14*(4), 201–204. https://doi.org/10.1016/0263-7863(95)00093-3

Baškarada, S. (2014). Qualitative Case Study Guidelines. *The Qualitative Report*. https://doi.org/10.46743/2160-3715/2014.1008

Becker, J., Knackstedt, R., & Pöppelbuß, J. (2009). Developing Maturity Models for IT Management. *Business & Information Systems Engineering*, *1*(3), 213–222. https://doi.org/10.1007/s12599-009-0044-5

Carnegie Mellon University. (2016). *Managing Technical Debt in Complex Software Systems*.

Christian B. Almazan University of Maryland, C. P. (n.d.). *A Comparison of Bug Finding Tools for Java*.

Chugh, D., Bazerman, M. H., & Milkman, K. L. (2008). *How Can Decision Making Be Improved?*

Daughtry III, J. M., & Kannampallil, T. G. (2005). Refactoring to Patterns. *The Journal of Object Technology*, *4*(4), 193. https://doi.org/10.5381/jot.2005.4.4.r2

Defense Science Board. (2018). February 2018 CLEARED FOR OPEN PUBLICATION DEPARTMENT OF DEFENSE OFFICE OF PREPUBLICATION AND SECURITY REVIEW. *2018*.

Ernst, N. A., Bellomo, S., Ozkaya, I., Nord, R. L., & Gorton, I. (2015). Measure it? Manage it? Ignore it? Software practitioners and technical debt. *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*. https://doi.org/10.1145/2786805.2786848

Fernández-Sánchez, C., Garbajosa, J., Yagüe, A., & Perez, J. (2017). Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. *Journal of Systems and Software*, *124*, 22–38. https://doi.org/10.1016/j.jss.2016.10.018

Guo, Y., Spínola, R. O., & Seaman, C. (2016). Exploring the costs of technical debt management – a case study. *Empirical Software Engineering*, *21*(1), 159–182. https://doi.org/10.1007/s10664-014-9351-7

Gustafsson, J., & Gustafsson, J. (n.d.). *Single case studies vs. multiple case studies: A comparative study*.

Harvard Business School. (2015). *Does IT Matter? An HBR Debate - HBS Working Knowledge - Harvard Business School*. https://hbswk.hbs.edu/archive/does-it-matter-an-hbr-debate

Henderson, J. C., & Venkatraman, N. (1993). Strategic alignment: Leveraging information technology for transforming organizations. In *REPRINTED FROM IBM SYSTEMS JOURNAL* (Vol. 32, Issue 1).

IPEK OZKAYA. (2019). *Managing the Consequences of Technical Debt: 5 Stories from the Field*. https://insights.sei.cmu.edu/blog/managing-the-consequences-of-technical-debt-5-stories-from-the-field/
Issue-Based Observation Form for Case Studies in Science Education. (n.d.).

Karlsson, M. (n.d.). *What Is a Case Study?*

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., & Arcelli Fontana, F. (2021). A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools. *Journal of Systems and Software*, *171*. https://doi.org/10.1016/j.jss.2020.110827

Lenarduzzi, V., & Fucci, D. (2019). *Towards an Holistic Definition of Requirements Debt*. http://arxiv.org/abs/1907.10887

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

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

MacCormack, A., & Sturtevant, D. J. (2016). Technical debt and system architecture: The impact of coupling on defect-related activity. *Journal of Systems and Software*, *120*, 170–182. https://doi.org/10.1016/j.jss.2016.06.007

Martin Fowler. (n.d.). *TechnicalDebtQuadrant*. Retrieved January 11, 2023, from https://www.martinfowler.com/bliki/TechnicalDebtQuadrant.html

Martini, A., Bosch, J., & Chaudron, M. (2014). Architecture technical debt: Understanding causes and a qualitative model. *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014*, 85–92. https://doi.org/10.1109/SEAA.2014.65

Nicolli, R. (2020). Hearing the Voice of Software Practitioners on Causes, Effects, and Practices to Deal with Documentation Debt. *Nicolli Rios*.

O'REILLY. (2022). *Managing Software Debt: Building for Inevitable Change by*.

Özcan-Top, Ö., & Demirors, O. (2019). Application of a software agility assessment model – AgilityMod in the field. *Computer Standards and Interfaces*, *62*, 1–16. https://doi.org/10.1016/j.csi.2018.07.002

Ramac, R., Mandic, V., Tausan, N., Rios, N., de Mendonca Neto, M. G., Seaman, C., & Spinola, R. O. (2020). Common Causes and Effects of Technical Debt in Serbian IT: InsighTD Survey Replication. *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, 354–361. https://doi.org/10.1109/SEAA51224.2020.00065

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., Izurieta, C., Seaman, C., & Spinola, R. (2022a). Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. *Journal of Systems and Software*, *184*. https://doi.org/10.1016/j.jss.2021.111114

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., Izurieta, C., Seaman, C., & Spinola, R. (2022b). Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. *Journal of Systems and Software*, *184*. https://doi.org/10.1016/j.jss.2021.111114

Rios, N., Spinola, R. O., de Mendonça Neto, M. G., & Seaman, C. (2019). Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. *Proceedings - 2019 IEEE/ACM*

International Conference on Technical Debt, TechDebt 2019, 3–12. https://doi.org/10.1109/TechDebt.2019.00009

Rios, N., Spínola, R. O., Mendonça, M., & Seaman, C. (2018, October 11). The most common causes and effects of technical debt: First results from a global family of industrial surveys. *International Symposium on Empirical Software Engineering and Measurement*. https://doi.org/10.1145/3239235.3268917

Rios, N., Spínola, R. O., Mendonça, M., & Seaman, C. (2020). The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. *Empirical Software Engineering*, *25*(5), 3216–3287. https://doi.org/10.1007/s10664-020-09832-9

ROBERT NORD. (2016). *The Future of Managing Technical Debt*. https://insights.sei.cmu.edu/blog/the-future-of-managing-technical-debt/

Rosser, L. A., & Norton, J. H. (2021). A Systems Perspective on Technical Debt. *IEEE Aerospace Conference Proceedings*, *2021-March*. https://doi.org/10.1109/AERO50100.2021.9438359

Rosser, L. A., & Ouzzif, Z. (2021a). Technical Debt in Hardware Systems and Elements. *IEEE Aerospace Conference Proceedings*, *2021-March*. https://doi.org/10.1109/AERO50100.2021.9438332

Rosser, L. A., & Ouzzif, Z. (2021b). Technical Debt in Hardware Systems and Elements. *IEEE Aerospace Conference Proceedings*, *2021-March*. https://doi.org/10.1109/AERO50100.2021.9438332

Shedd, C. (2015). *How to Effectively Address the Usability Debt Within Your Product | by Catriona Shedd | DesignIQ | Medium*. https://medium.com/iq-design/how-to-effectively-address-the-usability-debt-within-your-product-6b8693e6e853

Stephany Bellomo. (n.d.). *Got Technical Debt? Track Technical Debt to Improve Your Development Practices*. Retrieved July 3, 2022, from https://insights.sei.cmu.edu/blog/got-technical-debt-track-technical-debt-to-improve-your-development-practices/

Tomas, P., Escalona, M. J., & Mejias, M. (2013). Open source tools for measuring the Internal Quality of Java software products. A survey. In *Computer Standards and Interfaces* (Vol. 36, Issue 1, pp. 244–255). https://doi.org/10.1016/j.csi.2013.08.006

Vassallo, C., Panichella, S., Palomba, F., Proksch, S., Gall, H. C., & Zaidman, A. (2020). How developers engage with static analysis tools in different contexts. *Empirical Software Engineering*, *25*(2), 1419–1457. https://doi.org/10.1007/s10664-019-09750-5

Verdecchia, R. (n.d.). *Architectural Technical Debt: Identification and Management*. https://doi.org/10.13140/RG.2.2.11963.95525

Verdecchia, R., Kruchten, P., & Lago, P. (n.d.). *Architectural Technical Debt: A Grounded Theory*.

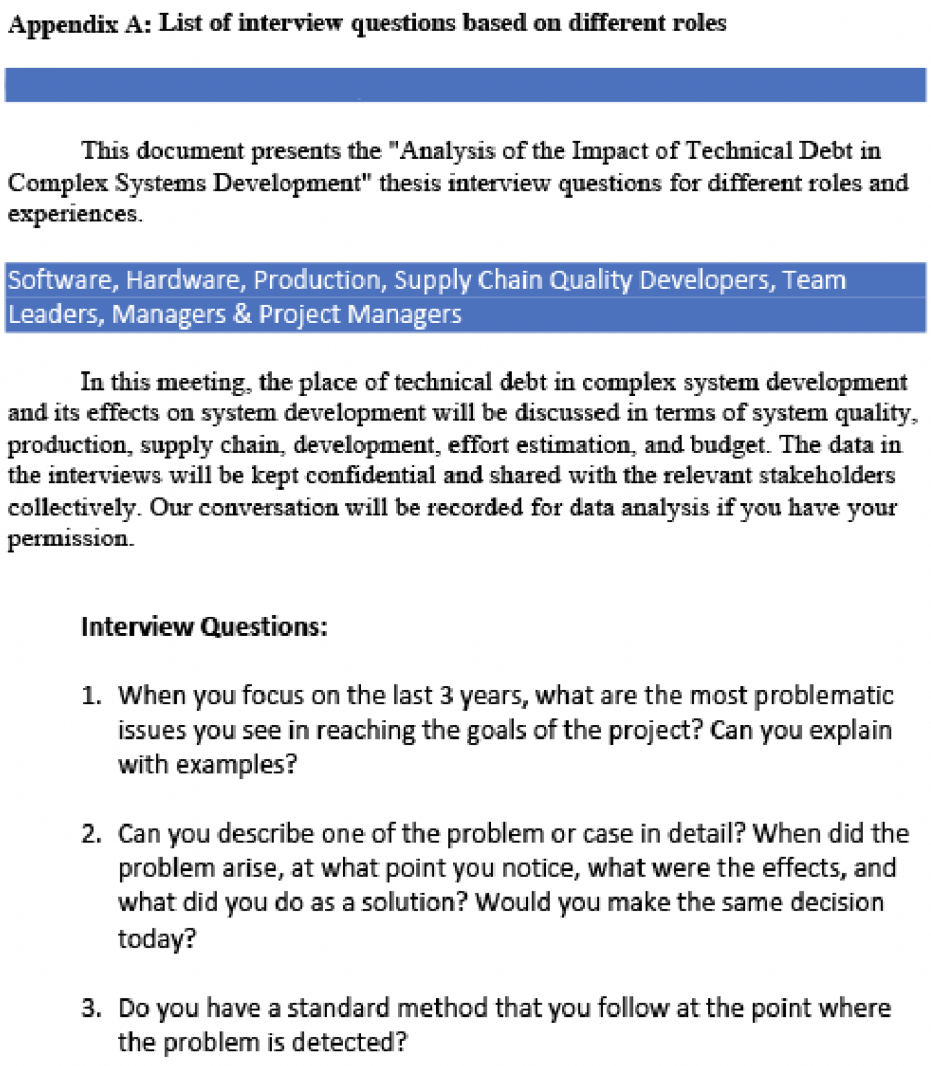
vom Brocke, J., Hevner, A., & Maedche, A. (2020). *Introduction to Design Science Research* (pp. 1–13). https://doi.org/10.1007/978-3-030-46781-4_1

Zampetti, F., Scalabrino, S., Oliveto, R., Canfora, G., & di Penta, M. (2017). How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines. *IEEE International Working Conference on Mining Software Repositories*, 334–344. https://doi.org/10.1109/MSR.2017.2

Zazworka, N., Spínola, R. O., Vetro, A., Shull, F., & Seaman, C. (2013). A case study on effectively identifying technical debt. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/2460999.2461005

APPENDIX

Appendix A: List of interview questions based on different roles



Appendix A: List of interview questions based on different roles

This document presents the "Analysis of the Impact of Technical Debt in Complex Systems Development" thesis interview questions for different roles and experiences.

Software, Hardware, Production, Supply Chain Quality Developers, Team Leaders, Managers & Project Managers

In this meeting, the place of technical debt in complex system development and its effects on system development will be discussed in terms of system quality, production, supply chain, development, effort estimation, and budget. The data in the interviews will be kept confidential and shared with the relevant stakeholders collectively. Our conversation will be recorded for data analysis if you have your permission.

Interview Questions:

1. When you focus on the last 3 years, what are the most problematic issues you see in reaching the goals of the project? Can you explain with examples?

2. Can you describe one of the problem or case in detail? When did the problem arise, at what point you notice, what were the effects, and what did you do as a solution? Would you make the same decision today?

3. Do you have a standard method that you follow at the point where the problem is detected?

Appendix B: IPMA project complexity participant -1-

IPMA - Measuring the Project Management Complexity: The Case of Information Technology Projects

Criteria:	Description of the criteria:		Very Low (1)	Low (2)	High (3)	Very High (4)
1. Objectives, Requirements and Expectations	Mandate and objective		■ Defined, obvious			Uncertain, vague
	Conflicting objectives		Few conflicts	■		Many conflicts
	Transparency of mandate and objectives		■ Quite transparent			Hidden
	Interdependence of objectives		Quite independent			■ Very interdependent
	Number and assesment of results		Low, monodimensional			■ Large, multidimensional
	Clear statement of requirements		Requirements perfectly clear	■		Requirements unclear
	Realistic expectations		Easily achievable	■		Unlikely to be achieved
	Clear strategic objectives(organizational)		Defined, obvious			■ Uncertain, vague
	Uncertain and changing regulatory requirements		■ Available, known			Uncertain, changing
	Complexity Rank	3	0,583333333			
2. Interested Parties, Integration	Interested parties, lobbies		Few parties			■ Numerous parties
	Categories of stakeholders		Few uniform categories			■ Many different
	Stakeholder interrelations		Few and well known relations		■	Unknown relations
	Interests of involved parties		Comparable interest			■ Divergent interests
	User involvement		■ User available and committed to the project			User uncommitted with the project
	Executive management support		Executive management committed to the project		■	Executive management uncommitted to the project
	Project sponsor supports project methodology		■ Sponsor committed with project methadology			Sponsor uncommitted with project methadology
	Complexity Rank	3	0,714285714			
3. Project Structure, Demand For Coordination	Structures to be coordinated		Few structures			■ Numerous structures
	Demand of coordination		Simple, straightforward			■ Demanding, elaborate
	Structuring of phases		Sequential			■ Overlapping, simultaneous
	Demand for reporting		Uni-dimensional, common			■ Multi-dimensional, comprehensive
	Complexity Rank	3	1			
4. Technology	Incompetence on using/applying technology		Technological competence in all of the project chain links		■	Technological Incompetence in any of the project chain links
	New technologies		Well known technologies used		■	Too many new technologies in place
	IT management support		Full IT management support			■ No IT management support
	Technology illiteracy		Stakeholders technology literacy	■		Stakeholders technology illiteracy
	Infrastructure, telecommunication constraints		Few			■ Many
	Complexity Rank	2	0,8			
5. Leadership, Teamwork, Decisions	Number of sub-ordinates		Few, small control span			■ Many, large control span
	Team structure		Static team structure		■	Dynamic team structure
	Leadership style		Constant and uniform		■	Adaptive and variable
	Decision-making processes		Few important decisions			■ Many important decisions
	Team motivated by the project		Highly motivated		■	Little motivation
	Hard-Working, Focussed Staff		Focused team		■	Dispersed team
	Near shore / offshore teams involved		Domestic teams		■	Offshore teams / Near shore teams involved
	Business aspects of project		Good know how in offshore / near shore teams		■	Teams unfamiliar with business / Technical aspects of the project
	Complexity Rank	3	0,8125			
6. Degree of Innovation, General Conditions	Technological degree of innovation		Known and proven technology	■		Unknown technology
	Demand of creativity		Repetitive approach		■	Innovative approach
	Scope for development		Limited			■ Large
	Significance on publica agenda		Public interest low		■	Large public interest
	Complexity Rank	1	0,75			
7. Project Organisation	Number of interfaces		Few			■ Many
	Demand for communication		■ Direct, not demanding, uniform			Indirect, demanding, manifold
	Hierarchical structure		Uni-dimensional, simple			■ Multidimensional, matrix structure
	Relations with permanent organisations		Few relations			■ Intensive, mutual relations
	Complexity Rank	4	**0,8125**			
8. Cultural and Social Context	Diversity of context		Homogenous		■	Diverse
	Cultural variety		Uniform, well known	■		Multicultural, unknown
	Geographic distances		■ Close, concentrated			Distant, distributed
	Social span		Small, easy to handle		■	Large, demanding
	Complexity Rank	4	0,5625			
9. Risk and Opportunities	Predictability of risks and opportunities		■ High, quite certain			Low, uncertain
	Risk probability, significance of impacts		Low risk potential, low impact	■		High risk potential, high impact
	Potential of opportunities		Many options for actions			■ Limited options for actions
	Options for action to minimise risks		■ Low potential of opportunities			Large potential of opportunities
	Complexity Rank	2	0,5			
10. Resources Including Finance	Availability of people, material, etc.		Available, known		■	Uncertain, changing
	Financial resources		■ One investor and few kind of resources			Many investors and kinds of resources
	Capital investment		Low(relative to project of the same kind)		■	Large(relative to project of the same kind)
	Quantity and diversity of staff		Low			■ High
	Complexity Rank	4	0,6875			
11. PM Methods, Tools and Techniques	Variety of methods and tools applied		Few, simple			■ Numerous, manifold
	Application of standarts		■ Common standarts applicable			Few common standarts applicable
	Availability of standarts		Much support available	■		No support available
	Propotions of PM to total project work		Low percentage		■	High percentage
	Incremental or iterative methodology used		Totally incremental methadology used	■		Totally iterative methadology used
	Complexity Rank	2	0,75			
Total Complexity Level (the complexity of the project against 62.5% of the minimum index in which a project is considered complex)			**0,723559908**			

Appendix C: IPMA project complexity participant -2-

IPMA - Measuring the Project Management Complexity: The Case of Information Technology Projects

Complexity

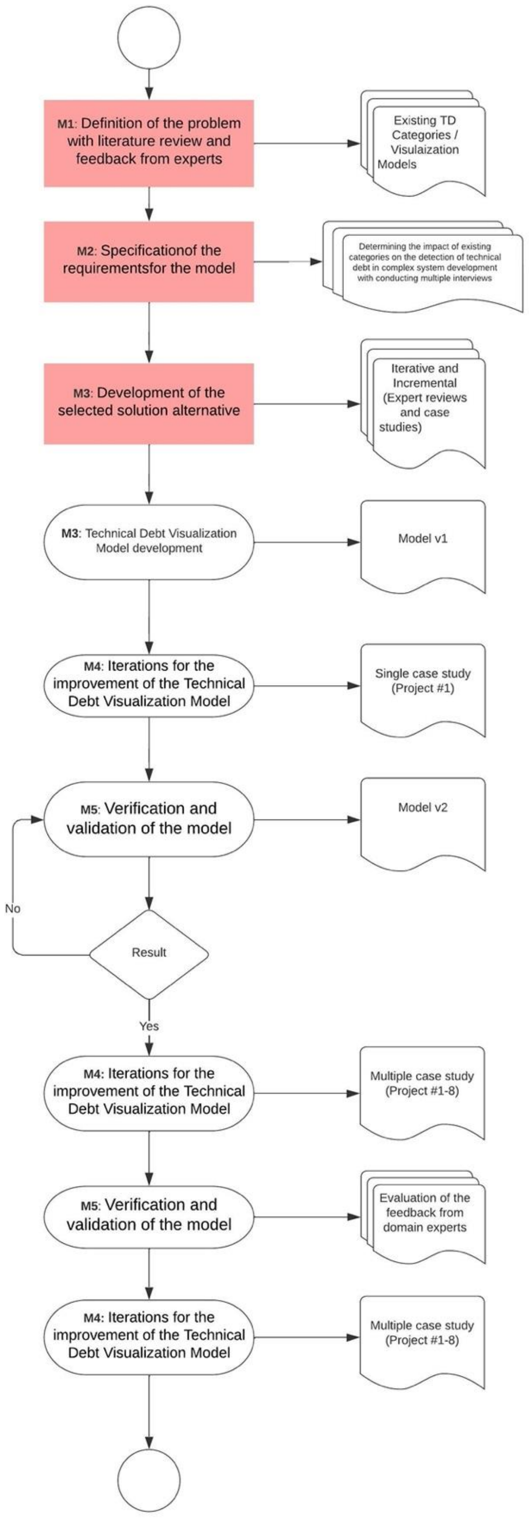
Criteria:	Description of the criteria:		Very Low (1)	Low (2)	High (3)	Very High (4)
1. Objectives, Requirements and Expectations	Mandate and objective		Defined, obvious	🟦		Uncertain, vague
	Conflicting objectives		Few conflicts			🟦 Many conflicts
	Transparency of mandate and objectives		Quite transparent		🟦	Hidden
	Interdependence of objectives		Quite independent			🟦 Very interdependent
	Number and assesment of results		Low, monodimensional			🟦 Large, multidimensional
	Clear statement of requirements		Requirements perfectly clear	🟦		Requirements unclear
	Realistic expectations		Easily achievable	🟦		Unlikely to be achieved
	Clear strategic objectives(organizational)		🟦 Defined, obvious			Uncertain, vague
	Uncertain and changing regulatory requirements		Available, known			🟦 Uncertain, changing
	Complexity Rank	3	🟩 0,611111111			
2. Interested Parties, Integration	Interested parties, lobbies		Few parties			🟦 Numerous parties
	Categories of stakeholders		Few uniform categories			🟦 Many different
	Stakeholder interrelations		Few and well known relations			🟦 Unknown relations
	Interests of involved parties		Comparable interest	🟦		Divergent interests
	User involvement		User available and committed to the project			🟦 User uncommitted with the project
	Executive management support		Executive management committed to the project		🟦	Executive management uncommitted to the project
	Project sponsor supports project methodology		Sponsor committed with project methadology	🟦		Sponsor uncommitted with project methadology
	Complexity Rank	3	🟩 0,821428571			
3. Project Structure, Demand For Coordination	Structures to be coordinated		Few structures			🟦 Numerous structures
	Demand of coordination		Simple, straightforward			🟦 Demanding, elaborate
	Structuring of phases		Sequential			🟦 Overlapping, simultaneous
	Demand for reporting		Uni-dimensional, common			🟦 Multi-dimensional, comprehensive
	Complexity Rank	3	🟩 1			
4. Technology	Incompetence on using/applying technology		Technological competence in all of the project chain links			🟦 Technological Incompetence in any of the project chain links
	New technologies		Well known technologies used			🟦 Too many new technologies in place
	IT management support		Full IT management support	🟦		No IT management support
	Technology illiteracy		🟦 Stakeholders technology literacy			Stakeholders technology illiteracy
	Infrastructure, telecommunication constraints		Few			🟦 Many
	Complexity Rank	2	🟩 0,75			
5. Leadership, Teamwork, Decisions	Number of sub-ordinates		Few, small control span			🟦 Many, large control span
	Team structure		Static team structure			🟦 Dynamic team structure
	Leadership style		Constant and uniform			🟦 Adaptive and variable
	Decision-making processes		Few important decisions			🟦 Many important decisions
	Team motivated by the project		Highly motivated		🟦	Little motivation
	Hard-Working, Focussed Staff		Focused team			🟦 Dispersed team
	Near shore / offshore teams involved		🟦 Domestic teams			Offshore teams / Near shore teams involved
	Business aspects of project		Good know how in offshore / near shore teams	🟦		Teams unfamiliar with business / Technical aspects of the project
	Complexity Rank	3	🟩 0,8125			
6. Degree of Innovation, General Conditions	Technological degree of innovation		Known and proven technology		🟦	Unknown technology
	Demand of creativity		Repetitive approach			🟦 Innovative approach
	Scope for development		Limited	🟦		Large
	Significance on publica agenda		Public interest low	🟦		Large public interest
	Complexity Rank	1	🟩 0,6875			
7. Project Organisation	Number of interfaces		Few			🟦 Many
	Demand for communication		Direct, not demanding, uniform			🟦 Indirect, demanding, manifold
	Hierarchical structure		Uni-dimensional, simple			🟦 Multidimensional, matrix structure
	Relations with permanent organisations		Few relations			🟦 Intensive, mutual relations
	Complexity Rank	4	🟩 1			
8. Cultural and Social Context	Diversity of context		Homogenous			🟦 Diverse
	Cultural variety		🟦 Uniform, well known			Multicultural, unknown
	Geographic distances		🟦 Close, concentrated			Distant, distributed
	Social span		Small, easy to handle		🟦	Large, demanding
	Complexity Rank	4	🟩 0,5625			
9. Risk and Opportunities	Predictability of risks and opportunities		High, quite certain		🟦	Low, uncertain
	Risk probability, significance of impacts		Low risk potential, low impact		🟦	High risk potential, high impact
	Potential of opportunities		Many options for actions	🟦		Limited options for actions
	Options for action to minimise risks		Low potential of opportunities		🟦	Large potential of opportunities
	Complexity Rank	2	🟩 0,6875			
10. Resources Including Finance	Availability of people, material, etc.		Available, known		🟦	Uncertain, changing
	Financial resources		One investor and few kind of resources	🟦		Many investors and kinds of resources
	Capital investment		Low(relative to project of the same kind)	🟦		Large(relative to project of the same kind)
	Quantity and diversity of staff		Low			🟦 High
	Complexity Rank	4	🟩 0,6875			
11. PM Methods, Tools and Techniques	Variety of methods and tools applied		Few, simple		🟦	Numerous, manifold
	Application of standarts		Common standarts applicable			🟦 Few common standarts applicable
	Availability of standarts		Much support available	🟦		No support available
	Propotions of PM to total project work		Low percentage			🟦 High percentage
	Incremental or iterative methodology used		Totally incremental methadology used			🟦 Totally iterative methadology used
	Complexity Rank	2	🟩 0,85			
Total Complexity Level (the complexity of the project against 62.5% of the minimum index in which a project is considered complex)			🟥 **0,774116743**			

Appendix D: Observation Form

Research Name: Analysis of the Impact of Technical Debt in Complex Systems Development
General Objective: To analyze employees' daily routines from different levels to understand their effects on technical debt management.

#:	Criterion:	Y:	N:	Observation:
1.	Did all relevant team members contribute to making critical decisions?			
2.	Was time pressure the most effective factor in making technical debt decisions?			
3.	When making the decision, have its effects on production, quality and overall project schedule been evaluated? If it was evaluated, was there a certain method followed?			
4.	Is the rate of exposure to different subsystems due to changes made in connected subsystems high?			
5.	In these scenarios was the technical debt cost measured?			
6.	Is the impact ratio of hardware and software on technical debt the same?			
7.	When making a redesign, refactor or implementation decision; did the managers take into account the maintenance and sustainability aspects of the system?			

Appendix E: TDVM model development strategy


M1: Definition of the problem with literature review and feedback from experts
Existing TD Categories / Visulaization Models
M2: Specificationof the requirementsfor the model
Determining the impact of existing categories on the detection of technical debt in complex system development with conducting multiple interviews
M3: Development of the selected solution alternative
Iterative and Incremental (Expert reviews and case studies)
M3: Technical Debt Visualization Model development
Model v1
M4: Iterations for the improvement of the Technical Debt Visualization Model
Single case study (Project #1)
M5: Verification and validation of the model
Model v2
No
Result
Yes
M4: Iterations for the improvement of the Technical Debt Visualization Model
Multiple case study (Project #1-8)
M5: Verification and validation of the model
Evaluation of the feedback from domain experts
M4: Iterations for the improvement of the Technical Debt Visualization Model
Multiple case study (Project #1-8)