TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Maria Bušujeva 230645IVGM

Measuring Technical Debt in the Public Sector: a Case Study of Estonia

Master's thesis

Supervisor: Richard Michael Dreyling III
Master’s degree

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maria Bušujeva 230645IVGM

“Tehnilise võla mõõtmine avalikus sektoris: Eesti juhtumiuuring”

Magistritöö

Juhendaja: Richard Michael Dreyling III
Magistrikraad

Tallinn 2024

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature, and the work of others have been referred to the best of my knowledge and ability. This thesis has not been presented for examination anywhere else.

Author: Maria Bušujeva

13.05.2024

Abstract

This master's thesis examines the measurement of technical debt in the public sector, focusing on a case study of Estonia. Through qualitative research methods, including thematic analysis of semi-structured interviews, the study highlights the challenges faced by public sector organizations in Estonia in addressing technical debt. The research emphasizes the importance of structured approaches and proactive measures to effectively manage technical debt in governmental IT projects. By implementing a framework for technical debt measurement, public organizations in Estonia can enhance decision-making, resource allocation, and overall digital government services' efficiency, reliability, and security. The findings underscore the perpetual nature of managing technical debt and the critical need for ongoing proactive measures and strategic management in the public sector.

Keywords: Technical Debt Measurement, Public Sector, Estonia, Information Technology, Governance, Risk Management.

This thesis is written in English and is 51 pages long, including 6 chapters, 2 figures and 1 table.

List of abbreviations and terms

COO	Chief Operating Officer
CTO	Chief Technical Officer
eGA	e-Governance Academy
IT	Information Technology
MKM	Ministry of Economic Affairs and Communications
RIA	Estonian Information System Authority
RIK	Centre of Registers and Information Systems
RIKS	State Infocommunication Foundation
RMIT	Information Technology Centre of the Ministry of Finance
SMIT	IT and Development Centre of the Ministry of the Interior
TEHIK	Health and Welfare Information Systems Centre

Table of contents

List of figures

List of tables

1 Introduction

The concept of technical debt has garnered significant attention as a critical factor influencing the quality and sustainability of software systems. Technical debt encapsulates the compromises made during the software development lifecycle, where hasty solutions may lead to deferred maintenance, increased complexity, and reduced agility over time. As software projects progress, the accumulation of technical debt poses challenges that can impede progress, increase costs, and undermine the overall viability of software products. Consequently, gaining a deep understanding of technical debt and developing effective strategies for its management have become paramount concerns for software development organisations.

This thesis aims to contribute to the scholarly discourse on technical debt by conducting a comprehensive investigation into its various dimensions, including conceptual foundations, measurement methodologies, and practical implications in real-world software projects. Through rigorous empirical research and critical analysis, this study seeks to look into the complexities of technical debt and offer insights that can inform decision-making and best practices in contemporary software engineering practices. By bridging theory and practice, this research aims to advance understanding of technical debt and provide actionable guidance for public organisations striving to achieve excellence and innovation in software development endeavours.

Furthermore, the significance of examining instances where technical debt management has faltered or measurement has been lacking. Numerous real-world examples highlight the adverse consequences of neglecting technical debt. For instance, cases of software systems plagued by accumulated technical debt have experienced reduced performance, increased downtime, and heightened vulnerability to security breaches. Moreover, inadequate measurement of technical debt has often resulted in unforeseen costs, project delays, and compromised software quality. Instances where technical debt has spiralled out of control underscore the critical need for effective measurement and proactive management strategies. By exploring these examples, this research aims to underscore

the urgency of addressing technical debt and provide insights into the potential ramifications of its neglect. Through this analysis, stakeholders can gain a deeper appreciation for the importance of incorporating robust technical debt management practices into software development processes.

This thesis intends to address two primary research questions concerning the measurement of technical debt in public organisations in Estonia. The first question explores the current approaches to measuring technical debt, with sub-questions delving into the methods and tools employed for measurement within projects in the Estonian public sector. The second question focuses on potential improvements in technical debt measurement, considering the applicability of suggested frameworks and identifying limitations specific to the public sector context in Estonia. The methodology is centred around semi-structured interviews and thematic analysis. Semi-structured interviews are utilised to gather insights from experts with experience in public sector IT projects regarding their approaches to measuring technical debt. These interviews allow for flexibility in exploring participants' perspectives while maintaining a focus on key research questions. Subsequently, thematic analysis is employed to identify recurring themes and patterns in the interview data, providing a structured framework for analysing and interpreting the findings.

1.1 Problem statement

In the realm of software development, technical debt refers to the potential expense of future revisions that may be needed as a result of choosing a quick but suboptimal solution over a more comprehensive one that may require additional time (Cunningham, 1992). The issue is multifaceted and includes a variety of types of debt (Melo et al., 2022). Similar to financial debt, failing to address technical debt can result in accruing "interest," leading to increased difficulty in implementing modifications or making them impossible. Technical debt can pose a significant risk to software systems by making them more difficult and expensive to maintain and update, and potentially leading to system failures or security vulnerabilities (Tom et al., 2013).

The public sector relies heavily on software systems to provide critical services to citizens, such as healthcare, education, and transportation. However, these systems often suffer from technical debt, resulting in increased maintenance costs, security

vulnerabilities, and reduced system performance. Despite the importance of managing technical debt in the public sector, there is a lack of research on how to measure technical debt in this context (Hansen, 2022). In order to provide strong arguments for funding and adequate attention it is necessary to find ways to measure technical debt and evaluate its impact.

The Estonian state institutions face challenges in maintaining existing IT systems due to a preference for allocating funds to new developments rather than upkeep (Lõvi, 2019). Consequently, operational systems quickly become outdated, necessitating a deliberate and separate budgeting approach for IT systems. This situation is exemplified by the ease of securing funds for new developments compared to maintaining or updating existing systems, with funds often originating from different sources (*Ibid.*). While significant investments in ICT projects are constantly being reported, there remains a lack of focus on long-term maintenance and support costs necessitating urgent action to address the growing technical debt and ensure the sustainability of the country's IT infrastructure.

The initial step to managing technical debt in software systems, particularly in large and complex legacy software systems, is to measure its levels (Melo et al., 2022). However, this can be a challenging task as technical debt is a broad and abstract concept. Establishing precise metrics is crucial in identifying the appropriate course of action to ensure that software systems can continue to support business and users over an extended period of time (Kruchten et al., 2012). This paper aims to address this gap by proposing a framework for measuring technical debt in public sector software systems, in order to help decision-makers prioritise technical improvements and allocate resources effectively.

This study will be analysing ways that technical debt is measured within the Estonian public sector and evaluating the viability of given methods. Unwisely used shortcuts during the software development process can lead to a variety of serious issues in the future. Therefore, the process of measurement of technical debt quantifies the costs and efforts required to create a comprehensive plan of eliminating the debt. The study will provide insights into the techniques and tools used in the public sector to measure technical debt in software systems and investigate possible improvements to adoption of effective measurement strategies.

2 Literature review

2.1 Understanding technical debt

Initially introduced by Ward Cunningham as a metaphor for describing the consequences of short-term expedient practices, technical debt refers to the accruing consequences of deferred tasks or suboptimal decisions (Cunningham, 1992). Technical debt not only impacts maintainability but also extends to various internal and external software qualities like performance, operability, and usability. Over time, definition has broadened to encompass deliberate or inadvertent actions and various forms, ranging from code deficiencies to architectural issues, anti-patterns, and even the impact on social structures and developer morale. While technical debt is inevitable to some extent, its management is crucial for software development success (Ciolkowski et al., 2021).

Technical debt is a metaphor used in software development to describe the consequences of taking shortcuts or making compromises during the development process. Just like financial debt, technical debt accumulates interest over time in the form of additional work that must be done to correct or address the shortcuts taken earlier. These shortcuts may include writing quick and dirty code, skipping proper documentation, delaying necessary refactoring, or neglecting testing procedures (Li et al., 2014). While in the short term these shortcuts may help in meeting deadlines or cutting costs, they can lead to increased complexity, reduced maintainability, and higher risks in the long run. Technical debt can slow down development, introduce bugs, and make future changes more difficult and costly. It is important for software development teams to manage and address technical debt proactively to maintain the health and sustainability of their software systems (Li et al., 2014).

Decision-making regarding technical debt prioritisation encompasses various proposed techniques derived from fields like finance and psychology, including methods such as cost-benefit analysis and the Analytic Hierarchy Process (Codabux et al, 2017). These approaches evaluate factors like severity, existence of workarounds, and the effort required for resolution, aiming to differentiate between potential and effective technical

debt. Despite the array of available techniques, their widespread adoption in the industry remains limited. Many companies opt for internal tools or bypass measuring their debts altogether, relying instead on singular factors like customer requests or issue severity, thereby lacking a comprehensive multi-criteria approach (Codabux et al, 2017).

The choice of technical debt measurement technique will vary based on the specific type of technical debt being assessed. The types of technical debt that were described in the tertiary study by Rios et al., 2018, are as follows:

- Design debt
- Code debt
- Architecture debt
- Test debt
- Documentation debt
- Defect debt
- Infrastructure debt
- Requirements debt
- People debt
- Build debt
- Process debt
- Automation test debt
- Usability debt
- Service debt
- Versioning debt

As pointed out by the study the most discussed types of technical debt are design, code and architecture debt (Rios et al., 2018). Ernst et al., 2015, note that bad architectural choices are the most commonly agreed source of technical debt among software engineers. Deliberate and unintended technical debt posed significant challenges, leading to ad-hoc decision-making and a lack of effective communication and rationale among stakeholders (Klinger et al., 2011). Therefore, this paper will also benefit from limiting research to primarily focus on these three types of technical debt and their measurement.

Definitions for the design, code and architecture technical debt will be based on the Rios et al., 2018 tertiary study and are presented in the following paragraph. Design debt refers

to debt that is identified by analysing the source code and identifying violations of good object-oriented design principles. This type of debt arises when the design of the codebase deviates from established best practices, leading to potential complications in maintenance and scalability. Code debt, on the other hand, pertains to issues found directly within the source code itself. This includes poorly written code that violates coding standards or best practices, resulting in reduced readability and increased complexity. Code debt can hinder the understanding of the codebase and make it more challenging to maintain and modify over time. Architecture debt encompasses problems encountered in the overall product architecture. This type of debt arises when architectural decisions are made hastily or without considering long-term implications. As a result, the architecture may not adequately meet evolving requirements or may become outdated as technologies and design patterns evolve. Architecture debt compromises internal quality aspects and may require significant refactoring. The full excerpt from the table with definitions and examples by Rios et al., 2018 can be found in the appendices section of the thesis.

2.2 Measuring technical debt

Further research becomes imperative to address the critical gap in the industry regarding the adoption of models for technical debt measurement. Industry-wide acceptance of these models remains restricted, emphasising the need for extensive research aimed at bridging this gap and crafting models tailored to suit the diverse needs of software development organisations (Codabux et al, 2017). Improved decision-making frameworks are essential, considering a combination of factors to prioritise debt items effectively. According to Klinger et al. (2011) technical debt decisions were being made by non-technical stakeholders rather than technical architects. This necessitates the development of more comprehensive strategies, leveraging methodologies to enhance decision-making in the context of technical debt management within software development.

A systematic mapping study conducted by Li et al. (2014) reveals a diverse range of measurement approaches for technical debt. These include calculation through mathematical formulas or models, utilising source code metrics, and estimation based on experiential knowledge, ranking among the most prominent methods. Quantifying

technical debt requires deciding on metrics of measurement. The majority of debt-related methodologies prioritise identifying and preventing defects rather than strategically managing critical infrastructure decisions, particularly concerning architecture (Nord et al., 2012).

Terms such as "code smells" and "spaghetti code" have been employed to tackle technical debt at the code level (Fowler et al., 1999). Refactoring techniques address the repayment of technical debt through localised changes to the codebase. These methods, often informed by defect analysis and software maintainability efforts, utilise metrics-based analysis to identify areas of potential debt within the system. For instance, static analysis tools can measure duplicate code, cyclomatic complexity, and the presence of god classes, offering insights into existing debt (Nord et al., 2012). However, these approaches have two limitations. Firstly, they primarily analyse code artefacts retrospectively, providing insights post-delivery without guiding adjustments during development to prevent debt accumulation. Secondly, while minor refactoring is feasible, addressing key design concerns after the fact often requires significant redesign and may not be straightforward (*Ibid.*). While these techniques offer avenues for addressing technical debt at the code level, their retrospective nature and limitations in handling significant design concerns post-delivery underscore the need for proactive strategies to prevent debt accumulation during development.

The Ministry of Interior 2018 report presents an approach to measuring technical debt, encompassing a two-phase methodology that involves locating technical debt in Fowler's quadrant and assessing its magnitude through static code analysis. The findings include the conversion of measurement results into hours, revealing an estimated technical debt size of approximately 71,100 software developer work hours for elimination or mitigation across analysed services and platforms (Ministry of Interior, 2018). Emphasising the importance of analysing technical debt from both the codebase and team perspectives, the report underscores the contextual nature of technical debt, highlighting the need for comparisons over time and across different services to effectively address and manage technical debt in software development projects (*Ibid*.).

The quantification approaches of technical debt encompass various methods and strategies used to measure and assess technical debt in software development projects. These approaches include the identification of code, design, or architecture smells as

indicators of potential technical debt, assessing the Return on Investment of refactoring activities, comparing the ideal state of software with its current state to understand quality gaps, evaluating alternative development paths to reduce technical debt accumulation, conducting cost and benefit analysis to estimate the impact of technical debt on projects, and performing risk assessments to understand the consequences of unaddressed technical debt (Perera et al., 2023).

Mayr et al. (2014) introduced a benchmark-driven approach to calculating technical debt. Their model for computing remediation costs of software integrated elements from three established technical debt calculation methodologies: the CAST model, SQALE model, and the SIG model. Authors managed to successfully derive metrics from these models and standardize them based on lines of code before incorporating into a framework. By utilising these diverse quantification approaches, organisations can make informed decisions to manage technical debt effectively and enhance software quality in the long term. Nevertheless, the choice of an approach will be solely dependent on the specific needs of the organisation.

2.3 Issues when measuring technical debt

Measuring technical debt presents several challenges that can complicate the assessment and management of software quality and maintainability. One of the primary issues lies in the ambiguity surrounding the definition of technical debt itself, as previously mentioned it encompasses various aspects of software development, including code quality, architecture, etc. (Melo et al., 2022). Additionally, technical debt is often subjective and context-dependent, making it difficult to establish universal metrics or criteria for measurement (Jaspan & Green, 2023). Furthermore, the multifaceted nature of technical debt requires a holistic approach that considers not only code-level issues but also architectural complexities, dependencies, and long-term implications.

Another challenge arises from the dynamic nature of software projects, where technical debt can accumulate and evolve over time, necessitating continuous monitoring and adaptation of measurement strategies (Kruchten et al., 2013). Finally, limited visibility and transparency into the accumulation of technical debt across different components and systems within an organisation can hinder effective decision-making and prioritisation of debt reduction efforts (Kruchten et al., 2012). At the same time on average, the cost of

managing technical debt in large software organisations is estimated to consume approximately 25% of the total development time (Martini et al., 2018).

Addressing these issues requires a nuanced understanding of technical debt and the development of robust measurement frameworks that account for its diverse manifestations and impacts throughout the software lifecycle. According to the Li et al., 2014, study some of the main issues in measuring technical debt include:

- Ambiguity in Terminology: The term "debt" is used in various ways by different individuals, leading to ambiguous interpretations of the concept of technical debt.
- Lack of Empirical Studies: There is a need for more empirical studies with high-quality evidence on the entire technical debt management process and the application of specific technical debt management approaches in industrial settings.
- Focus on Code-related Technical Debt: The study highlights that code-related technical debt and its management have received the most attention, indicating a potential lack of focus on other types of technical debt.
- Cost-Benefit Analysis: Technical debt is considered a risk in some studies and an investment in others. There is a need for further research on measuring the cost and benefit of technical debt to make informed decisions about incurring it.
- Differentiating Technical Debt from Non-Technical Debt: The study notes that there is limited effort in distinguishing between technical debt and non-technical debt in the existing literature, highlighting the importance of clear definitions and boundaries.

These issues underscore the complexity and challenges associated with measuring technical debt accurately and effectively in software development projects. Figure 1 illustrates a potential organisation of the technical debt landscape, depicting the progression of software enhancement from a given state (Kruchten et al., 2012). Within this depiction, discernible components such as new functionality and defects are evident, alongside elements perceivable only to software developers. The diagram highlights a distinction between evolutionary processes or associated challenges on the left-hand side and quality concerns, both internal and external, on the right-hand side. Furthermore, it

serves as evidence of complexity of understanding of technical debt, since it is mostly invisible.

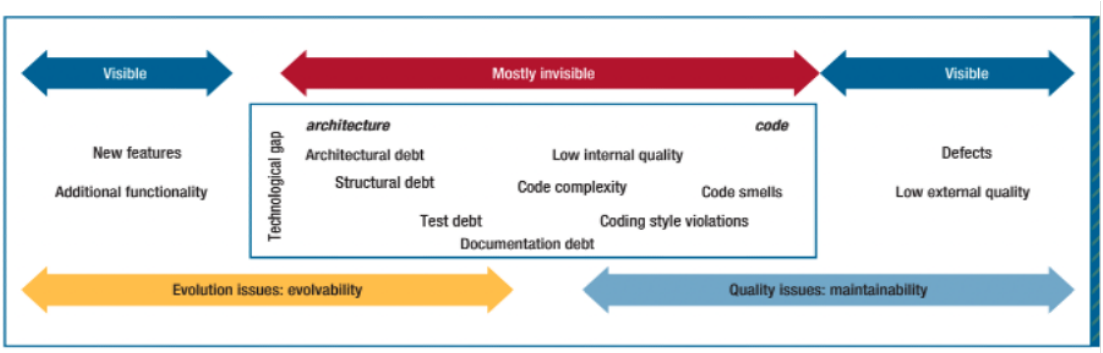

Figure 1. The technical debt landscape (Kruchten et al., 2012).

2.4 Tools and techniques for measuring technical debt

Conversely, automated code analysis tools represent another significant avenue in this domain. Numerous tools, including CAST, NDepend, SonarGraph, CodeMRI, SonarQube, SymphonyInsight, Code Inspector, and DV8, offer varying capabilities for technical debt measurement. These tools allow for flexibility in incorporating rules, defining metrics, and adjusting thresholds, though some, like Code Inspector and DV8, lack certain customization capabilities (Avgeriou et al, 2021). However, it's important to note the limitations inherent in automated tools, as they can sometimes generate false-positive results, necessitating cautious implementation (Melo et al, 2022).

The involvement of stakeholders, the creation of sustainable requirement lists, and the documentation of pending issues constitute the customer review approach. This method emphasises the importance of collaboration between the development team and clients, enabling effective management through thorough reviews of requirements and results. By integrating customer feedback, this approach facilitates adaptability in product development and requirement specifications, ultimately enhancing the identification of technical debt and overall efficiency (Melo et al, 2022; Kruchten et al., 2012).

Deserving a separate attention is artificial intelligence (AI), as it holds promising potential in revolutionising the landscape of technical debt measurement. AI technologies, particularly machine learning algorithms, can offer sophisticated analytical capabilities to identify, assess, and manage technical debt within software systems. These AI-driven approaches can automate the detection of code anomalies, architectural flaws, and other

indicators of technical debt. By analysing large volumes of code repositories, AI models can recognize patterns and correlations, enabling the prediction of potential areas susceptible to accruing technical debt (Srinivas et al., 2023). Moreover, AI-powered tools can assist in prioritising technical debt items by assessing their potential impact on system performance, security, and maintainability. Integrating AI into technical debt measurement can not only streamline the identification process but also empower developers and decision-makers with insights to proactively address and mitigate technical debt, thus fostering more resilient and efficient software systems.

Managing technical debt typically begins with the identification of technical debt items to compile a comprehensive list. Subsequently, the next phase involves assessing the debt items on the list by estimating their principal, interest amount, and interest probability (Spinola et al., 2019). While there's no universally accepted method for measuring technical debt, several approaches and metrics can help provide insights into its magnitude and consequences. Here are some common methods and techniques (Ernst et al., 2015):

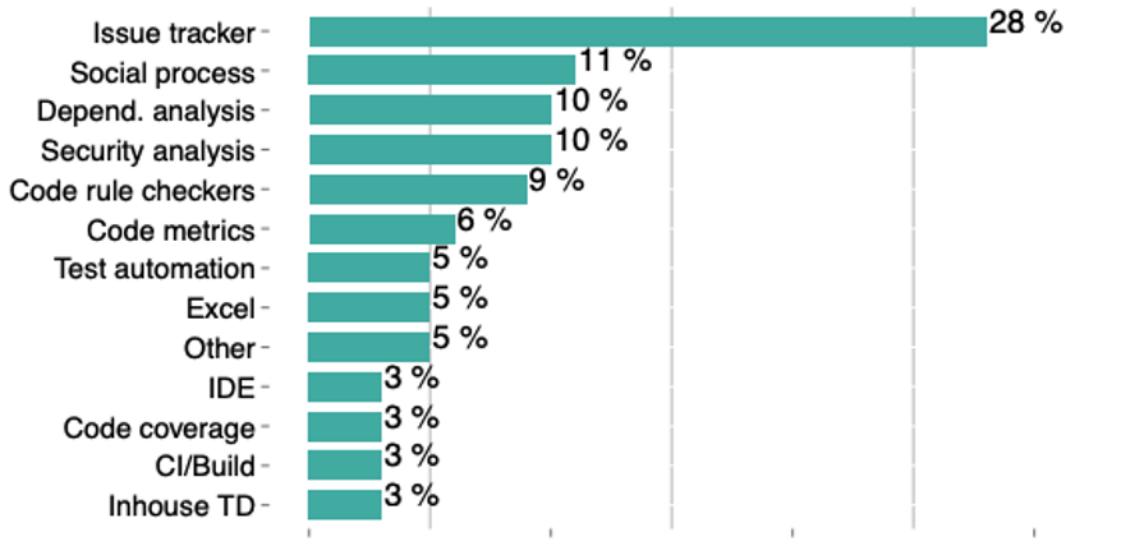

Figure 2. Tools used to analyse technical debt as percentage (Ernst et al., 2015).

According to study by Ernst et al., 2015, in the realm of technical debt measurement, issue trackers emerged as the predominant category of tools, with platforms like Redmine, Jira, and Team Foundation Server being the most widely utilised. Following this, other tool categories such as dependency analysis (e.g., SonarQube, Understand), code rule checking (e.g., CPPCheck, Findbugs, SonarQube), and code metrics (e.g., Sloccount) did not yield significant results.

Worth mentioning is also the SQALE (Software Quality Assessment based on Life-cycle Expectations) method, which offers a structured framework for estimating and managing technical debt in source code by identifying quality characteristics, calculating debt indexes for associated requirements. It is utilising pyramid indicators for debt distribution visualisation, and analysing debt to provide technical rationale for decisions. Organisations can deploy SQALE using compatible tools, monitor technical debt daily, and plan refactoring activities to improve code quality and reduce long-term maintenance costs, making it a valuable approach for enhancing software development practices and mitigating the impact of technical debt in software projects (Letouzey & Ilkiewicz, 2012). The limitations of the SQALE method include its focus on source code technical debt, dependency on compatible tools, subjectivity in debt assessment, training requirements, and scalability challenges in larger projects (Ibid.). These factors may impact the method's effectiveness and adoption in organisations seeking to manage technical debt beyond source code and at scale.

3 Methodology

3.1 Research questions

The goal of this research is to assess the current processes in measuring technical debt in the public sector in Estonia. Furthermore, to evaluate the effectiveness of the framework and tools and to look into any flaws in the process. This will eventually lead to developing a comprehensive framework for measuring technical debt in public sector software systems.

The following research questions (RQ) and sub-questions (SQ) were formulated to attain the research goals:

RQ1. What are the current approaches to measuring technical debt?

> **SQ1.1.** How is technical debt measured and monitored within projects in the Estonian public sector?
>
> **SQ1.2.** How are available tools or measurement techniques used?

RQ2. How would it be possible to improve measurement of the technical debt in public organisations in Estonia?

> **SQ2.1.** What framework for technical debt measurement would be valid for public sector organisations?
>
> **SQ2.2.** What are the limitations in the public sector for technical debt measurement?

3.2 Research method

The methodology adopted for this research is rooted in qualitative methods, focusing on an exploratory case study approach to investigate technical debt measurement within the public sector, specifically in Estonia. The methodology integrates semi-structured interviews as the primary data collection technique (Boyce & Neale, 2006). Qualitative research is based on non-numerical data collection and analysis from semi-structured interviews with decision-makers and software developers in governmental organisations.

The aim is to gain a deeper understanding of technical debt measurement techniques in its natural setting and to identify and explore the range of factors that contribute to it. The case study design provides an opportunity to investigate technical debt in a real-world context, while the semi-structured interviews with experts in the organisation offer a rich source of qualitative data on their perceptions of technical debt and the challenges of managing it (Yin, 2018).

The thematic analysis of qualitative data, derived from semi-structured interviews, serves as the basis for findings of this master's thesis. Thematic analysis constitutes a method centred on the identification, analysis, and reporting of patterns or themes inherent within the dataset (Braun & Clarke, 2006). It is a commonly used approach across various qualitative research designs and is particularly useful for exploring complex phenomena and understanding the meaning behind the data (Castleberry & Nolen, 2018). Through an examination of participants' narratives and perspectives, this study uncovers the underlying themes and patterns inherent in their experiences. By employing a systematic approach to coding and categorization, the analysis unveils the complexity of the data, allowing for a nuanced understanding of the phenomenon under investigation (*Ibid*.). Through this thematic lens, the thesis seeks to illuminate key insights, contribute to existing knowledge, and offer practical implications for theory, research, and practice in the field.

In qualitative research, the researcher plays a significant role in shaping the study outcomes, making it crucial to acknowledge and address personal biases openly (Castleberry & Nolen, 2018). Through the utilisation of reliable and esteemed data collection and analysis methodologies, researchers have the opportunity to establish trustworthiness and credibility among the audience (Yin, 2011). By providing detailed descriptions of the coding procedures and interpretations, researchers can invite scrutiny into decision-making processes (Castleberry & Nolen, 2018).

3.3 Interview design

In qualitative research, semi-structured interviews present an advantage by uncovering new insights and expanding upon established knowledge, even though participants may demonstrate response bias (Karatsareas, 2022). The interviews for this study were crafted to elicit rich insights from participants while ensuring the reliability and validity of the

data collected. To achieve this, the interviews were structured around a set of 15 open-ended questions, allowing for in-depth exploration of key themes and topics related to the research objectives. Questions can be found in the Appendix 3. The questions were designed to encourage participants to share their experiences, perspectives, and expert knowledge on the subject matter, thereby facilitating a comprehensive understanding of the phenomenon under investigation. Additionally, the interview design incorporated probing techniques to delve deeper into specific areas of interest and to clarify any ambiguities that arose during the conversation. Semi-structured interviews offer flexibility with set questions, allowing for follow-ups and clarifications, enhancing depth in qualitative research without the rigidity of structured interviews (Osborne & Grant-Smith, 2021).

Expert interviews can serve as a valuable primary data source in case studies and enrich the study by offering insights into intricate decision-making processes, connecting macro and micro perspectives, mitigating biases, and bolstering the evidential strength (Von Soest, 2022). The interviews were recorded and transcribed to facilitate thorough analysis. Prior consent was obtained from all participants for both recording and citation in the ensuing research findings. Although the interviews were conducted in English, potential communication challenges were acknowledged. Interviewees were encouraged to convey their ideas in their mother tongue if they felt more comfortable doing so, ensuring clarity and effective communication throughout the process. Overall, the interview design was tailored to facilitate the generation of rich qualitative data that would contribute to the robustness and depth of the study findings.

The interviewees for the study were chosen for their background in the public sector and perspectives on the technical debt measurement. Among them are individuals with extensive experience in technology leadership, architecture, and governmental IT projects. Their combined expertise provides valuable insights into the challenges and strategies associated with managing technical debt in public sector organisations. The list mostly consists of experts from Estonia, with the notable exception of Julia Richman, who offers an expert perspective from the United States.

4 Research findings and analysis

4.1 Context of research

The research focusing on technical debt measurement within the Estonian public sector is situated within the broader context of digital transformation and modernization initiatives undertaken by the government. Estonia is widely recognized for its advanced and sophisticated online public services, earning accolades for its digital innovations (Capgemini, 2007). Virtually all public services in Estonia feature an e-service component, underscoring the country's commitment to digitalization and accessibility. Notably, state and local government agencies, as well as entities in public and private law executing public law functions, are mandated to accept digitally signed documents (Kalvet, 2012). This comprehensive integration of e-services across governmental bodies demonstrates Estonia's proactive approach towards leveraging technology to enhance public service delivery and streamline administrative processes.

However, with the proliferation of complex IT systems and the rapid pace of technological advancements, public sector entities face a myriad of challenges in managing technical debt effectively. These challenges include balancing the need for continuous innovation with the imperative to maintain legacy systems, ensuring compliance with evolving regulatory frameworks, mitigating cybersecurity risks, and optimising resource allocation amidst budgetary constraints. Moreover, the inherently collaborative nature of public sector projects necessitates robust frameworks for measuring and addressing technical debt across diverse stakeholders and departments. Thus, understanding and effectively managing technical debt emerge as critical imperatives for sustaining the efficiency, reliability, and security of digital government services in Estonia.

Larger ministries have their own major departments for information systems (information technology development centres), the main ones being KeMIT, RMIT, RIK, SMIT, TEHIK, and RIA. These IT competence centres create public sector ICT services in a variety of fields, such as law enforcement, work, culture, health, environment, and social security. Establishing a dedicated IT competency centre within government infrastructure is instrumental in streamlining and optimizing IT operations but still maintaining flexibility (Grauer & Sipelgas, 2019). Government ministries and their agencies directly

manage ICT strategies, investments, data, and information architecture. Such IT hub consolidates specialised expertise, standardised practices, and promotes cost-efficient resource management. It plays a pivotal role in bolstering innovation, research, and development in the public sector, while also enhancing project management, security, and compliance efforts. Moreover, these centres foster skill development, vendor relationships, and aids in prioritising IT projects, aligning them with the government's strategic objectives. Nevertheless, admittedly the main risks for Estonia's e-government are a shortage of IT-skilled individuals and struggling IT systems (Lõvi, 2019).

The issues with Estonian IT systems are multifaceted and deeply rooted. Despite Estonia's reputation as an e-state, the technological infrastructure lags significantly behind (Hindre, 2022). For example, the existing systems in the social sector SKAIS-1 and SKAIS-2 developed by TEHIK, suffer from deficiencies, highlighting the challenges of maintaining outdated systems alongside developing new ones (*Ibid*.). This fragmented approach results in a patchwork system that lacks cohesion and efficiency, leading to complexities in service delivery. Lack of attention to maintenance of the systems and constant pressure from the legislators leads to even deeper problems with systems (Rudi, 2023).

The problem with financing IT systems in Estonia lies in the need for consistent funding that allows for flexibility, enabling continuous improvement of systems and the use of modern IT services (Ilves, 2022). The resource-intensive nature of developing and implementing individualised support systems further exacerbates the situation, requiring continuous manual intervention for validation and verification (Hindre, 2022). Moreover, the lack of seamless integration among various systems leads to operational inefficiencies and increases the likelihood of errors or discrepancies. The shortage of skilled IT professionals exacerbates these challenges, as the demand for expertise exceeds the available workforce, hindering timely and effective solutions. Despite efforts to modernise and streamline processes, the pervasive reliance on manual intervention and outdated systems continues to impede progress in delivering essential services to citizens (Rudi, 2023).

4.2 Interviewees

Below is a compilation of experts interviewed presented in alphabetical order.

Interviewee	Duration (min)	Recent relevant experience	Interview style
Andres Kütt	49.43	CTO at Jio Egov Center of Excellence (2018 - present), consulted TEHIK (2023)	Video conference call
Artur Novek	22.11	IT Architect at TEHIK (2017-present)	Video conference call
Heiko Vainsalu	38.47	Programme Director of Technology, eGA (2020-present); RIA IT Architect (2017-2018); RIA Domain Manager (2013-2017)	Video conference call
Jevgeni Krutov	21:17	Architect at Nortal, working on TEHIK projects (2019-present)	Video conference call
Julia Richman	25:25	COO of Colorado Governor's Office of Information Technology (2020-2023)	Video conference call
Kristo Vaher	59:14	Technology Director at Digital Nation (2023-present); MKM CTO (2018-2023)	Video conference call
Marti Lung	33:52	Head Of Development at SMIT (2020-present)	Video conference call
Martin Õunap	41:32	Head Architect at TEHIK (2018-present)	In person
Tarmo Hanga	36:00	Head Architect at RIA (2007-present)	Video conference call
Kaimar Karu	39:41	Mindbridge CTO (2020-present); Minister of Entrepreneurship and Information Technology of Estonia (2019–2020)	Video conference call

Table 1. List of expert interviews.

4.3 Thematic analysis of the interviews

The following chapter presents thematic analysis results from the conducted interviews, offering an exploration of the key themes and insights gathered during the discussions

with experts. The interviews focused on the current methodologies employed to measure technical debt in the public sector. Technical debt measurement emerges as a challenging issue for many of the experts emphasising the need for ongoing vigilance required to combat technical debt effectively. For example, Tarmo Hanga says: "*About the technical debt for me it's a never-ending question. You must fight it all the time and there are no correct answers.".*

The mentioned approaches to technical debt underscore the significance of continuous monitoring. Although currently organisations in Estonia do not have a formalised or structured approach towards measuring technical debt, there are still certain techniques and tools in use. Among tools mentioned by the experts were SonarQube, PMD, Jira, Dependency track, AI based tools and Dynatrace, which facilitate proactive identification of code deficiencies and issues with libraries. Tools such as SonarQube play a pivotal role in the Estonian public sector in code analysis, aiding in the detection of code smells and other indicators of technical debt accumulation.

However, the quest for improvement does not end with a mere understanding of current practices; it extends to envisioning a more effective future state. Here, the discourse shifts towards organisational culture and its profound impact on technical debt management and measurement. Kristo Vaher has highlighted the influence of organisational culture on technical debt: *"And, and I do think that culture is absolutely huge in terms of technical debt as well. It's going to be frustrating for engineers that are going to say that, hey, but we are unable, like... it, it's inconvenient for us to do these changes for an ever-expanding sort of legacy system that we have. We would like to build it again. And if our organisation says no, but we don't have time or money for this, just add those things. It creates this compound technical debt."* Interviewees overall agreed that culture emerges as a potential catalyst for enhanced technical debt measurement. Fostering a culture of adaptability and responsiveness to change can drastically improve the overall situation with measuring technical debt.

Blame and responsibility have been brought out as factors influencing the facing of technical debt within organisations. In order to facilitate an open discussion and proactive approach to measuring technical debt, it is imperative to enhance accountability and promote a culture of collaboration within organisations. This entails transcending bureaucratic inertia and instilling an organisational mindset of openness to change and

continuous improvement. *"Organisational culture, including factors like fear of judgement and blame culture, can significantly influence how technical debt is recognized and addressed within a team or organisation"* (Kaimar Karu). However, such endeavours are not devoid of challenges; resistance to change, bureaucratic impediments, and a lack of prioritisation pose formidable obstacles.

In regards to the technical debt measurement framework, interviewees agreed that it is a beneficial strategy to approach the issue of measuring technical debt. In addition to agreeing with the necessity of the framework, interviewees brought out several suggestions, such as a long-term approach to managing IT products and providing metrics and a list of available tools. As Kristo Vaher had put it: *"It could focus on certain sort of principles of a healthy organisation that is more tolerant to technical debt and is actually looking ahead more than five years at a time."* Furthermore, organisations need to grow expertise towards product-based thinking rather than project-based thinking. Andres Kütt emphasized the necessity to have a certain mind shift towards product-based approach in development: *"The, these issues are caused by people not thinking in terms of products, but in terms of projects."* The reliance on project-based funding, particularly from the European Union, exacerbates the challenge of sustaining IT systems beyond their initial development.

One of the notable challenges discussed was the allocation of budgets for projects related to technical debt within the public sector. Effectively managing technical debt requires adequate financial resources, yet navigating budgetary constraints in the public sector can prove to be particularly complex. *"Management decisions have a profound impact on technical debt, and it is crucial to consider the consequences of prioritisation and resource allocation in managing debt effectively."* (Kaimar Karu). Striking a balance between allocating funds for maintaining existing systems and investing in new solutions that resonate with constituents can pose a significant dilemma for decision-makers. In this context, prioritising technical debt initiatives may often take a backseat to projects with more immediate political appeal, potentially exacerbating long-term technical debt accumulation and compromising the resilience and efficiency of public sector IT infrastructure.

A significant number of interviewees highlighted the pivotal role of cyber security audits in shedding light on the extent of technical debt within systems. These audits serve as

crucial mechanisms for uncovering vulnerabilities and weaknesses in IT systems, thereby providing invaluable insights into the magnitude of technical debt present. By assessing the security posture of the organisation's infrastructure, applications, and processes, cyber security audits reveal areas where technical debt has accumulated, particularly in relation to outdated or insecure software components, inadequate security controls, and unresolved vulnerabilities. Moreover, these audits offer an opportunity to prioritise remediation efforts and allocate resources effectively to address the identified security risks. Thus, interviewees underscored the importance of integrating cyber security audits into the broader framework for measuring and managing technical debt, emphasising the need for a comprehensive approach to ensure the resilience and security of organisational IT systems.

In addition to the themes outlined, another noteworthy aspect deserving attention is the variance in understanding technical debt among the interviewees. This discrepancy in comprehension may stem from differing professional backgrounds, levels of experience, or organisational contexts. Some interviewees may perceive technical debt solely in terms of code quality and architecture, while others may consider broader factors such as infrastructure, security, or human resources. Understanding these varying perspectives is crucial for effectively addressing technical debt within organisations, as it influences decision-making processes, resource allocation, and prioritisation strategies. Acknowledging and reconciling these differences can lead to more comprehensive and informed approaches to managing technical debt and promoting organisational resilience and innovation.

While none of the experts explicitly identified the definition of technical debt as a concern, certain data suggests that there is room for refinement in understanding this concept. Interviewees have indicated that technical debt is a multifaceted and intricate issue, encompassing various dimensions beyond mere code quality or architecture. This recognition underscores the complexity inherent in managing technical debt within organisations. As such, refining the understanding of technical debt to encompass its broader implications, including infrastructure, security, and compliance considerations, is essential for devising effective strategies to mitigate its impact. By acknowledging the multifaceted nature of technical debt, organisations can adopt more holistic approaches to address this challenge and promote long-term sustainability and innovation.

4.3.1 Current approaches to measuring technical debt

The complex nature of technical debt poses challenges in effectively measuring and addressing it. Through analysis of the qualitative interviews, this study aims to contribute to the advancement of understanding and methodologies surrounding the measurement of technical debt in software development contexts. The techniques brought out by experts during the interviews can be divided into following categories:

- Manual reviews
- Automated tool utilisation
- Business owner's/user feedback

The tools highlighted by experts offer practical solutions and insights into effective technical debt management strategies. Metrics mentioned include vulnerabilities, dependencies, code smells, user satisfaction, bug fixing time, and time to implementation of new features. Technical debt is monitored by tracking metrics such as security issues, licensing compliance, and dependency updates. Regular code scans and analysis are conducted to identify areas of concern. As Tarmo Hanga aptly expresses, *"Actually, you have to choose some auditing tools and key points where to audit your software."* This highlights the strategic selection process involved in adopting auditing tools tailored to an organisation's specific needs and software architecture.

Nevertheless, Martin Õunap admitted that merely monitoring by developers is not enough and eventual responsibility also lies on the architects and manual revision: *"And yes, we are using SonarQube, we see there are problems over there. And as architect on TEHIK's side, when some kind of work is given to our server, this is one point where architect should go and see what is the SonarQube metrics are."* He also mentioned the metrics that are tracked using Dependency track: *"In Dependency track, I have, I'm using three types of measures. I'm measuring how many security issues is, I'm measuring what kind of licences are used and I measure if there are new releases for the dependencies."*.

The interviewees described the current approaches to measuring technical debt in their organisation, emphasising the importance of proactive prevention rather than reactive fixes. They highlighted the need for collaboration between architects and product owners to assess new requirements against existing technical landscapes.

Utilising cyber security audits for measuring technical debt proves highly beneficial for organisations. However, the procurement process involved in conducting these audits can be both costly and time-consuming. Additionally, a common challenge arises from the tendency to focus these audits primarily on new systems, inadvertently neglecting older systems that may serve as dependencies. Although, many techniques have been mentioned the process of measuring technical debt within public sector is not formalised and is something that comes to mind when systems start failing.

4.3.2 Opportunities for improvement in technical debt measurement

In exploring opportunities for improvement in technical debt measurement, it becomes evident that a multifaceted approach is necessary to navigate the complex landscape of debt accumulation and mitigation within the public sector. The interviewees discussed opportunities for enhancing the measurement of technical debt in public organisations in Estonia. They emphasised the potential benefits of implementing a framework for technical debt measurement to improve decision-making and resource allocation.

In the realm of technical debt management within the public sector, a resounding theme emerges: the necessity of continuous effort and strategic foresight. This underscores the perpetual nature of the battle against technical debt, emphasising the need for ongoing proactive measures. Kaimar Karu further emphasises the importance of strategic management with the assertion that the two-phase approach could involve revising existing technical debt issues and instituting ongoing monitoring to prevent future accumulation.

Security and risk mitigation are paramount concerns, as highlighted by Tarmo Hanga's emphasis on prioritising security patches: *"Our first concern is always security patches for libraries for software itself."* This recognition underscores the critical importance of safeguarding systems against vulnerabilities and potential threats. Additionally, Marti Lung's observation regarding the impact of legislation on complexity echoes the need to navigate regulatory frameworks while mitigating technical debt effectively.

Kristo Vaher proposed a valuable insight into the detection of technical debt, emphasizing the importance of monitoring specific metrics. *“So, I think measuring these indicators how quickly bugs get fixed and how quickly new features get implemented into the products or services, there are two indicators for seeing when technical debts might*

actually appear and start affecting your systems." (Kristo Vaher). He highlighted the significance of tracking the speed at which bugs are addressed and the efficiency with which new features are integrated into products or services. According to Vaher, these two metrics serve as crucial indicators for identifying the emergence of technical debt and its potential impact on systems. By closely monitoring these indicators, organizations can proactively address technical debt issues before they escalate, ensuring the long-term health and performance of their systems.

Cultural shifts and organisational mindsets play a pivotal role in shaping approaches to technical debt. Tarmo Hanga's assertion that an agile mindset fosters openness to change emphasises the need for organisational adaptability. Moreover, Kaimar Karu's reflection on how organisational culture influences debt recognition and mitigation underscores the significance of fostering a culture that encourages transparency and accountability. Marti Lung also emphasised the importance of enhancing internal capabilities to gain a deeper understanding of the systems. Increasing the level of involvement and knowledge among team members is crucial for improving the overall situation.

Lastly, on the adoption of AI for technical debt measurement Julia Richman said: *"But I think what's really cool about the opportunity in AI is like using AI to refactor systems at scale and sort of recode and, you know, upgrade from version to version.".* The long-term risks and future technological shifts inherent in technical debt management are articulated by Kristo Vaher's warning about the risks for the Estonian public sector in the era of AI. As he puts it, entering the actual AI era of the digital world poses significant challenges, necessitating proactive strategies to navigate these shifts: *"Let's order this AI type of thing and a lot of systems are not ready for implementing AI in this way as we expect so immediately technical debt flags are going to be erased.".* This highlights the imperative for public sector entities to not only address immediate technical debt but also anticipate and prepare for future technological advancements and associated risks.

5 Discussion

The interview results indicate that organisations often choose to utilise internal tools or completely overlook the measurement of their technical debt. Instead, they rely on singular factors such as customer requests or the severity of cybersecurity issues, resulting in a lack of a comprehensive multi-criteria approach (Codabux et al., 2017). A case study conducted involving a mid-sized company, interviewing 27 software practitioners with the goal of comprehending their definitions, characterizations, and prioritizations of technical debt revealed that developers often have their own taxonomy for technical debt (Codabux & Williams, 2013). This surely correlates with findings in the current study and highlights the importance of formalising the approach in order to know how healthy systems actually are.

Firstly, manual management of technical debt measurement involves a hands-on, non-tool-dependent procedure executed by software professionals. This method entails the identification and measurement of technical debt through the establishment of metrics and benchmarks, complemented by manual analysis of system performance and the application of formulas to calculate diverse ratios (Melo et al, 2022).

Automated management involves the utilisation of software and automated resources to detect and assess technical debt. When considering these resources, a key issue is the prevalence of false positives: the number of instances incorrectly identified as technical debt. An excess of false positives can divert attention away from genuine technical debt. It might be beneficial to look into already successful approaches that used multiple systems to measure technical debt (Mayr et al., 2014).

Furthermore, customer feedback plays a crucial role in ensuring the success of a project's quality requirements. This involves engaging stakeholders progressively, compiling lists of sustainable requirements, and documenting any outstanding issues. Consequently, it is imperative for all management to review requirements with both the client and the development team. This collaborative approach allows to adapt the product and refine specified requirements in response to customer feedback, facilitating the identification of technical debts more efficiently.

Many software codebases exhibit complexity, making them difficult to comprehend and costly to modify and enhance. Prioritising technical debt presents a formidable challenge, particularly in modern systems comprising millions of lines of code and involving multiple development teams, thereby lacking a holistic overview. Moreover, the perpetual dilemma between enhancing existing code and introducing new features necessitates rigorous management (Kruchten et al., 2013).

Readily accessible version-control data can unveil the behaviour and patterns within the development organisation. The focus of the tools is primarily on code technical debt, with minimal attention given to design debt and architectural debt. However, architectural debt, in particular, can have a significant impact on maintenance efforts. There is also a lack of consensus on standardised rules and metrics for measuring technical debt, resulting in discrepancies among tools and creating confusion regarding the importance of certain rules and how to customise them to suit individual needs.

Quantifying technical debt through static analysis provides a glimpse into its scale. While measurement may shed light on the extent of technical debt, its utility in guiding actionable decisions remains limited. Instead, prioritising technical debt must extend beyond mere code analysis to encompass its broader business impact. Specifically, the additional time and resources required for feature development due to existing technical debt. This crucial consideration, unmeasurable from the code alone, underscores the necessity for a comprehensive perspective. Ultimately, striking a balance between improving existing code and introducing new features is paramount, highlighting the perpetual trade-off inherent in technical debt management and the need for meticulous project prioritisation aligned with long-term objectives.

Legacy and technical debt are often added up together, but they represent distinct concepts. Legacy commonly refers to old code lacking in quality that was not authored by the current team. Facing legacy code can have significant repercussions, particularly if a key contributor has left the organisation. If one of these primary contributors leaves and their work is not adequately understood by others, their portion of the codebase may swiftly transition into legacy code (Dedeke, 2012).

Escalating complexity within software development brings forth multifaceted consequences. Notably, the impact on the roadmap becomes visible as cycle times

gradually lengthen over time, transforming what used to be manageable tasks into protracted endeavours. This elongation diminishes predictability, exacerbating the already intricate task of software estimation. Simultaneously, the team bears the brunt of this complexity surge, experiencing a surge in turnover rates as the allure of working on needlessly convoluted tasks wanes. Moreover, the team's dependency on key individuals intensifies, amplifying the risk associated with sole experts in particular code segments a phenomenon often referred to as. Meanwhile, users face the ramifications first hand, encountering a surge in bugs that manifest as a tangible decline in software quality.

Translating technical debt measurement results into an understandable language for decision-makers and ministry officials serves several crucial purposes. Firstly, it facilitates effective communication and comprehension of the implications of technical debt within the organisation. By presenting technical debt metrics in a language that resonates with decision-makers, such as financial terms or risk assessments, it enhances their ability to grasp the significance of technical debt and its potential impact on project timelines, budgets, and overall organisational objectives. Secondly, it fosters informed decision-making by providing decision-makers with actionable insights derived from technical debt analysis.

Clear and comprehensible explanations of technical debt metrics enable decision-makers to prioritise investments, allocate resources effectively, and make strategic decisions that mitigate risks associated with technical debt. Moreover, translating technical debt measurement results into understandable language helps bridge the gap between technical experts and non-technical stakeholders, fostering collaboration and alignment across different levels of the organisation. Ultimately, by translating technical debt measurement results into an understandable language, decision-makers and ministry officials are empowered to make informed decisions that drive organisational success and mitigate the adverse effects of technical debt.

Effectively conveying risks to decision-makers entails framing them in a language that resonates with their priorities and responsibilities. Decision-makers are typically focused on organisational objectives, financial outcomes, and strategic initiatives. Therefore, risks should be articulated in terms of their potential impact on these areas, such as project delays, budget overruns, reputational damage, or hindrance to achieving strategic goals.

Moreover, it's essential to emphasise the urgency and severity of these risks, highlighting the consequences of inaction or inadequate mitigation efforts. By contextualising risks within the broader organisational context and aligning them with decision-makers' objectives, they are more likely to recognize the importance of addressing them and take ownership of mitigation strategies. Additionally, fostering a culture of accountability ensures that decision-makers understand their role in risk management and actively participate in implementing and monitoring mitigation measures. This combination of clear communication, contextualization, and accountability empowers decision-makers to proactively address risks and safeguard the organisation's interests.

Various methods exist for measuring technical debt within software development efforts. One approach involves utilising system health analysis tools to quantify and visualise technical debt within code bases. These tools provide relevant, objective, and actionable data to help identify, quantify, and resolve technical debt within software systems. Technical assessments conducted on code bases using these tools can assess the level of technical debt present.

Another method involves defining a set of metrics to measure the extent of technical debt accumulation within software systems. Metrics may include bug fix time, architectural cyclicality, propagation cost, and other indicators of code health. Coded results can then be used to indicate the performance levels of code bases. The main challenge would be agreeing on the necessary metrics and benchmarks. Each public organisation could develop a personal approach. Nevertheless, a universal baseline understandable equally by everyone can significantly benefit the overall system.

Additionally, quantifying the cost of technical debt is crucial for understanding its impact on software development efforts. This involves projecting the programmatic and economic impacts of technical debt and mapping the output from architectural health analysis tools to associated financial outcomes. By linking technical characteristics of code bases to business outcomes such as productivity, defect density, staff turnover, growth rates, cost performance, and schedule performance, organisations can gain insights into the true cost of technical debt.

Measuring technical debt over time is essential for recognizing its evolution within software systems. Technical debt can be present from the beginning of a design or

accumulate over time as software functionality changes. By monitoring changes in coding practices, market pressures, and system objectives, organisations can assess how technical debt evolves and take proactive measures to manage it effectively.

Technical debt management tools play a crucial role in addressing technical compromises during software evolution by providing support for informed decision-making and facilitating the mitigation of technical debt. Some tools help in quantifying code metrics related to technical debt, allowing teams to measure the impact of technical debt on software quality and make informed decisions (Saraiva et al., 2021). Allowing to cross check with multiple tools and techniques can significantly improve the accuracy in measuring technical debt (Mayr et al., 2014).

5.1 How would it be possible to improve measurement of the technical debt in public organisations in Estonia?

According to Nord et al., 2012, measuring technical debt involves assessing the trade-off between short-term benefits and long-term consequences in software development. Here are some approaches to measure technical debt:

- Code Analysis Tools: Utilise static code analysis tools to identify potential technical debt in the codebase. These tools can detect code smells, duplicate code, complexity metrics, and other indicators of poor code quality.
- Architectural Analysis: Conduct architectural analysis to understand the dependencies and structural elements contributing to technical debt. Metrics based on architecture structure and dependency analysis can help quantify technical debt outcomes.
- Rework Cost Calculation: Calculate the rework cost associated with implementing new architectural elements or making changes to the system. This cost can be based on detecting changing dependencies that create interest payments in the form of rework.
- Dependency Analysis: Analyse the dependencies within the system to identify areas where technical debt may be accumulating. Understanding the dependencies can help in quantifying the impact of changes and potential rework costs.

- Economic Models: Develop economic models to account for the cost of paying back technical debt. Consider the future cost of paying back debt, make the debt visible, and analyse the consequences of payback or carrying the debt.
- Metrics for Refactoring: Use metrics to guide the refactoring process and assess the quality of the system at the architecture level. Metrics such as duplicate code, cyclomatic complexity, and code smells can provide insights into potential technical debt.
- Empirical Studies: Conduct empirical studies to assess the impact of technical debt on software projects. Gather data on induced and unintentional debts, challenges faced, and decision-making processes related to managing technical debt.
- Quantifying Value: Consider quantifying the value of infrastructure and quality-related tasks, especially architectural ones, to understand the trade-off between short-term and long-term value in software development.

By employing these measurement approaches and considering the various aspects of technical debt, organisations can gain insights into the impact of debt on their software projects and make informed decisions to manage and mitigate technical debt effectively.

Transitioning from a project-based mentality to a product-based mentality is crucial for effectively managing technical debt and fostering sustainable development practices. Unlike projects, which have defined start and end dates, products are continuous entities that evolve over time. By embracing a product-based mindset, organisations shift their focus from short-term project deliverables to long-term value creation and ongoing improvement. This approach encourages teams to prioritise the maintenance and enhancement of existing systems alongside the development of new features, ensuring that technical debt is continuously monitored and addressed throughout the product life cycle. Moreover, adopting a product-based mentality promotes collaboration, agility, and customer-centricity, as teams work iteratively to deliver value and meet evolving user needs. Ultimately, this shift enables organisations to build resilient, adaptable systems that can effectively navigate the complexities of modern software development and drive sustainable growth.

Managing technical debt effectively in software development projects can be tricky for ensuring the quality, maintainability, and success of the software product. Software developers and project managers should be aware of the meaning of technical debt and

its implications on software projects. Educating team members about the importance of managing technical debt can help foster a proactive approach to addressing potential issues (Melo et al., 2022). Harnessing data with advanced digital tools has the potential to enhance both decision-making processes and the management of service quality (Morgareidge et al., 2014).

5.2 Framework recommendations

Some of the recommendations could include conducting regular reviews of requirements documentation to identify and address any instances of technical debt (Melo et al., 2022). Refine requirements continuously to ensure they are clear, complete, and aligned with stakeholder needs (*Ibid*.). Encouraging collaboration and feedback from all project stakeholders to improve requirement quality. Prioritising requirements based on their criticality and impact on the project. Develop a clear roadmap for requirement implementation, considering dependencies, constraints, and trade-offs. Establishing a structured process for managing changes to requirements to minimise the accumulation of technical debt (*Ibid*.).

Utilise tools and metrics to assist in the identification and measurement of technical debt. Implement automated resources or software solutions that can streamline the management of technical debt in requirements. Track relevant metrics to monitor the impact of technical debt on project progress and quality (Nord et al., 2012). Foster open communication and collaboration among team members, stakeholders, and clients to ensure a shared understanding of requirements and project goals. Encourage regular feedback loops and discussions to address any ambiguities, conflicts, or changes in requirements promptly. Incorporate risk management practices into requirement engineering processes to proactively identify and mitigate potential sources of technical debt. Anticipate and address risks associated with incomplete, ambiguous, or poorly defined requirements to prevent future challenges and rework (Melo et al., 2022).

A comprehensive framework for measuring technical debt in the public sector necessitates careful consideration of the unique characteristics and requirements inherent in governmental organisations. To begin, it is essential to define technical debt within the context of public sector IT projects, taking into account factors such as regulatory

compliance, security, and long-term sustainability. This clear definition has to be agreed upon by all of the organisations and maintained centrally to avoid disagreements.

Once the technical debt definition has been agreed upon, the framework must establish measurable metrics and indicators to quantify its extent, considering aspects like code quality, system complexity, security vulnerabilities, and maintenance effort. Precisely defining metrics becomes imperative in delineating the most suitable path forward, thereby safeguarding software systems' ability to sustainably support business operations and user needs over the long term (Kruchten et al., 2012).

Selecting appropriate measurement tools and techniques, such as code analysis tools, dependency scanners, and documentation assessments, is to be left for the IT organisations to decide internally based on the list of tools that are most suitable for the task. Approaches predominantly centred on code analysis overlook the distinction between various design challenges that result in increased rework expenses, thus leaving unanswered the critical inquiry into which issues carry the most substantial weight (Ozkaya & Nord, 2019). A multitude of companies choose to utilise internal tools or forgo measuring their technical debt entirely, instead relying on singular factors such as customer requests or issue severity. This approach overlooks the necessity of adopting a comprehensive multi-criteria approach to effectively assess technical debt (Codabux et al., 2017).

Li et al. (2014) introduced a technical debt management approach comprising five key stages: identification, measurement, prioritisation, repayment, and monitoring. Integration with existing project management processes, such as Agile or waterfall methodologies, is crucial for effective implementation. Defining thresholds and prioritisation criteria based on impact, risk levels, and strategic alignment with organisational goals enables informed decision-making. Additionally, establishing a governance structure for oversight, continuous monitoring, and transparent reporting of technical debt is essential. One of the best practices for managing technical debt is dedicating an iteration within a 10-week release cycle specifically for debt reduction efforts (Codabux & Williams, 2013).

Managing and mitigating technical debt risks require the development of proactive strategies, including risk mitigation plans and resource allocation. Education and

awareness programs aimed at project teams, managers, and decision-makers are vital to emphasise the importance of technical debt management and its implications for project success. Continuous evaluation of the framework's effectiveness, coupled with adjustments based on lessons learned and emerging best practices, ensures its ongoing relevance and effectiveness. Ultimately, alignment with organisational goals ensures that the framework supports the broader mission, vision, and strategic priorities of the public sector organisation.

SonarQube used by organisations is among one of the most popular tools utilised for technical debt measurement (Avgeriou et al., 2020). While all tools assess maintainability issues to some extent, not all of them consider the consequences of these issues, such as extra maintenance costs and the probability of additional work, thereby limiting the effectiveness of technical debt as a communication medium. Secondly, most tools rely solely on static analysis in their calculation models, overlooking valuable sources of information from version history, issue trackers, and email exchanges (Avgeriou et al., 2020).

While empirical evidence currently lacks to support the implementation of enhanced processes and tools for managing technical debt, according to Martini et al., 2018, existing literature on technical debt and related research on change management suggest potential future maturity steps that organisations could achieve with the implementation of research findings. These steps encompass a progression from merely measuring technical debt to institutionalising processes for its management and, ultimately, to fully automating decisions regarding refactoring. For instance, companies may start by adopting tools for identifying technical debt and implementing indicators to aid in its estimation and prioritisation.

Subsequently, they may institutionalise these processes across the organisation, enabling aligned prioritisation of technical debt and resource allocation. Finally, as the organisation matures, it may transition to fully data-driven decision-making processes, leveraging statistics collected from historical data or benchmarking against reference systems. Achieving such advanced stages of technical debt management necessitates a comprehensive approach integrating research insights and practical implementation strategies.

5.3 Recommendations for further research

For further research, several avenues could be explored to enhance understanding and address gaps in the current knowledge of technical debt measurement in the public sector. Firstly, conducting a larger-scale study involving a more extensive sample of public sector organisations and IT professionals could provide broader insights into the challenges and best practices associated with managing technical debt. Additionally, incorporating quantitative analysis alongside qualitative methods would offer a more comprehensive understanding of the quantitative impact of technical debt on project outcomes and organisational performance.

Furthermore, investigating in an experimental setting the effectiveness of different measurement frameworks and tools in the public sector context could help identify practical strategies for assessing and prioritising technical debt. Lastly, exploring the role of organisational culture, leadership, and governance structures in influencing technical debt management practices could provide valuable insights into the socio-technical aspects of this phenomenon and inform the development of tailored interventions and strategies.

Future directions in technical debt management encompass several areas, including the need for holistic methods to assess and prioritise technical debt, especially considering its economic and long-term impacts. Additionally, there's a call for better understanding and managing technical debt induced by emerging technologies like cloud-native approaches, machine learning, and agile frameworks (Ciolkowski et al., 2021). Moreover, integrating technical debt measurement seamlessly into agile and DevOps practices and mitigating the risk of overlooking technical debt while emphasising feature development remain key areas for further exploration.

5.4 Limitations and future work

The conducted research has unveiled deeper underlying issues that warrant attention and resolution. Despite its insights, the study is constrained by several limitations. These include a relatively small sample size of interviewees and the absence of quantitative data analysis. While the qualitative findings offer valuable insights, the lack of quantitative data may limit the generalizability of the study's conclusions. Additionally, the research

may benefit from broader participant representation to ensure a comprehensive understanding of the subject matter. As well as providing more comparisons by introducing cases from other countries. Addressing these limitations through future research endeavours could enhance the validity and robustness of the findings, thereby contributing to a more nuanced understanding of the complexities surrounding the topic at hand.

Moving forward, future research endeavours could explore avenues to address the identified limitations and further enhance the validity and robustness of findings in technical debt measurement within the public sector. This may involve expanding the sample size of interviewees to capture a more diverse range of perspectives and experiences. Additionally, integrating quantitative data analysis methods alongside qualitative approaches could provide a more comprehensive understanding of the phenomenon. Furthermore, comparative studies across different public sector organisations or longitudinal studies tracking the evolution of technical debt over time could offer valuable insights into patterns and trends. By addressing these avenues, future research can contribute to advancing knowledge and informing strategies for effectively managing technical debt in public sector contexts.

6 Conclusion

In conclusion, this master's thesis on measuring technical debt in the public sector, with a focus on Estonia, sheds light on the critical importance of effectively managing technical debt in governmental IT projects. The research findings highlight the challenges faced by public sector organisations in Estonia in measuring and addressing technical debt, emphasising the need for structured approaches and proactive measures. The study underscores the significance of continuous effort and strategic foresight in navigating the complex landscape of debt accumulation and mitigation within the public sector.

Moreover, the research findings underscore the importance of implementing a framework for technical debt measurement to improve decision-making and resource allocation in public organisations in Estonia. The study highlights the perpetual nature of the battle against technical debt, emphasising the need for ongoing proactive measures and strategic management. Security and risk mitigation emerge as paramount concerns, underscoring the critical importance of safeguarding systems against vulnerabilities and potential threats.

In conclusion, this thesis provides insights into the challenges, successes, and potential strategies for managing technical debt effectively in governmental IT projects in Estonia. The research outcomes offer a comprehensive understanding of the current approaches to measuring technical debt and opportunities for improvement within the public sector. By addressing these challenges and leveraging the opportunities identified, public sector organisations in Estonia can enhance their digital government services' efficiency, reliability, and security, ultimately contributing to sustainable digital transformation and modernization initiatives.

References

Avgeriou, P., Taibi, D., Ampatzoglou, A., Arcelli Fontana, F., Besker, T., Chatzigeorgiou, A., Tsintzira, A. (2021). An Overview and Comparison of Technical Debt Measurement Tools. IEEE Software, 38(3), 61-71.

Avgeriou, P., Taibi, D., Ampatzoglou, A., Fontana, F. A., Besker, T., Chatzigeorgiou, A., Lenarduzzi, V., Martini, A., Moschou, A., Pigazzini, I., Saarimäki, N., Sas, D., De Toledo, S. S., & Tsintzira, A. A. (2020). An overview and comparison of technical debt measurement tools. IEEE Software, 38(3), 61–71. https://doi.org/10.1109/ms.2020.3024958

Boyce, C. & Neale, P. (2006). Conducting in-depth Interviews: A Guide for Designing and Conducting In-Depth Interviews. Pathfinder International Tool Series.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Capgemini, B. NV/SA. (2007). The User Challenge: Benchmarking the Supply of Online Public Services. 7th Measurement, European Commission, Directorate General for Information Society and Media, Available online at: http://www.ch.capgemini.com/m/ch/tl/EU_eGovernment_Report_2007.pdf

Castleberry, A., & Nolen, A. (2018). Thematic analysis of qualitative research data: Is it as easy as it sounds? Currents in Pharmacy Teaching and Learning, 10(6), 807-815. https://doi.org/10.1016/j.cptl.2018.03.019

Ciolkowski, M., Lenarduzzi, V., Martini, A. (2021). 10 Years of Technical Debt Research and Practice: Past, Present, and Future. IEEE Software, vol. 38(6), 24-29. https://doi.org/10.1109/ms.2021.3105625

Codabux, Z., & Williams, B. (2013). Managing technical debt: An industrial case study. 2013 4th International Workshop on Managing Technical Debt (MTD). https://doi.org/10.1109/mtd.2013.6608672

Codabux, Z., Williams, B., Bradshaw, G. L., & Cantor, M. (2017). An empirical assessment of technical debt practices in industry. Journal of Software: Evolution and Process, 29(10). https://doi.org/10.1002/smr.1894

Cunningham W. (1992). The WyCash portfolio management system. Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications.

Dedeke, A. (2012). Improving Legacy-System Sustainability: A Systematic Approach. IT Professional, 14(1), 38–43. doi:10.1109/mitp.2012.10

Ernst, N.A., Bellomo, S., Ozkaya, I., Nord, R.L., & Gorton, I. (2015). Measure it? Manage it? Ignore it? software practitioners and technical debt. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). Refactoring: Improving the Design of Existing Code. Boston: Addison-Wesley Professional.

Grauer, M., Sipelgas, K. 2019. How to set up a competence centre for innovation. Procure2innovate

Hansen, M. E. K (2022). Technical Debt Management in the public sector. IT University of Copenhagen, Computer Science.

Hindre, M. (2022, September 7). SKA juht kaheaastasest IT-arendusest: meil on majas Lotte porgandimasin. ERR. https://www.err.ee/1608708193/ska-juht-kaheaastasest-it-arendusest-meil-on-majas-lotte-porgandimasin

Ilves, L. (2022, September 8). *Kindlasti tuleb IT-le kulutada rohkem.* ERR. https://www.err.ee/1608708700/luukas-ilves-kindlasti-tuleb-it-le-kulutada-rohkem

Jaspan, C., & Green, C. (2023). Defining, measuring, and managing technical debt. IEEE Software, 40(3), 15–19. https://doi.org/10.1109/ms.2023.3242137

Kalvet, T. (2012). Innovation: a factor explaining e-government success in Estonia. Electronic Government, an International Journal, 9(2), 142. doi:10.1504/eg.2012.046266

Karatsareas, P. (2022). Semi-Structured Interviews. In R. Kircher & L. Zipp (Eds.), Research Methods in Language Attitudes (pp. 99–113). chapter, Cambridge: Cambridge University Press.

Klinger, T., Tarr, P., Wagstrom, P., & Williams, C. (2011). An enterprise perspective on technical debt. Proc. 2nd Work. Managing Technical Debt (MTD '11), ACM Press, May 2011, pp. 35–38, doi: 10.1145/1985362.1985371.

Kruchten, P., Nord, R. L., Ozkaya, I., (2012). Technical Debt: From Metaphor to Theory and Practice. IEEE Software.

Kruchten, P.B., Nord, R.L., Ozkaya, I., & Falessi, D. (2013). Technical debt: towards a crisper definition report on the 4th international workshop on managing technical debt. ACM SIGSOFT Softw. Eng. Notes, 38, 51-54.

Letouzey, J., & Ilkiewicz, M. (2012). Managing Technical Debt with the SQALE Method. IEEE Software, 29(6), 44–51. https://doi.org/10.1109/ms.2012.129

Li, Z., Avgeriou, P., & Liang, P. (2014). A Systematic Mapping Study on Technical Debt and Its Management. Journal of Systems and Software, 101, 12-31. DOI: 10.1016/j.jss.2014.12.027

Lõvi, S. (2019, November 11). Riigikontroll: IT-süsteemide käigushoidmine vajab teadlikku eelarvestamist. ERR. https://www.err.ee/1001671/riigikontroll-it-susteemide-kaigushoidmine-vajab-teadlikku-eelarvestamist

Martini, A., Besker, T., & Bosch, J. (2018). Technical Debt tracking: Current state of practice. Science of Computer Programming, 163, 42–61. https://doi.org/10.1016/j.scico.2018.03.007

Mayr, A., Plosch, R., & Korner, C. (2014). A Benchmarking-Based Model for Technical Debt Calculation. 2014 14th International Conference on Quality Software. doi:10.1109/qsic.2014.35

Melo, A., Fagundes, R., Lenarduzzi, V., Santos, W. (2022). Identification and Measurement of Technical Debt Requirements in Software Development: a Systematic Literature Review. Journal of Systems and Software. https://doi.org/10.1016/j.jss.2022.111483

Ministry of Interior. (2018). Final report: The development of ICT services in the administrative area of the Ministry of the Interior and the sustainability of management financing and the impact on ensuring internal security. 17.08.2018 [WWW] https://www.smit.ee/files/ikt-finantseerimine-pwc-lopparuanne-veeb.pdf?a8072e9c5b

Morgareidge, D. L., Cai, H., & Jun, J. (2014). Performance-driven design with the support of digital tools: Applying discrete event simulation and space syntax on the design of the emergency department. Frontiers of Architectural Research, 3(3), 250– 264. https://doi.org/10.1016/j.foar.2014.04.006

Nord, R.L., Ozkaya, I., Kruchten, P.B., & Gonzalez-Rojas, M. (2012). In Search of a Metric for Managing Architectural Technical Debt. 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, 91-100.

Osborne, N., & Grant-Smith, D. (2021). In-Depth interviewing. In Cities research series (pp. 105–125). https://doi.org/10.1007/978-981-16-1677-8_7

Ozkaya, I., & Nord, R. (2019, December 16). Data-Driven Management of Technical Debt. Retrieved March 30, 2024, from https://insights.sei.cmu.edu/blog/data-driven-management-of-technical-debt/.

Perera, J., Tempero, E., Tu, Y., & Blincoe, K. (2023). Quantifying Technical Debt: a systematic mapping study and a conceptual model. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.06535

Rios, N., Mendonça, M., & Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Information & Software Technology, 102, 117–145. https://doi.org/10.1016/j.infsof.2018.05.010

Rudi, H. (2023, February 6). *Poliitikute kihk toetusi ja pensione tõsta pani SKA keerulisse olukorda.* ERR. https://www.err.ee/1608876080/poliitikute-kihk-toetusi-ja-pensione-tosta-pani-ska-keerulisse-olukorda

Saraiva, D., Kulesza, U., Freitas, G., Almeida, R. R., et al. (2021). Technical Debt Tools: A Systematic Mapping Study.

Spínola, R., Zazworka, N., Vetrò, A., Shull, F., & Seaman, C. (2019). Understanding automated and human-based technical debt identification approaches-a two-phase study. Journal of the Brazilian Computer Society, 25(1). https://doi.org/10.1186/s13173-019-0087-5

Srinivas B. P., Binta, S., & Kaushal, S. (2023). Artificial Intelligence for Technical Debt Management in Software Development. ArXiv.org, ArXiv.org, 2023.

Tom, E., Aurum, A., Vidgen, R. (2013). An exploration of technical debt. Journal of Systems and Software.

Von Soest, C. (2022). Why do we speak to experts? Reviving the strength of the expert interview method. Perspectives on Politics, 21(1), 277–287. https://doi.org/10.1017/s1537592722001116

Yin, R. K. (2011). Qualitative Research from Start to Finish. The Guilford Press.

Yin, R. K. (2018). Case Study Research and Applications - Design and Methods. Los Angeles: SAGE Publications, Inc.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis

I, Maria Bušujeva (Date of Birth: 23.04.1991)

1. Allow the Tallinn University of Technology without any charges (Plain licence) my work

"Measuring Technical Debt in the Public Sector: Case Study of Estonia",

supervised by Richard Michael Dreyling III,

1.1. to be reproduced for the purpose of conservation and electronic publication, including the digital repository of the Tallinn University of Technology, until the end of copyrighted time limit;

1.2. to be available to the public through the Tallinn University of Technology online environment, including the digital repository of the Tallinn University of Technology, until the end of the copyrighted time limit.

2. I am aware, that all rights, named in section 1, will remain to the author.

3. I confirm that by allowing the use of the Plain licence, no intellectual rights of third parties will be violated as set in the personal data protection act and other legislation.

Signed digitally

13.05.2024

Appendix 2 – Table from Rios et al., 2018, on technical debt types

TD types.

Type	Definition	Situations where debt itens can be found
Design Debt	Refers to debt that can be discovered by analyzing the source code and identifying violations of the principles of good object-oriented design.	- Violations of the principles of good object-oriented design; - Some types of code smells; - Complex classes or methods; - Excessive design complexity.
Code Debt	Refers to the problems found in the source code (poorly written code that violates best coding practices or coding rules) that can negatively affect the legibility of the code making it more difficult to maintain.	- Unnecessary code duplication and complexity; - Bad style that reduces the readability of code; - Over-complex code.
Architecture Debt	Refers to the problems encountered in product architecture, which can affect architectural requirements. Usually, architectural debt could be the result of sub-optimal upfront solutions, or solutions that become sub-optimal as technologies and patterns become superseded, compromising some internal quality aspects, such as maintainability.	- Violation of modularity; - Complex architectural behavioral dependencies; - Architectural compliance issues; - System-level structure quality issues; - Non-uniferm usage of architectural policies and patterns; - Lack of handling interdependent resources; - Lack of addressing non-functional requirements; - Implementation of immature architecture techniques.

Appendix 3 – Interview questionnaire

Intro

Tell me about your relevant professional experience.

Could you describe your understanding of technical debt?

Let's build common ground for the interview and define technical debt. Architectural debt.

Section 1: Current Approaches to Measuring Technical Debt

1. What are the current approaches to measuring technical debt that you are aware of?
    1.1. Can you describe how technical debt is currently measured and monitored within projects?
    1.2. How are available tools (ex. SonarQube) or measurement techniques utilised in assessing technical debt within projects?

Additional Probing Questions:

    1.2.1. Can you provide specific examples of technical debt encountered in projects?
    1.2.2. How do you prioritise addressing technical debt within projects?
    1.2.3. Are there any notable challenges or successes related to measuring technical debt?
    1.2.4. Can you provide any relevant metrics for measuring technical debt?

Section 2: Opportunities for Improvement in Technical Debt Measurement

2. How can the measurement of technical debt in public organisations in Estonia be enhanced?
    2.1. Do you believe a framework for technical debt measurement would be beneficial for public sector organisations in Estonia? Why?
    2.2. What do you perceive as the primary limitations or barriers to effectively measuring technical debt within the public sector?

Additional Probing Questions:

    2.2.1. How do you envision the integration of technical debt measurement practices?

2.2.2. What support or resources do you think would facilitate more effective measurement and management of technical debt in the public sector?

2.2.3. In your opinion, how might organisational culture influence the recognition and mitigation of technical debt?

3. Any other remarks relating to measurement of technical debt?