Leiden University

ICT in Business and the Public Sector

A Performance Measurement System for Scrum/DevOps Environments

Name:	Nikolay E. Pavlov
Student ID:	s4150759
Date:	18/09/2025
1st supervisor:	Joost Visser
2nd supervisor:	Paul van Leeuwen
Company supervisor:	Klaas Pier Mulder

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Einsteinweg 55
2333 CC Leiden
The Netherlands

Abstract

Background: Scrum and DevOps have revolutionized the software delivery process at an industry level by emphasizing iterative development, automation, and continuous deployment. While Scrum focuses on team efficiency through accurate planning and stable velocity, DevOps prioritizes system reliability and operational stability. Hybrid Scrum/DevOps teams are in need of a unified set of KPIs to effectively measure their performance.
Aim: This thesis aims to identify the most relevant KPIs that accurately reflect the software delivery performance of hybrid Scrum/DevOps teams. With our research, we propose a hybrid performance measurement framework that integrates Scrum and DevOps metrics, balancing process efficiency and reliability.
Method: This study employs a Design Science Research approach unfolding into three steps - problem exploration, solution design, and evaluation. Data collection involves survey distribution, interviews and existing dashboard analysis. By combining them with state-of-the-art solutions, we design and evaluate a set of KPIs which will form a holistic framework. We consider external factors such as collaboration and culture in the analysis to account for the in-situ setting of the experiment. We review existing tools for performance tracking, eventually defining best practices for integrating Agile and DevOps metrics into a cohesive performance measurement framework.
Results: Findings suggest that Agile teams often manipulate Scrum-based KPIs to maintain favorable metrics, while DevOps-oriented KPIs provide a more objective assessment. The data collection findings further indicate conflicting points in the presently implemented performance measurement standards in the form of conflicting perspectives or measurements based on convenience. The study identifies DORA metrics and the SPACE framework as crucial to evaluating software delivery success.
Conclusion: Our research shows that a hybrid Performance Measurement Framework can be used to offer a balanced reflection of both development process efficiency and operational reliability. The gathered insights provide actionable recommendations for organizations seeking to optimize software delivery.

Keywords:
Agile, DevOps, Performance Measurement, KPIs, DORA Metrics, Agile-DevOps Teams, Software Delivery

"Measurement is the first step that leads to control and eventually to improvement. If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it."

— H. James Harrington

Contents

Chapter 1

Introduction

In the perpetual evolution of the contemporary digital landscape, organizations face unprecedented pressure to live up to customer requirements by delivering software components of sufficient quality within restrictive time periods. Such attempts to maximize performance often result in unmet deadlines or accumulated technical debt, which builds tension between management representatives and development teams. To prevent this, a better look is needed towards the existing performance measurement frameworks to evaluate how they can be adapted to the modern software development approaches – Scrum and DevOps, and the perspectives of all parties involved in the production of software.

1.1 Software Development Methodologies

Over the past two decades, software development methodologies have evolved significantly to meet the increasing demands for faster delivery, higher quality, and better alignment with business needs. Traditional approaches like the Waterfall model, which is characterized by linear and sequential development, were largely succeeded by Agile methodologies that emphasize flexibility, customer collaboration, and iterative delivery. Among these, Scrum emerged as a dominant framework due to its structured yet adaptable process based on roles, ceremonies, and sprints (Schwaber and Sutherland, 2011).

Parallel to Agile's rise, the DevOps movement gained traction, seeking to bridge the gap between development and operations. DevOps introduces practices such as Continuous Integration (CI), Continuous Deployment (CD), infrastructure as code, and real-time monitoring to enable rapid and reliable software delivery. Its focus on automation and cross-functional collaboration complements the iterative and feedback-driven nature of Agile (Leite et al., 2019; Offerman et al., 2022).

As organizations increasingly combine Agile and DevOps practices, hybrid Scrum/DevOps teams are becoming the norm. On this point, Digital.ai (2023) report that 49% of large organizations implement a hybrid form of software development methodologies. In their paper, Samarawickrama and Perera (2017) share that such teams integrate Scrum planning and review cycles with DevOps automation and operational efficiency to deliver software in a continuous and collaborative manner.

1.2 Existing Challenges

While the integration of Scrum and DevOps offers substantial benefits, it also introduces new complexities, especially in performance measurement. Traditional metrics such as velocity, lines of code, or defect counts fail to capture the nuanced dynamics of modern development teams. Metrics may overlook essential qualitative aspects such as team morale, cross-functional collaboration, or system resilience (Forsgren et al., 2021).

Furthermore, organizations often struggle with fragmented or misaligned measurement practices. KPIs may be defined in isolation from strategic objectives or used inconsistently across different teams (Bouwers et al., 2012). Overemphasis on output metrics can incentivize superficial performance, while under-representing enablers like automation maturity, deployment health, or developer satisfaction (Georgsson, 2024).

This ambiguity leads to several challenges: lack of visibility into team dynamics, difficulty in benchmarking or tracking progress, and inability to use measurement systems as tools for learning and improvement. As performance dashboards become increasingly central to strategic execution, their design and metric selection require careful attention (Bugwandeen and Ungerer, 2019; Brahimi et al., 2019).

1.3 Problem Statement

Hybrid Scrum/DevOps teams operate in a fast-paced, collaborative, and continuously evolving environment. Despite the widespread adoption of Agile and DevOps principles, organizations lack an effective, comprehensive, and customized performance measurement system that reflects the realities of these teams (Ayyash, 2024).

Most existing metrics are either too abstract to drive team-level insights or too narrowly focused on delivery speed and operational throughput. Consequently, decision makers are left with incomplete information and teams may pursue performance indicators that do not align with organizational goals.

While results are the ultimate measure of achievements, they are often a derivative of operational environment conditions. In the context of DevOps, Davis and Daniels (2016) label these conditions as pillars and summarize them in the following list:

- Collaboration - The ability of individuals and teams to work together toward a shared goal, thus emphasizing the importance of strong interactions both within and between different teams.
- Affinity - The strength of relationships between different teams that is developed by building empathy, trust and mutual understanding.
- Tooling - Tools serve as the enabler of change in DevOps by focusing on automation and process improvements. Without proper analysis in tool selection, additional friction is added to the workflows of the given entity.
- Scaling - The implementation of DevOps across multiple layers of an organization with the goal of maintaining agility, quality and effective collaboration during periods of growth.

There is a clear need for a well-structured performance measurement system that:

- Captures both delivery outcomes and enabling factors (e.g. collaboration, affinity, tooling and scaling (Davis and Daniels, 2016)),
- Covers both effectiveness and efficiency,
- Supports continuous improvement,
- Dynamically aligns with organizational strategy, and
- Provides actionable insights through well-designed dashboards.

This thesis aims to address this gap by designing a performance measurement system specifically for hybrid Scrum/DevOps teams, using the existing SPACE and DORA frameworks as starting points. Additionally, we are guided by design principles which we discover during the literature review part of the process. This is described in Chapters 5& 6.

A large financial institution provides the research context as the host organization of a research internship. We will take the Process Automation Department, which consists of ten DevOps/Scrum teams organized in Chapters, as a case study and a source of data input and validation. Each employee is part of a Chapter aligned with their core area of expertise, providing role-based guidance and development, while their main responsibilities remain within their assigned Scrum team. We define the most representative employee roles within the case study company as follows:

- Developer - All software engineers who are part of a development team and take part in all artifacts and rituals of the implemented software development strategy.
- Product Owner - All employees who manage the priority of the backlog items for development teams, steer the decisions regarding the scope of work the team will work on, and have direct contact with all relevant clients of the development team.

- IT Lead - All managers who coordinate, regulate and monitor the development teams while ensuring alignment with organizational and technical strategy.

1.4 Research Questions

To address the above problem, this study is guided by the following research questions:

- **Main Research Question** Which key performance indicators (KPIs) from Agile, DevOps, and team health frameworks are most suitable for hybrid Scrum/DevOps environments?
- **SRQ 1.** How can the design of performance dashboards be tailored to the distinct perspectives and responsibilities of developers, product owners, and IT leads?
- **SRQ 2.** What are the main challenges and cultural factors influencing the adoption and effectiveness of hybrid performance measurement systems?
- **SRQ 3.** How can performance measurement systems be designed to enable continuous improvement and stakeholder engagement?
- **SRQ 4.** How can the Performance Measurement System (PMS) be validated in practice to ensure it provides meaningful, actionable insights and avoids common pitfalls such as metric gaming or dashboard fatigue?

These questions will be explored through literature review, framework analysis, and practical design application using a Design Science Research approach (Hevner et al., 2004).

1.5 Structure of the Thesis

In Chapter 2, we discuss the existing challenges in the domain of measuring productivity in the scope of software delivery. The reviewed literature indicates the multidimensionality of productivity itself and how it is defined in different software development strategies. In addition, existing design principles for performance measurement systems are being reviewed. The data collection and data analysis methods used to answer the research questions are then presented in Chapter 3 together with the context of the case study company. Subsequently, the findings from the experiment are reported in Chapter 4. In Chapter 5 we evaluate the previously identified design principles and discuss the intended architectural structure, and features of the proposed framework. Consequently, in Chapter 6 we present the metrics which align with the metric usefulness criteria that was formulated as a result of the literature review and data collection. We map them to the dimensions of the SPACE framework, while embedding the DORA framework within its five dimensions. After finalizing the development of the research prototype, in Chapter 7, we will evaluate its utility, acceptance, and theoretical soundness. Chapter 8 will focus on interpreting the results, the key findings from the research process, the theoretical and practical utility of the designed framework, and any applicable limitations. Lastly, in Chapter 9 all findings from the literature review and the experiment will be combined to answer all stated research questions, together with the identified propositions for future work on the topic.

Chapter 2

Background and Related Work

This chapter begins by exploring the literature around the existing software development methodologies - Scrum and DevOps, and reviews their core values. The topic of existing principles and methods for designing performance measurement frameworks and tools is then reviewed in depth in order to identify state-of-the-art best practices for building performance measurement frameworks.

2.1 Agile and Scrum Methodologies

Agile methodologies have significantly transformed the landscape of software development by addressing the inefficiencies and inelasticity of traditional approaches such as the Waterfall model (Royce, 1987). Agile promotes adaptive planning, early delivery, and continuous improvement, emphasizing flexibility and responsiveness to change (Beck et al., 2001). This paradigm shift was driven by the need to meet rapidly changing business requirements, enhance customer satisfaction, and reduce time-to-market.

Among the various Agile frameworks, *Scrum* has gained widespread popularity due to its structured yet flexible process. Scrum introduces defined roles (Product Owner, Scrum Master, Development Team), time-boxed iterations called sprints, and a set of ceremonies including sprint planning, daily stand-ups, reviews, and retrospectives (Schwaber and Sutherland, 2011). It encourages transparency, inspection, and adaptation, fostering a team-centric approach to delivering incremental value.

The application of Scrum has demonstrated numerous benefits. Examples include stakeholder collaboration, as reported by Verwijs and Russo (2023), while Alami and Krancher (2022) report higher product quality and communication. However, limitations persist, particularly in large-scale and complex software environments. Challenges include integrating non-development roles (such as operations and infrastructure), maintaining sprint cadence in the face of changing priorities, and measuring performance beyond simplistic metrics like story points or sprint velocity (Kuusinen et al., 2018).

These limitations have led to the exploration of complementary methodologies that address areas where Scrum alone may fall short — most notably, in bridging the gap between development and operations.

2.2 DevOps: Principles and Practices

DevOps, a combination of "Development" and "Operations", emerged as a response to the siloed nature of traditional software development organizations (Mezak, 2018). While Agile Scrum successfully improved development processes, DevOps extends the focus to include software deployment, operations, and system administration. It emphasizes automation, continuous delivery, and a culture of shared responsibility across the software lifecycle.

Key principles of DevOps, as presented by Davis and Daniels (2016), include:

- **Automation of the software pipeline:** including building, testing, and deployment.
- **Continuous Integration and Continuous Delivery (CI/CD):** ensuring that changes are integrated and deployed rapidly and reliably.

- **Monitoring and observability:** providing feedback loops from production to development.
- **Collaboration across roles:** breaking down organizational silos between developers, testers, and operations staff.

DevOps has shown significant impact on organizational performance, including faster time-to-market, increased deployment frequency, lower change failure rates, and shorter recovery times from incidents (Leite et al., 2019; Offerman et al., 2022).

A study on the adoption of DevOps by Blinde (2022) reveals that implementation of many of its practices are probable to increase organizational performance. They also discuss that practices primarily focused on automation and continuity – such as automated testing, automated and continuous monitoring and automated deployments – positively impact software delivery performance. Nevertheless, organizations face challenges in the adoption of DevOps, such as cultural resistance, lack of standardized practices, and technical debt.

2.3 Hybrid Scrum/DevOps Teams: Convergence and Complexity

With Agile and DevOps sharing many complementary values, such as iterative improvement, collaboration, and customer focus, many organizations have begun integrating the two into what we will call *hybrid Scrum/DevOps* teams, including the case study for this thesis. These teams combine Scrum structured planning and feedback cycles with DevOps' automation and focus on operational efficiency.

In such hybrid environments, Scrum ceremonies continue to guide product development, while DevOps pipelines automate deployment and monitoring tasks. The combination allows for fast, high-quality, and continuous delivery of software. However, integrating these methodologies is not without friction. Teams must manage both planned sprint cycles and real-time operations work, align responsibilities across evolving roles, and ensure that the tools and processes support both agility and stability.

A study examining the best practices for implementing Agile and DevOps in a complementary manner by Ayyash (2024) points out that Agile principles enhance the speed of software delivery and customer satisfaction, while DevOps contributes to improving the deployment process and infrastructure management, thus creating a sustainable culture. Samarawickrama and Perera (2017) also propose a method of combining Scrum and DevOps (primarily prioritizing Continuous integration and Continuous Delivery) that is called Continuous Scrum.

2.4 Performance Metrics in Agile and DevOps Contexts

Effective performance measurement is crucial in managing software development and operations teams. However, traditional metrics, such as Lines of Code (LoC), Defect Counts, or Burndown rates, often fail to capture the multifaceted nature of modern team performance (Forsgren et al., 2021). As an example, Forsgren et al. (2021) argue that a developer's productivity cannot be measured with only activity indicators.

To address this, several new frameworks have emerged. The most widely adopted in DevOps environments is the DORA metrics framework originally defined by Velasquez et al. (2014), which includes:

- Deployment Frequency - The frequency at which code changes are deployed to production or released to end users, indicating the team's ability to deliver value rapidly.
- Lead Time for Change - The amount of time it takes to get a developed software unit (or a code change) from a *committed* to *deployed to Production* state.
- Change Failure Rate - The percentage of deployments that result in a failure in production—such as incidents, outages, or service degradations—reflecting the quality and stability of releases.
- Time to Restore Service - The average time required to recover from a production failure, often called Mean Time to Recovery (MTTR), indicating system resilience and the efficiency of incident response.

These metrics are empirically linked to software delivery and organizational performance and have become industry standards for benchmarking DevOps performance (Georgsson, 2024).

In parallel, the SPACE framework by Forsgren et al. (2021) provides a broader view of developer productivity. It measures:

- Satisfaction and well-being - The level of work-related fulfillment among developers.
- Performance (both output and outcomes) - The outcome of the delivered work, such as business value, quality and impact.
- Activity - The volume of actions completed, such as commits, tests, reviews, or deployments.
- Communication and collaboration - The quality of information sharing and cooperation between peers.
- Efficiency and flow - The ability of employees to maximize productivity, while minimizing delays, friction, and interruptions.

Unlike traditional frameworks, SPACE recognizes the human and team-centric aspects of software engineering, thus making it particularly useful for assessing hybrid team environments (Forsgren et al., 2021).

2.5 Goal-Oriented Measurement Approaches

To ensure that metrics are actionable and aligned with organizational goals, goal-oriented approaches such as the Goal Question Metric (GQM) framework by Caldiera and Rombach (1994) are frequently applied. There are numerous examples of the application of this method in its original shape or in modified and enhanced forms (Lindstrom, 2004; Basili et al., 2007). GQM advocates a top-down methodology:

1. Define high-level business goals.
2. Formulate questions that assess the achievement of these goals.
3. Identify metrics that provide data to answer the questions.

This structured approach ensures that performance measurement is not arbitrary but purpose-driven. It also aids in avoiding metric overload and supports the integration of both quantitative and qualitative insights.

2.6 Performance Dashboards: Design and Best Practices

Dashboards are increasingly used to visualize key performance indicators and support decision-making at multiple organizational levels. In Agile and DevOps environments, dashboards must strike a balance between simplicity and depth, ensuring that they offer actionable insights without overwhelming users.

Design principles for effective dashboards include (Eckerson, 2010):

- Alignment with strategic and operational goals
- Clear, concise visualizations
- Real-time or near real-time data updates
- User-centric customization and accessibility

Literature suggests that dashboards succeed when they are developed with stakeholder involvement and tailored to specific decision contexts (Bugwandeen and Ungerer, 2019; Brahimi et al., 2019; Boon et al., 2023). Conversely, poor dashboard design – characterized by data overload, irrelevant metrics, or lack of interactivity – can lead to disengagement or misinformed decisions (Bouwers et al., 2013; Ghafoori, 2025).

2.7 Design Principles for Performance Measurement Systems

A growing body of research emphasizes the need for Performance Measurement Systems (PMS) that are not only technically robust, but also strategically aligned and contextually sensitive. De Oliveira and Proença (2019) conducted a systematic literature review to consolidate and formalize design principles specifically for performance measurement in research and development (R&D) environments.

R&D environments, like hybrid software teams, are characterized by high levels of complexity and uncertainty, reliance on intangible assets such as knowledge and motivation, and non-linear workflows involving multiple stakeholders. These factors complicate performance measurement and necessitate adaptable, goal-driven systems (de Oliveira and Proença, 2019). As such, the design principles developed for R&D PMS offer valuable guidance for software engineering contexts.

The review by De Oliveira and Proença (2019) identifies the following guiding principles for designing an effective performance measurement system (PMS):

- **Goal-orientation:** To establish an effective PMS, one should have a goal that is known and measurable (Kerssens-van Drongelen and Bilderbeek, 1999).
- **Use of Evaluation Models:** To use a PMS as a management tool, one should be in possession of an evaluation model in the form of a balanced scoreboard or audit model (Kaplan et al., 2005; Chiesa et al., 1996).
- **Managing Multiple Domains:** To manage performance across multiple dimensions, one should utilize a dashboard-based PMS (Sawhney et al., 2006; Cohn, 2013; Skerlj, 2014).
- **Avoid PMS as a Control Mechanism:** A PMS should not be introduced, perceived or promoted as a controlling element over employees (Chiesa and Frattini, 2009; Saunila et al., 2014).
- **Promoting Positive Behavioral Change** A PMS should provide insights that lead to positive behavioral changes and improvements in existing processes (Benaim, 2015).
- **Time and Cost Indicators:** When aiming to improve development performance, indicators related to time and cost must be selected (Hauser and Zettelmeyer, 1997).
- **Quantifying and Measuring productivity:** When monitoring productivity, one should use internal and in-process KPIs which Schumann Jr et al. (1995) describe as measuring the internal productivity of an organization and the technical improvements around it. On the topic of performance tracking, Schumann Jr et al. (1995) recommends the use of internal and end-of-process KPIs, as they show the change in performance of a single entity across two separate points in time.
- **Syncronising R&D with Strategy:** To align with operational and strategic goals, R&D processes must be synchronized with innovation management objectives (Rothwell, 1994). The link between this principle and this thesis is the need to avoid measuring performance in an isolated setting (Bouwers et al., 2012). By aligning the development units with the business representatives, we can achieve a level of clarity that will elevate the level of understanding behind business strategy of the given technical entity.
- **Balanced KPI orientation:** If performance is viewed from both financial and non-financial perspectives, process inputs and outputs should be viewed via both qualitative and quantitative KPIs (Cavalcante, 2014).

These guiding principles demonstrate strong alignment with existing and well-established software performance measurement frameworks. A vivid example is the GQM (Goal-Question-Metric), which emphasizes traceability from high-level goals to measurable indicators (Berander and Jönsson, 2006). Similarly, the SPACE framework encourages multidimensional and human-centric assessments of developer productivity from both quantitative and qualitative perspective (Forsgren et al., 2021).

In the context of hybrid Scrum/DevOps teams, applying these design principles is essential to address the measurement challenges that arise from integrating planning-oriented Scrum practices with the automation and deployment focus of DevOps. For example, aligning KPIs with higher-level strategic objectives ensures that operational metrics such as those from the DORA framework, discussed by Georgsson (2024), are contextually meaningful and support innovation goals (Rothwell, 1994). Furthermore, utilizing time and cost related indicators for development activities, as proposed by Hauser and Zettelmeyer (1997), together with the suggestions of Schumann Jr et al. (1995) for internal and in-process KPIs guarantees a holistic view of performance that transcends simplified and/or isolated output measures which Bouwers et al. (2012) describe.

In summary, while originating in the R&D domain, the design principles consolidated by de Oliveira and Proença (2019) offer a solid foundation for developing a context-sensitive and goal-aligned performance measurement system tailored to the realities of hybrid software teams. This thesis draws directly on these principles to guide the design and evaluation of its proposed measurement and dashboard framework.

2.8 Design Challenges

Smeds et al. (2015) report that the transition to DevOps is likely to impose workflow difficulties on multiple aspects. Vivid examples are disruption in customer testing habits, developer resistance to picking up new technologies, and technical impediments such as monolithic infrastructure. This suggests that during the design phase, we should focus on designing a system that invites continuous improvement. To do this, we will propose dynamic adjustments of KPI thresholds so that developers and other relevant parties do not feel urged or micromanaged.

Laukkarinen et al. (2018) further addresses the regulatory limitation of measuring delivery using DevOps metrics. The example they provide observes highly regulated organizations where code releases at high frequency can be prohibited due to highly conservative or ambitious Service Level Agreements (SLAs), which further imply low risk appetite. This is a valid point for the case study company in this paper, as it is operating in the financial sector, where risk appetite is low, rapid changes are not accepted, and the Code of Conduct revolves around thoroughly defined procedures.

The literature underscores the growing complexity of software development in hybrid Scrum/DevOps environments and the necessity for integrated, goal-aligned performance measurement. Existing metric frameworks like DORA and SPACE, combined with structured approaches such as GQM and existing design principles, provide a strong foundation. However, practical implementations often fall short due to inadequate customization, poor dashboard design, and lack of contextual awareness.

Chapter 3

Methodology

This chapter will detail the methodology used in this thesis to design and evaluate a tailored performance measurement system and dashboard for hybrid Scrum/DevOps teams, using design science as the guiding research paradigm.

3.1 Research Method

To address the challenges faced in performance measurement within hybrid Scrum/DevOps environments, this study employs a Design Science Research (DSR) methodology (Peffers et al., 2007). This approach is appropriate for the context of this research, as its focus lies in the creation and evaluation of artifacts, that solve organizational problems while using scientific knowledge as a basis. Unlike the documented six-step-procedure in the paper from Peffers et al. (2007), the methodology follows three sequential phases: Problem Identification and Validation through empirical data collection, Solution Design based on problem findings and theoretical frameworks, and Demonstration and Evaluation via stakeholder assessment to ensure practical viability.

3.2 Research Design Overview

This study addresses the core challenges of developing a performance measurement system for hybrid Scrum/DevOps teams through systematic problem exploration, solution design, and evaluation. This approach responds to the complexity of performance measurement in environments where conflicting goals are likely to arise, thus requiring empirical validation prior to proposing any solutions.

3.2.1 Phase 1 - Problem exploration and validation

This phase, combining Problem Identification and Defining Solution objectives from Peffers et al. (2007), employs a systematic problem analysis through triangulated data collection. This is done to avoid solution design based only on subjective findings, such as stakeholder perceptions. These perceptions are collected through a distributed survey and optional interviews. As a mitigation measure, the analysis of existing dashboards provides additional confirmation of all the insights received from the previous two sources.

This shape of the data collection approach is supported by multiple authors. Forsgren et al. (2021) states that only a single data source (e.g. surveys) is likely to provide skewed or subjective insights. Leite et al. (2019) indicates that cross-referencing different data sources ensures consistency in findings. Additionally, Riihimäki (2024) describes how organizations often perceive software delivery success differently from what the operational metrics say.

3.2.2 Phase 2 - Solution Design

In this phase, the findings of the problem exploration are combined with the design principles and existing frameworks identified in the literature review to design a comprehensive performance measurement framework which incorporates role-based perspectives, appropriate metrics and dashboard visualization principles. The design of the final product employs the Goal-Question-Metric (GQM) methodology as a practical tool to ensure that each of the proposed KPIs addresses documented stakeholder needs. Thus, the traceability between the observed dysfunctions and the proposed solution is improved.

3.2.3 Phase 3 - Validation and Evaluation

This phase combines the Demonstration and Evaluation steps from the paper of Peffers et al. (2007) and involves the stakeholder assessment of the proposed solution to ensure its practical validity and theoretical soundness. Unlike the data collection in Phase 1, the evaluation will prioritize the depth and quality of the findings. More specifically, the feedback of the aforementioned stakeholders, being the only data source, will be collected in an iterative manner until improvement points and suggestions diminish, while the theoretical soundness remains untouched.

3.2.4 Survey Design

The survey aims to collect the subjective perception of the Agile/DevOps practitioners within the case study department of ABN AMRO Bank - Process Automation Grid. The design of the survey is strongly influenced by the principles outlined in Effective DevOps by Davis and Daniels (2016), with emphasis mostly on collaboration, autonomy, knowledge-sharing and cultural factors. The goal of the survey is to investigate the work dynamics from multiple angles in order to provide necessary insights that we will use in the second phase of the problem exploration - the interviews. Structurally, there are five sections which focus on the following topics:

- Demographics,
- Levels of affection (Engagement, autonomy, contextual awareness)
- Collaboration levels
- Planning and Continuous Improvement
- Open points of improvement regarding work dynamics, communication with management

All survey questions can be found in Appendix A.

3.2.5 Survey Analysis

The various sections of the survey require different methods of analysis. The first four sections of the survey contain only categorical and ordinal data that will be analyzed using descriptive and inferential statistics to present appropriate frequencies and correlations. The last part of the survey, consisting of three open-ended questions, will be analyzed via categorical coding, thus aiming to capture patterns and new insights that were not taken into consideration during the design of the survey.

3.2.6 Interview Design

While the objective of the survey is to identify possible correlations or causalities between any traits or certain practices, interviews are necessary to understand the reasoning behind the views of the respondents on Agile Scrum and DevOps performance measurement. The interviews are semi-structured, allowing flexibility while ensuring that key topics are covered:

- How do practitioners perceive performance measurement from the Agile SCRUM and DevOps perspective?
- What are the biggest challenges with existing KPIs?
- Have teams experienced incentive-driven performance metric manipulation?
- What KPIs need to be adopted to measure performance more accurately?

Each interview lasts approximately 20-30 minutes and is conducted with professionals at multiple levels of seniority (Developers, Chapter Leads, Product Owners). Due to access restrictions to the network of the case study organization, the interviews are not recorded and instead notes are taken during the conversation with the participant.

In a pilot test, two interviews were conducted in an unstructured format in order to evaluate in what direction the discussion should be steered.

3.2.7 Dashboard Analysis

A critical aspect of this research is validating perception-based survey and interview data with empirical performance metrics. To achieve this, existing Agile and DevOps dashboards are analyzed. This study explores two actively monitored dashboards: Software Delivery Performance Dashboards and the Azure DevOps portal which contains Agile SCRUM indicators such as New/Active/Completed work items, planning accuracy, and sprint burndown charts.

In Chapters 5 & 6 we elaborate more on Phase 2 from the DSR research process. There we make the comparison of survey and interview findings with dashboard KPIs, and evaluate the insights from a governance perspective. We select the most relevant KPIs and integrate them into a framework, adhering to the design principles established in the literature review. Consequently, we develop and present a dashboard prototype that implements all framework elements will be developed and presented.

In Chapter 7 we dive into Phase 3 from the DSR research process. There we describe the structure of the evaluation process and present the transition of the designed dashboard from initial prototype to its finalized version.

Chapter 4

Results

This chapter describes the data analysis performed after the survey distribution and the interviews with employees at the case study company. Furthermore, it looks at the current implementation of performance measurements within the case study company in order to determine the points of improvement that will be used in the next steps of the research.

4.1 Survey Analysis Results

The survey collected responses from 49 participants working across hybrid Scrum/DevOps teams. Eight of the survey questions are measured on a 4-point Likert scale, while six other questions required a Multiple Choice answer to indicate a range or frequency (similar to a Likert scale). The primary statistical method we apply for the survey analysis is **Spearman rank correlation coefficient** because it is suitable for ordinal data. Even the questions which did not ask about quantifiable information were arranged in a way to enable usage of the selected correlation analysis.

4.1.1 Data Transformation and Cleaning

The question *What is the core focus of the chapter you participate in?* ranks the roles of individuals based on their orientation, where the Development is marked as 1, while the Business Analytics – 5. We rank the questions related to *Years of Experience* by assigning a lower rank to less professional experience (e.g. 0-2 years = 1, 3-5 years = 2, etc.). We implement the same approach with the questions which are answered with a certain percentage range(e.g. 0-20% = 1, 21% - 40% = 2, etc.) or relative frequency (e.g. Rarely = 1, Sometimes = 2, Often = 3, etc.). Questions which are answered with a Likert scale preserve their value without further transformation. The question asking about retrospective result, for which participants were able to select more than one answer is transformed in the following manner:

- No changes in team dynamics = 1
- No change in team dynamics + Temporary Changes in team dynamics = 2
- Temporary changes in team dynamics = 3
- Temporary changes in team dynamics + Long-lasting changes in team dynamics = 4
- Long-lasting changes in team dynamics = 5

We transform the question *How does your team treat unfinished user stories during sprint opening/closure?* based on its ethical orientation towards the measurement validity:

- Creating clone user stories in new sprint and closing current ones = 1
- Closing user stories in previous sprint and creating an Action item to finish the implementation = 2
- Taking the unfinished stories to the new sprint = 3

We transform the question *How do you typically approach tasks where your skillset is less applicable or underdeveloped?* based on the proneness of the participant to delegate a task:

- I let a more competent colleague resolve the issue faster = 1
- I complete the task at the expense of extra time = 2
- I work with a more competent colleague to learn and resolve the issue = 3

After cleaning and filtering, 18 survey questions with ordinal answer types were analyzed.

The list of questions is as follows:

1. What is the core focus of the chapter you participate in?
2. How long have you been working in your current role?
3. How long have you been working in the IT industry?
4. What percentage of your total work consists of tasks directly related to your core expertise (your chapter's focus)? (e.g. per week / sprint)
5. When working on tasks within your core expertise, which approach typically applies to you?
6. On scale of 1 to 4, how confident are you when working on tasks outside your core expertise?
7. On scale of 1 to 4, when working, what is your attitude when taking up tasks outside your core expertise?
8. What is your level of engagement in the company vision and strategy?
9. What is the level of your functional understanding of the products that you develop?
10. What is the level of your contextual knowledge regarding Management decisions and targets?
11. What is the level of autonomy - the degree of freedom you have in making decisions and managing your tasks or responsibilities without external control?
12. How do you typically approach tasks where your skillset is less applicable or underdeveloped?
13. How frequently do you collaborate with colleagues from other teams to resolve and issue/task/incident?
14. How frequently do you collaborate with colleagues in your team to resolve an issue/task/incident?
15. On a scale of 1 to 4, how confident are you to speak up during retrospectives?
16. What are the typical results from the retrospectives in your team?
17. What percentage of the planned user stories does the team typically close within a sprint?
18. How does your team treat unfinished user stories during sprint opening/closure?

These questions can be revisited in Appendix A where the answer choices for all questions are displayed.

4.1.2 Descriptive Statistics

While the pool of respondents comprises approximately 30% of the employees within the case study, the distribution of roles is preserved and is considered representative. The ratio between developers, quality assurance engineers, information analysts, and operations representatives remains at 5:1:1:1, which is the actual distribution of chapters within the department. The pie chart for this question is presented in figure 4.1.

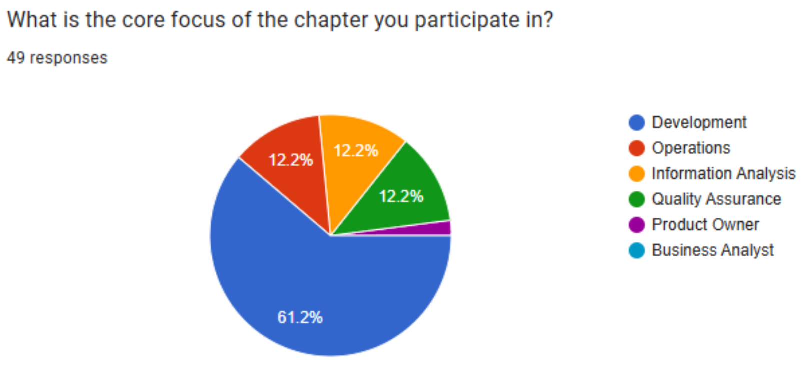
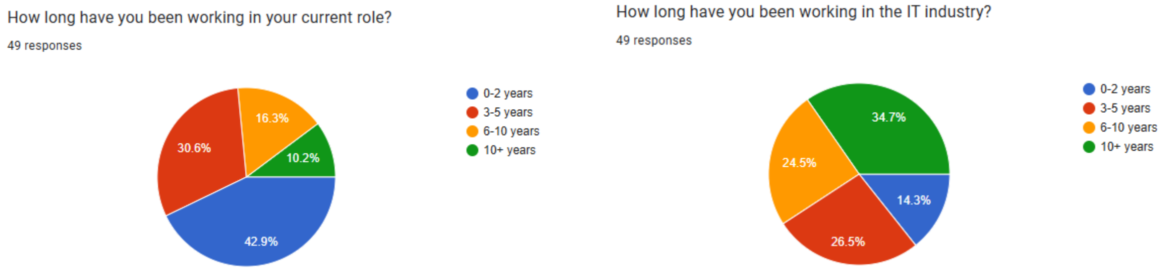


Figure 4.1: Role distribution within pool of respondents.

59% of respondents indicate that they have worked in the IT industry for more than five years. While the junior employees are less than 15%. When we zoom in on their experience within the case study, we see a rapid shift in the distribution. 72% of respondents indicate that they have been working at their current role for less than five years, while the contextually senior employees with experience greater than five years stay at 27%. This ratio does not account for internal role changes. The pie charts representing the distribution of experience within the case study and generally within the IT industry are resented in figure 4.2.



(a) Experience in current role

(b) Experience within IT Industry

Figure 4.2: Two figures side by side.

Many of the respondents indicate a phenomenon very similar to what Bouwers et al. (2012) describes as treating the metrics. In the question asking *How does your team treat unfinished user stories during sprint open/close?*, only 33% of the survey respondents demonstrate proper Scrum implementation in the form of planning the work that is expected to be completed within the sprint boundaries. The remaining 67% show premature marking as complete at sprint's end combined with either replication of user stories or putting a reminder in following sprint but only as an Azure Boards task under an Action point. The pie chart for this question is presented in figure 4.3.

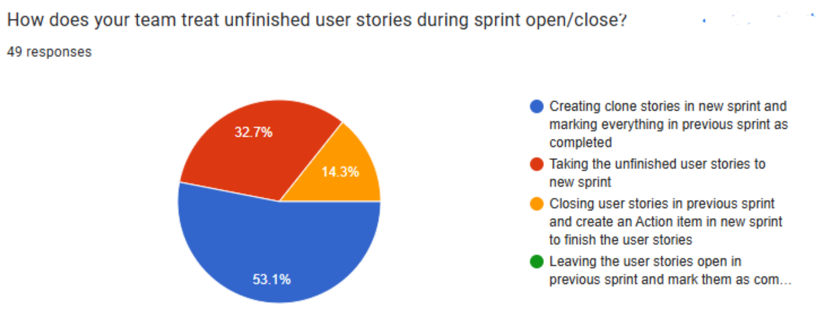


Figure 4.3: Sprint Closure practices within pool of respondents.

As an interesting remark on figure 4.3, the same ratio from the participants in the survey (67%) indicate that they consider Planning Accuracy as an important KPI, thus putting it at first place among nine other metrics. The chart for this can be found in figure 4.4.

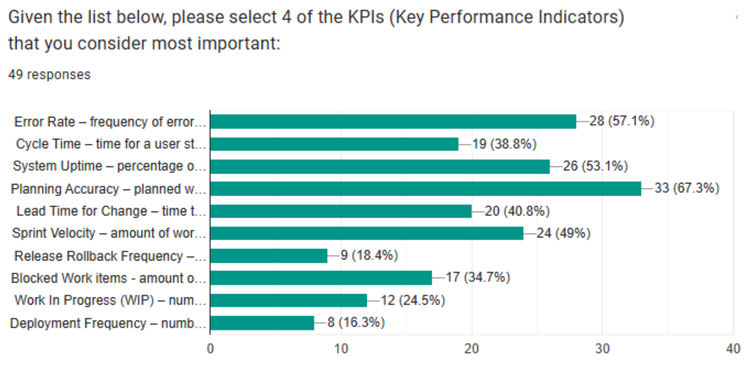


Figure 4.4: Sprint Closure practices within pool of respondents.

4.1.3 Spearman Correlation Analysis

Spearman correlation analysis was performed to uncover relationships between key aspects of team performance, collaboration, confidence, and workflow dynamics. Both the correlation coefficients and their corresponding p-values were computed.

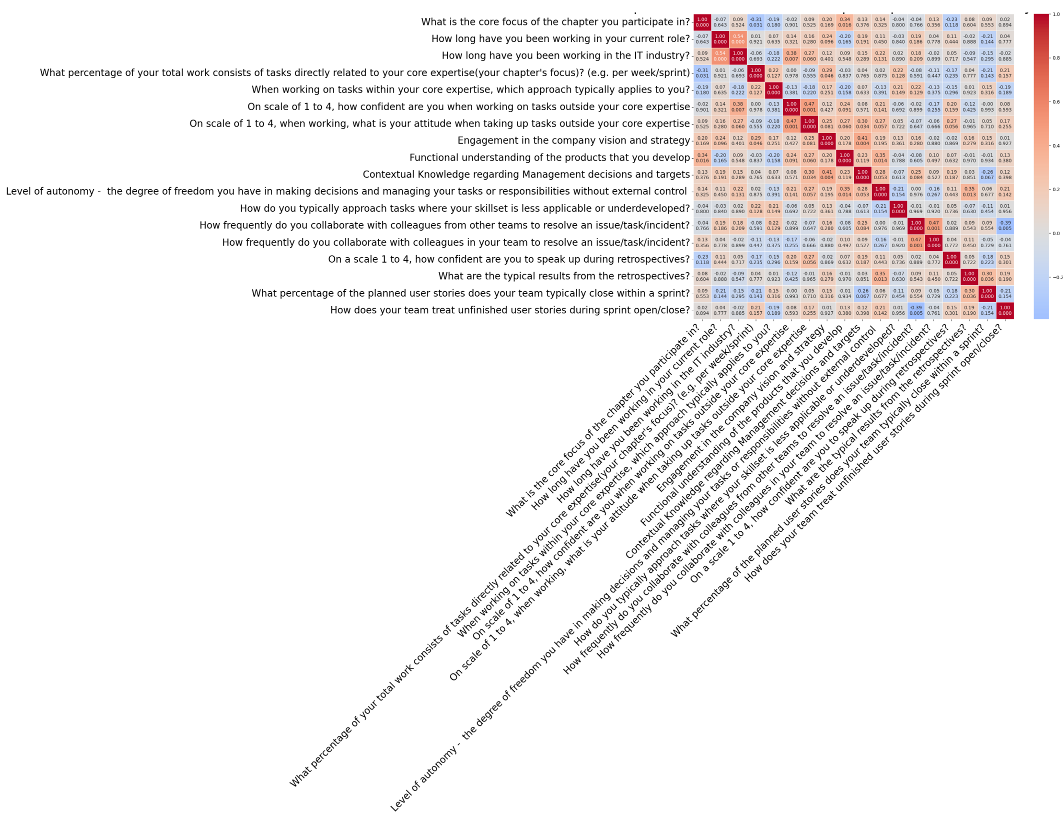


Figure 4.5: Spearman correlation heatmap of survey results.

On *Figure 4.1* each square from the matrix represents the correlation between the pair of questions that lie next to the corresponding row and beneath the respective column. The first number in each question is the correlation coefficient, while the second number represents the p-value for the given correlation. The color schema is determined based on the strength of the correlation. If the correlation is strongly negative, thus approaching the value of **-1**, the color of the matrix element moves to dark blue. While, if the correlation is strongly positive, thus approaching the value of **1**, the color of the matrix element moves to dark red. For non-correlated pairs, the color is light gray, since the correlation index is **0**.

4.1.4 Key Findings

We summarize our interpretation of the correlation results in the following key findings.

1. Practitioners who feel more confident when working on tasks that are not part of their core expertise are likely to approach them with a more positive attitude. This indicates that psychological safety and confidence can promote proactive behavior in cross-functional work.

2. Practitioners with longer professional experience tend to feel more confident while operating outside their main area of expertise. This indicates that more years in the field are likely to build flexibility and adaptability.

3. As practitioners' duties move away from technical software development, their functional understanding of the actual purpose of the resulting software pieces tends to increase. This indicates that promoting cross-functional personas can result in better understanding of the purpose behind the tools that are being built.

4. Practitioners who are highly engaged with the company's vision are slightly more likely to operate on a narrow set of duties. This suggests that the breadth of responsibilities within one's work is negatively correlated with their engagement in the case study company goals.

5. Practitioners who are aware of the reasons why the company is moving in a given direction are more likely to be more intrinsically motivated to take up tasks outside of their main area. This can imply that transparent communication can boost confidence and motivation.

Table 4.1: Top Significant Spearman Correlations (p < 0.05)

No.	Survey Item Pair	Correlation (ρ)	p-value
1.	Confidence working outside core expertise vs. Positive attitude while taking up tasks outside expertise	0.47	< 0.001
2.	Professional Experience vs. Confidence working outside core expertise	0.38	0.007
3.	Core competence (Chapter) vs. Functional understanding	0.34	0.016
4.	Engagement in company vision vs. (lack of) diversity in duties	0.29	0.046
5.	Level of contextual knowledge regarding management's steering decisions vs. Attitude when working outside of core competence	0.30	0.034
6.	Level of contextual knowledge regarding management's steering decisions vs. engagement in company vision	0.41	0.004
7.	Level of autonomy vs. Functional understanding of developed products	0.35	0.014
8.	External collaboration index vs. internal collaboration index	0.47	0.001
9.	Retrospective outcome vs. level of autonomy	0.35	0.013
10.	Retrospective outcome vs. Percentage of successfully closed stories	0.30	0.036
11.	Treating sprint closure vs. External collaboration	-0.39	0.005
12.	(Lack of) diversity in duties vs. core competence	-0.31	0.031

6. Practitioners who are aware of the reasons why the company is moving in a given direction are more likely to feel connected to the overall vision. This implies that transparent communication can strengthen employee engagement.

7. Practitioners who show greater autonomy perception are likely to better understand the purpose behind the products they develop. This implies that individuals who can manage their work without external guidance can explore and comprehend the broader system functionalities.

8. Practitioners who collaborate with the peers from their own team are more likely to collaborate with practitioners from other teams. This implies that collaborative culture is likely to permeate both within and out of the boundaries of a development team.

9. The more persisting the changes from retrospectives, the higher the perceived autonomy among practitioners. This implies that continuous improvement resulting from constructive feedback during reflection moments are likely to empower practitioners to self-manage their work better.

10. Practitioners who report effective retrospectives also close higher percentage of the planned user stories during a sprint. This shows a direct link between continuous improvement and steady performance.

11. Practitioners from teams that manage sprint closure poorly have lower external collaboration scores. This implies that inefficient internal practices may weaken cross-team collaboration.

12. As practitioners' duties move away from technical software development, the diversity in their duties decreases and vice versa. This implies that the developer and Quality Assurance engineer are more cross-functional/multifaceted than Operations or Information analysts.

4.1.5 General Observations from Quantitative Data

These statistically significant correlations reveal relationships between:

- Confidence, intrinsic motivation and professional experience
- Transparency of management decisions and employee engagement
- Collaboration patterns internally and externally
- Effectiveness of retrospectives and sprint success rates.

4.1.6 Thematic Analysis of Open-Ended Survey Responses

To complement the quantitative analysis, the survey included three open-ended questions targeting workflow improvements, management support and management awareness in the DevOps / Scrum practices. The most repeating topics in the responses are listed below per question.

Workflow Dynamics

1. Communication was emphasized as the most important factor for optimal workflow dynamics within or between development units. The most vivid quote from the responses says: *"I would suggest for all teams to communicate better with each other. All teams involved need to know and understand what was agreed, and should also be transparent if things are difficult or not possible at all."*
2. Planning is the second most recurring topic with calls from participants that sprint capacity and software delivery need to be discussed more frequently and in-depth to avoid conflicts. This coincides with one point of the interviews in which the conflict between developers and product owners during velocity planning causes misalignment - with the product owners planning ambitiously, while the developers advocating for a more realistic sprint backlog.
3. Shared ownership is the third theme which advocates the need for an innovative performance measurement system. A clear example for this is the following response: *"Accountability and responsibility within team is crucial. After all it is not only PO who should be having an eye on the target"*.

Support and Guidance from Management

Management involvement appears as a crucial factor for smooth processes in the work environment. This theme appears in more than one form in the responses under this question and due to their similarities, we list out the most appropriate sub-topics:

1. Feedback loops have been mentioned by ten respondents with requests for better following up on pressing matters. Example response quotes: *"Listen to the DevOps teams and actually work on their feedback"*.
2. Eleven of the responses demand an improved role-definition within the case study department. The respondents mention the need to have a dedicated Scrum Master to manage responsibilities of the team around capacity planning, handling of incidents, and other tasks from the Scrum ceremonies which Schwaber and Sutherland (2011) describe.
3. Four of the responses clearly state the need for inclusive decision-making processes that invite all developers and relevant stakeholders to decide democratically on the choice of approach.

Awareness of Management

In this question the respondents largely return back to the feedback they provided in the previous two questions. High emphasis is placed on continuous communication and feedback between managers and engineers which can be achieved by regular meetings, critical feedback loops and active involvement.

4.1.7 Summary

Shortly, in spite of the lack of strong correlation and limited pool of respondents, the survey findings suggest that building a cross-functional mindset, transparent communication, good collaboration and effective reflection are critical drivers for high-performing development teams.

The open-ended questions from the survey further provide inspiration for the solutions that can be implemented to build a more collaborative and efficient environment where duties are clearly communicated, feedback is implemented in follow-up iterations, and all operations are occurring in a transparent manner.

4.2 Interview Analysis Results

During the survey distribution, eight of the respondents voluntarily added their names to the last question of the survey, indicating their willingness to participate in an interview. To contextualize the representation in the interview phase, we present the role distribution in Table 4.2. The duration of all interviews varied between 20 and 30 minutes.

-	Chapter Lead	Engineer
Development	2	3
Operations	1	1
Product Owner	1	

Table 4.2: Roles of participants in interview sessions.

4.2.1 Overview of Coding and Themes

The qualitative analysis of the interviews revealed several key themes organized into broader categories aligned with the research aims. Across all interviews, participants consistently highlighted critical success factors, challenges, cultural shifts, and practical recommendations related to Scrum and DevOps transformations. The coding framework structured findings into main themes and sub-themes, summarized as follows:

- **Agile and DevOps Integration**
  - Importance of automation, continuous integration, and deployment.
  - Strong emphasis on cross-functional collaboration.
  - Recognition that DevOps is a mindset and cultural shift, not merely a tooling exercise.
- **Organizational Culture and Change**
  - Organizational resistance as a major barrier.
  - Need for strong leadership support.
  - Importance of psychological safety and an experimentation culture.
- **Metrics and Performance Measurement**
  - Preference for team-level, flow-based metrics (e.g., Cycle Time, Lead Time) over KPIs like Planned vs Actual Delivery.
  - Challenges with quantifying cultural and collaboration improvements.
- **Challenges and Barriers**
  - Legacy systems and technical debt as major inhibitors.
  - Difficulties in scaling DevOps practices across larger, traditional organizations.
  - Misinterpreting the core values of DevOps which may lead to chasing wrong targets.
- **Success Factors and Best Practices**
  - Incremental adoption strategies (small wins) were seen as more successful than big-bang approaches.
  - Training, mentoring, and continuous learning initiatives strongly supported adoption.
  - Alignment between development and operations goals critical for lasting change.

4.2.2 Detailed Thematic Findings

Agile and DevOps Integration

Participants stressed that Agile and DevOps should not be treated as separate initiatives. Integration of Agile ways of working (iterative development, feedback loops) with DevOps practices (automation, deployment pipelines) was crucial for success. Several participants highlighted the use of "Continuous Scrum" practices to achieve faster delivery and higher product quality.

Representative quote:

> "Without integrating Agile mindsets into DevOps pipelines, you only automate chaos faster."

Organizational Culture and Change

Culture emerged as the dominant enabler or barrier. Successful transformations foster trust, autonomy, and blameless handling of failure. Resistance, fear of losing control, and traditional *command-and-control* structures were commonly cited as impediments.

Representative quote:

> "DevOps won't fix a broken culture. It amplifies what is already there."

Metrics and Performance Measurement

Participants were skeptical of traditional KPIs that mainly favor outcome-oriented measures. The SPACE framework dimensions (Satisfaction, Performance, Activity, Communication, Efficiency) were cited as more aligned with modern Scrum/DevOps environments.

Commonly cited metrics included Deployment Frequency, Lead Time for Change, Mean Time To Recovery (MTTR), and employee satisfaction scores.

Representative quote:

> "If you measure only output, you miss the collaboration and learning that make Scrum and DevOps work."

Challenges and Barriers

Resistance from middle management, unclear vision from leadership, and overemphasis on tools over cultural change were recurring themes. Technical complexity, especially around integrating legacy systems into continuous delivery pipelines, was another major barrier.

Representative quote:

> "We bought the tools, but without mindset change, it just became another expensive project."

Success Factors and Best Practices

Interviewees emphasized the importance of top-down and bottom-up alignment, early wins through pilot teams, continuous learning through communities of practice (CoPs), and the use of visual management tools like dashboards for transparency and engagement.

Representative quote:

> "You can't mandate DevOps from a boardroom. You have to show how it solves real problems for teams."

4.2.3 Summary of Key Insights

- **Culture first, tools second**: Sustainable Agile and DevOps transformations are people-driven.
- **Metrics should reflect flow and collaboration**: Traditional productivity measures are insufficient.
- **Leadership commitment is non-negotiable**: Active, visible support matters.

- **Start small, scale deliberately**: Pilots allow learning and adaptation prior to organization-wide rollouts.
- **DevOps is continuous**: It is not a project with an end date but a way of working.

4.2.4 Summary of Themes and Example Codes

For clarity and for easy reference, we summarize all topics of the interviews together with the example codes for each in Table 4.3.

Theme	**Example Codes**
Agile and DevOps Integration	• Continuous integration and delivery (CI/CD) • Cross-functional collaboration • DevOps as a mindset, not just tooling
Organizational Culture and Change	• Resistance to change • Leadership support and sponsorship • Psychological safety and experimentation
Metrics and Performance Measurement	• Preference for flow-based metrics (cycle time, lead time) • Difficulty measuring collaboration and cultural change • Use of SPACE framework dimensions
Challenges and Barriers	• Legacy systems and technical debt • Misalignment between development and operations • Superficial DevOps adoptions focusing only on tools
Success Factors and Best Practices	• Pilot teams and small wins • Continuous learning and communities of practice • Transparent performance dashboards

Table 4.3: Summary of Themes and Example Codes from Interview Analysis

4.3 Software Delivery Performance Dashboard

Due to privacy regulations and the inevitable sharing of data regarding the actual performance within the case study, the existing dashboard is only textually described and not shared in this thesis. The Software Delivery Performance Dashboard is constructed of three main sections:

4.3.1 Software Delivery Performance

- Cycle Time (Average Days) - This metric visualizes the number of days it took for a user story to go from "In progress" to "Completed" state. In the dashboard it is interpreted as a measure of how fast solutions can be delivered. High values of this metric indicate the existence of bottlenecks or improper segmentation of the work into digestible parts.
- Change Failure Rate - This metric visualizes the percentage of failed software releases to Production. In the dashboard it is interpreted as a measure of quality of delivered software. High values are linked to high technical debt, non-automated release processes, and suboptimal solutions.

- Deployment Frequency - This metric visualizes the number of times code was taken to Production Environment during a given period (month or quarter). Higher values of this metric indicate that the PR environment is expected to be less risk-prone because the changes are deployed in shorter steps and smaller size.
- MTTR (Mean Time To Recover in Hours) - This metric visualizes the average hours it takes to recover from High Priority incidents. IT does not account for incidents of medium or low significance. Higher values indicate that improvements in the way incidents are handles should be implemented.

4.3.2 Operational

- Planned vs. Actual Rate (%) - This metric visualizes the percentage of user stories finished in the sprint they were planned in. In the dashboard it is interpreted as a measure of how accurate the software delivery forecasting can be. High scores indicate efficient planning and resource allocation.
- Availability - Business Service Offerings within Service Level Agreements - This metric visualizes the percentage of components (BSOs) that met their availability commitment. High values mean that software components were available to business users within the predefined limit.

4.3.3 Team Health Survey

Team Health Score - This metric visualizes an average "Health" score per development team. This score is calculated by taking the responses of engineers from a survey that aims to collect their perceptions on aspects such as quality of work, speed of delivery, learning, team work, autonomy. If this indicator is close to 0, this would imply that a development team requires attention. And if it is close to 5, then the team can be considered as happy due to feeling autonomous and fully in control of their dynamics.

4.3.4 Summary

The Software Delivery Performance Dashboard provides a comprehensive overview of key performance indicators across software delivery, operational efficiency, and team health dimensions. It enables visibility into delivery speed, quality, operational stability, and internal team dynamics through a balanced set of metrics such as Cycle Time, Change Failure Rate, Deployment Frequency, MTTR, Planned vs. Actual Rate, Service Availability, and Team Health Score. While the dashboard offers a valuable foundation for linking technical outcomes with operational effectiveness and human factors, it remains relatively basic and limited in its depth of analysis and actionable insights. Consequently, although it marks an important step towards performance transparency, it does not fully leverage modern best practices in DevOps performance management. Therefore, the goal of the case study is to design and propose an enhanced performance management dashboard that provides more comprehensive, targeted, and actionable insights to better support high-performing software development teams.

4.4 Validation of Findings

An interview with the IT Lead of the department at the case study company was conducted to obtain expert validation of the gathered findings from the previous interviews, the survey distribution, and existing dashboard analysis, combined. The insights from this meeting discuss existing gaps and necessary improvements in the existing dashboard set-up and performance framework. All identified subtopics are presented in separate paragraphs.

4.4.1 Interview Insights

Misalignment between metrics and reality

The IT Lead criticizes the commonly used metrics, such as cycle time and deployment frequency, for not being able to reflect actual blockers and impediments. This reflects a gap between what is measured and what is truly important. Furthermore, he points the attention to a "because we can" mindset in selecting KPIs for measurement, thus pointing to a lack of critical thinking in design of existing metrics. Concisely, metrics are often selected due to the availability of data rather than actual value or relevance.

Lack of Participatory Measurement Design

The description that best fits the current norms is "one-way metric imposition". The IT Lead emphasizes that development teams have little influence in defining performance indicators. This lack of participation can not only undermine the relevance of the metrics, but also decreases the sense of ownership and engagement within a team. Another important point on this topic is the level of attention that each developer pays to existing performance dashboards and whether this can influence the speed and quality of software delivery. This is a relevant suggestion for future work on the topic.

Importance of Leadership Presence

The IT Lead links strong leadership with better guidance, that is, increased ability to follow through with impediments that should result in fewer recurring problems and greater accountability. This suggests that metrics alone are not enough to determine the success of a performance measurement system (PMS); rather, it is the ability to provide actionable insights that foster active leadership engagement. The findings presented in the literature review under Design principles align closely with this point.

Preference for simplicity

The interview presents another critique of performance measurement design that supports the need to avoid the phenomenon *'metrics galore'* (Bouwers et al., 2012). The IT Lead stresses that metrics should be few, meaningful, and actionable.

4.4.2 Relevant recommendations for PMS Design

Goal-Oriented Metrics

The interview with the IT Lead supports the design principle that metrics should reflect goals - for the context of this study the goals are of an operational essence (Rothwell, 1994). This further supports the use of Goal - Question - Metric (GQM) approach for PMS design.

Participatory Design

Development teams should be involved in the process of selecting the metrics that are important to measure. This aligns with open feedback input in survey distribution and interviews.

1. "Developers and other technical people should be involved in decision making, since they have the knowledge to determine technical feasibility".
2. "As a manager, I believe the presence of feedback loops is crucial for continuous improvement".

Balanced and Multidimensional View

According to the IT Lead, both qualitative and quantitative metrics that affect different dimensions should be present in a dashboard. This aligns with design principles concerning multidimensionality and essence of KPIs (leading/in-process or lagging/outcome-based) (Cho, 2018; Schumann Jr et al., 1995; Cavalcante, 2014). Forsgren et al. (2021) additionally specify that the SPACE framework is a highly relevant tool for problems of this kind, as it offers a view over the software delivery from the desired panoramic perspective.

Chapter 5

Design of Performance Measurement Framework

This chapter revisits the design principles that we distilled from the Literature Review in Chapter 2 and provides further theoretical motivation behind the design choices that we will follow in the development of the final prototype.

5.1 Motivation

While performance measurement via time-oriented metrics such as Planning Accuracy, Mean Time to Recover (MTTR) or Change Failure Rate can provide insights into the operational efficiency of different development units – individual engineers, development team or even entire departments – their utility remains limited. These metrics fail to account for external variables, thus staying relative and constrained if used alone.

As observed in empirical data from interviews and surveys, the process of aligning the goals of stakeholders from diverse backgrounds is a complex endeavor that is often obscured by unanticipated circumstances. A vivid example is the behavior of Product Owners (POs), who prioritize pursuing the maximization of generated business value through client-centricity and speedy delivery of user stories. This often leads to accumulation of technical debt as speed takes precedence over architectural soundness. As observed in the IT world and in the case study company, high emphasis on client satisfaction can lead to suboptimal solutions that generate bugs and additional load on teams' backlog.

This phenomenon reflects the psychological term **hedonic treadmill** which describes how one quickly adapts to achievements of a given magnitude, only to raise their expectations and continue pursuing new gains with short-term satisfaction (Brickman and Campbell, 1971). In software development, each successful release can generate temporary excitement that will quickly become the new baseline. This leads to continuously increasing demands and expectations that form a perpetual cycle of overcommitment and diminishing returns.

Conversely, developers frequently advocate for investing more in learning, automation, and optimization of existing code bases. Such initiatives, despite failing to demonstrate benefits immediately in terms of business value, aim to mitigate the existing technical debt and ensure long term sustainable value delivery. The divergence in these strategies presents the risks of isolated/siloed goal setting, thus escalating misalignment across the business unit.

To handle this, the most crucial prerequisite is transparent and continuous communication between development teams, product owners and similar managerial representatives. The method of achieving this is to unite the viewpoints of both parties in a centralized location. This concept aligns with the architectural principle of "viewpoints", where different stakeholders interpret a system through lenses relevant to their roles (Kotusev, 2018).

Accordingly, this study proposes a dashboard-based performance measurement system that incorporates key performance indicators (KPIs) valued by different stakeholder groups. Dashboards are not only monitoring tools but also serve as vehicles for shared understanding. As noted by Bugwandeen and Ungerer (2019), "Performance dashboards are an information system tool used to translate the organization's strategy into objectives, metrics, initiatives, and tasks for each group and individual… [They] should provide employees with the right information to optimize decisions, enhance efficiency, and increase profits" (p. 162). In this sense, the dashboard becomes a communication interface that reflects the current state of the organization across multiple dimensions.

To contextualize this approach, we borrow from the "fog of war" metaphor commonly used in strategic studies. In this analogy, development teams and business leaders operate with limited visibility into each other's domains. Developers may be unaware of how their outputs influence business outcomes, while business leaders may not grasp the implications of architectural decisions. The proposed dashboard enables proactive signaling—allowing any stakeholder to raise alerts when pre-defined service-level agreements (SLAs) are at risk. This mechanism supports ad hoc, issue-driven discussions, replacing rigid, schedule-based meetings with more efficient and contextually relevant engagement.

A practical illustration comes from the Wayland project, where stakeholder conflicts arose around what constituted core versus extension-based specifications. To manage this, the governance team developed a protocol process requiring multiple approvals and cross-project alignment, essentially functioning as a governance-oriented dashboard to monitor the frequency and intensity of disagreement. When consensus was difficult to reach, governance mechanisms enabled rotating or replacing representatives to foster progress (Wayland Project, 2024).

Ultimately, the issue is not technological, but communicative. Dashboards act as shared reference points for the current state of complex systems. Referring back to the paper of Bugwandeen and Ungerer (2019), effective dashboards enable organizations to align strategy with execution by making relevant information visible and actionable to all stakeholders. This reinforces cross-functional communication and minimizes the risk of isolated decision making.

This paradigm also supports the premise of Conway's law, which holds that the design of a software system reflects the communication structure of the organization that built it (Conway, 1968). In environments such as Microsoft, where different teams (e.g., operating system vs. graphics subsystem developers) must collaborate to deliver integrated solutions, communication becomes as critical as technical proficiency. Thus, a well-designed dashboard becomes a foundational element in enabling such alignment.

5.2 Design Principles & Guidelines

5.2.1 General Principles

An effective Performance Measurement System (PMS) serves not only as a monitoring tool but also as an enabler of continuous improvement, learning, and strategic alignment. This study uses the systematic review by De Oliveira and Proença (2019) as a conceptual foundation, enriching it with principles from established performance management literature to guide the design of a PMS suitable for hybrid Scrum/DevOps teams.

A key foundation of PMS design is the presence of a clearly defined and measurable goal (Kerssens-van Drongelen and Bilderbeek, 1999). Without such a goal, metrics risk becoming arbitrary or disconnected from team objectives. The PMS must also include a structured evaluation model such as a balanced scorecard or audit-based approach to guide strategic interpretation of performance indicators (Kaplan et al., 2005; Chiesa et al., 1996).

Given the multidimensional nature of software development environments, especially in Agile and DevOps contexts, a dashboard-based PMS is recommended to provide real-time insights across the delivery, collaboration, and value dimensions (Sawhney et al., 2006; Cohn, 2013; Skerlj, 2014). However, it is essential that such a system is not used as a control mechanism. A PMS should be introduced in a way that avoids perceptions of surveillance or micromanagement, as this can harm motivation and autonomy (Chiesa and Frattini, 2009; Saunila et al., 2014).

Furthermore, Benaim (2015) states that the system must be designed to generate insights that lead to positive behavioral change and process improvements, especially in complex environments where a multidimensional perspective is crucial (Cho, 2018). To improve performance, time and cost related indicators should be included, but balanced with in-process and internal KPIs to monitor internal operations productivity (Hauser and Zettelmeyer, 1997; Schumann Jr et al., 1995).

Finally, aligning development work with larger organizational objectives requires synchronizing performance indicators with innovation management goals (Rothwell, 1994). This alignment should include both financial and non-financial measures, integrating qualitative and quantitative KPIs to support holistic evaluation (Cavalcante, 2014).

This framework is further strengthened by incorporating the Goal-Question-Metric (GQM) approach, which structures measurement activities around specific goals, derived questions, and targeted metrics. GQM ensures that every measurement has traceable value and direct alignment with both team-level and organizational objectives.

5.2.2 Dashboard Design Principles

The visual and functional design of the PMS dashboard plays a crucial role in determining how effectively it is used. Dashboards must not only present data, but do so in a way that supports understanding, engagement, and timely action.

One fundamental design principle is minimalism. Each role-specific dashboard should include only between 4 to 6 core metrics, clearly labeled and interpreted. This ensures focus and reduces cognitive load, in line with Benaim (2015)'s recommendation to prioritize behaviorally relevant information.

Metrics should be contextualized with historical trends and annotated thresholds. Time-based charts, such as line graphs or sparklines, help users detect positive or negative movement over time. Highlighting deviations or targets enables quick situational awareness and informed discussions.

Custom views tailored by role further increase usability. Developers, Product Owners, Team Leads, and Executives should each see data most relevant to their scope of influence, while maintaining access to broader information when needed. This preserves both specificity and transparency.

To avoid purely quantitative interpretation, dashboards should support brief qualitative inputs. For instance, teams may annotate a spike in cycle time due to cross-team dependencies or urgent technical debt. These contextual notes can be valuable in retrospectives or root cause analyses.

Finally, feedback mechanisms must be embedded in the dashboard. Users should be able to flag metrics as unclear, outdated, or misaligned with current team goals. This participatory feature aligns with agile principles and supports continuous system refinement.

5.2.3 Roles and Custom Views

Performance measurement becomes more meaningful when metrics are tailored to the needs and responsibilities of different roles within the organization. A one-size-fits-all approach tends to obscure relevant insights and reduce engagement. To address this, the PMS framework provides custom dashboard views for different roles, each aligned with their respective goals, questions, and decision-making needs.

For developers, the focus is primarily on delivery efficiency and quality. Metrics such as Cycle Time, Change Failure Rate, Deployment Frequency, and Team Health Score help engineers assess their workflow performance and stability. These indicators enable real-time reflection and foster a culture of technical accountability.

Product Owners are more concerned with value delivery and customer alignment. For them, key indicators include Sprint Goal Achievement Rate, Perceived Business Value, and Customer Satisfaction. These metrics offer insight into whether the product roadmap and development output meet business priorities.

Team leads and Scrum Masters rely on collaboration and organizational health metrics. The Cross-Team Collaboration Index and the ability to have feedback loops are especially useful in identifying friction points and guiding team development. Not only seeing the metrics in the dashboard, but being able to direct the team's attention to the existing issue, and facilitate a Root Cause Analysis (RCA) discussion, can elevate the PMS effectiveness for this stakeholder group. MTTR is also relevant, particularly when dealing with unplanned work or operational disruptions.

Engineering and delivery managers require a broader, aggregated view to monitor systemic patterns across multiple teams. Metrics such as Deployment Frequency, Change Failure Rate, and Health-of-Team-Dynamics Index (aggregated) help them identify trends and prioritize coaching or support.

At the strategic level, executive leaders are focused on alignment with company goals and overall engineering health. Strategic Alignment Scores, Innovation Throughput, and macro trends drawn from lower-level dashboards support strategic planning, resource allocation, and organizational change initiatives.

This role-based segmentation ensures that each stakeholder receives relevant and actionable insights, while still drawing from a shared measurement foundation. It also supports the GQM principle of mapping different questions to different actors in the system.

5.2.4 Metric Selection Principles

The selection of performance metrics must follow a structured and thoughtful process to ensure alignment with organizational goals, team context, and decision-making relevance. First and foremost, metrics must be goal-driven and relevant. According to Kerssens-van Drongelen and Bilderbeek (1999), performance indicators must be derived from clear, measurable objectives to avoid the pitfall of tracking data simply because it is available. The GQM approach operationalizes this by requiring that every metric supports a specific question, which in turn links directly to a predefined goal.

Equally important is the need for multi-dimensional and balanced measurement. In complex environments, a narrow set of indicators may fail to capture the full spectrum of team and process performance. As noted by Kaplan et al. (2005) and Cho (2018), a comprehensive PMS should include both leading and lagging indicators across process, people, and outcome domains.

Simplicity and interpretability are also essential. In line with Benaim (2015), only a small set of high-impact metrics should be used for each role or view. This avoids dashboard fatigue and improves the clarity of the system. Metrics must also be presented in a way that team members can readily understand and act upon.

The principle of participatory and adaptive design addresses the need to involve teams in the creation and ongoing refinement of their performance indicators. Drawing from Chiesa and Frattini (2009), metrics that are imposed without input risk undermining psychological safety and relevance. Instead, team-level engagement in retrospectives and feedback loops ensures that measurement systems remain dynamic and context-sensitive.

Finally, a well-designed PMS integrates both quantitative and qualitative inputs. As Cavalcante (2014) emphasizes, combining objective performance data with subjective insights allows for a more nuanced understanding of progress, challenges, and team dynamics.

5.2.5 Governance and Feedback Loop

To remain effective, a performance measurement system must evolve alongside the teams and environments it supports. This requires a formalized governance structure that balances consistency with adaptability, and strategic oversight with team-level input.

At the team level, regular metric retrospectives are held—ideally every month or per development cycle. In these sessions, team members discuss which metrics are still valuable, which are not, and whether new metrics are needed. This aligns with the GQM principle of iteratively refining the link between goals, questions, and metrics.

At the leadership level, quarterly reviews are conducted to identify trends across teams. These reviews consider aggregate indicators such as Deployment Frequency or Team Health, and may highlight cross-team challenges or opportunities. Importantly, leadership also considers feedback gathered from teams to understand how metrics are being used, perceived, or resisted (Kaplan et al., 2005).

Supplementing these cycles, lightweight pulse surveys are administered every few sprints. These brief questionnaires assess autonomy, perceived usefulness of current metrics, and unmeasured pain points. Their results can guide both local and systemic improvements.

Metric evolution is governed by a lightweight protocol. Proposed changes are piloted in one team, observed for clarity and utility, and then scaled where appropriate. Change history is tracked and made visible to reinforce transparency and trust.

Together, these governance mechanisms ensure that the PMS remains aligned, credible, and adaptable—reflecting both empirical insights and lived experiences.

5.3 Tooling

The success of a PMS is not determined solely by its conceptual design; it also depends on its technical implementation and organizational adoption. Poor tooling or misapplication can undermine even the most thoughtful frameworks.

Tooling should prioritize integration and ease of access. Dashboards must be embedded within platforms that teams already use, such as Jira, Confluence, or Azure. Data pipelines should be automated to pull from version control systems, CI/CD tools, and survey platforms. This reduces manual workload and ensures that metrics remain current and credible.

Role-based dashboards can be created using flexible platforms such as Power BI. This tool allows filtered views, historical trend visualization, and interactive feedback components. Importantly, access permissions should encourage transparency while still preserving relevance.

5.4 Anti-Patterns

Several implementation anti-patterns must be explicitly avoided. First, metrics should never be used to evaluate individual performance. Doing so erodes trust and encourages gaming behavior (Chiesa and Frattini, 2009; Saunila et al., 2014). Second, including too many metrics can overwhelm users, causing dashboards to be ignored entirely (Benaim, 2015). Third, applying the same set of metrics to every team ignores the diversity of contexts and reduces engagement.

Another risk is failing to include qualitative interpretation. Without space for explanation, metrics may be misinterpreted or viewed in isolation (Cavalcante, 2014). Finally, treating the PMS as a fixed entity ignores the dynamic nature of Agile and DevOps environments. Regular review, feedback, and iterative adjustment are essential to keeping the system relevant.

By combining thoughtful tooling with awareness of these anti-patterns, the PMS can maintain both technical robustness and user trust.

The proposed PMS aims to align strategy, collaboration, and delivery within hybrid Agile/DevOps settings. By anchoring the framework in both practitioner insight and academic guidance, and structuring it through the GQM lens, the design supports learning, transparency, and measurable improvement across all levels of the organization.

Chapter 6

Development of Performance Measurement Framework

In this chapter we present the technical realization of the proposed Performance Measurement System. We do this by describing the tools we use for the purpose, as well as the actual building blocks of the PMS, the metrics. We map each metric to its corresponding dimension from the SPACE framework and apply the Goal-Question-Metric approach to motivate our choice.

6.1 Tooling & Architecture

The software selected to build the dashboard prototype is Microsoft Power BI. This platform was selected because the dashboard described in Chapter 4 has been built using it and the infrastructure is already existing. The sources from which the existing software delivery performance dashboard takes the data used to visualize all of the metrics currently in use are Microsoft Azure DevOps and ServiceNow. In the prototype that we develop we also use a live connection to Azure DevOps. Due to licensing limitations, the connection to ServiceNow is limited in the prototype to an excel sheet which contains a number of items with attributes closely mimicking a real-life scenario.

The prototype does not connect to any administrative unit of the case study company due to existing data protection policies. Instead, a sample synthetic organization was created in Azure DevOps under student licensing from Leiden University for the Microsoft suite of tools. The organization in question is comprised of two teams, whose backlogs run for 12 sprints in the period 1st January, 2025 until 6th July, 2025. Each of the teams operates a backlog of user stories which have been generated in a CSV file and imported into the Azure DevOps.

6.2 Metrics & Views

The selection of metrics to be displayed in the dashboard prototype have been heavily/solely derived from the feedback that was collected during the problem exploration phase in the surveys and interviews. Having the DORA metrics that are being utilized in the existing dashboard, the participants in the interviews additionally indicated that employee satisfaction is an important measure for Root Cause Analysis in impediment situations. Additionally, referring to the final interview from the problem exploration which was done with the IT Lead of the case study department, the selection process aims to avoid easy-to-implement metrics if there was no grounds for using them as a (cor)-relating factor for other important metrics.

It is also important to mention that many of the metrics enumerated below have been custom defined in Azure DevOps, thus demonstrating that the final prototype is highly flexible and adaptable to all organizations that are administering their backlog alike. Since the initial version of the prototype differs from the one at the end of the evaluation phase, the list of metrics will not be presented in a role-based fashion but according to the SPACE framework dimension they fall into. Additionally, the Goals and Questions that have led to the choice, after applying the Goal-Question-Metric Approach, are presented.

6.2.1 Satisfaction and Well-being

1. Average Developer Satisfaction
    - Goal: Promote team morale, retention, and productivity by understanding and responding to developer needs.
    - Question: Are you happy with the user stories we work on?
    - Metric: Average Developer Satisfaction - A custom metric which measures how satisfied a developer was while working on a user story. High values indicate that description, requirements and peer-review cycle are smooth, while lower scores can serve as discussion facilitators to improve existing impediments.
2. Value-driven Orientation
    - Goal: Maintain focus on delivering customer or business-relevant features over internal-only work.
    - Question: What percentage of our sprint backlog is driven by business value vs. code optimization?
    - Metric: Value-driven orientation - Indicates how much effort is put in optimizing existing software components vs. amount of effort out in working on direct business requirements.

6.2.2 Performance

1. Lead Time for Change
    - Goal: Measure how quickly changes move from code commit to deployment, driving faster delivery and shorter feedback loops.
    - Question: How long does it take to implement and deploy software to production?
    - Metric: Lead Time for Change - Part of the DORA metrics, heavily embraced by employees within the case study company. Measures the delivery speed and responsiveness of development units.
2. Lead Time
    - Goal: Measure the total time it takes for a work item to go from creation to completion, helping teams reduce delays and improve delivery predictability.
    - Question: In how many days does a user story go from Refined to Completed?
    - Metric: Lead Time - Direct indicator of the backlog status and timely planning. Indicates how much time it takes to get a user story from refined to completed state.
3. Epic Completion Progress
    - Goal: Track large product delivery to ensure strategic alignment and timely progress.
    - Question: What percentage of the work on a selected project has been delivered?
    - Metric: Epic Completion Progress - Tracks the delivery progress of high-level business goals, such as separate software projects.
4. Cumulative Business Value Delivered
    - Goal: Quantify business impact over time to align work with strategic goals and outcomes.
    - Question: How much business value has the selected product generated?
    - Metric: Cumulative Business Value Delivered - Outcome-oriented measure of the value that has been shipped over time. Avoids the context of introducing technical employees into financial value of software units and puts a measure similar or identical to the development effort of different work items.

6.2.3 Activity

1. Technical Debt Contribution Ratio
   - Goal: Ensure that technical debt is actively managed without overwhelming the sprint, thus maintaining long-term system health.
   - Question: What proportion of our capacity is dedicated to optimizing / fixing existing software?
   - Metric: Technical Debt Contribution Ratio - Indicates how much of the sprint capacity is spent addressing quality of existing software components
2. User Story Cycle Time
   - Goal: Improve development speed and predictability by identifying delays in the workflow.
   - Question: Are we completing user stories in a timely manner?
   - Metric: User story Cycle Time - Measures the duration of user stories from the moment development work has been initiated until the moment they are marked as complete (Reviewed and Tested).
3. User Story Cycle Time based on Effort Points
   - Goal: Assess whether large/more complex user stories are disproportionately slower, supporting better estimation and prioritization.
   - Question: How does delivery speed change in terms of user story scope?
   - Metric: User story Cycle Time based on Effort points - Presents a more nuanced view of the correlation between workload and speed of delivery. Can be used as an indicator to demonstrate how smaller user stories are completed in exponentially less time than larger ones.
4. Planning Accuracy
   - Goal: Increase sprint commitment reliability by aligning planned work with what is actually delivered, supporting better forecasting and stakeholder trust.
   - Question: How well can we plan our delivery capacity?
   - Metric: Planning Accuracy - Compares planned vs. completed work ration to reflect the demand forecasting and reliability of development units.

6.2.4 Communication and Collaboration

No metrics that directly address this dimension of the SPACE framework have been added. Following Conway (1968), the state of communication and collaboration can be derived from any of the other dimensions, as it is likely to be a root cause for other operational problems. To mitigate this, instead, we develop an alert mechanism that can be triggered by any user of the final product.

1. None included. Instead, an alert mechanism is implemented as described in Section 6.3.

6.2.5 Efficiency and Flow

1. Test Finding Distribution per Environment
   - Goal: Identify at which stage issues are being caught to strengthen initial delivery quality.
   - Question: What is the quality of our review/test cycles illustrated by test findings along the OTAP?
   - Metric: Test Finding Distribution per Environment - Indicates where test findings are detected and reflects the quality and efficiency of the feedback loops within the development unit.
2. Number of Incidents per Team

- Goal: Pinpoint stability issues or process gaps in specific teams to target better support and improvement.
- Question: Which teams are facing more incidents?
- Metric: Number of Incidents per Team - Show the impact of inefficiencies or (in)stability per development unit. No grouping of incidents has been implemented, hence each incident is treated as a unique case.

3. Number of Incidents per Day
    - Goal: Track system stability and trends over time to proactively address quality fluctuations.
    - Question: Are there peaks in incident volumes on specific dates / periods?
    - Metric: Number of Incidents per Day - Measures flow stability and defect trends over time.
4. Lead Time based on Business Value of User Story
    - Goal:Ensure high-value items are flowing quickly through the delivery process to maximize ROI.
    - Question: In how many days do we complete a user story put int order of its business value?
    - Metric: Lead Time based on Business Value of User Story - Illustrates how effectively high-value items move through the delivery cycle.
5. Closed Incidents within SLA
    - Goal: Ensure service reliability and responsiveness by resolving incidents within agreed time limits, improving user satisfaction and operational maturity.
    - Question: How often do we solve an issue within the timeframe as specified in the SLA?
    - Metric: Closed incidents before deadline - Measures the responsiveness of teams with regards to predefined SLA.

6.3 Integration with Power Automate

In addition to displaying the metrics and serving only as a visualizing method, the prototype dashboard facilitates an integration point with Microsoft Power Automate. Since many of the metrics provide a target to which development teams should aim, one of the functionalities of the dashboard is an Alert mechanism, which sends a notification to the employees from the corresponding development unit with a snapshot of the dashboard, requesting a meeting moment to discuss how improvements can be implemented to rectify the problem. The motivation for this functionality was discussed in chapter 5 where one of the goals is to develop a solution that will allow stakeholders to raise awareness about inefficiencies that are directly influenced by the operational activities of some of the other parties.

Thus, the prototype dashboard transitions its utility from a merely informative tool, which presents meaningful numbers about the dynamics of a development unit, into a discussion facilitator for the enhancements that can be implemented.

Since the initial design differs significantly from the final version, all versions of the developed prototype can be located in Appendix B.

Chapter 7

Evaluation

This chapter presents the evaluation of the performance measurement system for hybrid Scrum/DevOps environments implemented as a prototype dashboard. Following the DSR methodology from chapter 3, this part of the research process focuses on the validation of the practical utility and acceptance, and the theoretical soundness of the proposed artifact. More specifically, it aims to address the dashboard's ability to solve the existing measurement challenges by combining the viewpoints of Developers, Product Owners and IT Governance Leads.

7.1 Evaluation Approach

The evaluation process used an iterative stakeholder-centered approach. Instead of conducting a single assessment, the evaluation evolved through three distinct iterations, each incorporating stakeholder feedback to refine the usability, comprehensiveness, and relevance of the dashboard. This iterative methodology aligns with the continuous improvement philosophy which we can find in both Agile and DevOps practices, thus ensuring that the final product reflects authentic user needs rather than purely theoretical assumptions.

This approach was chosen because the prototype dashboard represents a proof-of-concept implementation with synthetic data instead of direct integration into the workflow dynamics of the case study company. By assessing factors such as usability (how effectively stakeholders can navigate and understand the dashboard and the information it provides), comprehensiveness (the coverage of stakeholder information requirements), and relevance (whether the selected metrics address genuine decision-making needs), we get the answer to SRQ. 1.4 and SRQ. 1.4.

7.2 Participant Selection

For the evaluation process, participants were selected based on their job position within the case study company: Developers, Product Owners, Chapter Leads, Business Analysts, and IT Leads. Having introduced the research topic to the employees in the case study department, the evaluation aimed to collect feedback from individuals who had not heard about the development of the prototype as well. Hence, the pool of participants included a mix of first-time witnesses and familiar participants who participated in the problem exploration phase. With this approach, our aim was to ensure that the feedback collected is not affected by bias in any of the participant groups' feedback. The common trait between all participants was their involvement in a hybrid Agile/DevOps environment to ensure familiarity with all relevant terminology.

Each iteration involved between 5-6 participants, providing sufficient diversity of perspectives while maintaining manageable feedback volume for a sustainable iterative refinement. The group composition also varied across the iterations to ensure that there is no stakeholder group whose feedback becomes a majority.

7.3 Data Collection

The evaluation moments combined a structured demonstration of the dashboard prototype with an introduction to the research problem together with semi-structured interviews focusing on metric relevance, visualization effectiveness, and implementation barriers. Open-ended questioning allowed participants to provide their genuine concerns and suggestions that were not anticipated during the design and development phase.

Participants were invited to think out loud about the following questions:

- Does the presented viewpoint on the dashboard provide information that is sufficient for you as a developer/PO/Lead to monitor the performance of your entity?
- Seeing the other viewpoints, do you see metrics that are relevant for your contextual awareness as a developer/PO/Lead?
- Would you prefer to have a single viewpoint that is tailored to your role, or would you like to have the opportunity to observe all performance indicators?
- What can be improved about any of the metrics you are looking at now?
- Do you identify value in the presented set of metrics on the dashboard?
- What challenges do you foresee in the adoption of this dashboard as a tool to steer performance of Scrum/DevOps entities?

We took notes actively during the discussions and did not record the sessions to minimize disturbance of the participants. In the end of each discussion, we repeated the notes we had taken and asked the participants if anything was omitted.

7.4 Evaluation Implementation & Iterations

In this section, each iteration will be presented separately and incremental improvements will be displayed to enhance the traceability of the iterative cycle.

7.4.1 Iteration 1: Foundation Validation

The initial evaluation involved five participants: two developers, one chapter lead, one business analyst, and one product owner. This iteration focused on validating the initial design that implemented tailored viewpoints based on the job position of the user, and identifying usability issues that required structural changes.

Key Findings & Implemented Changes:
Participants confirmed that splitting the KPIs into separate viewpoints was useful, as it reduced the cognitive load. However, the limitation of this was the matchmaking with job descriptions. In the case study company, the majority of the employees indicated that they perform duties from more than one background (e.g. development, operations, business analysis, scrum master role, chapter leads), hence limiting their view to a specific set of approximately five metrics did not bring significant value.

Additionally, suggestions for reshuffling the metrics across the different viewpoints were given, such as incident list to be moved from IT Lead's perspective into the developer's perspective. Developer satisfaction index was moved in the opposite way into the IT Lead's viewpoint as an appropriate metric to track the satisfaction of developers within separate entities.

Furthermore, participants criticized the design of certain data visualizations such as Lead Time, Lead Time for Change, and Planning Accuracy. The initial design as a card with a single number did not turn the data into valuable information. Instead, we changed the design format to a gauge visual which utilized a target value, a value range with minimum and maximum bounds, a color change of the value based on the distance from the target, a title of the metric and a subtitle with the question to which the metric answers based on the GQM framework.

Finally, the viewpoints were changed from job-oriented to topic-oriented. The developer's perspective transformed into a technical viewpoint, the product owner's perspective into a business-oriented viewpoint, and the IT Lead's perspective into a governance viewpoint.

7.4.2 Iteration 2: Comprehensive Context

The second evaluation iteration involved six participants who were different from the ones in the previous cycle: three chapter leads (who manage Information Analysis and Operations), two developers, and one product owner. This iteration assessed whether the structural improvements implemented after the first iteration had properly addressed contextual concerns while identifying any remaining comprehensive gaps.

Key Findings & Implemented Changes:
In this round, participants confirmed the change from job-based to topic-oriented viewpoints addresses contextual limitations better. The updated visualizations were perceived as more comprehensible and the metric selection better aligned with the corresponding viewpoint at which it was placed. However, a new limitation was mentioned by the two chapter leads and the product owner: there was no option to observe the historical change of the metrics. The ability to see if there are any trends that can be further matched to fluctuations in other metrics was deemed as crucial to understand performance patterns over time.

To cater for the addition of the needed temporal context, the following changes were implemented. Firstly, an indicator – which is called a ticker in the PowerBI toolbox – was added to the appropriate visualizations to accommodate displaying the change in value on a metric compared to the last finished sprint cycle. Additionally, a button was added to each of the viewpoints that leads to a new page on the dashboard where the user can look at the relevant historical performance of the team over the last sprints (defined by the user). These additions transformed the dashboard from a current-state monitoring tool into a platform which supports both operational awareness and strategic analysis.

7.4.3 Iteration 3: Refinement and Finalization

The final iteration from the evaluation process involved five participants: three developers, one IT Lead, and one product owner. This iteration focused on final refinement and validation that the dashboard achieves stakeholder acceptance without major structural limitations.

Key Findings & Implemented Changes:
The participants showed satisfaction with the further refined version of the dashboard and appreciated the choice of visualization for all metrics, the historical trend functionality. The IT Lead and the product owner both commented that the ability to reach out to development teams when some of the metrics are going out of the bounds of existing Service Level Agreements (SLAs) brings additional business value. The points for improvement that were introduced in this iteration concerned the addition of measurement units to the metrics which did not appear as a percentage ratio. The other finding which came out of this iteration concerned the metric 'Value-driven orientation', where the legend on the diagram did not present a sufficiently clear description.

Final Implementation:
Shortly after completing all interviews from the third iteration cycle, we implemented the addition of measurement units to all relevant metrics and optimized the legends of visualizations, thus ensuring their unambiguous interpretation. These changes completed the dashboard refinement process, as no further functional or structural concerns were revealed.

7.5 Validation Against Design Requirements

The evaluation results demonstrate that the dashboard addresses the core problems identified in Chapter 4. The iterative feedback process confirmed that participants from all relevant stakeholder groups consider the metrics as relevant to their operational needs, thus addressing the problem of measurement misalignment between the different organizational roles. The integrated viewpoint design additionally mitigates the concern that performance dashboards create a "silo goal setting" by allowing stakeholders to measure their performance in relation to a diverse set of organizational goals.

During the evaluation we validate that the framework successfully cover the four DevOps pillars from Davis (2015) which were introduced in Chapter 1. The cross-functional viewpoints address the *Collaboration & Affinity* pillars as they enable shared understanding between Developers, Product Owners, and IT Leads. We demonstrate the *Tooling* via the connection to Azure DevOps and the integration of Power Automate mechanisms which participants found valuable for proactive issue management. Finally, we can demonstrate the *Scaling* capabilities of the PMS through the topic-oriented design (instead of role-specific), thus being able to adapt to role evolution or organizational changes.

The addition of historical trends, which was missed in the initial proof-of-concept design of the dashboard, further tackles the problem of measurement dysfunction, where teams focus solely on short-term metric optimizations rather than slow but sustainable improvements. Thus, supporting one of the significant correlations from the survey distribution, where thorough retrospectives link to better planning accuracy, teams can self-manage their historical performance and plan their capacity better.

7.6 Validation Against Design Principles

The evaluation process utilizes the participatory design principle mentioned in Chapter 5. The iterative refinement approach ensured that the final version of the dashboard reflects the authentic needs of the stakeholders rather than the initial assumptions of the researcher. Each iteration demonstrated the principle that "development teams should be involved in the process of selecting the metrics that are important to measure."

The gradual convergence of the feedback across the three iterations validates the goal-oriented measurement principle introduced by the GQM approach. Participants focused on metrics that supported their contextual awareness instead of requesting data on an availability principle. This alignment between stakeholder needs and metric selection confirms that the GQM methodology successfully guided the framework design.

Moreover, during the evaluation process, we did not record a request about missing metrics. This validates the simplicity principle mentioned in Chapters 4 and 5. The participants were satisfied with the set of selected metrics, suggesting that the dashboard achieves a balance between cognitive load and holistic comprehensiveness.

Chapter 8

Discussion

In this chapter we describe how this thesis contributes to the field of performance measurement both from theoretical and practical perspectives. Additionally, we illustrate the link between the points we drew from the literature review in Chapter 2 and data collection, and analysis in Chapter 4. Lastly, we present the limitations and threats to validity of the research.

8.1 Key Findings

In this thesis we address the challenge of measuring performance in hybrid Scrum/DevOps environments where traditional metrics often fail to capture the complexity of sustainable software delivery. By applying a Design Science Research approach through survey distribution, interviews, analysis of current performance measurement systems, and iterative evaluation of the delivered PMS prototype, this study demonstrates that effective measurements in hybrid teams requires more sophisticated frameworks that integrate multiple perspectives while maintaining practical utility.

The quantitative findings from the survey, in which 49 participants filled in their subjective views, revealed some statistically significant correlations illustrating the team dynamics in the case study company. For instance, one of the correlations indicates that constructive feedback moments (retrospectives with deep discussion of work dynamics) are likely to be linked to elevated autonomy levels. Due to the weak statistical significance, and the limited pool of responses, causality between the two cannot be confirmed. This example supports the theory of Davis and Daniels (2016) who state that feedback loops and culture that embraces open discussion are crucial for building autonomy within teams.

A more concerning finding from the interviews is the fact that 67% of the respondents engage in premature sprint closure to maintain favorable metrics. This exposes a dysfunction inherent to poorly designed measurement systems, confirming the point of Bouwers et al. (2012) about metric manipulation or lack of quality assurance in the measurement process.

The qualitative analysis further reinforced these quantitative insights while revealing measurement challenges from a cultural perspective. Many of the interview participants stated that the equal integration of Scrum and DevOps practices is subject to cultural transformation within the environment rather than the adoption of certain tools, which aligns with the reasoning of Leite et al. (2019) that DevOps success is based on sustainable changes in organizational culture. Additionally, Forsgren et al. (2021) further validates the framework we propose due to its emphasis on human factors and technical metrics.

Circling back to the requirements stated in Chapter 1, the delivered product directly covers three of the four the pillars introduced by Davis and Daniels (2016). The affinity aspect is tackled by centralizing the communication point for all parties, that are relevant to software development, within the case study scope. By bringing light over the different aspects of performance, from business, technical or governance level, the employees will be able to build a better mutual understanding that can lead to higher trust and empathy. Secondly, the integration of tools like Power Automate, Azure DevOps, and the mimicked functionality for ServiceNow, we demonstrate how the functionality of different platforms can be combined to enable desired process improvements. Next, by bridging the gap between operational and governance performance measurements, we show how scalable the solution is, thus confirming its fit within the hybrid Scrum/DevOps nature of the case study environment.

Additionally, the SPACE framework caters for both effectiveness and efficiency metrics, as shown in the metric selection in Chapter 6. Good examples for the efficiency aspect are the metrics monitoring *Cycle Time Planning Accuracy.* As for the effectiveness part, sufficiently illustrative are the *Closed Incidents within SLA* and *Test Finding Distribution per Environment*, as they both show the reliability of the development unit.

8.2 Contribution to Theory

In this thesis, we extend the existing theory on performance measurement in several important ways. To start with, we provide empirical validation for integrating DORA metrics (Georgsson, 2024) within the SPACE framework dimensions (Forsgren et al., 2021) in environments where Scrum and DevOps are implemented simultaneously. Thanks to the stakeholder acceptance of this integration we can further confirm that this paper supports the principles for design of performance measurement systems in complex R&D environments (de Oliveira and Proença, 2019).

By utilizing the Goal-Question-Metric (GQM) methodology (Berander and Jönsson, 2006) in a hybrid environment, this paper contributes to measurement theory by demonstrating how structured approaches can maintain the theoretical validity of designed models while adapting to the needs of a diverse audience of stakeholders. During the evaluation process, the transition from role-based to topic-oriented dashboard views shows how participatory design can refine theoretical frameworks for better application in practice (Benaim, 2015).

Seeing the stakeholder feedback progressed from structural points down to implementation details suggests a predictable adoption pathway for performance measurement systems, thus extending the theory on designing such systems to match organizational objectives (Brahimi et al., 2019; Bugwandeen and Ungerer, 2019).

8.3 Practical contributions

This paper provides actionable advice for companies that struggle with designing optimal performance measurement systems in hybrid environments by addressing the gaps described by Ayyash (2024) in scaling Agile and DevOps practices. The separation of metrics in different viewpoints provides a template for balancing individual responsibility with cross-functional collaboration, thus supporting the Continuous Scrum approach (Samarawickrama and Perera, 2017).

The iterative evaluation methodology provides a replicable process for organizational adoption, thus addressing the point of Smeds et al. (2015) about the difficulties in large-scale DevOps transitions. Instead of using a top-down approach for steering performance, organizations can use a multi-iteration refinement process to ensure practical relevance and stakeholder acceptance.

Observations from data collection in the exploration phase showed that approximately half of the survey respondents indicated a practice of gaming metrics in a minor or more severe form, a phenomenon that closely aligns with the research of Bouwers et al. (2012) on measurement dysfunctions. The designed framework in this study addresses this by proposing participatory design and feedback loops which will involve teams in metrics selection. The basis for this choice can be found in the research of Chiesa and Frattini (2009) where we see arguments that careful design of PMS can prevent perceptions for controlling mechanisms via involving multiple stakeholders.

8.4 Limitations and Threats to Validity

The evaluation was carried out with a proof-of-concept dashboard using synthetic data. This limits the assessment to conceptual design and user experience rather than real-life performance impact. Participants could reason on the relevance of metrics and the comprehensiveness of visualizations but could not assess whether this is sufficient for operational improvements or better decision making in practice.

Despite mimicking a diverse set of scenarios, including critical situations with mock data to represent sufficient deviations from optimal performance, and vice versa, the same could not be implemented for stakeholder perceptions. The absence of pressure from actual deadlines, real-life priorities, and tool integration could have resulted in a more positive feedback than would occur in a release of a minimum viable product of the dashboard in a production environment.

While the participant pool involved employees from different departments in the case study company (with representatives of IOS Development teams, Platform Engineering and others), it was still limited in size and organizational diversity. Despite the maximally diversified selection, the case study company has employed general guidelines for implementing Scrum and DevOps practices, which could differ in other organizations.

The general findings from the research also fall into certain generalizability constraints. Since the case study context stays within a financial services organization, this may limit the applicability to other industries, especially considering the observations of Laukkarinen et al. (2018) about the regulatory constraints in DevOps adoption. The highly regulated environment of the banking industry and the inherent low risk appetite are likely not present in other industries and domains.

Lastly, while sufficient for extracting valuable correlations, the survey sample size with only 49 participants limits the statistical power of the study to detect smaller effect sizes. As a suggestion for future work, following the recommendations of Forsgren et al. (2021), a larger sample size should be used to uncover more and novel relationships with potential causality between measurement practices and team performance.

Chapter 9

Conclusion

This chapter presents the accomplishments of this research by summarizing the answers to each of the research questions stated in Chapter 1. Furthermore, we provide suggestions for future research that can build on the framework proposed in this thesis.

9.1 Answers to the research questions

Main Research Question
Which key performance indicators (KPIs) from Agile, DevOps, and team health frameworks are most suitable for hybrid Scrum/DevOps environments?
This thesis addressed the challenge of designing a performance measurement system (PMS) tailored to the needs of development teams implementing Scrum and DevOps methodologies together. Through a Design Science Research approach we demonstrate that optimal performance measurement can be achieved by combining the DevOps Research & Assessment framework with the dimensions of the SPACE framework.

Specifically, with our research, we validate that metrics addressing Satisfaction and Well-being (such as Average Developer Satisfaction and Value-driven orientation), Performance outcomes (including Lead Time for Change, Lead Time, and Cumulative Business Value Delivered), Activity measures (like Technical Debt Contribution Ratio and Planning Accuracy), and Efficiency and Flow indicators (such as Test Finding Distribution per Environment and Closed Incidents within SLA) provide the most comprehensive and actionable performance assessment (Forsgren et al., 2021; Velasquez et al., 2014).

This heterogeneous combination of technical delivery and human-centric idicators addresses both the operational efficiency of DevOps and the collaborative, iterative nature of Scrum methods. The empirical validation through stakeholder feedback confirmed that the integrated approach captures performance dimensions that single-framework approaches consistently miss.

SRQ 1. How can the design of performance dashboards be tailored to the distinct perspectives and responsibilities of developers, product owners, and IT leads?
With regards to the first subresearch question - which aims to reveal how dashboards can be designed to address the distinct perspectives of developers, product owners, and IT managers - it was found that topic-oriented structure provides better support than job-oriented one. The iterative evaluation of the developed dashboard prototype revealed that distinguishing between technical, governance, and business-centric KPIs addresses information needs much better by mitigating the risk of fragmentation between the aforementioned stakeholder groups, thus reintroducing silo-based performance measurements. This is supported by Bugwandeen and Ungerer (2019), where we find evidence that, while requests for dashboards come mainly from higher management, the design and development must be done in a bottom-up approach to preserve the relevance, reliability, and subsequent accuracy of the information displayed.

SRQ 2. What are the main challenges and cultural factors influencing the adoption and effective ness of hybrid performance measurement systems?
Both surveys and interviews presented evidence that measurement systems are accepted by employees

when they are accompanied by high transparency, leadership support, and psychological safety. The support for this finding is discovered in Effective DevOps by Davis and Daniels (2016) whose theory puts a high emphasis on cultural transformation in the adoption of DevOps. Hence, the main challenges and cultural factors that influence the adoption of hybrid performance measurement systems include lack of leadership support in the review of metrics and fear of performance data misuse. In addition, regulatory concerns must be considered before defining optimal performance due to the diversity of sectors in which hybrid Scrum/DevOps teams can operate (Laukkarinen et al., 2018).

By embedding participatory design and structured feedback loops during the development of the PMS (as a dashboard), continuous improvement and engagement became a resulting factor.

SRQ 3. How can performance measurement systems be designed to enable continuous improvement and stakeholder engagement?

Answering SRQ 3, this study demonstrates how feedback should be integral to the design of the PMS. First, the relevant stakeholders (in the case study being developers, product owners and IT leads) should be involved in KPI workshops where they will be able to state what KPIs cover the metrics they deem important for their operational effectiveness - a direct implementation of the Goal-Question-Metric approach (Berander and Jönsson, 2006). Secondly, facilitating regular feedback moments, like a sprint retrospective or a quarterly review, create a formal adaptation cycle which drives continuous improvement (Davis and Daniels, 2016). Thirdly, topic-oriented viewpoints that present performance from technical, business-centric and governance perspective with historical trends support cross-functional dialogues and collaborative problem solving. This is backed by Bugwandeen and Ungerer (2019) who state that structured stakeholder involvement is essential for the relevance of dashboards.

SRQ 4. How can the Performance Measurement System (PMS) be validated in practice to ensure it provides meaningful, actionable insights and avoids common pitfalls such as metric gaming or dashboard fatigue?

Finally, referring to SRQ. 4, practical validation through iterative evaluation rounds demonstrated that the designed framework provides insights which are both meaningful, manageable, and contextually rich, all while avoiding the well-described phenomenon - 'metrics galore' (Bouwers et al., 2012). The final iteration of the evaluation process did not record additional improvement points, thus confirming that the participatory GQM-driven structure maintained balance and usability.

9.2 Future work

To further confirm the validity of this research and further development of the proposed framework, we will draw a pathway for later research on this topic. Due to time constraints at the case study company, we did not test the performance of the proposed prototype in a live business setting with real data. Therefore, validation through field experiments is required to assess whether the proposed PMS contributes to performance improvements at a desired level. Additionally, replication of the experiment within another case study environment is required to ensure actual practical utility that is not limited to the setting of the current case study.

The work on this research concludes with the following list of research recommendations:

Integration of Advanced Analytics and Automation

To optimize the PMS presented in this thesis, future research can be conducted on the use of machine learning techniques for automated anomaly detection, performance forecasting, and root-cause analysis within the performance dashboards. The current data collection measured correlations between job focus, years of experience, DevOps familiarity, cross-functional orientation and other factors which were largely attributable to the single individual. With more advanced analytics, performance forecasting can take into account time of year, resource allocation, and other factors that can account for fluctuations in the quantity or quality of delivered software.

The Impact of the Hedonic Treadmill Phenomena Within Performance Measurements in the Software Delivery context.

By investigating this phenomenon, more light can be thrown on sustainable management of human resources within software development environments. With this research, one could examine whether static Service Level Agreements (SLAs) continue to bring valuable insights about the performance of development units or if gradual increases of the same SLAs serve as motivating factors for development teams to optimize their operations and efficiency.

Integration of Large Language Models (LLMs) within Performance Measurement Systems
On this topic, one can work on integrating an LLM inside a performance dashboard and using all present metrics as input to present a narrative on what improvements can be made to optimize the software delivery efficiency.

Appendix A

Survey Results

This appendix contains the full list of survey questions used in the study. We used Google Forms as the distribution tool. We made this choice due to existing privacy policies that prevent the flow of information between the case study company devices and external tools. Each participant received a QR code containing the link to the survey along with instructions to get acquainted with the purpose of the study, which is displayed before the first section of questions.

Measuring Performance within SCRUM and DevOps

This survey is part of a Master’s thesis focused on understanding and improving team dynamics and workflow efficiency. Your input will help identify patterns and areas for enhancement within collaborative work environments. **The survey does not collect any personal information, ensuring your complete anonymity.**

You are encouraged to answer all questions as honestly as possible, as your genuine insights are critical to the study's success. The results will contribute to recommendations that aim to create more productive and satisfying work environments for your teams.

The individual responses from this survey **WILL NOT** be shared with any party / individual other than the researcher (Nikolay Pavlov).

!!! The reason why I am collecting this information is to evaluate what is the DevOps state of our grid. With the data collected I will be able to design a dashboard which will show in the most objective and accurate way the software delivery performance of the grid and each team within it. **!!!**

For any questions, please contact the researcher (Nikolay Pavlov)!

Thank you for your time and participation!

1. What is the core focus of the chapter you participate in?

   Mark only one oval.

   - ◯ Development
   - ◯ Operations
   - ◯ Information Analysis
   - ◯ Quality Assurance
   - ◯ Product Owner
   - ◯ Business Analyst

2. Have you worked in a DevOps manner prior to joining your current role?

   Mark only one oval.

   - Yes
   - No

3. How long have you been working in your current role?

   Mark only one oval.

   - 0-2 years
   - 3-5 years
   - 6-10 years
   - 10+ years

4. How long have you been working in the IT industry?

   Mark only one oval.

   - 0-2 years
   - 3-5 years
   - 6-10 years
   - 10+ years

5. What percentage of your total work consists of tasks directly related to your core expertise(your chapter's focus)? (e.g. per week/sprint)

   Mark only one oval.

   - ⬭ 0% - 20%
   - ⬭ 21% - 40%
   - ⬭ 42% - 60%
   - ⬭ 61% - 80%
   - ⬭ 81% - 100%

6. When working on tasks within your core expertise, which approach typically applies to you?

   Mark only one oval.

	1	2	3	4	
Ana	⬭	⬭	⬭	⬭	Dive into the task with a trial and error approach / e before implementing anything

7. On scale of 1 to 4, how confident are you when working on tasks **outside** your core expertise

   Mark only one oval.

	1	2	3	4	
Unc	⬭	⬭	⬭	⬭	Very Confident

8. On scale of 1 to 4, when working, what is your attitude when taking up tasks **outside** your core expertise

Mark only one oval.

	1	2	3	4	
Very	◯	◯	◯	◯	Very positive

9. Given the list below, please select 4 of the KPIs (Key Performance Indicators) that you consider most important:

Check all that apply.

- ☐ Error Rate – frequency of errors/incidents across environments
- ☐ Cycle Time – time for a user story to get from started to finished
- ☐ System Uptime – percentage of time a system is operational and running
- ☐ Planning Accuracy – planned work vs actually completed work ratio
- ☐ Lead Time for Change – time taken from code commit to production release
- ☐ Sprint Velocity – amount of work completed in story points per sprint
- ☐ Release Rollback Frequency – number of rollbacks required due to failed deployment
- ☐ Blocked Work items - amount of work items blocked during a sprint
- ☐ Work In Progress (WIP) – number of tasks actively being worked on
- ☐ Deployment Frequency – number of deployments within a specific period

Section 2

During Progress Reporting (e.g. sprint review, Grid sync, Big Demo), do you perceive changes in any of the following areas:

10. Engagement in the company vision and strategy

Mark only one oval.

	1	2	3	4	
Decı	◯	◯	◯	◯	Increased

11. Functional understanding of the products that you develop

Mark only one oval.

	1	2	3	4	
Decı	◯	◯	◯	◯	Increased

12. Contextual Knowledge regarding Management decisions and targets

Mark only one oval.

	1	2	3	4	
Decı	◯	◯	◯	◯	Increased

13. Level of autonomy - the degree of freedom you have in making decisions and managing your tasks or responsibilities without external control

Mark only one oval.

	1	2	3	4	
Decı	◯	◯	◯	◯	Increased

Section 3

14. How do you typically approach tasks where your skillset is less applicable or underdeveloped?

Mark only one oval.

◯ I let a more competent colleague within my team handle it to resolve the issue faster

◯ I complete the task at the expense of extra time

◯ I work with a more competent colleague to learn and resolve the issue

15. How frequently do you collaborate with colleagues from other teams to resolve an issue/task/incident?

Mark only one oval.

- ◯ Continuously (on daily basis)
- ◯ Almost always (weekly)
- ◯ Often (several times a month)
- ◯ Sometimes (once or twice a month)
- ◯ Rarely (once in a few months)

16. How frequently do you collaborate with colleagues in your team to resolve an issue/task/incident?

Mark only one oval.

- ◯ Continuously (on a daily basis)
- ◯ Almost always (weekly)
- ◯ Often (several times a month)
- ◯ Sometimes (once or twice a month)
- ◯ Rarely (once in a few months)

Retrospectives and Planning

17. On a scale 1 to 4, how confident are you to speak up during retrospectives?

Mark only one oval.

	1	2	3	4	
Unc	◯	◯	◯	◯	Confident

18. What are the typical results from the retrospectives?

Check all that apply.

- ☐ Long-lasting changes in the team dynamics
- ☐ Temporary changes in the team dynamics with diminishing effects
- ☐ No change in team dynamics

19. What percentage of the planned user stories does your team typically close within a sprint?

Mark only one oval.

- ◯ 0% - 25%
- ◯ 26% - 50%
- ◯ 51% - 75%
- ◯ 76% - 100%
- ◯ > 100%

20. How does your team treat unfinished user stories during sprint open/close?

Mark only one oval.

- ◯ Creating clone stories in new sprint and marking everything in previous sprint as completed
- ◯ Taking the unfinished user stories to new sprint
- ◯ Closing user stories in previous sprint and create an Action item in new sprint to finish the user stories
- ◯ Leaving the user stories open in previous sprint and mark them as completed late

Untitled Section

For the next three questions, please provide information on what works best at the moment, indicate any existing pitfalls, practices that could be eliminated or optimized.

21. What suggestions do you have for improving workflow dynamics within or outside your team?

22. What suggestions do you have to help management provide better support and guidance for development teams?

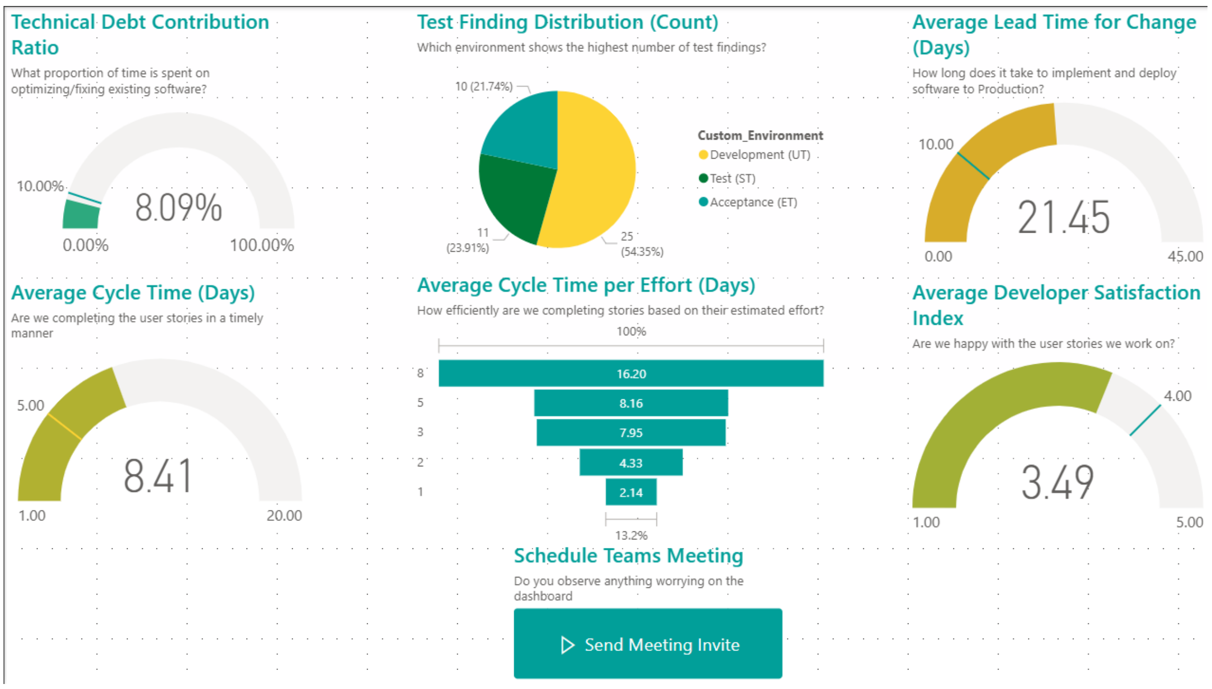
23. What suggestions do you have to help increase the awareness of management on team and workflow dynamics?

24. Please, fill in your name you would like to participate in an interview where you can provide further insights you might consider appropriate (Not mandatory)

Appendix B

Dashboard Prototype

Iteration 1



Technical Debt Contribution Ratio
What proportion of time is spent on optimizing/fixing existing software?
10.00%
8.09%
0.00%
100.00%
Test Finding Distribution (Count)
Which environment shows the highest number of test findings?
10 (21.74%)
11 (23.91%)
25 (54.35%)
Custom_Environment
● Development (UT)
● Test (ST)
● Acceptance (ET)
Average Lead Time for Change (Days)
How long does it take to implement and deploy software to Production?
10.00
21.45
0.00
45.00
Average Cycle Time (Days)
Are we completing the user stories in a timely manner
5.00
8.41
1.00
20.00
Average Cycle Time per Effort (Days)
How efficiently are we completing stories based on their estimated effort?
100%
8
16.20
5
8.16
3
7.95
2
4.33
1
2.14
13.2%
Average Developer Satisfaction Index
Are we happy with the user stories we work on?
4.00
3.49
1.00
5.00
Schedule Teams Meeting
Do you observe anything worrying on the dashboard
Send Meeting Invite

Figure B.1: Developer Perspective

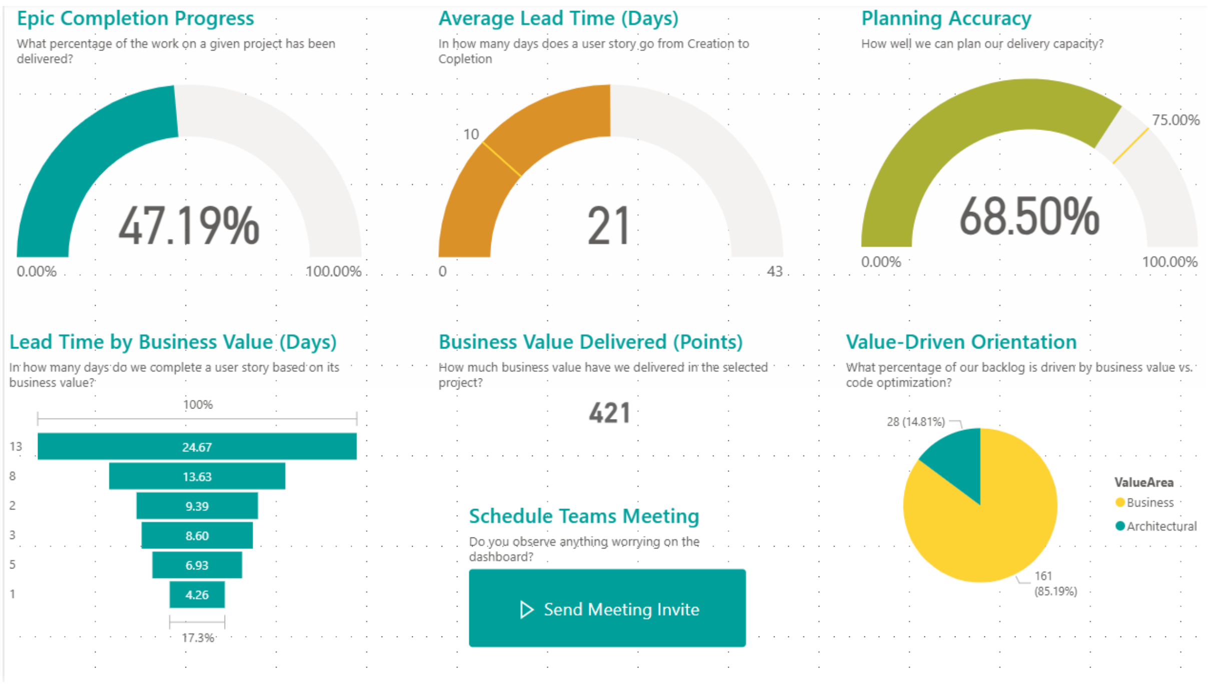
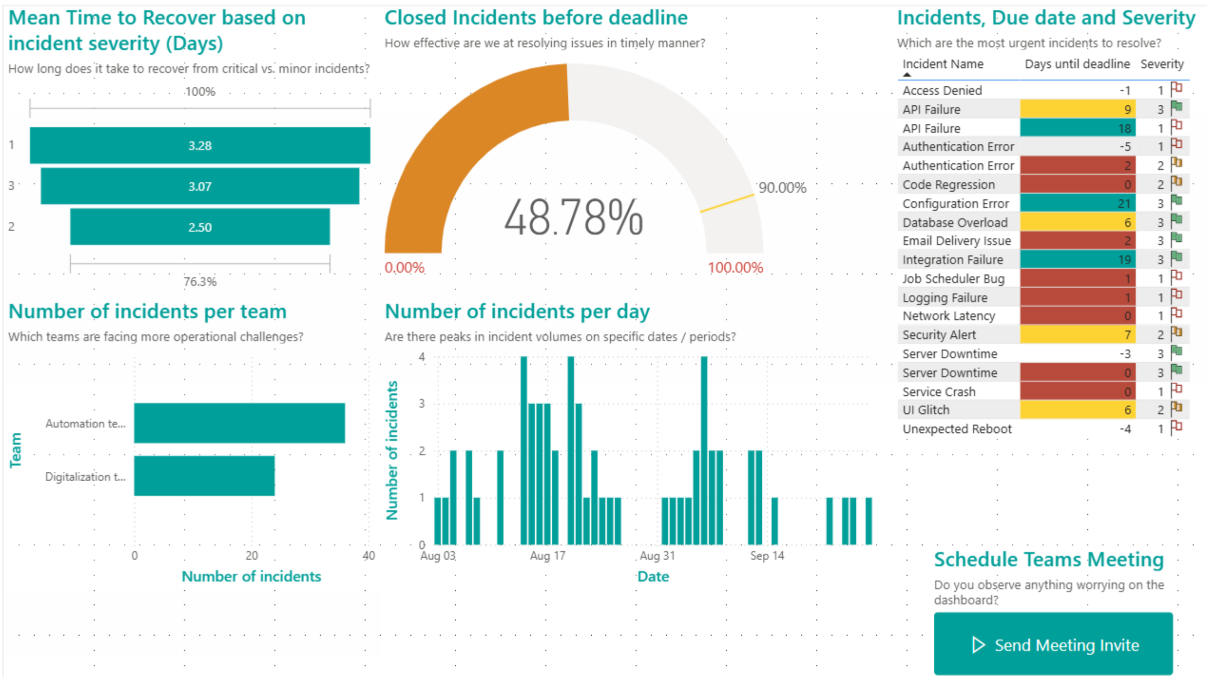

Epic Completion Progress
What percentage of the work on a given project has been delivered?
47.19%
0.00%
100.00%
Average Lead Time (Days)
In how many days does a user story go from Creation to Copletion
10
21
0
43
Planning Accuracy
How well we can plan our delivery capacity?
75.00%
68.50%
0.00%
100.00%
Lead Time by Business Value (Days)
In how many days do we complete a user story based on its business value?
100%
13
24.67
8
13.63
2
9.39
3
8.60
5
6.93
1
4.26
17.3%
Business Value Delivered (Points)
How much business value have we delivered in the selected project?
421
Schedule Teams Meeting
Do you observe anything worrying on the dashboard?
Send Meeting Invite
Value-Driven Orientation
What percentage of our backlog is driven by business value vs. code optimization?
28 (14.81%)
161 (85.19%)
ValueArea
Business
Architectural

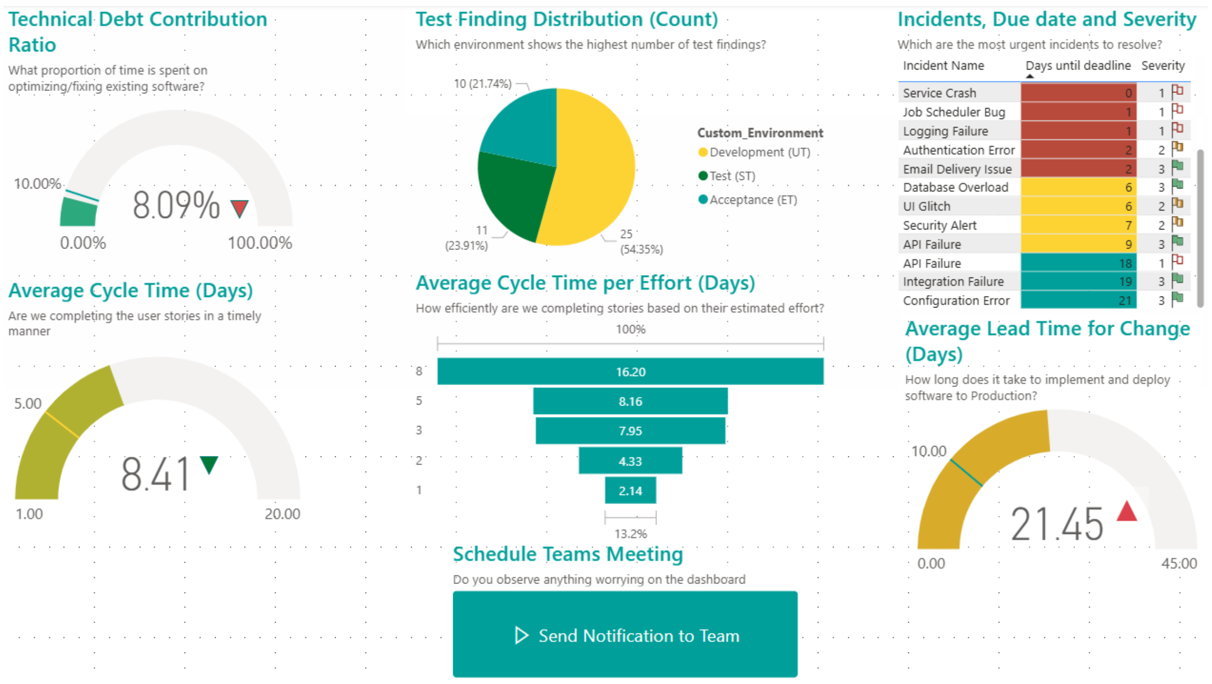
Figure B.2: Product Owner Perspective


Mean Time to Recover based on incident severity (Days)
How long does it take to recover from critical vs. minor incidents?
100%
1
3.28
3
3.07
2
2.50
76.3%
Closed Incidents before deadline
How effective are we at resolving issues in timely manner?
48.78%
90.00%
0.00%
100.00%
Number of incidents per team
Which teams are facing more operational challenges?
Team
Automation te...
Digitalization t...
0
20
40
Number of incidents
Number of incidents per day
Are there peaks in incident volumes on specific dates / periods?
Number of incidents
4
3
2
1
0
Aug 03
Aug 17
Aug 31
Sep 14
Date
Incidents, Due date and Severity
Which are the most urgent incidents to resolve?
Schedule Teams Meeting
Do you observe anything worrying on the dashboard?
Send Meeting Invite

Incident Name	Days until deadline	Severity
Access Denied	-1	1
API Failure	9	3
API Failure	18	1
Authentication Error	-5	1
Authentication Error	2	2
Code Regression	0	2
Configuration Error	21	3
Database Overload	6	3
Email Delivery Issue	2	3
Integration Failure	19	3
Job Scheduler Bug	1	1
Logging Failure	1	1
Network Latency	0	1
Security Alert	7	2
Server Downtime	-3	3
Server Downtime	0	3
Service Crash	0	1
UI Glitch	6	2
Unexpected Reboot	-4	1

Figure B.3: IT Lead Perspective

Iteration 2



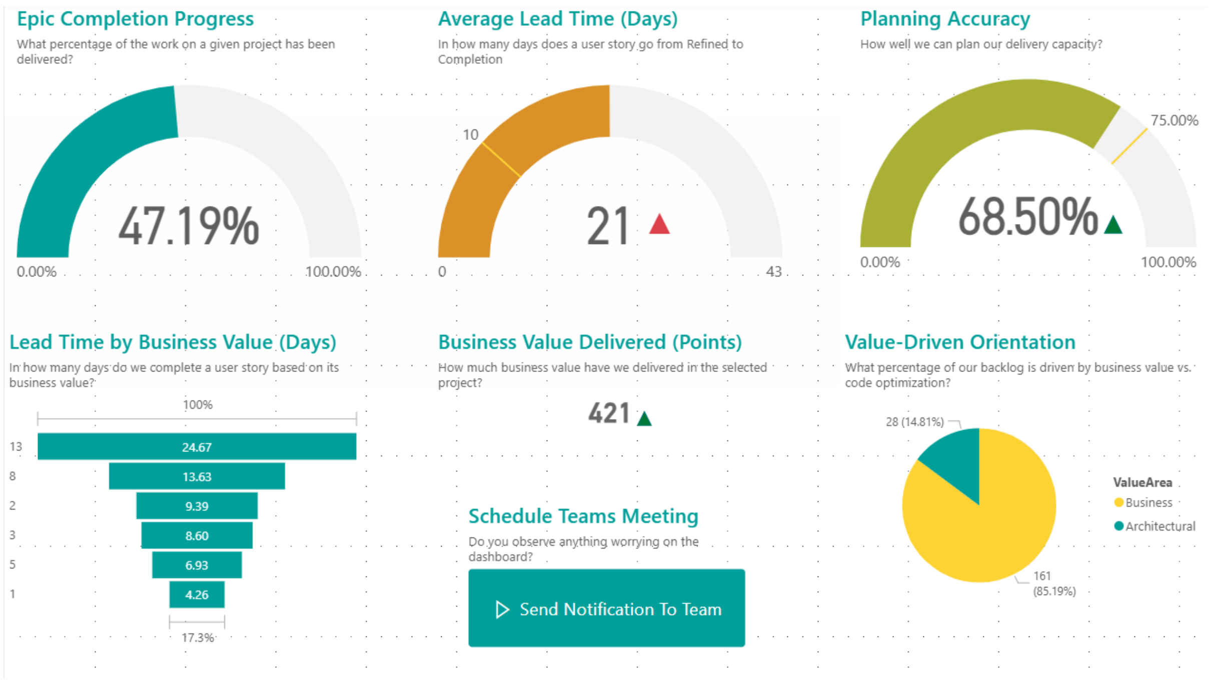
Technical Debt Contribution Ratio
What proportion of time is spent on optimizing/fixing existing software?
10.00%
8.09%
0.00%
100.00%
Test Finding Distribution (Count)
Which environment shows the highest number of test findings?
10 (21.74%)
11 (23.91%)
25 (54.35%)
Custom_Environment
Development (UT)
Test (ST)
Acceptance (ET)
Average Cycle Time (Days)
Are we completing the user stories in a timely manner
5.00
8.41
1.00
20.00
Average Cycle Time per Effort (Days)
How efficiently are we completing stories based on their estimated effort?
100%
8
5
3
2
1
16.20
8.16
7.95
4.33
2.14
13.2%
Schedule Teams Meeting
Do you observe anything worrying on the dashboard
Send Notification to Team
Average Lead Time for Change (Days)
How long does it take to implement and deploy software to Production?
10.00
21.45
0.00
45.00

Incidents, Due date and Severity

Which are the most urgent incidents to resolve?

Incident Name	Days until deadline	Severity
Service Crash	0	1
Job Scheduler Bug	1	1
Logging Failure	1	1
Authentication Error	2	2
Email Delivery Issue	2	3
Database Overload	6	3
UI Glitch	6	2
Security Alert	7	2
API Failure	9	3
API Failure	18	1
Integration Failure	19	3
Configuration Error	21	3

Figure B.4: Technical Viewpoint



Epic Completion Progress
What percentage of the work on a given project has been delivered?
47.19%
0.00%
100.00%
Average Lead Time (Days)
In how many days does a user story go from Refined to Completion
10
21
0
43
Planning Accuracy
How well we can plan our delivery capacity?
75.00%
68.50%
0.00%
100.00%
Lead Time by Business Value (Days)
In how many days do we complete a user story based on its business value?
100%
13
8
2
3
5
1
24.67
13.63
9.39
8.60
6.93
4.26
17.3%
Business Value Delivered (Points)
How much business value have we delivered in the selected project?
421
Schedule Teams Meeting
Do you observe anything worrying on the dashboard?
Send Notification To Team
Value-Driven Orientation
What percentage of our backlog is driven by business value vs. code optimization?
28 (14.81%)
161 (85.19%)
ValueArea
Business
Architectural

Figure B.5: Business Oriented Viewpint

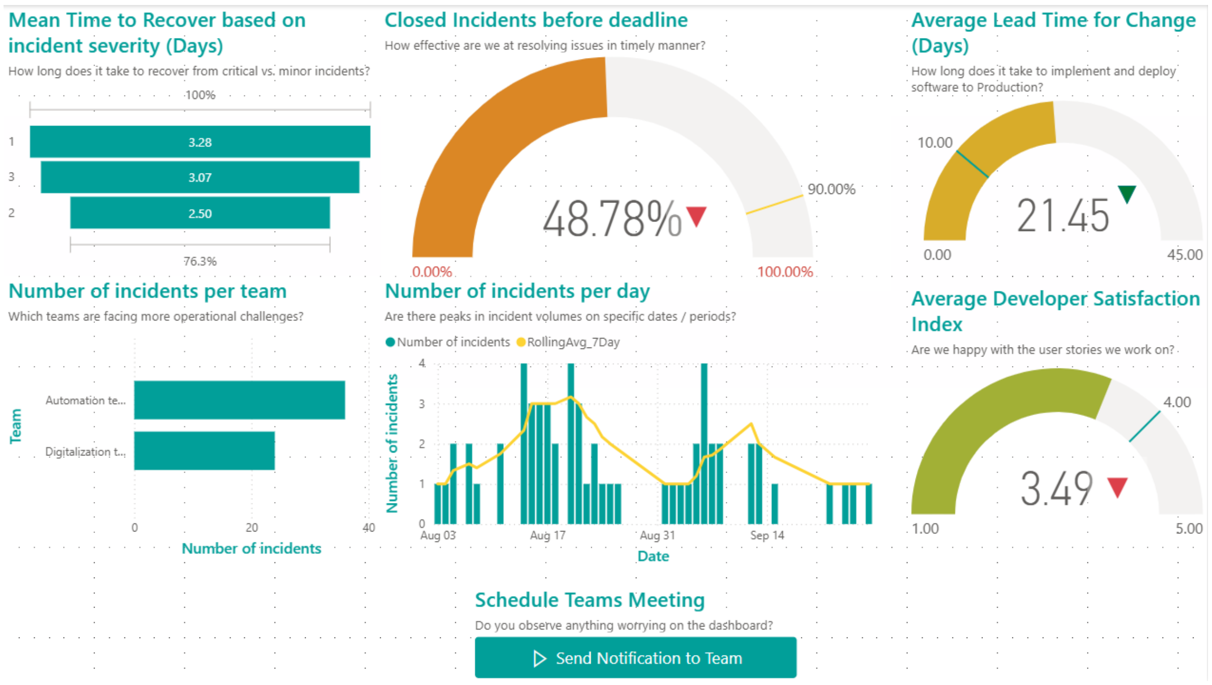
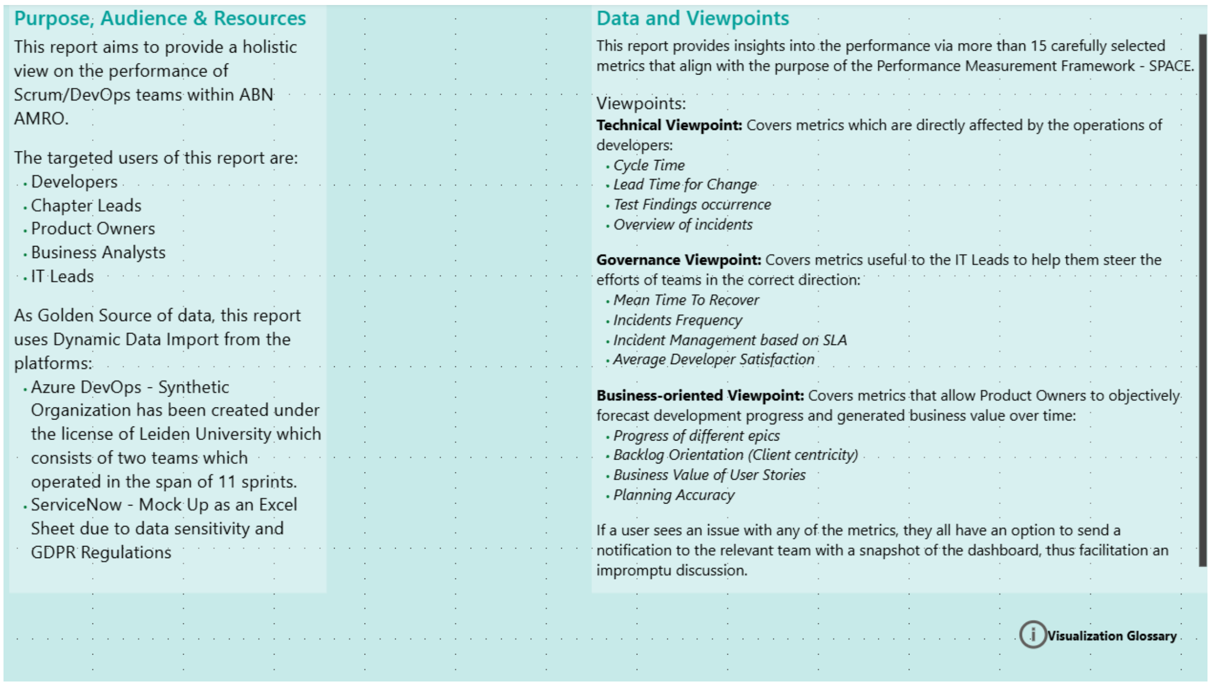

Mean Time to Recover based on incident severity (Days)
How long does it take to recover from critical vs. minor incidents?
100%
1
3.28
3
3.07
2
2.50
76.3%
Closed Incidents before deadline
How effective are we at resolving issues in timely manner?
48.78%
90.00%
0.00%
100.00%
Average Lead Time for Change (Days)
How long does it take to implement and deploy software to Production?
10.00
21.45
0.00
45.00
Number of incidents per team
Which teams are facing more operational challenges?
Team
Automation te...
Digitalization t...
0
20
40
Number of incidents
Number of incidents per day
Are there peaks in incident volumes on specific dates / periods?
● Number of incidents ● RollingAvg_7Day
Number of incidents
4
3
2
1
0
Aug 03
Aug 17
Aug 31
Sep 14
Date
Average Developer Satisfaction Index
Are we happy with the user stories we work on?
4.00
3.49
1.00
5.00
Schedule Teams Meeting
Do you observe anything worrying on the dashboard?
Send Notification to Team

Figure B.6: Governance Viewpoint



Purpose, Audience & Resources

This report aims to provide a holistic view on the performance of Scrum/DevOps teams within ABN AMRO.

The targeted users of this report are:
- Developers
- Chapter Leads
- Product Owners
- Business Analysts
- IT Leads

As Golden Source of data, this report uses Dynamic Data Import from the platforms:
- Azure DevOps - Synthetic Organization has been created under the license of Leiden University which consists of two teams which operated in the span of 11 sprints.
- ServiceNow - Mock Up as an Excel Sheet due to data sensitivity and GDPR Regulations

Data and Viewpoints

This report provides insights into the performance via more than 15 carefully selected metrics that align with the purpose of the Performance Measurement Framework - SPACE.

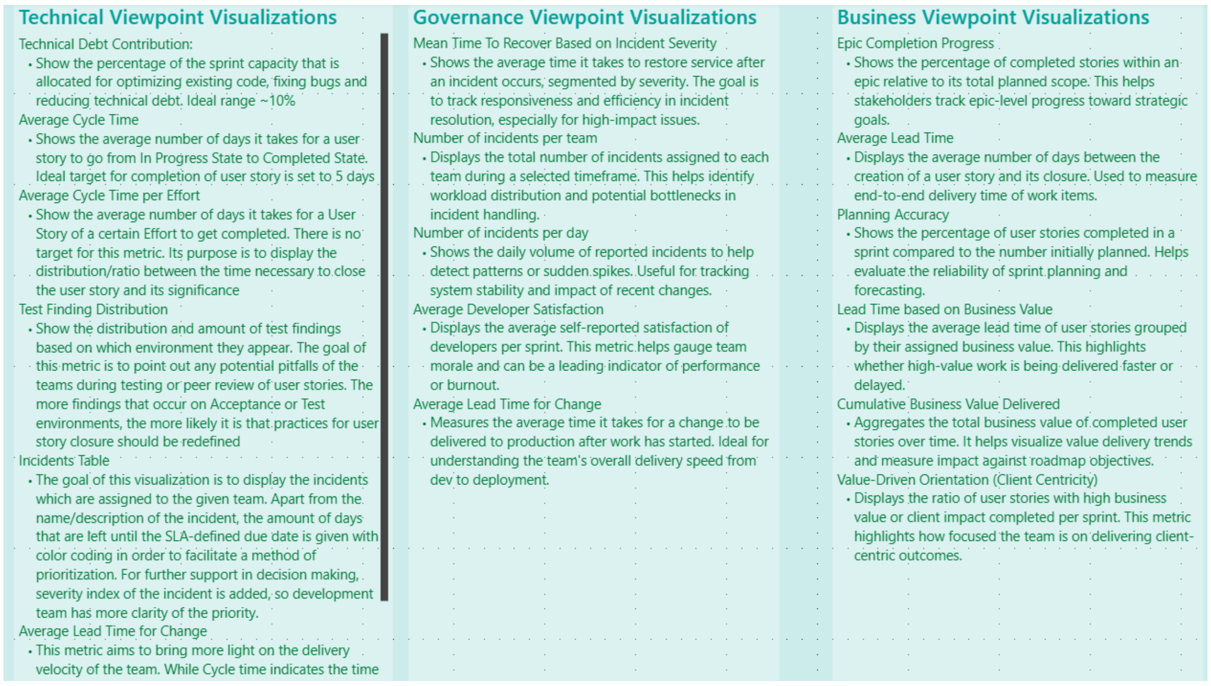
Viewpoints:
Technical Viewpoint: Covers metrics which are directly affected by the operations of developers:
- *Cycle Time*
- *Lead Time for Change*
- *Test Findings occurrence*
- *Overview of incidents*

Governance Viewpoint: Covers metrics useful to the IT Leads to help them steer the efforts of teams in the correct direction:
- *Mean Time To Recover*
- *Incidents Frequency*
- *Incident Management based on SLA*
- *Average Developer Satisfaction*

Business-oriented Viewpoint: Covers metrics that allow Product Owners to objectively forecast development progress and generated business value over time:
- *Progress of different epics*
- *Backlog Orientation (Client centricity)*
- *Business Value of User Stories*
- *Planning Accuracy*

If a user sees an issue with any of the metrics, they all have an option to send a notification to the relevant team with a snapshot of the dashboard, thus facilitation an impromptu discussion.

Figure B.7: Information regarding the viewpoints on the dashboard together, the target audience, and the data used for the demonstration



Technical Viewpoint Visualizations

Technical Debt Contribution:

- Show the percentage of the sprint capacity that is allocated for optimizing existing code, fixing bugs and reducing technical debt. Ideal range ~10%

Average Cycle Time

- Shows the average number of days it takes for a user story to go from In Progress State to Completed State. Ideal target for completion of user story is set to 5 days

Average Cycle Time per Effort

- Show the average number of days it takes for a User Story of a certain Effort to get completed. There is no target for this metric. Its purpose is to display the distribution/ratio between the time necessary to close the user story and its significance

Test Finding Distribution

- Show the distribution and amount of test findings based on which environment they appear. The goal of this metric is to point out any potential pitfalls of the teams during testing or peer review of user stories. The more findings that occur on Acceptance or Test environments, the more likely it is that practices for user story closure should be redefined

Incidents Table

- The goal of this visualization is to display the incidents which are assigned to the given team. Apart from the name/description of the incident, the amount of days that are left until the SLA-defined due date is given with color coding in order to facilitate a method of prioritization. For further support in decision making, severity index of the incident is added, so development team has more clarity of the priority.

Average Lead Time for Change

- This metric aims to bring more light on the delivery velocity of the team. While Cycle time indicates the time

Governance Viewpoint Visualizations

Mean Time To Recover Based on Incident Severity

- Shows the average time it takes to restore service after an incident occurs, segmented by severity. The goal is to track responsiveness and efficiency in incident resolution, especially for high-impact issues.

Number of incidents per team

- Displays the total number of incidents assigned to each team during a selected timeframe. This helps identify workload distribution and potential bottlenecks in incident handling.

Number of incidents per day

- Shows the daily volume of reported incidents to help detect patterns or sudden spikes. Useful for tracking system stability and impact of recent changes.

Average Developer Satisfaction

- Displays the average self-reported satisfaction of developers per sprint. This metric helps gauge team morale and can be a leading indicator of performance or burnout.

Average Lead Time for Change

- Measures the average time it takes for a change to be delivered to production after work has started. Ideal for understanding the team's overall delivery speed from dev to deployment.

Business Viewpoint Visualizations

Epic Completion Progress

- Shows the percentage of completed stories within an epic relative to its total planned scope. This helps stakeholders track epic-level progress toward strategic goals.

Average Lead Time

- Displays the average number of days between the creation of a user story and its closure. Used to measure end-to-end delivery time of work items.

Planning Accuracy

- Shows the percentage of user stories completed in a sprint compared to the number initially planned. Helps evaluate the reliability of sprint planning and forecasting.

Lead Time based on Business Value

- Displays the average lead time of user stories grouped by their assigned business value. This highlights whether high-value work is being delivered faster or delayed.

Cumulative Business Value Delivered

- Aggregates the total business value of completed user stories over time. It helps visualize value delivery trends and measure impact against roadmap objectives.

Value-Driven Orientation (Client Centricity)

- Displays the ratio of user stories with high business value or client impact completed per sprint. This metric highlights how focused the team is on delivering client-centric outcomes.

Figure B.8: Glossary Page explaining the meaning of the visualizations in the dashboard

Iteration 3

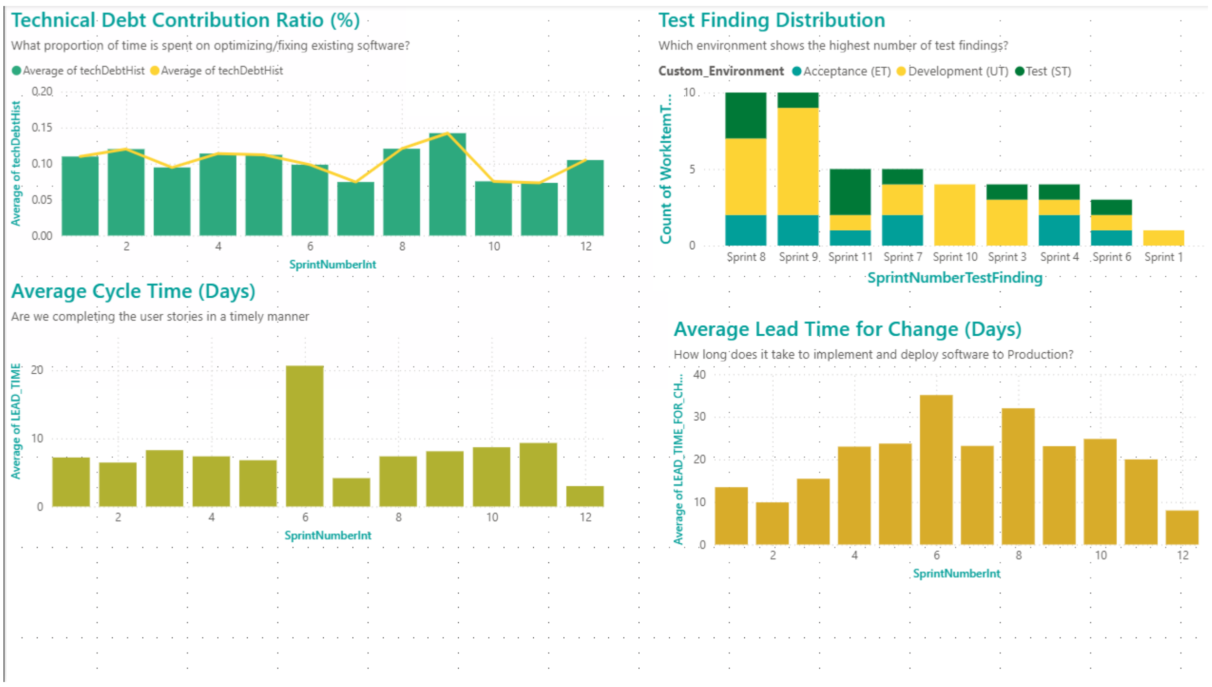


Figure B.9: Historical Overview of the Technical Viewpoint

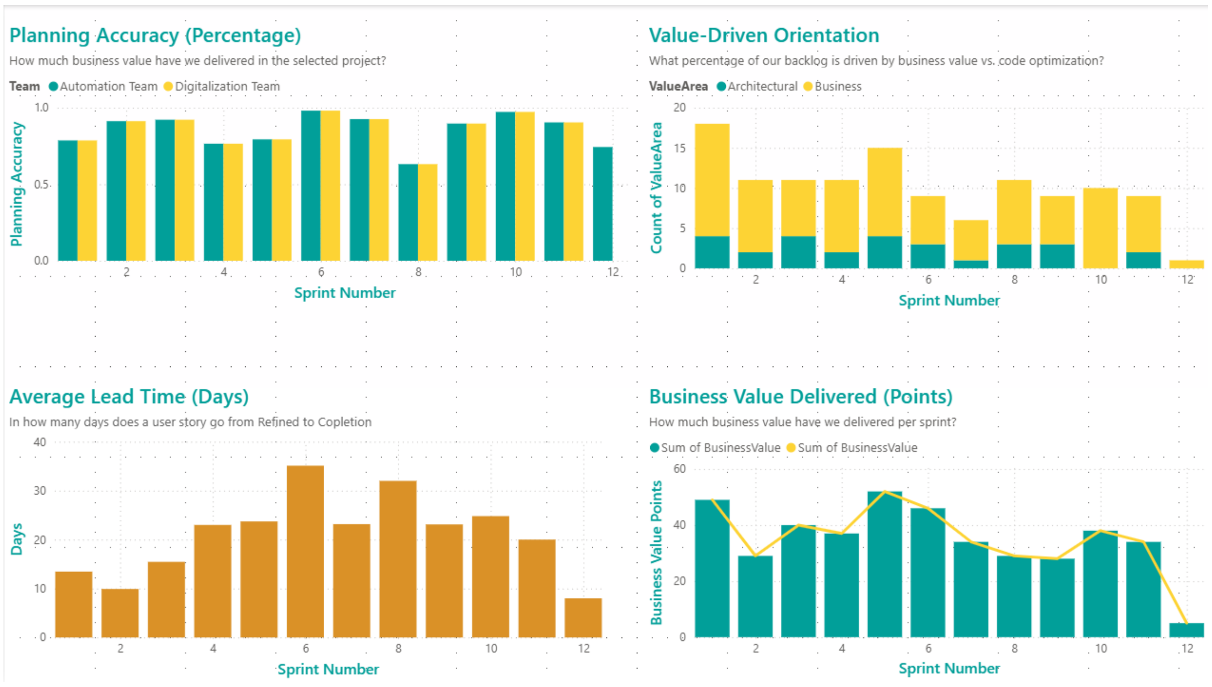


Figure B.10: Historical Overview of the Business Oriented Viewpoint

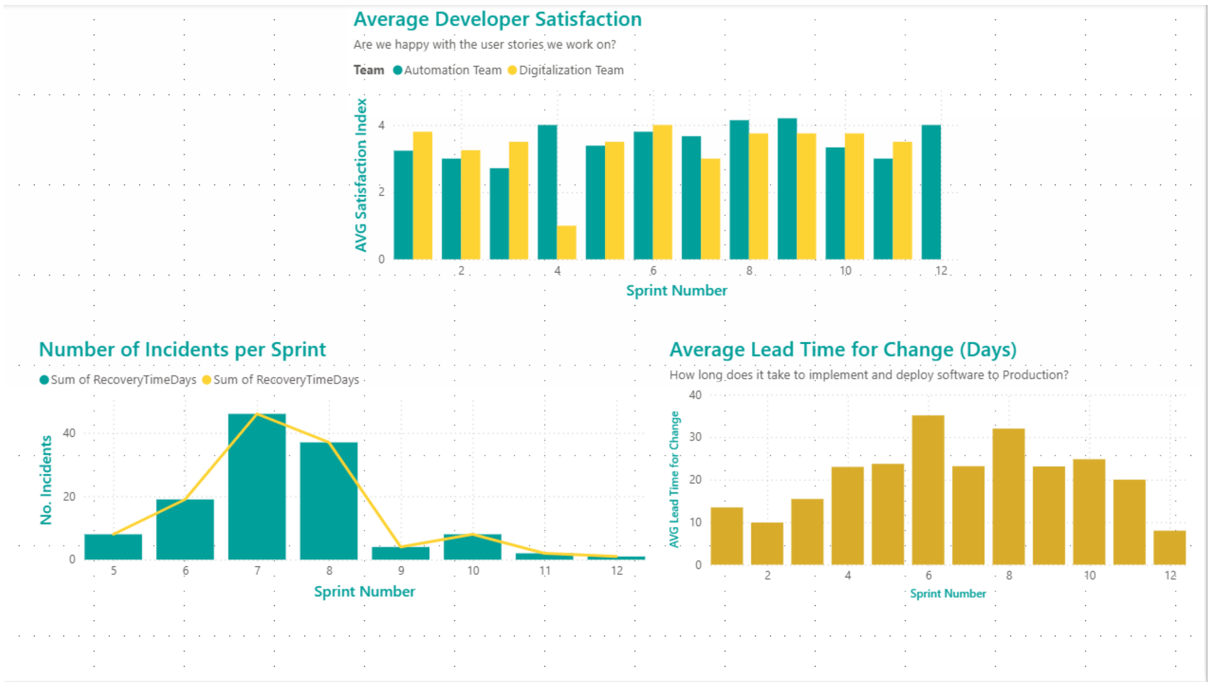

Average Developer Satisfaction
Are we happy with the user stories we work on?
Team ● Automation Team ● Digitalization Team
AVG Satisfaction Index
Sprint Number
Number of Incidents per Sprint
● Sum of RecoveryTimeDays ● Sum of RecoveryTimeDays
No. Incidents
Sprint Number
Average Lead Time for Change (Days)
How long does it take to implement and deploy software to Production?
AVG Lead Time for Change
Sprint Number

Figure B.11: Historical Overview of the Governance viewpoint

Bibliography

Alami, A. and Krancher, O. (2022). How scrum adds value to achieving software quality? *Empirical Software Engineering*, 27(7):165.

Ayyash, M. A. I. A. (2024). *Implementing Agile and DevOps at Scale: Identifying Best Frameworks, Practices, and Success Factors.* PhD thesis, Al-Quds University.

Basili, V., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, H., Seaman, C., and Trendowicz, A. (2007). Gqm strategies®: A comprehensive methodology for aligning business strategies with software. *MetriKon, Kaiserslautern*, pages 1–14.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). The agile manifesto.

Benaim, A. (2015). Innovation capabilities–measurement, assessment and development.

Berander, P. and Jönsson, P. (2006). A goal question metric based approach for efficient measurement framework definition. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 316–325.

Blinde, R. (2022). Devops unravelled: A study on the effects of practices and technologies on organisational performance.

Boon, G. C., Stettina, C. J., Visser, J., and El-Baz, Y. (2023). Beyond dashboards: Operationalising a measurement framework for agile teams. In *International Conference on the Quality of Information and Communications Technology*, pages 130–146. Springer.

Bouwers, E., van Deursen, A., and Visser, J. (2013). Evaluating usefulness of software metrics: an industrial experience report. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 921–930. IEEE.

Bouwers, E., Visser, J., and Van Deursen, A. (2012). Getting what you measure. *Communications of the ACM*, 55(7):54–59.

Brahimi, S., Aljulaud, A., Alsaiah, A., AlGuraibi, N., Alrubei, M., and Aljamaan, H. (2019). Performance dashboards for project management. In *International Conference on Computing*, pages 228–240. Springer.

Brickman, P. and Campbell, D. T. (1971). Hedonic relativism and planning the good society. *Adaptation level theory*, pages 287–301.

Bugwandeen, K. and Ungerer, M. (2019). Exploring the design of performance dashboards in relation to achieving organisational strategic goals. *South African Journal of Industrial Engineering*, 30(2):161–175.

Caldiera, V. R. B. G. and Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532.

Cavalcante, L. R. (2014). An analysis of the business enterprise research and development expenditures composition in brazilan analysis of the business enterprise research and development expenditures composition in brazil. *Revista Brasileira de Inovação*, 13(2):433–458.

Chiesa, V., Coughlan, P., and Voss, C. A. (1996). Development of a technical innovation audit. *Journal of Product Innovation Management: an international publication of the product development & management association*, 13(2):105–136.

Chiesa, V. and Frattini, F. (2009). *Evaluation and Performance Measurement of Research and Development: Techniques and Perspectives for Multi-level Analysis*. Edward Elgar Publishing.

Cho, Y. (2018). Assessing the r&d effectiveness and business performance: A review of their mechanisms and metrics. *STI Policy Review*, 9(1):1–29.

Cohn, S. (2013). A firm-level innovation management framework and assessment tool for increasing competitiveness. *Technology Innovation Management Review*, 3(10).

Conway, M. E. (1968). How do committees invent. *Datamation*, 14(4):28–31.

Davis, C. (2015). *Agile Metrics in Action: How to measure and improve team performance*. Simon and Schuster.

Davis, J. and Daniels, R. (2016). *Effective DevOps: building a culture of collaboration, affinity, and tooling at scale*. " O'Reilly Media, Inc.".

de Oliveira, A. R. and Proença, A. (2019). Design principles for research & development performance measurement systems: a systematic literature review. *Brazilian Journal of Operations & Production Management*, 16(2):227–240.

Digital.ai (2023). 17th state of agile report. `https://digital.ai/resource-center/analyst-reports/state-of-agile-report/`. Accessed: 2025-07-19.

Eckerson, W. W. (2010). *Performance dashboards: measuring, monitoring, and managing your business*. John Wiley & Sons.

Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., and Butler, J. (2021). The space of developer productivity: There's more to it than you think. *Queue*, 19(1):20–48.

Georgsson, B. M. (2024). *Design, development and adoption of a DevOps performance dashboard: Action Design Research at Sidekick Health*. PhD thesis.

Ghafoori, M. S. Z. (2025). *AI-Driven Business Performance Assessment: A Case Study*. PhD thesis, Technische Universität Wien.

Hauser, J. R. and Zettelmeyer, F. (1997). Metrics to evaluate r, d&e. *Research-Technology Management*, 40(4):32–38.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

Kaplan, R. S., Norton, D. P., et al. (2005). *The balanced scorecard: measures that drive performance*, volume 70. Harvard Business Review Boston, MA, USA.

Kerssens-van Drongelen, I. c. and Bilderbeek, J. (1999). R&d performance measurement: more than choosing a set of metrics. *R&D Management*, 29(1):35–46.

Kotusev, S. (2018). Togaf version 9.2: what's new. *British Computer Society (BCS)*, 21.

Kuusinen, K., Balakumar, V., Jepsen, S. C., Larsen, S. H., Lemqvist, T. A., Muric, A., Nielsen, A. Ø., and Vestergaard, O. (2018). A large agile organization on its journey towards devops. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 60–63. IEEE.

Laukkarinen, T., Kuusinen, K., and Mikkonen, T. (2018). Regulated software meets devops. *Information and Software Technology*, 97:176–178.

Leite, L., Rocha, C., Kon, F., Milojicic, D., and Meirelles, P. (2019). A survey of devops concepts and challenges. *ACM Computing Surveys (CSUR)*, 52(6):1–35.

Lindstrom, B. (2004). A software measurement case study using gqm. *J. Lund Univ., USA*.

Mezak, S. (2018). The origins of DevOps: What's in a name? `https://devops.com/the-origins-of-devops-whats-in-a-name/`. Accessed: 2025-07-19.

Offerman, T., Blinde, R., Stettina, C. J., and Visser, J. (2022). A study of adoption and effects of devops practices. In *2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference*, pages 1–9. IEEE.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Riihimäki, R. (2024). Productivity metrics and their integration into devops.

Rothwell, R. (1994). Towards the fifth-generation innovation process. *International marketing review*, 11(1):7–31.

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*, pages 328–338.

Samarawickrama, S. S. and Perera, I. (2017). Continuous scrum: A framework to enhance scrum with devops. In *2017 Seventeenth international conference on advances in ICT for emerging regions (ICTer)*, pages 1–7. IEEE.

Saunila, M. et al. (2014). Performance management through innovation capability in smes.

Sawhney, M., Wolcott, R. C., and Arroniz, I. (2006). The 12 different ways for companies to innovate. *MIT Sloan management review*.

Schumann Jr, P. A., Ransley, D. L., and Prestwood, D. C. (1995). Measuring r&d performance. *Research-Technology Management*, 38(3):45–54.

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21(1):1–38.

Skerlj, T. (2014). Measuring innovation excellence: Measurement framework for pwc's wheel of innovation excellence concept. In *Human Capital without Borders: Knowledge and Learning for Quality of Life; Proceedings of the Management, Knowledge and Learning International Conference 2014*, pages 221–229. ToKnowPress.

Smeds, J., Nybom, K., and Porres, I. (2015). Devops: a definition and perceived adoption impediments. In *International conference on agile software development*, pages 166–177. Springer.

Velasquez, N. F., Kim, G., Kersten, N., and Humble, J. (2014). State of devops report. *URL https://puppet. com/resources/report/2014-state-devops-report.*

Verwijs, C. and Russo, D. (2023). A theory of scrum team effectiveness. *ACM Transactions on Software Engineering and Methodology*, 32(3):1–51.

Wayland Project (2024). Wayland protocols governance document. `https://gitlab.freedesktop.org/wayland/wayland-protocols/-/blob/main/GOVERNANCE.md`. Accessed: 2025-05-15.