Sini Annamaa

CHATBOTS IN SOFTWARE RELEASE OPTIMIZATION: CASE STUDY

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2024

ABSTRACT

Annamaa, Sini
Chatbots in Software Release Optimization: Case Study
Jyväskylä: University of Jyväskylä, 2024, 84 pp.
Information Systems, Master's thesis
Supervisors: Marttiin, Pentti & Vakkala-Malinen, Suvi

The optimization of software release processes presents a challenge for today's organizations, especially those using Agile methodologies like the Scaled Agile Framework (SAFe). This thesis investigates the potential of Large Language Model (LLM)--based chatbots in enhancing software release processes within a case organization. Given the growing interest in AI tools such as LLM chatbots, this study investigates a relevant topic. However, the lack of peer-reviewed studies on chatbot applications in software release management highlights the importance of this study. This study aims to explore how an LLM-based chatbot can improve software release processes and identify the key features required for its effective implementation. The study begins with a literature review that examines software release processes through the perspective of Agile methods, release management, and chatbot functionality. The empirical part of the research was conducted within a case organization to determine the software requirements for an LLM-based chatbot. The findings indicate that LLM-based chatbots could optimize software release processes, as they can streamline tasks, automate repetitive actions, and enhance information sharing. However, successful implementation depends on addressing user needs and ensuring smooth integration with existing tools. By applying LLMs, organizations can adopt chatbot solutions without disturbing software release processes.

Keywords: agile, software release management, large language models, chatbot, case study

TIIVISTELMÄ

Annamaa, Sini
Chatbotit ohjelmistojen julkaisemisen optimoinnissa: Tapaustutkimus
Jyväskylä: Jyväskylän yliopisto, 2024, 84 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaajat: Marttiin, Pentti & Vakkala-Malinen, Suvi

Ohjelmistojen julkaisuprosessien optimointi on haasteellista nykypäivän organisaatioille, erityisesti niille, jotka käyttävät ketteriä menetelmiä, kuten Scaled Agile Frameworkia (SAFe). Tässä tutkielmassa tutkitaan suuriin kielimalleihin (LLM) perustuvien chatbottien potentiaalia ohjelmistojen julkaisuprosessien tehostamisessa case-organisaatiossa. Kun otetaan huomioon kasvava kiinnostus tekoälytyökaluja, kuten LLM-chatbotteja, kohtaan, tämä tutkimus tutkii ajankohtaista aihetta. Vertaisarvioitujen tutkimusten puute chatbottien soveltamisesta ohjelmistojen julkaisunhallinnassa korostaakin tämän tutkimuksen tärkeyttä. Tutkimuksen tavoitteena on tutkia, miten LLM-pohjainen chatbot voi parantaa ohjelmistojen julkaisuprosesseja, ja tunnistaa sen tehokkaan käyttöönoton edellyttämät keskeiset ominaisuudet. Tutkimus alkaa kirjallisuuskatsauksella, jossa tarkastellaan ohjelmistojen julkaisuprosesseja ketterien menetelmien, julkaisunhallinnan ja chatbotin toiminnallisuuden näkökulmasta. Tutkimuksen empiirinen osa suoritettiin case-organisaatiossa LLM-pohjaisen chatbotin ohjelmistovaatimusten määrittämiseksi. Tulokset osoittavat, että LLM-pohjaiset chatbotit voivat optimoida ohjelmistojulkaisuprosesseja, koska ne pystyvät virtaviivaistamaan tehtäviä, automatisoimaan toistuvia toimintoja ja tehostamaan tiedon jakamista. Onnistunut käyttöönotto riippuu kuitenkin käyttäjien tarpeiden huomioimisesta ja sujuvan integroinnin varmistamisesta olemassa olevien työkalujen kanssa. Soveltamalla LLM:iä organisaatiot voivat ottaa käyttöön chatbot-ratkaisuja häiritsemättä ohjelmistojen julkaisuprosesseja.

Asiasanat: agile, ohjelmistojen julkaisuhallinta, suuret kielimallit, chatbotti, tapaustutkimus

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

Artificial Intelligence (AI) is becoming common daily in today's ever-changing technological landscape. Abbass (2021, p. 94) states, "AI is everywhere, touching and blending with everything in our life, including all fields of science." This statement underlines the broad role of AI in transforming various areas of people's lives. The emergence of Generative Pre-trained Transformer (GPT) models has marked a milestone that has spurred the development and adaptation of generative AI tools across industries and academia (Stokel-Walker & Van Noorden, 2023).

Agile methodologies have transformed software development and increased the demand for faster and more efficient software delivery (Putta et al., 2018). Frameworks, such as the Scaled Agile Framework (SAFe), have embraced this demand and introduced new product delivery concepts that have adapted to the evolving practices (Scaled Agile Framework, 2024b). This shift has led to innovative approaches to software release and management, including introducing frameworks such as DevOps.

AI-based tools are playing an important role in software development. Large Language Models (LLMs), which are advanced AI-based language processing tools, have gained attention because they can be utilized in versatile ways from chatbots to task automation (Chang et al., 2024). For instance, LLMs are central to the design of chatbot systems that assist in improving collaboration and productivity (Cīrule & Bērziša, 2019).

This study aims to investigate the potential of an LLM-based chatbot for the optimization of software release processes. The study aims to understand how such chatbots can streamline workflow in both the development and release phases and to identify the software requirements for their effective implementation. The study answers the following research questions:

RQ1: How can an LLM chatbot optimize software release?

RQ2: What are the software requirements for an LLM chatbot in software release processes?

This study combines a literature review and a case study, conducted with the help of Design Science Research (DSR) and the Balanced Scorecard (BSC), to address the research questions. The study explores the potential of LLM chatbots and aims to provide insights into how software release processes can be optimized through such a chatbot in a case study organization.

The literature review follows the guidelines presented by Templier and Paré (2015), for guiding and evaluating literature reviews in the information systems literature, to ensure a thorough assessment of existing research. The review is used to synthesize insights from scientific articles, books, and conferences retrieved from databases such as IEEE Xplore, JYKDOK, Google Scholar, ScienceDirect, and Scopus. Terms used to search for the literature included *Agile*, *Scrum*, *Kanban*, *SAFe*, *LLMs*, *chatbot*, *software development*, *software release*, *DevOps,* and *ITIL*.

When gathering the literature, the number of references and their impact on the field has been considered. In addition, the literature must be relevant to the research and support answering the research questions. In total, hundreds of literatures were found from which 114 were selected for this thesis. The literature chosen for the review has been verified with the help of the Publication Forum, operating under the Federation of Finnish Learned Societies. The forum is a classification and rating system that supports the quality assessment of research findings. It uses a four-level classification, where the number zero indicates that the scientific journal does not meet the needed criteria and the number three is the highest level (Federation of Finnish Learned Societies, 2022). Selected literature must meet at least level one of the Publication Forum rating systems. This increases trust and ensures that literature has been chosen from high-quality peer-reviewed sources. However, this could have left out important information. The Publication Forum's rating is not absolute, as the rating does not consider, for example, the societal impact of the research being evaluated, leading to the possibility of exclusion of relevant literature. However, six pieces of gray literature have been included to address the limited number of peer-reviewed sources on this emerging topic, especially when opening the technologies behind chatbots.

The literature review aims to answer the first research question, which examines how LLM-based chatbots can optimize software releases. The results of this review suggest that chatbots can automate repetitive tasks, streamline information sharing, and facilitate team communication. They can also improve Continuous Delivery (CD) and DevOps processes through efficient software deployment and management.

The empirical study aims to answer the second research question with the help of the DSR methodology. This involves problem identification, definition of solution objectives, chatbot design and development, and collection and analysis of data. The BSC framework is used to assess the impact of the chatbot, focusing on functional requirements and non-functional requirements. Based on the results, the most valued features for the chatbot were information sharing and internal process optimization, whereas the least valued features were automation, learning, and financial aspects. These suggest that the primary need for a chatbot

is to serve as a reliable and integrative tool for information access and management, rather than as a new tool for instance for project management or task automation.

The structure of the study is as follows. After the introduction, the study's literature review is from Chapters 2 to 5. Chapter 2 consists of Agile and SAFe methodologies in software development. Chapter 3 consists of software release through release management, DevOps, and ITIL, as well as Agile Product delivery. The literature review in Chapter 4 is about Large Language Models, chatbots in software development and project management, and the best practices for chatbot design. Lastly, Chapter 5 summarizes the literature review.

Chapter 6 consists of the research setting. Based on the literature review chapters, this chapter presents the research questions. It also discusses the organization in which the case will be conducted. Lastly, this Chapter introduces the Design Science Research (DSR) methodology and the Balanced Scorecard (BSC) framework chosen for the study. Chapter 7 presents the case study by following the DSR phases. The structure of the Chapter is the following: problem identification, solution objectives, design and development, data collection, data analysis, validity, and reliability, as well as results and evaluation.

Chapter 8 is the discussion part of the study. In this, the research questions are brought up again and the hypothesis related to them. It will also give implications and future research possibilities. Lastly in the Chapter, the limitations of the study will be discussed. Chapter 9 concludes the thesis. It wraps up the research and brings up the most important notes. This is done by summarizing the study and its findings.

2 AGILE SOFTWARE DEVELOPMENT

This chapter gives an overview of Agile software development. It introduces Agile methodology and its subsets Scrum, Kanban, and SAFe. The chapter concludes with an overview of challenges related to Agile software development.

2.1 Agile Methodology

Agile methodology is a software development and project management approach. It is based on the Agile Manifesto introduced by Beck and others (2001). It was developed as a response to the limitations of the Waterfall model, which struggled with the dynamic and iterative nature of the latest software development (Thesing et al., 2021). Through iterative development and team collaboration, Agile emphasizes adaptability to evolving customer needs.

Agile methodologies are impactful. Due to their flexibility, adaptiveness, customer-centricity, and emphasis on continuous improvement, they can enhance software projects (Aouni et al., 2024). The core values of the Agile Manifesto emphasize individuals and interactions over processes and tools and working software over comprehensive documentation (Figure 1). Šmite and others (2021) suggest that the values enable teams to address problems quickly while promoting creativity and maintaining a user-centric approach. However, the effectiveness of Agile methods depends on their implementation, as challenges such as customer engagement and team coordination can weaken its principles (Thesing et al., 2021).

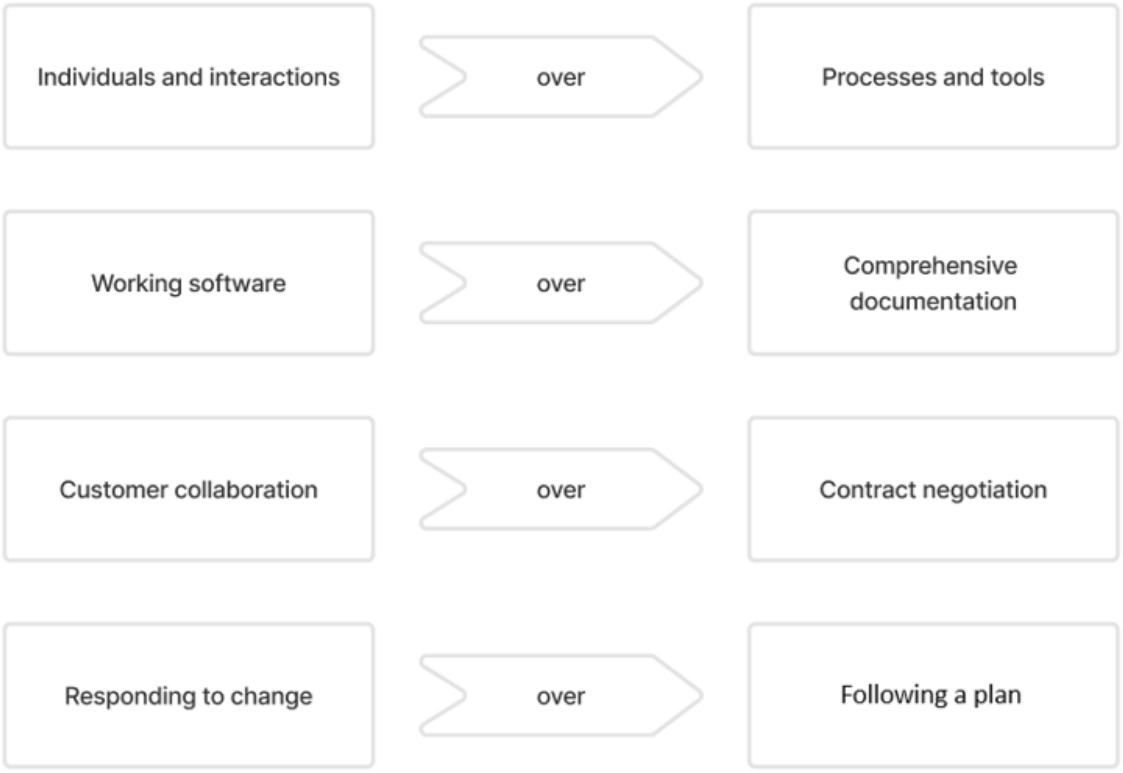


FIGURE 1 The Values of Agile Manifesto (adapted from Beck et al., 2001)

In addition, the Agile Manifesto introduced the Twelve Principles of Agile Software Development (Beck et al., 2001). The principles were designed to guide teams in their daily tasks. They provide a framework for the operationalization of Agile values and the promotion of a people-centric and flexible approach to software environments.

Table 1 presents the principles and their descriptions. The principles highlight the importance of iterative processes, sustainable development practices, and customer satisfaction. For instance, the second principle of welcoming changing requirements even in the later stages of development illustrates the flexibility that characterizes Agile as distinct from traditional methods, like Waterfall. Even though these principles are widely accepted, their implementation is not without difficulties. There has been criticism that some principles can be difficult to implement in remote or distributed teams (Šmite et al., 2021). Furthermore, the tenth principle can require a significant cultural shift, which can cause resistance in organizations moving to Agile ways of working.

TABLE 1 The Twelve Principles of Agile Software Development (adapted from Beck et al., 2001)

Principle Number	Principle Description by Beck and others (2001)
Principle 1	Customer satisfaction through early and continuous delivery of valuable software.
Principle 2	Welcome changing requirements, even when they are late in development. Agile processes harness change for the customer's competitive advantage.
Principle 3	Deliver working software frequently, with a preference for shorter timescales.
Principle 4	Collaboration between businesspeople and developers is key throughout the project.
Principle 5	Build projects around motivated individuals, providing them with the support they need and trusting them to get the job done.
Principle 6	Face-to-face conversation is the most effective form of communication.
Principle 7	Working software is the primary measure of progress.
Principle 8	Sustainable development, where developers, sponsors, and users should be able to maintain a constant pace indefinitely.
Principle 9	Continuous attention to technical excellence and good design enhances agility.
Principle 10	Simplicity, the art of maximizing the amount of work not done—is essential.
Principle 11	The best architecture, requirements, and designs emerge from self-organizing teams.
Principle 12	At regular intervals, the team reflects on how to become more effective and adjusts its behavior accordingly.

Agile development follows an iterative cycle. In it, projects are divided into smaller increments or sprints, lasting anywhere from one to four weeks (Fagarasan et al., 2021). During each sprint, development teams focus on delivering functional software, allowing frequent adaption and evaluation (Rodríguez et al., 2019). Also, Fagarasan and others (2021) highlight that it also allows teams to identify and resolve bugs or usability issues early in the development process, contributing to producing higher-quality software that meets user needs. Agile also promotes better communication, coordination, and decision-making within development teams through its focus on cross-functional collaboration and regular internal reviews (Schmidt, 2016). Moreover, Rodríguez and others (2019) point out Agile's adaptability, which makes it particularly suited to fast-paced environments where requirements evolve frequently.

Still, the approach does not come without its challenges. Gandomani and Nafchi (2016), point out that the simultaneous workflow can lead to duplication of tasks or mismatches between teams, weakening the collaboration and capability that Agile aims to foster. Also, the lack of definitive deadlines can delay project delivery and disrupt the expectations of stakeholders (Schmidt, 2016). The challenges suggest that even if the Agile approach facilitates collaboration and responsiveness, it requires careful management to avoid any inability.

Despite the challenges, Agile has not lost its popularity in the field as it meets the needs of today's software development. It supports iterative workflow, stakeholder collaboration, and adaptability, which are essential in the management of dynamic project environments. This makes Agile an ideal candidate for integration with tools such as AI-based chatbots that can improve communication and task coordination in iterative workflows.

The flexibility of Agile has also led to the development of specialized subsets, designed to meet specific needs of project management. These methods and scalable Agile frame models are extreme programming (XP), Scrum of Scrums and Large Scale Scrum (LESS). Nowadays the most used are Scrum, Kanban, and the Scaled Agile Framework (SAFe). These methodologies, which retain the core values and principles of Agile, are discussed in the following sections.

2.1.1 Scrum

The first Agile method to be looked at in this study is Scrum, a management framework designed to improve the effectiveness of teams in delivering projects. Originally introduced in 1995, it provides both a structured and flexible approach to managing complex problems through defined roles, artifacts, and ceremonies (Schwaber & Sutherland, 2020). Scrum emphasizes processes where decisions are guided by experimentation, observation, and learning from experience (Sachdeva, 2016).

Scrum relies on self-organizing teams. To achieve these teams, it follows defined core elements, described in the Scrum Guide (Schwaber & Sutherland, 2020). Figure 2 illustrates the framework of Scrum, showcasing its incremental nature that supports continuous development and delivery.

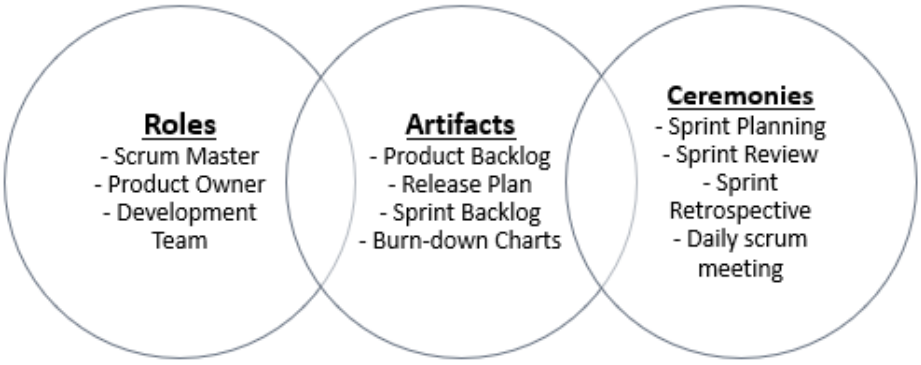


FIGURE 2 The Framework of Scrum (adapted from Sachdeva, 2016, p. 16793)

Scrum outlines three primary roles: Scrum Master, Product Owner, and Developer (Figure 3). These roles are coequal and responsible for delivering high-quality, phased solutions within the Scrum Framework. The Scrum Master facilitates and coaches the team, ensuring it implements and follows the Scrum framework properly (Nilsson Tengstrand et al., 2021). The Product Owner acts as a bridge between the stakeholders and the development team by defining the product vision, gathering, and prioritizing requirements, and managing the product

backlog (Berntzen et al., 2019). Developers are part of a cross-functional and self-organized team, with all the skills needed to create and deliver a product, which plans, implements, and evaluates work to achieve goals and ensure continuous improvement (Sachdeva, 2016). These roles ensure that Scrum teams work cohesively but independently while balancing project vision, technical execution, and team collaboration (Nilsson Tengstrand et al., 2021). When the responsibilities are defined clearly, Scrum allows flexibility and accountability needed for iterative development and continuous delivery.

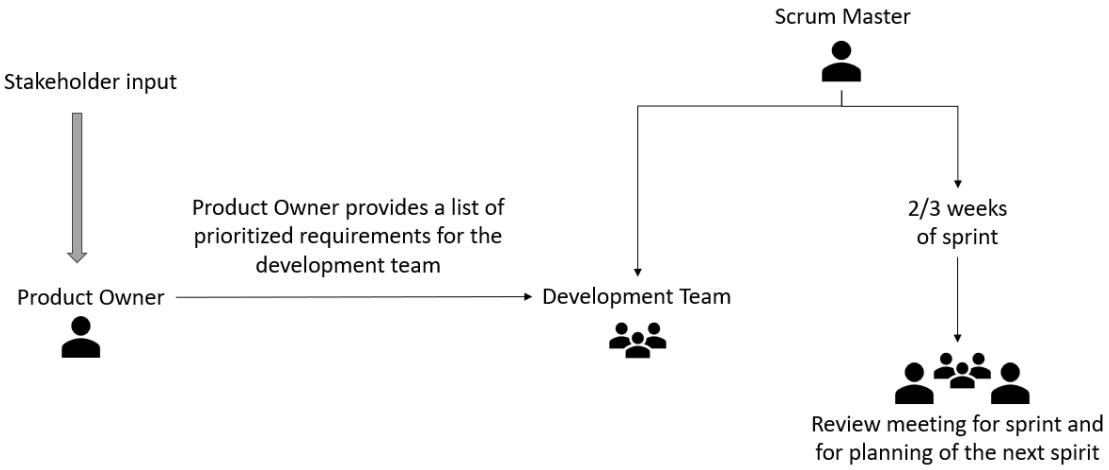


FIGURE 3 The Roles and Their Relationships in Scrum (adapted from Sachdeva, 2016, p. 16794)

Scrum contains four key artifacts that guide the development process and enhance team collaboration: product backlog, release plan, sprint backlog, and burn-down charts. Product Backlog is the list of requirements for the developed product in a user story format. The Release Plan describes the release goal, the item with the highest priority, the risks, and the functionality and overall features of the product. The content of the Sprint Backlog comes from the Sprint Planning Meetings, and it is both owned and modified by the team. It includes the tasks for each Sprint, as well as estimated and assigned tasks. Lastly, Burn-down Charts are visual artifacts to motivate the team. It is often a graph showing the amount of remaining work from the Sprint Backlog. (Jacobson et al., 2022).

Scrum also incorporates four core ceremonies that structure the iterative sprint process: sprint planning, daily scrum, sprint review, and sprint retrospective. Sprint planning is a meeting where the iteration is planned by first determining the aim of the Sprint and how the product increment will be built. Sprint review is also a meeting where the development teams present and discuss the work done in the sprint. Sprint retrospective is a discussion where the whole team evaluates the finished work. In daily scrum meetings, each team member explains their tasks for the day and if any obstacles are stopping them from executing them. (Sachdeva, 2016).

Scrum offers several benefits for software release processes. Paasivaara and others (2009) note its ability to increase transparency in distributed projects,

early problem detection, promote the sharing of knowledge, and encourage informal communication between teams. These benefits are useful in Agile software development, where adaptability and responsiveness are critical.

However, Scrum also has its downsides. For instance, Marchenko and Abrahamsson (2008) point out that the excessive focus on Scrum practices by Scrum Masters can decrease team productivity. In addition, unclear management and expectations can weaken the key principles of Scrum, communication, and transparency (Paasivaara et al., 2009). Marchenko and Abrahamsson (2008) also bring up the risks of excessive overly individualistic and managerial interference, which can compromise the autonomy of self-organized teams and disrupt the iterative feedback loop that is central to Scrum's success.

Scrum provides an iterative and collaborative framework for software development, facilitating cross-functional collaboration between teams. However, as the literature suggests, optimizing the implementation of Scrum could improve the effectiveness of managing software release processes.

2.1.2 Kanban

Kanban, the second Agile method presented in this study, offers a simpler and more visual approach when compared to Scrum. It is widely used in DevOps and Agile software development, described in later chapters. Kanban was originally developed by Toyota in the 1940s and was applied to software development in 2004, providing a versatile framework for improving and managing team productivity (Wakode et al., 2015).

The core of Kanban is the Kanban board (Figure 4), a visual tool showing work items and their progression through workflow. The board usually consists of defined columns representing the different stages of the workflow, like backlog, test and released. Kniberg and Skarin (2010) describe that the visual representation allows team members to keep track of the current tasks and facilitate smooth transitions between steps. Also, Kanban supports the automation of workflow by providing a clear overview of both incomplete and completed tasks (Wakode et al., 2015).

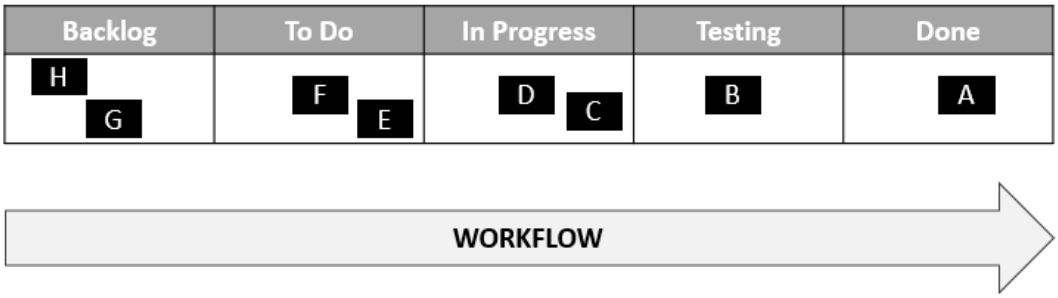


FIGURE 4 The Kanban Board (adapted from Kniberg & Skarin, 2010, p. 42)

Similarly to Agile, Kanban is based on six key practices: visualizing workflow, limiting work in progress, measuring and managing flow, clarifying process practices, introducing feedback loops, and promoting collaborative

improvement (Ahmad et al., 2013). These enable on-demand delivery, where tasks are completed on demand, in distinction to time-limited approaches like Scrum, where work schedules are determined by predefined sprints and events (Anderson, 2010).

In addition, Kanban also follows four guiding principles that correspond to the philosophy of the Agile Manifesto. The first principle, starting with what you are doing, emphasizes identifying and leveraging existing processes and identifying areas for improvement. The second principle encourages incremental and evolutionary improvement, which aligns with the emphasis on continuous development and adaptation in Agile. The third principle is to respect existing roles, responsibilities, and processes, thereby facilitating incremental and seamless improvements. Lastly, Kanban encourages leadership at all levels, emphasizing the importance of ownership in organizations. (Ahmad et al., 2013).

Kanban offers several advantages. Visualization through tools, such as the Kanban board, gives stakeholders and development teams a clearer understanding of task progress and workflows. Dennehy and Conboy (2017) highlight that this visual clarity increases productivity and facilitates problem-solving, improving communication and coordination between teams and stakeholders. Also, Kanban's independence from heavy documentation allows frequent iteration and rapid implementation of changes, making it well-suited for dynamic development environments (Alaidaros et al., 2018).

Nevertheless, Kanban also faces challenges, in its limited scope as an independent framework. Ahmad and others (2013) indicate that Kanban does not inherently support project outputs but requires complementary practices or frameworks to ensure successful project management. This places Kanban as a workflow tracking tool, rather than a comprehensive project management method. Similarly, Alaidaros and others (2018) argue that Kanban lacks the needed mechanism for detailed progress monitoring, further supporting its role as a complementary rather than an independent method.

Overall, Kanban executes well for visually representing workflows and facilitating coordination between teams. Its flexibility and focus on continuous improvement make it a valuable tool for improving productivity and responsiveness. Still, Kanban's reliance on complementary frameworks points out its role as a support method rather than a complete project management solution. This reflects its primary strengths to visualize and optimize processes in Agile methodologies.

2.1.3 Scaled Agile Framework

The last subset of Agile discussed in this thesis is the Scaled Agile Framework (SAFe). It is designed primarily for large organizations to effectively scale Agile practices across the enterprise while maintaining responsiveness and flexibility, the core principles of Agile (Putta et al., 2018). According to Alqudah and Razali (2016), SAFe emphasizes collaboration, alignment, and delivery at all levels of the organization, and addresses the complexity of cross-functional projects without having to compromise Agile's core values.

SAFe is built on seven core competencies that organizations use to deliver customer-centric, cross-organizational solutions. The competencies are Lean Portfolio Management, Organizational Agility, Continuous Learning Culture, Lean-Agile Leadership, Agile Product Delivery, and Enterprise Solution Delivery. These competencies are illustrated and described in Figure 5, to highlight their role in applying SAFe to organizations. As Putta and others (2018) brought up, customer centricity is the fundamental element on which all SAFe competencies are based.

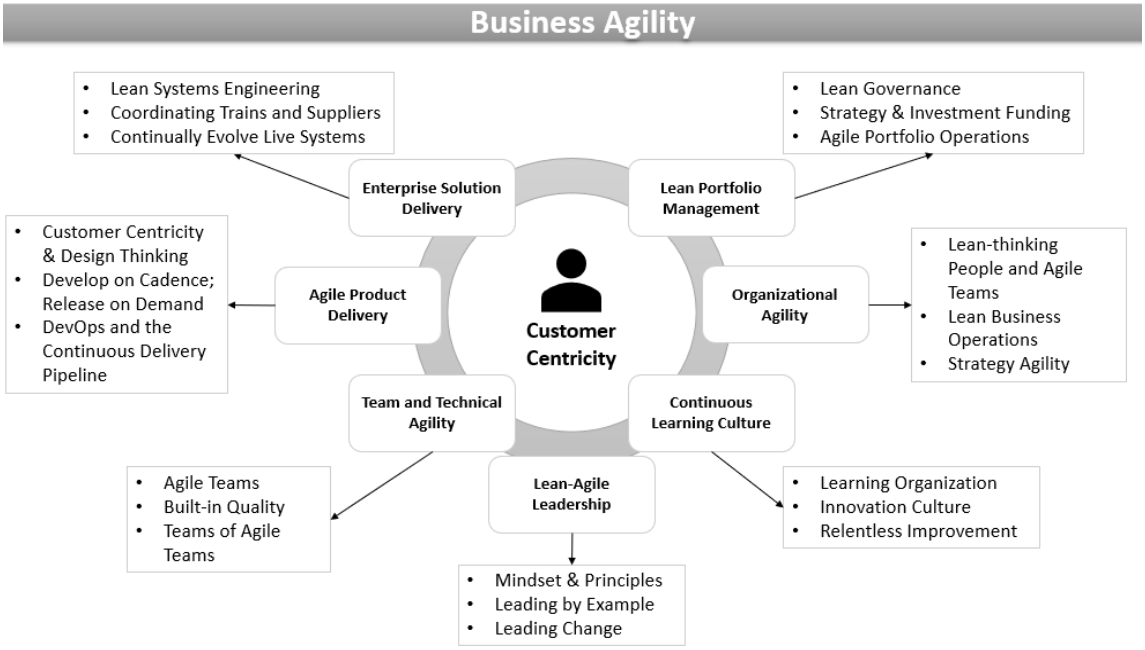


FIGURE 5 Core Competencies of SAFe 6.0 Framework (adapted from Scaled Agile Framework, 2024b)

In addition to the competencies driving organizational agility, SAFe includes layers to ensure the scalability of the framework at different organizational levels: team, program, large solution, and portfolio. Nilsson Tengstrand and others (2021) mention that these layers are crucial to adapting and scaling Agile practices to the needs of large enterprises, ensuring the alignment and facilitation of collaboration at each level. The roles associated with each layer are illustrated in Figure 6 and discussed in detail below.

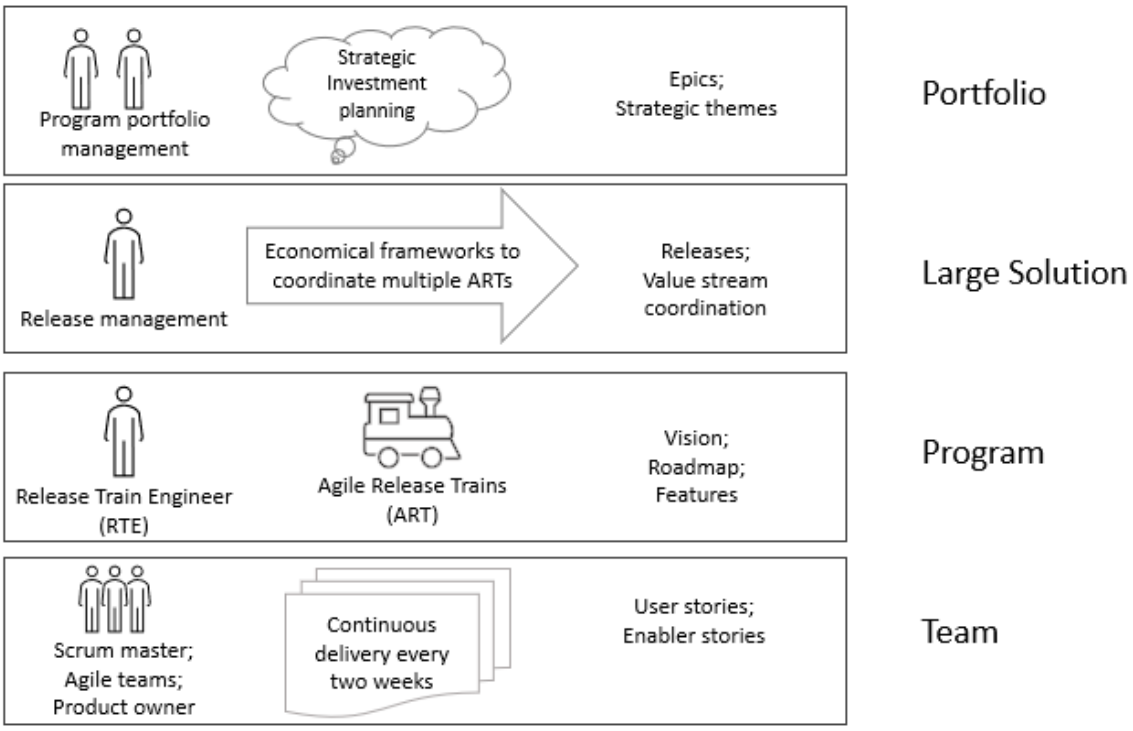


FIGURE 6 The Levels, Roles, and Activities in SAFe (adapted from Nilsson Tengstrand et al., 2021, p. 159)

At the Team level, development teams apply Agile practices through frameworks such as Kanban and Scrum. The teams use the frameworks to manage tasks and workflow and follow the roles and processes defined in the chosen methodology. (Duncan, 2018). In Figure 6 the illustrated team level gives an example of a development team following Scrum.

The program level introduces the Agile Release Train (ART). It is a cross-functional team where development teams work together on the same solution. ART teams share a common vision, roadmap, and a set of capabilities. A key role at the program level is played by the Release Train Engineer (RTE), who oversees the progress of ART. They facilitate collaboration, manage risks, and ensure that ART remains aligned with business objectives. (Nilsson Tengstrand et al., 2021).

At the Large Solution level, the focus is on managing and coordinating multiple ART activities and value streams. The level often relates to release management, where the Solution Train Engineer (STE) synchronizes the work of ARTs to ensure team alignment. The role of the STE is to manage dependencies and maintain cohesion between ARTs using a financial framework to prioritize initiatives and deliver value efficiently. (Duncan, 2018).

The portfolio level ensures the alignment of Agile and strategic objectives. Program Portfolio Management (PPM) is responsible for monitoring resource allocation and budget management to achieve business objectives. PPM also ensures that the work of ARTs and development teams is aligned with the broader organizational strategy, optimizing the value delivered to the business. (Nilsson Tengstrand et al., 2021).

SAFe uses a variety of Agile practices, such as iterative development and continuous delivery. Theobald and Schmitt (2020) note that SAFe places a strong emphasis on continuous delivery, but more on a structured and controlled approach. One of SAFe's key events is Program Increment (PI) Planning occurring every 8-12 weeks, in which all teams of each ART work together to define goals and identify dependencies, ensuring that the teams are aligned with the shared goals before moving forward (Putta et al., 2019). This is a crucial event to better foster shared understanding and coordinate efforts across teams.

SAFe is particularly useful for large enterprises because it allows Agile to be scaled at multiple levels. According to Putta and others (2018), it enables alignment between business units and development teams, facilitating collaboration across organizational levels, improving risk management, and speeding time to market. This makes SAFe especially suitable for larger organizations, needing to scale Agile beyond the team level.

However, the implementation of SAFe does not come without its own challenges. Dikert and others (2016) highlight that the implementation of SAFe requires significant cultural and organizational changes, often needing the allocation of resources and commitment of management. This change can lead to additional costs and potential resistance at organizations' different levels, hindering the implementation. As in any Agile framework, employees are the cornerstone of SAFe's success, making their engagement critical to overcoming these challenges.

In summary, SAFe provides a comprehensive and structured approach to scaling Agile practices in larger organizations. Its emphasis on alignment, continuous delivery, and communication across organizational levels makes it a valuable methodology for organizations looking to scale Agile practices into large enterprises. While its implementation may require significant resources and face cultural resistance, SAFe holds its place as one of the most used Agile methods in today's software development.

2.2 Challenges

Scaling Agile practices in organizations poses its own challenges. Research suggests that success in scaling Agile practices depends on effective coordination and communication at different levels of the organization (Putta et al., 2018; Theobald & Schmitt, 2020). Without frequent updates and established communication channels, bottlenecks can occur that hinder the pipeline of software development.

Resistance to change is one of the most common challenges when adopting Agile practices. The time-consuming nature of adapting to a new set of processes can cause resistance, particularly if employees do not agree with the proposed changes (Kalenda et al., 2018). Resistance can occur at any organizational level, but it is particularly complex at higher levels with the decision-making power.

Moe (2013) brings up that resistance at this higher level can create barriers when moving to Agile methods, further leading to delays or disruptions.

From a psychological perspective, the challenges of adopting Agile methods stem from the strong emphasis on openness. While Agile transparency promotes collaboration and accountability, some employees may view it as a tool to control or criticize their performance. Conboy and others (2011) identified a fear among developers of transparency, implying that some may perceive it as a threat to their professional competence. Similarly, Kalenda and others (2018) argue that relying on Agile development in self-organized teams and increased individual accountability can overwhelm some employees, especially if responsibilities accumulate over time.

Agile methods also face obstacles in decentralized working environments, which are common in today's working life after the impact of COVID-19. Studies show that the team members' physical separation can erode interpersonal relationships, reducing team cohesion and hindering effective communication (Paasivaara et al., 2013; Vallon et al., 2013). The lack of face-to-face interaction can make it difficult for the flow of information to the right stakeholders, weakening the transparency on which Agile methods rely.

Quality assurance (QA) can also face problems when adopting Agile practices. Vallon and others (2013) describe a case where the introduction of Agile quality led to the degradation of software quality. In this case, increased workload, and pressure to meet deadlines were the main identified factors contributing to the negligence of testing and debugging, resulting in technical debt. Vallon and others (2013) also brought up that in extreme cases, teams have falsified reports to cover up neglected QA practices, such as untested features or uncorrected tests. Such failures in QA not only compromise the stability of the production environment but can also weaken the confidence of customers, potentially damaging the organization's reputation.

Overall, Agile software development has become essential due to its way of enabling adaptation to changing requirements and economic conditions, while promoting collaboration and openness. Its challenges should not be underestimated. Agile development is not a one-size-fits-all solution and it requires careful planning and risk mitigation to overcome its challenges. As software delivery has been accelerated through Agile development, customers are increasingly expecting updates and value-based releases. To meet this rising demand, organizations have adopted software release management practices to complement Agile methodologies.

3 SOFTWARE RELEASE MANAGEMENT

The objective of software release management is to ensure stability and quality. Sometimes this objective cannot be achieved at the required level, thus creating situations where assistive tools are needed, hence providing a footing for LLMs chatbot incorporation into software release. This chapter introduces and defines software release management and its supporting frameworks, DevOps and ITIL. The chapter aims to establish a context for release management within Agile software development through Agile product delivery.

3.1 Overview of Release Processes

Software release management, often called release management in short, is a technique used to plan, manage, and control software releases through various phases. It aims to improve the speed, quality, and efficiency of software delivery, thereby increasing the likelihood of a successful and stable release (Heikkilä et al., 2017). This method is essential for balancing the need for fast deliveries in complex environments where multiple teams work on the same solution (Samer, 2016).

Release management can be executed through different methodologies, the most famous ones being Agile or Waterfall. Each method represents its own philosophy and approach to managing software release processes. The Agile method promotes adaptability and flexibility through its incremental processes, allowing both iterative development and continuous delivery of software (Thesing et al., 2021). The Waterfall method promotes sequential and linear approaches where each phase must be completed before the next one begins, making the method effective for projects with static and well-defined requirements (Schaefer et al., 2012). Due to its rigid nature, the Waterfall method is less adaptable to changes and is not suitable for projects with changing requirements, which is why Agile is preferred over it in today's software release management (Thesing et al., 2021).

Still, it is good to note that the original Waterfall model by Royce (1970) was not strictly linear, as it included feedback mechanism allowing earlier steps to be revisited. This is often an overlooked aspect in discussions where the method is considered rigid. Also, McConnell (1996) describes several methods designed for different project contexts. These include strategies, which divide systems according to priorities or requirements, and iterative approaches, allowing refinement of steps early on. Yet, the comparison between Agile and Waterfall model is still common and often based on oversimplified comparisons. Such descriptions can easily ignore the flexibility of the model and overstate the current dominance of Agility. By recognizing these organizations can choose the best approach suiting best their needs.

Agile release management is guided through an iterative process cycle, ensuring the delivery of quality software by continuous improvement. This cycle consists of the following steps: planning, building, testing, preparation, and deploying, described in detail in Table 2. In these stages, Agile release management prioritizes documentation, coordination, and repeatability to ensure the efficiency and streamlining of the software release processes (Kajko-Mattsson & Yulong, 2005).

TABLE 2 Release Management Process Cycle (adapted from Kajko-Mattsson & Yulong, 2005, p. 851)

Phase	Objective
1. Planning	Scope definition, stakeholder alignment, and risk management.
2. Building	Version control, Continuous Integration (CI), and quality control.
3. Testing	Testing strategies and automation.
4. Preparing	Documentation and readiness, and environment setup.
5. Deploying	Deployment models, and monitoring and post-release testing.

Release management ensures high-quality and stable software releases through a structured approach, avoiding disruptions in the production environment. It aims to minimize the risks associated with errors and instability and ensure a smooth experience for end-users (Heikkilä et al., 2017). A successful release is one in which end-users do not encounter difficulties, strengthening the software's trust. This structured approach seeks to optimize the efficiency of software delivery by managing resources, avoiding delays, and standardizing processes, further improving consistency (Schaefer et al., 2012). With Agile values, release management plays an important role in today's software development practices.

Agile release management leverages DevOps practices, especially continuous integration (CI) and continuous delivery (CD), in supporting frequent and reliable releases. CI automates code changes from multiple sources into a single software (Ståhl & Bosch, 2014; Virmani, 2015). CD builds, tests, and deploys software in smaller cycles, improving the quality of software (Samer, 2016; Zhao et al., 2017). Together, these practices create the CI/CD pipeline, which is critical for Agile release management. This pipeline automates repetitive tasks, such as

testing, application orchestration, and deployment, ensuring seamless coordination of quality control and delivery (Virmani, 2015). In the context of Agile, the CI/CD pipeline bridges the gap between operations and development, contributing to faster and safer releases (Samer, 2016).

Dependency management is a critical challenge in Agile release management. As software is often dependent on numerous dependencies, release managers must coordinate them to ensure the systems stay operational even after the release (Heikkilä et al., 2017). Studies have identified that a single application can have several dependencies (Dietrich et al., 2022; Zhao et al., 2017), underlining the complexity of ensuring compatibility and stability. The challenge of dependencies is further emphasized when teams collaborate on the same solution. Effective coordination requires alignment of development and release schedules, which need to handle consecutive or simultaneous releases and prevent disturbances in services (Heikkilä et al., 2017). Weak coordination can lead to system failures, further affecting negatively user experience.

Release management plays an essential role in delivering reliable, high-quality, and secure software in Agile environments. Through its structured processes, it ensures stability while considering both technical and business requirements. Still, the challenges of maintaining speed and quality highlight the need for supporting tools and strategies to help in managing these complexities. Next, two widely used frameworks supporting software release management, DevOps and ITIL, are introduced.

3.1.1 DevOps

DevOps is a philosophy and practice, which integrates both software development and IT functions into a cohesive, automated, and collaborative workflow (Ebert et al., 2016). The term DevOps is a combination of words development and operations, reflecting the integration of these two disciplines. Through its emphasis on team empowerment, automation, and cross-functional communication, DevOps has become an ideal framework for managing releases in Agile environments (Banica et al., 2017).

DevOps follows four principles, collaboration and communication, automation, continuous integration (CI), and continuous delivery (CD), as well as monitoring and feedback, which enhance its effectiveness and adaptability in Agile contexts. Collaboration and communication facilitate a cross-functional culture between operations and development teams, promoting shared responsibility for product quality and performance. Automation acts as the cornerstone of DevOps, driving faster processes for development and deployment, minimizing human errors, and improving consistency. CI ensures early detection of integration problems through frequent integration of code into a shared repository, while CD automates the deployment and supports iterative releases. Monitoring and feedback provide actionable feedback and enable proactive problem identification and resolution, ensuring software performance and reliability. Together these four principles promote faster and quality releases while focusing on continuous improvement and responsiveness. (Ebert et al., 2016).

DevOps lifecycle consists of eight phases: planning, coding, building, testing, releasing, deploying, operating, and monitoring. These phases ensure a smooth flow from the design of the software to the end-user experience. The objectives of each phase are presented in Table 3. According to Alnafessah and others (2021), this lifecycle is similar to the traditional software development lifecycle but differs from it with its highly automated and continuous release process.

TABLE 3 DevOps Lifecycle (adapted from Alnafessah et al., 2021, p. 3)

Phase	Objective
1. Planning	Scoping out new features and functions based on user feedback and cases.
2. Coding	Coding and building new features on user stories and work items in the backlog.
3. Building	The new code is integrated into the existing code, then tested and packaged for release and deployment.
4. Testing	Ensuring that the new application meets the set of standards and requirements.
5. Releasing	Runtime to deploy, check quality and compliance, and run security tests.
6. Deploying	The solution is deployed to the production environment for end-users to access.
7. Operating	Monitoring performance, behavior, and availability of the features. This ensures that the feature provides value for end-users.
8. Monitoring	Feedback gathering from stakeholders on functions, features, performance, and business value.

DevOps offers benefits to organizations, especially in the context of Agile release management and development. DevOps promotes shorter cycles of releases, supporting frequent deliveries that meet the needed requirements (Faustino et al., 2022). Also, the use of CI/CD pipelines ensures that the code changes are tested and functional, improving the overall reliability of the software (Ståhl & Bosch, 2014). DevOps improves efficiency and productivity through its enhancement of closer communication between development and operations teams (Aouni et al., 2024). Through its proactive approach to ensure compliance with standards, DevOps minimizes vulnerabilities and reduces risk in the early stages (Faustino et al., 2022). Through its automation, it improves resource allocation and scalability by reducing manual workload and saving both time and money (Aouni et al., 2024).

On the other hand, DevOps comes with its own challenges. Switching to DevOps can be challenging for teams who are not used to collaborative and cross-functional workflows (Leite et al., 2019). To enable effective DevOps practices, organizations may need to invest in new tools, automation technologies, and cloud platforms, which can be both costly and time-consuming (Shahin et al., 2016). To effectively use DevOps, teams should update their set of skills, which require additional resources from the organization (Leite et al., 2019; Shahin et al., 2016).

In conclusion, DevOps meets today's technological environment. While there are challenges that need to be considered, such as costs, infrastructure, and organizational changes, the benefits can be seen as outweighing the challenges. In release management, DevOps enables teams to adopt Agile approaches. In the following section, ITIL is being looked at from the point of view of improving IT services and supporting release management.

3.1.2 ITIL

ITIL (Information Technology Infrastructure Library) is a set of practices for IT service management (ITSM). It was developed by the Central Computing and Telecommunications Agency (CCTA) in the 1980s. Its most recent iteration is ITIL 4, released in 2019 (Obwegeser et al., 2019). ITIL aligns IT services with organizations' business objectives.

Compared to the previous iterations, ITIL 4 integrates with current approaches, such as Agile and DevOps. It adopts an iterative, holistic, and value-driven approach to ITSM, making it more suitable for fast-paced Agile environments, allowing organizations to respond effectively to customer needs and pressures from competition (Agutter, 2020).

ITIL 4 organizes ITSM with the help of four dimensions, forming a balanced view. The first dimension is organizations and people, which focuses on the competence, roles, responsibilities, and culture needed for value deliverance. The second dimension is information and technology, emphasizing the data, technologies, and tools required to support services. The third is the dimension of partners and suppliers, which underlines the dependencies and relationships between external entities and organizations involved in delivering services. The last dimension is value streams and processes, focusing on processes, activities, and workflows creating and delivering value. (Obwegeser et al., 2019).

ITIL provides a structured framework, offering benefits for the ITSM. It provides practices that enable reliable and predictable IT delivery. It also provides risk management strategies, reducing the possibility of disruptions in services and overall improving system performance. ITIL improves user satisfaction and service relevance through the alignment of IT services with both business and customer needs. In addition, it helps organizations to reduce operational costs without having to compromise on quality while keeping itself adaptable to changing requirements. (Marrone & Kolbe, 2010).

ITIL provides a structured approach to managing IT services. With ITIL 4, it has become relevant for organizations due to its integration of Agile principles. As new iterations are formed, the relevance of ITIL can be expected to grow and further provide organizations with tools to deliver valuable IT services. The next section provides details about Agile Product Delivery and its relevance to software release management.

3.2 Agile Product Delivery

Agile Product Delivery (ADP) is one of the core competencies of SAFe, employing efficient, fast, and incremental delivery. It ensures that development teams can respond effectively to feedback, adapt to changes, and deliver constant value (Knaster & Leffingwell, 2020). Similarly to Agile, ADP emphasizes customer satisfaction through frequent software delivery.

ADP consists of three key dimensions: customer centricity and design thinking, development on cadence and release on demand, and DevOps and continuous delivery pipeline (CDP), where each dimension plays a critical part in ensuring successful product delivery. A detailed description of each dimension can be found in Table 4. These dimensions ensure that organizations can deliver quality products while maintaining adaptability to market needs (Cvejič, 2022).

TABLE 4 Three Dimensions of Agile Product Delivery (adapted from Scaled Agile Framework, 2024a)

Dimension	Description
Customer Centricity and Design Thinking	Aims to put customers at the center of every decision. By applying design thinking, it ensures that the solution is sustainable, feasible, and viable.
Development on Cadence, Release on Demand	Helps manage product development by releasing so that customers get what they want when they need it.
DevOps and the Continuous Delivery Pipeline (CDP)	Creates the foundation, enabling organizations to release at any time to meet market and customer demand.

ADP provides several benefits that align with today's software delivery practices. It helps in reducing the time to market through shorter release cycles, further allowing organizations to deliver features and updates fast. Loops of regular customer feedback ensure that the product meets user needs, increasing customer satisfaction. ADP also promotes flexibility, enabling quick responses to change requirements and promoting collaboration through cross-functional teamwork. (Knaster & Leffingwell, 2020).

While both ADP and release management share the common goal of the delivery of quality products, there are some differences. ADP promotes iterative development and continuous improvement, while release management emphasizes planning, coordination, and software deployment. The key activities of ADP enable customer-centric development and speedy delivery while release management's focus is on governance, ensuring stability and risk mitigation. (Heikkilä et al., 2010).

Still, the integration of ADP into release management can enhance both methods. Together they reduce risks through frequent and smaller releases which ensures better stability and reliability. They also improve quality through continuous testing and iterations. In addition, the integration of ADP into release management enhances customer responsiveness by responding to customer feedback and deploying changes without compromising quality. (Samer, 2016).

In summary, the nature of ADP and release management allows organizations to achieve balanced agility and control. This is seen valuable, especially in regulated environments where rapid deployment must be balanced with intense requirements. Together, they support a customer-centric, competitive, and efficient approach.

In this chapter, software release management, DevOps, ITIL, and Agile Product Delivery were introduced. The Chapter looked at the frameworks and methodologies essential to successful software deployment. Despite their differences, these frameworks and methodologies contribute to a bigger goal of delivering high-quality solutions in both structured and customer-centered ways. This effective management does not require only firm IT management but also adaptive product delivery processes.

The following chapter looks at the role of Large Language Models (LLMs) and chatbots in assisting release management practices. It investigates how these tools can address the challenges and complexities of today's software delivery environments.

4 LLMS AND CHATBOTS IN SOFTWARE RELEASE MANAGEMENT

Large Language Models (LLMs) based chatbots could be a feasible solution to tackle the common challenges related to software release processes. This chapter introduces Large Language Models in the context of software development and investigates how chatbots are already utilized in project management. The chapter concludes with the best practices for a chatbot design to get insights into what to consider when designing your own chatbot.

4.1 Overview of LLMs

Large Language Models (LLMs) are advanced language models. They use deep learning techniques, are trained in vast amounts of data, and can produce, analyze, and understand human language (Chang et al., 2024). LLMs architecture, based on the deep learning architecture called the Transformer introduced by Vaswani and others (2017), allows them to process input text as numerical representation named tokens, and contextualize the relationship between them, ensuring both high performance and efficient training.

LLMs are suitable for a wide range of natural language tasks. As Brown and others (2020) highlight, these models can translate content between languages as well as produce, complete, condense, and analyze texts. These features allow LLMs to perform tasks that previously required human intervention (Vaswani et al., 2017).

Through automating and streamlining key processes, LLM programs have contributed to software development. Sridhara and others (2023) point out that some notable LLM applications in software development are related to code generation, bug fixing, documentation, code/database queries, testing, and validation. These applications bring several benefits. According to Chang and others (2024), LLMs improve efficiency, support learning, reduce errors, and improve collaboration between stakeholders.

Despite their potential, adopting LLM programs brings their own challenges. They can misinterpret contexts leading to inaccurate results, generate code with security vulnerabilities, and output nonrelevant or inaccurate answers if the training data is outdated or biased (Xu et al., 2023). Incorporation of LLMs can also require significant resources, which may be impossible for some organizations to allocate (Chang et al., 2024).

Through its vast options and implementation possibilities, LLMs have proven to be revolutionary in various fields. In software development, they can streamline processes, improve collaboration, and foster new innovations. While there are still challenges that need to be addressed, their potential to increase efficiency and transform workflows is unquestionable.

Besides software development, LLMs play a crucial role in the development of conversational AI, like chatbots. These will be introduced in the following section, focusing on their application in software development and project management.

4.2 Chatbots in Software Development and Project Management

Chatbots are computer programs that mimic human conversation with technologies, like LLMs and natural language processing (NLP), becoming increasingly common tools in improving communication, automating routine tasks, and increasing efficiency (Cīrule & Bērziša, 2019). Their integration into business tools is increasing, which has transformed the role of chatbots from a simpler query handler to an agent capable of managing complex workflows (Stanica et al., 2018).

Chatbots are valuable for both software development and project management, where tools like Jira, Trello, and Slack can be integrated. According to Bodea and others (2020), these integrations enable functionalities, such as automation of routine tasks, enhancement of workflows, management of project and development pipelines, and facilitation of planning and collaboration. These integrations offer several opportunities. Chatbots can free up time for developers and project managers through task automation, improve communication by sharing real-time information between stakeholders, reduce cognitive load thus improving productivity, and streamline bug reporting and task sharing further reducing human errors (Cīrule & Bērziša, 2019).

As conversational AI technologies keep evolving, so can chatbots become more capable and integral parts of development pipelines. In the future, chatbots can be seen more in adapting project planning, personalizing task recommendations based on employees' workload and performance, and predicting insight for project managers through intelligent analytics (Bodea et al., 2020). Further supporting the idea that the integration of chatbots into software development and project management can be expected.

Through versatile implementations, chatbots are evolving to essential components, rather than simple conversational tools. As technology matures, the

potential of chatbots can further transform the ways of working and blend into Agile and DevOps environments. Although, it should be noted that the application of chatbots in software development and project management is still an emerging field, explaining the briefness of this chapter. However, as LLM technologies are adopted and integrated into workflows, more research and case studies can be anticipated to rise, further bringing more insight into the topic.

The following section reviews the best practices for chatbot development, underlining factors such as usability, design, and integration into existing workflows.

4.3 Best Practices for Chatbot Design

Developing a chatbot is a complex process. It requires both advanced technical and programming skills, as well as an understanding of user needs and behavior (Setiaji & Wibowo, 2016). There is no one-size-fits-all approach as a variety of methods can be applied. Still, chatbots usually consist of three main components: classifier, graphmaster, and responder (Figure 7).

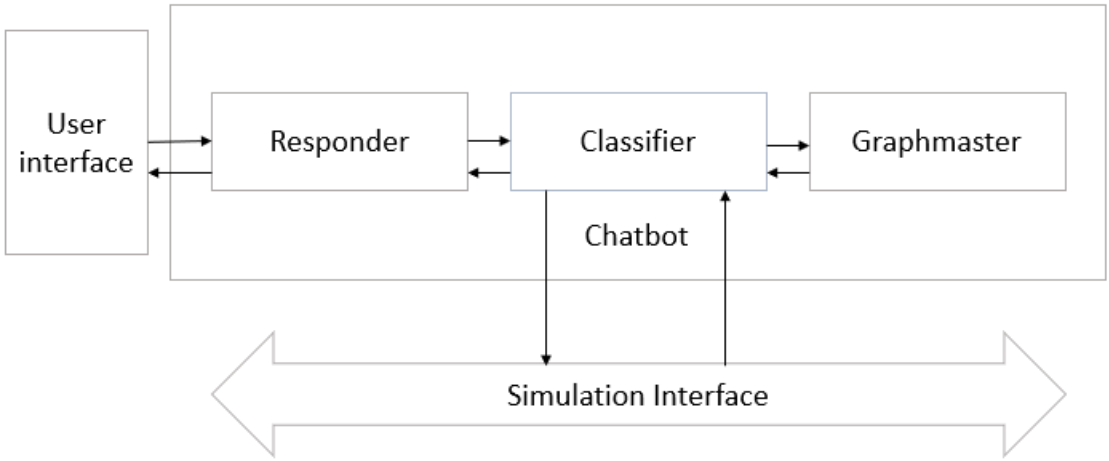


FIGURE 7 Chatbot Components (adapted from Abdul-Kader & Woods, 2015, p. 73)

Responder acts between the user and chatbot. It manages the flow of data by relaying user input into the classifier and delivering the output back to the user. The classifier normalizes and processes the user's input. It filters the data, decomposes the input into components, and translates it to graphmaster, playing a crucial role in processing and interpreting user queries. Graphmaster organizes content and uses pattern-matching algorithms to identify the most suitable response to user queries. It ensures that the chatbot's output matches its programmed knowledge base. These components work together in processing user input and generating responses, allowing chatbots to perform a wide range of tasks. Developers can further tailor the design of the chatbot to meet specific

functionalities and user expectations and requirements. (Abdul-Kader & Woods, 2015).

Even if there are numerous ways to develop a chatbot there are some general best practices in the field. The best practices for developing a chatbot include various techniques. The techniques are parsing, pattern matching, AIML, chat script, SQL and relational database, Markov Chain, language tricks, and ontologies (Setiaji & Wibowo, 2016).

Parsing analyzes the input text and manipulates it with several natural language processing (NLP) functions. Pattern matching defines and applies regular expressions to identify patterns in user input. AIML (Artificial Intelligence Markup Language) is a rule-based description language. Its main task is to provide automated responses to users' questions. Chat script helps when no match can be found in AIML. It builds a sensible default answer by concentrating on the best syntax. It gives sets of functionalities such as facts, logical and/or, and variable concepts. (Singh & Beniwal, 2022).

SQL and Relational Database (RDB) help the chatbot remember previous conversations by building a database for the chatbot, making the conversation more meaningful and continuous. Markov Chain ensures that the response of the chatbot is more applicable probabilistically and, consequently, more correct. Language tricks are phrases, paragraphs, and sentences in chatbots that add variety to the knowledge base and make it more convincing. The language tricks are used to satisfy a specific purpose and provide alternative answers to questions. Ontologies compute the relation between concepts, such as synonyms, hyponyms, and other relations. (Setiaji & Wibowo, 2016).

The success of a chatbot does not solely rest on the technical side of the bot. When designing and integrating a new chatbot the intuitiveness of user interaction plays a crucial role (Hill et al., 2015). This means that when developing a chatbot it requires an approach that considers both the functionality as well as the users.

When thinking about intuitive user interaction, the user-centered design (UCD) can be utilized. Firstly, it's crucial to understand the users by identifying target users and their needs. Secondly, the chatbot's personality and tone need to be defined. For example, the chatbot should be conversational but not too casual. Thirdly, the chatbot should be easy to use and not create a cognitive overload with hard navigation or too much information. Fourthly, the chatbot should behave consistently, for instance, in its language and response times. (Bahja et al., 2020).

To provide value for users, there should be context awareness, meaning that the chatbot should be able to retain the context of the conversation, reducing the need for repetitive questions or clarifications from the users. In addition, it should recognize when it does not understand the input correctly or when it does not have any answer to it. In this case, it should be able to ask the user to, for example, reformulate their given input or clarify what information they are missing. (Haugeland et al., 2022).

The chatbot should also be able to automate tasks (Sridhara et al., 2023). For it to optimize the workflow inside an organization, the chatbot should be able to help in creating Jira tickets and generate sprint reports. It should also be able to interpret various query types (Hill et al., 2015). Also, chatbots could help in the approval process. If the chatbot could help in retrieving the approval status it could help with the deployment of a new software version.

The main idea behind a successful chatbot lies in the integration of other tools. For it to be useful and value-adding the chatbot should have seamless integration with existing tools such as Jira and Confluence. This way users could interact and retrieve information from these tools through chatbot. This can be seen as one of the biggest possibilities that the chatbot could provide, as it can reduce the need for manual navigation. (Cīrule & Bērziša, 2019).

Overall, the best practices for chatbot design and integration need both technical and user aspects. It is important to prioritize the technical side for the chatbot to be successfully integrated but also the user experience is a vital part in the design of it. With the configuration of the internal tools, the chatbot could add value to streamlining the release process and software development.

In short, LLMs and chatbots in software development are becoming essential. As LLM-based chatbots are recognized as more useful tools, many organizations can leverage them in a versatile way. This can be either in project management, software development processes, or even by creating a solution of their own. With all of this, it cannot be denied the impact that AI-based tools will have in the future for many fields in software development.

Although chatbot development and integration is growing, it is still a relevantly under-researched topic, compared to other areas such as Agile or DevOps. Much of the available literature focuses on technical components or broader applications of such conversational AI, whereas there are significant gaps in studies from software development contexts or organizational case studies. Therefore, this chapter gave an overview of key aspects of best practices and techniques, while remaining in a limited scope.

This chapter concludes the literature review of the thesis. The review started with Agile and SAFe methodologies that gave ground for software release. Software release follows processes, which are defined in every organization. Software releases can, for instance, follow an Agile way of releasing which was introduced through Agile Product Delivery. Software release should also be management, which was then closer looked at through ITIL and DevOps. In the last chapter of the literature review LLMs and chatbots were introduced. With these topics, the literature review can answer the study's first research question and help in the empirical research of the study. The summary of the literature review is introduced in the following chapter.

5 SUMMARY OF LITERATURE REVIEW

The literature review looked at the crossroads of Agile, DevOps, and ITIL in software release management, highlighting the challenges and shortcomings of traditional release processes. It examined the role of LLM-based chatbots in optimizing these processes, providing a basis for addressing the study's first research question: How can an LLM chatbot optimize software release?

The literature suggests that software release management often struggles to adapt to the rapid pace of Agile development. Teams struggle to maintain consistency between iterations, while release managers struggle with overwhelming workloads and scattered communication. These challenges are particularly acute in large organizations where multiple teams work on the project, often struggling with inefficient flow of communication. Frameworks, such as ITIL, Agile, and DevOps offer their own solutions, but they are not always sufficient to bridge the gaps in highly dynamic environments.

The review identified the potential of chatbots in addressing these challenges. Chatbots can improve productivity and reduce bottlenecks in software solution management, by automating repetitive tasks, streamlining workflow, and facilitating communication. In particular, the chatbots could help teams by automating routine queries and tasks, ensuring smooth coordination of the team by acting as a central for sharing real-time updates, and improving deployment processes by providing insights and communicating key events to relevant stakeholders. These allow LLM chatbots to create value and not replace human professionals by guiding team efforts to more meaningful and productive work.

While the potential is considerable, there are also critical aspects to the integration of LLM chatbots. To avoid misunderstandings, chatbots need to interpret inputs accurately and preserve the context of the conversation. The scope and identification of sensitive data during the development and release management requires solid security measures. Also, to maintain trust between team members, the scope of the chatbot should be carefully defined to avoid any duplicative roles.

Empirical studies are necessary to validate these claims and to gain insights into the implementation of chatbots. Still, the literature indicates that LLM

chatbots can optimize software release processes. However, its success depends on thoughtful implementation, regular monitoring, and clear boundaries to ensure that value is added without disruptions.

As digitalization continues to proceed, the integration of LLM chatbots into software release management is a growing area to research. While the current literature supports a promising outlook, it remains limited. The insights of this literature review provide a foundation for the empirical research of this study. By combining both theoretical background and real-world applications, the following chapters explore and validate the practical potential of LLM chatbots in optimizing software release processes.

6 RESEARCH SETTING

This chapter describes the research question, case organization, and methodology of the study. It presents the research method used and justifies the choice and the steps taken.

6.1 Research Questions

The research questions have been formulated in the context of a case study and focused on the potential application of LLMs in software release processes. The objective of the research is to explore the optimization potential of an LLM chatbot in the area and to identify the software requirements needed to implement it. The research questions and hypothesis are as follows:

RQ1: How can an LLM chatbot optimize software release?

> *Hypothesis: LLM chatbot can optimize software release by enhancing communication between teams and stakeholders.*

RQ2: What are the software requirements for an LLM chatbot in software release processes?

> *Hypothesis: The software requirements for an LLM Chatbot include features especially related to project management, information sharing, and task automation.*

The objective of this study is to investigate how an LLM chatbot can optimize software release management and to identify the essential features and requirements of such a chatbot. This objective is addressed by answering the research question. The first question is answered by the literature review, providing theoretical insights into the potential optimization. The second question is answered through an empirical study which includes a survey in a case organization.

6.2 Case Description

Nowadays, AI is the language of the generation. This makes the need to catch up on the continuous development necessary for an organization to stay relevant. According to Abbas (2021), the interest in AI solutions has skyrocketed within organizations, with AI-related projects increasing across sectors as companies aim to leverage their potential. One area of focus is the management of software solutions. Organizations are seeking faster and more efficient release processes to deliver quality software. This drive has led to the development and research of AI-based solutions for Agile software release processes, aiming to optimize both speed and quality (Perkusich et al., 2020).

One of these kinds of organizations is the study's case organization, operating in the financial sector and creating technological solutions for both B2B (business-to-business) and B2C (business-to-consumer) businesses. The organization is part of an international large-scale company serving customers primarily across the Nordic countries. The chosen case organization consists of thousands of full-time employees working in various Agile Release Trains (ARTs). This facilitates cross-functional teamwork in delivering customed technical solutions to customers.

The case organization's management of its software solutions has been growing more complex due to the increasing number of employees and solutions being developed. This has made it challenging to deploy SAFe frameworks as effectively as in the past. A key problem identified relates to the availability of information. Although the organization has extensive documentation and protocols, accessing and effectively using this information has become more difficult over time.

The main factors contributing to this problem are silent information and quantity and distribution of information. Silent information collected by long-standing employees is difficult to document or share, particularly in fast-changing environments with frequent staff turnover and changing requirements. It is also challenging for workers to find the right information at the right time. This lack of easily accessible information creates inefficiencies in the releasing process, which could be overcome by an AI-based solution, such as a chatbot. This approach contrasts with concerns about AI creating barriers to accessing information or replacing human roles. Instead, the chatbot would make the release processes more seamless and efficient improving information accessibility and transparency.

To address these challenges, a team within the organization developed a Proof of Concept (PoC) for an LLM-based chatbot. A PoC is evidence demonstrating that a product, business proposal, or idea is feasible (Banerjee et al., 2017). It was designed to assist the release management process by integrating with existing databases and tools, effectively retrieving, and presenting relevant information. The team behind the chatbot aims to identify features and functionalities that the chatbot could support for more effective software release processes.

This thesis intends to produce workable software requirements for the chatbot's development team, guiding future iterations. By optimizing the release process with AI, the organization can better manage the release of its digital products. The empirical research is conducted iteratively in collaboration with the chatbot development team, ensuring consistency between results and practical application. The case organization has provided the resources needed to support the research, providing a well-established framework for exploring the potential of LLM chatbots for optimizing software release processes.

6.3 Design Science Research Methodology

The study's second research question focuses on the design and validation of chatbot, acting as the study's' artifact, and its software requirements. This artifact is a response to an underlying phenomenon of the growing challenges faced by organizations in managing fast and complex software release cycles. The challenges arise for example from fragmented information, silent knowledge, and ineffective communication and task management during releases. This study examines the challenges and explores whether they can be addressed by an LLM-based chatbot.

Design Science Research (DSR) was chosen as the research method. This is because, unlike traditional qualitative case studies, seeking to describe or explain phenomena, DSR is well suited to solving problems through the creation and validation of practical artifacts. In this study's context, the artifact is the chatbot.

DSR is a research methodology, focusing on the development and validation of knowledge in the field of information systems (IS) (Peffers et al., 2007). It provides a structured framework for solving practical problems through a constructive and iterative process (Hevner et al., 2004). DSR was chosen because it enables the combination of theoretical research and the creation of solutions. In addition, it contributes to the understanding of how AI-based tools can address broad challenges in managing software.

The DSR consists of six phases: problem identification, definition of the objective of a solution, design and development, demonstration, evaluation, and communication, illustrated in Figure 8, and described below. According to Peffers and others (2007), the process iterates mainly between phases two (definition of the objectives of a solution) and five (evaluation), ensuring that the solution is aligned and refined with the identified problem and resulting in a usable artifact.

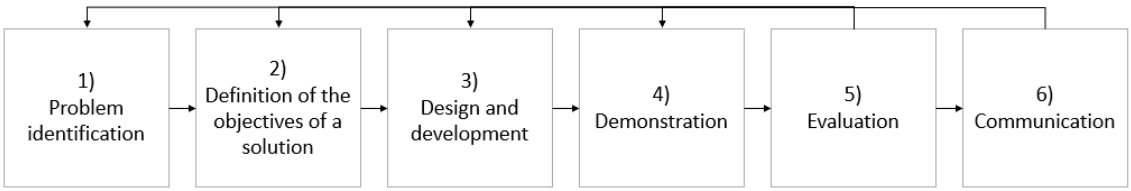


FIGURE 8 The DSRM Process (adapted from Peffers et al. 2007, p. 54)

The first phase of DSR is problem identification which identifies and defines the problem to be solved. This is done by asking supporting questions, such as "*What would solve the problem?*" and "*Is this a valid problem, or does routine design solve it?*" (Hevner et al., 2004). The expected outcome of this phase is to identify research problems and justifications as well as create a research plan.

The second phase dives into the definition of the objectives of a solution. The phase should create specifications of the targeted solution in line with the chosen theoretical baseline (Peffers et al., 2007). This is executed by studying the focus and scope of the perceived problem, and by implementing knowledge from relevant theoretical sources.

The third phase is the design and development of the solution. The result of the phase is an artifact which is a solution to the presented problem. According to Hevner and others (2004), this phase specifies the knowledge applied and justifies the choices. The artifacts are often derived from previous research and state-of-the-art practices of industries.

The fourth phase, demonstration, tests the solution. The idea is to test the solution in the designed environment to see if it is feasible there (Peffers et al., 2007). The outcome should be knowledge and evidence of the viability of the solution. Also, the artifact is portrayed within its context, demonstrating its functionalities in the intended environment.

The fifth phase is often seen as one of the most important phases in DSR as it evaluates the solution. The solution is evaluated on how it performs and how it works with the specifications. The assessment of the designed and built solution is done with different metrics, benchmarks, or techniques (Hevner et al., 2004).

The sixth phase of DSR is communication. It can be a report, paper, presentation, or press release (Peffers et al., 2007). This phase presents the results, the final solution, and the insights gained at each step. This phase is important as it enables a broad knowledge base on design science.

In this master's thesis, empirical research is based on the DSR. It aims to guide the creation of software requirements for a chatbot that addresses the challenges of software release management. This research contributes to the field of information systems (IS) research by expanding knowledge of how LLM-based tools can help in solving bottlenecks in software release processes.

While the study cannot implement all the phases in the ideal depth within the constraints of a master's thesis, it prioritizes the creation, presentation, and evaluation of software requirements to better ensure usable results. Partial implementation of DSR is common (Stange et al., 2022), which can be seen

particularly in academia at the master's and doctoral levels, due to time and resource constraints. By applying DSR in a critical way, the study answers the research problem while acknowledging the methodological limitations of its scope, rather than creating a fully deployed and validated artifact.

The reason DSR was chosen as the methodology for this study is based on its focus on solving business problems by developing technology-based solutions. It was considered more suitable than alternative methodologies such as Action Design Research (ADR), which according to Sein and others (2011), emphasizes a collaborative, change-oriented approach where iterative design is deeply embedded in the organizational environment. Although ADR could theoretically fit into the context of the research, it was not selected as the method because it focuses on organizational changes rather than problem-solving. For this study, DSR's narrower focus on the design and validation of a functional artifact makes it a more suitable method for exploring and addressing the problem of optimizing software release management.

Prior to the study, the organization had already developed a Proof of Concept (PoC) for the chatbot. Thus, this study's focus is on the evaluation phase, aiming to define the requirements for future iterations of the solution. The empirical study is based on a scientific framework presented by Hevner and others (2004) called the DSR framework.

The DSR framework is derived from real-world scenarios, in which people's environment, organizations, and technology form the basis for exploring both business and organizational needs. The knowledge base serves as a repository of best practices and established insights that future IS research can draw from. The IS research process focuses on developing and evaluating artifacts and theories, addressing the identified needs. (Hevner et al. 2004). This framework is presented in Figure 9.

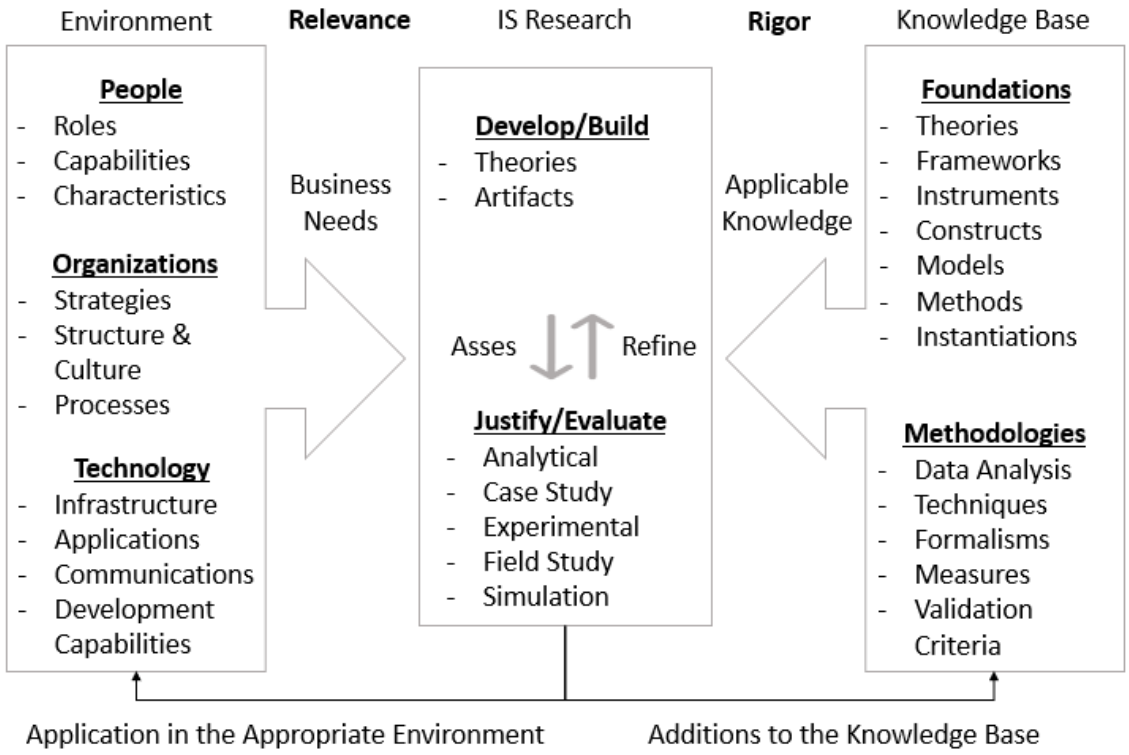


FIGURE 9 Design Science Research Framework (adapted from Hevner et al. 2004, p. 80)

The contribution of this research to the knowledge base is presented next. The knowledge base is structured around specific inputs. These inputs developed through the DSR can be classified into three levels based on their nature and maturity.

Level one comprises the implementation of artifacts, such as applied processes or software products, delivering concrete, complete, and mature outputs. Level two includes design theories, such as principles, frameworks, methods, models, and design principles, thus delivering often a balanced knowledge of both abstract and concrete. The third level represents well-developed theories, resulting in specific and limited knowledge with the lowest level of maturity. (Gregor & Hevner, 2013). The levels are illustrated in Figure 10.

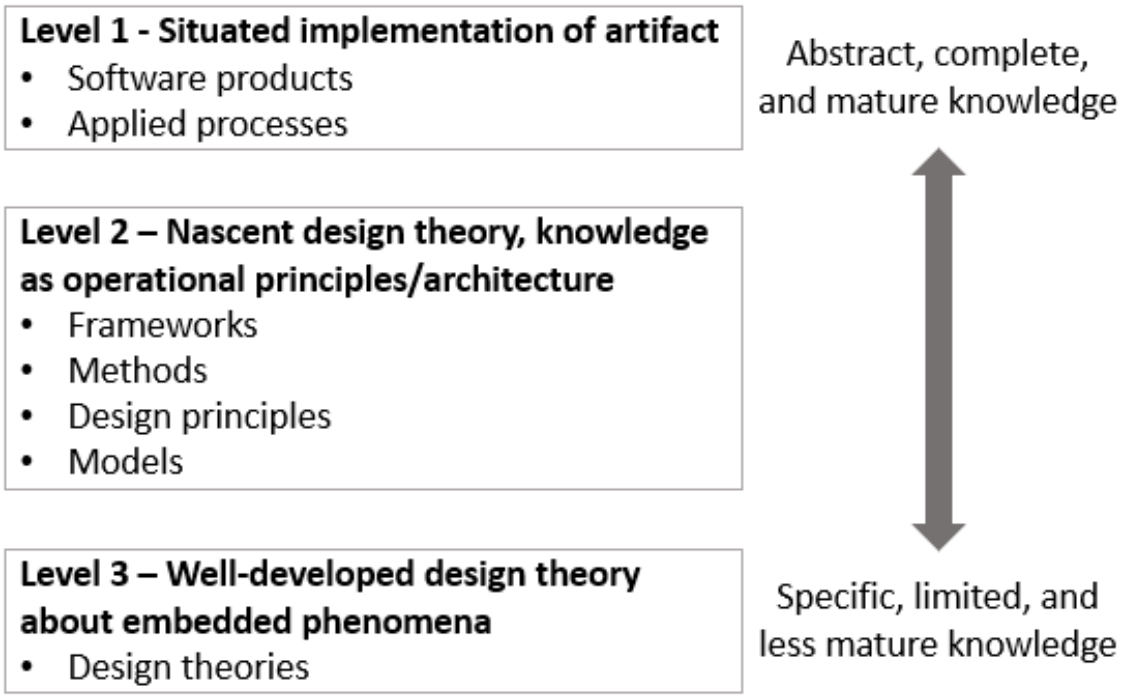


FIGURE 10 Design Science Research Contribution Types (adapted from Gregor & Hevner, 2013, p. 342)

When revisiting the research question and research objective, the second research question corresponds to the second level of the DSR contribution types. This is suitable because the research aims to investigate the use of a chatbot and develop its software requirements, which are neither design theories nor a fully developed product or process. The second level characterizes the research outcome to be neither fully abstract nor specific while being in between mature and immature development stages.

The DSR framework studies information technology (IT) artifacts, which in this study is the IT artifact being the chatbot. According to March and Smith (1995), these artifacts can be divided into four types: constructs, models, methods, and instantiations. Based on this division the study's artifact type is instantiation type. This is because the chatbot is a concrete implementation of a solution designed to optimize software release processes, showcasing how an LLM-based tool can function in practice.

In the demonstration phase, the software requirements are presented with the framework on which the artifact is based. In addition, the workflow and the decision-making processes leading to the results are presented. In the communication phase, the study results are publicly shared. The thesis will be published in the research repository JYX-library managed by the University of Jyväskylä. This ensures that the thesis will be publicly accessible in a digital format.

6.4 Balanced Scorecard Framework

Organizations leveraging Agile often evaluate the value they create (Hartmann & Dymond, 2006). The Balanced Scorecard (BSC) was created for this reason. Introduced by Kaplan and Norton in 1992, it serves as a metric measurement of strategic management performance. It helps organizations monitor and measure both success and past performance, identify and improve internal operations, assign priority, and provide feedback for better decision-making (Kaplan, 2009).

The scorecard consists of four perspectives: customer relationship, finances, education and growth, and internal processes (Figure 11). The framework divides organizational performance into these perspectives, allowing organizations to gain a comprehensive view of their business. The BSC links the perspectives to the organization's vision and strategy (Kaplan, 2009). In the framework success in one aspect drives improvement in others, creating a cumulative positive effect.

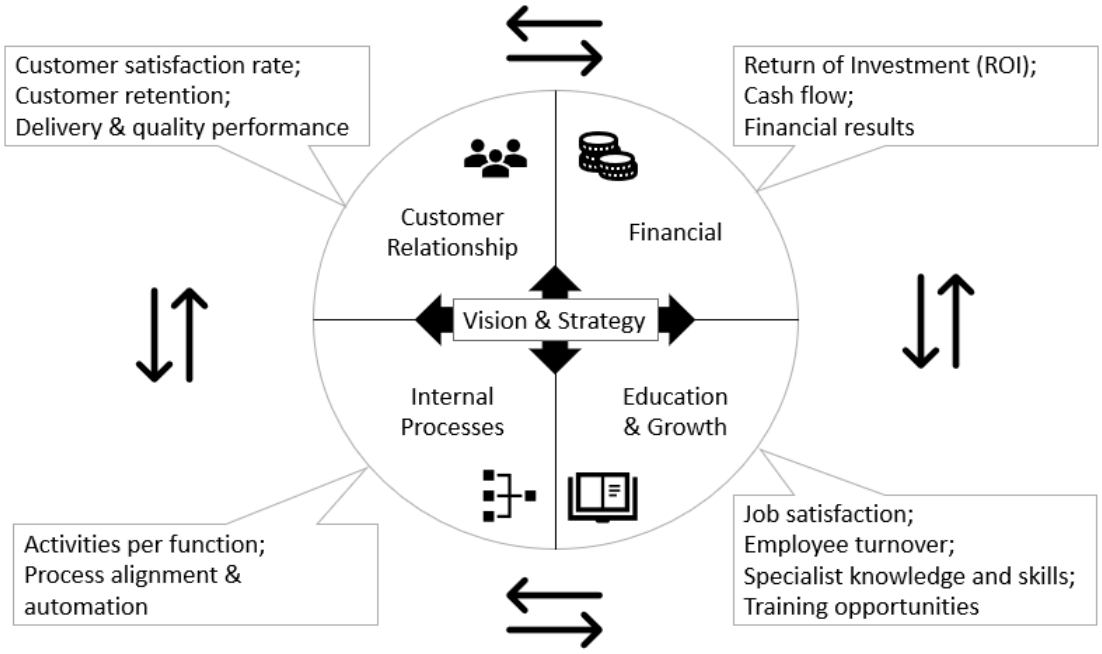


FIGURE 11 The Balanced Scorecard (adapted from Kaplan & Norton, 1992, p. 72)

The customer relationship perspective evaluates customer satisfaction and retention, market share, and customer loyalty (Nørreklit, 2003). Its objective is to understand how effectively the organization serves its customers. The financial aspect focuses on profitability, revenue, cost management, and return on investment (Kaplan, 2009). The third perspective, education, and growth, emphasizes organizational improvement and employee development. It emphasizes learning and skill development as a drive for employee satisfaction (Nørreklit, 2003). The fourth perspective, internal processes, investigates the efficiency and

effectiveness of internal operations focusing on functional activities, process alignment, and automation (Kaplan, 2009).

The BSC helps companies align their organizational structure with strategic objectives. It provides insights into financial performance while assessing the quality of the organization's services and the efficiency of its operations. In short, the BSC enables streamlined strategic reporting while supporting a business-oriented approach to managing performance. (Nørreklit, 2003).

On the other hand, BSC comes with its own disadvantages. It does not consider possible external factors such as competitors, nor does it consider risk analysis or time consideration. BSC has also been criticized for its way of focusing too much on internal processes and for the lack of clarity in implementation. (Chavan, 2009).

The BSC framework was chosen for this study because it meets the research objectives. As the study's aim is to optimize organization's internal processes, it was seen appropriate to focus the framework on internal processes. This allows the framework to address the organizational aspects, central to the study.

The BSC is used in this study in two different ways: data collection and evaluation. In the data collection, the BSC framework guided the structure and context of the survey, ensuring its consistency with the study's objectives. In the evaluation, the framework was used to evaluate the survey's results and to develop the software requirements for the chatbot. The software requirements are divided into two categories: functional and non-functional requirements. Functional requirements are the tasks that the solution should perform whereas the non-functional requirements are the qualities that the solution should have (Dabbagh et al., 2015).

With this, the research setting chapter is concluded. It provided an overview of the study and presented the utilized method and framework. The chapter introduced research questions and case study design, providing a detailed description of the background of the study, including the IT artifact. The following chapter describes the case study in detail.

7 CASE STUDY

This chapter describes the study's empirical research, focusing on the evaluation of an existing Proof of Concept (PoC) chatbot. Its objective is to answer the second research question, investigating the software requirements of the chatbot. The research process is structured and guided by the phases of Design Science Research (DSR), outlined in the previous chapter.

7.1 DSR Project

This section outlines the use of DSR in this study. The methodology was followed to ensure consistency with the research objectives and the needs of the case organization. The DSR helped to address the research questions and create software requirements for the chatbot. Table 5 presents the main steps of the DSR project in chronological order. It highlights the overlapping phases of design and development and demonstrates further underlining the iterative nature of the study.

TABLE 5 The Timeline of the DSR Project

Schedule (2024)	DSR Phase	Activity
June and July	Problem identification	Meetings and stakeholder interviews to identify challenges in the release process.
August and September	Definition of the objectives of a solution	Definition of chatbot's objectives.
September and October	Design, development, and demonstration	Development of the PoC of the chatbot and presentation to relevant stakeholders for feedback.
October and November	Evaluation	Construction and distribution of the survey to collect data for the empirical study. Analyzation, and validation of survey responses from which iterated software requirements.
December	Communication	Documentation of the findings in the form of a master's thesis throughout the whole timeline. Presentation of key findings to the case organization. Publishing the thesis internally inside the organization and publicly to the JYX library.

The problem identification phase started in June and continued until July. During this time, meetings and informal interviews were held to understand the challenges associated with the current release processes. These highlighted the key bottlenecks such as micromanagement and ineffective communication.

The solution objectives were defined in August and September. This is when the concept of a chatbot solution evolved, and its objectives were defined. Based on this the second research question was also identified.

In September and October, the design and development, and the demonstration phases were carried out simultaneously. The chatbot's Proof of Concept (PoC) was developed and presented to relevant stakeholders, whose feedback provided insights into refining the solution. This feedback also helped in developing the survey for the empirical study.

The evaluation phase took place in October and November. The PoC was evaluated through a survey from which the study's data was collected, analyzed, and validated. These findings contributed to the formulation of the software requirements for the chatbot.

Lastly, the communication phase takes place in December. The results of the DSR project are presented to the case organization in a hybrid meeting that can be attended both offline and online. This is to ensure the reachability of as many stakeholders as possible. The meeting focuses on presenting the key findings of the DSR project and the study. The master's thesis will be shared within the organization to promote the practical application of the research. This is to support future chatbot iterations and to encourage wider use of the results. In addition, the thesis will be made publicly available in the JYX library. This is to improve its accessibility to a wider audience and contribute to the academic field.

The DSR method proved to be a suitable approach in addressing the identified problem and in answering the research question. Its iterative nature

ensured that the software requirements were developed through continuous refinement and incorporated feedback from stakeholders. The following sections examine empirical research in detail. These sections bring insights into how the results of the survey and stakeholder engagement were used to develop and evaluate software requirements for the chatbot.

7.2 Problem Identification

The problem identification phase is the first step in the DSR process. This involved meetings and informal interviews with stakeholders to identify the challenges facing the release processes in the case organization. Factors such as communicating status updates and deadlines between project teams in Agile Release Trains (ARTs) were identified to be challenging. This is due to the rapid growth in the size of ARTs, making the current processes become stressful and time-consuming. It was discovered that employees found it challenging to identify who to contact, which processes to follow or to keep up to date with the progress of the project. This lack of communication has contributed to a culture of micromanagement and interruptions to work, disrupting the workflows.

Also, one identified challenge was related to the daily flood of Microsoft Teams messages and emails, especially for those working closely in the later stages of the release processes. These inquiries about status updates and pipeline processes were identified to be distracting employees from their core tasks.

To address these challenges, a team within the organization participated in a hackathon, an event in which individuals collaborate intensively for days to solve problems or explore new opportunities (Komssi et al., 2015). During this event, the team produced the idea of developing a chatbot leveraging on Large Language Models (LLMs) to assist employees with release processes. The chatbot was seen as an assistant to help streamline information retrieval and communication. During the hackathon, the team successfully developed a Proof of Concept (PoC) for the chatbot. Following this, a question was raised about how to further develop the chatbot and determine the features needed to make it both successful and useful in optimizing the release processes.

A planning meeting was held to define a roadmap for the chatbot's development. In it, the team decided that the first step would be to define the software requirements for the chatbot. This shaped the second research question of the study: what are the software requirements for an LLM chatbot in software release processes? In the following section, the objectives behind the solution are discussed.

7.3 Objectives of the Solution

The second phase of DSR is the definition of the solution's objectives. The objective of this empirical research is to answer the second research question. By answering the second research questions the study aims to help the team behind the chatbot understand what features they should consider for future increments of the chatbot. This is achieved by creating software requirements for a chatbot that assists employees in the release processes inside the case organization.

Insights from the literature review suggest that chatbots can automate tasks and streamline communication between stakeholders. For smoother communication and user experience, integration with existing tools such as Jira and Confluence is crucial. Based on the meetings with the chatbot's development team and gathered stakeholder feedback, the chatbot should also answer other priorities as well. It should optimize the release processes by helping with problem-solving, identifying obstacles, and reducing time-consuming, non-value-adding tasks for employees. The chatbot's role should be a digital assistant, improving communication between ARTs and supporting employees in navigating the release processes efficiently. The next section details the design and development of this solution.

7.4 Design and Development

The third phase of DSR is design and development. As the PoC for the chatbot has been already created, this phase presents the solution in general terms. For confidentiality reasons, detailed descriptions of the design, specific implementations, and used data will not be disclosed. Instead, the focus will be on the publicly available, open-source tools and technologies that were used when building the chatbot to provide an overview of its development process.

The chatbot is based on LLMs and uses widely adopted open-source tools such as LangChain, Ollama, Python, and Streamlit. These tools were selected by the development team through an evaluation, highlighting their relevance to the development of an LLM-based chatbot.

LangChain is a software library designed for applications leveraging LLMs. It facilitates answering questions by connecting integrated data sources and retrieving relevant information (Topsakal & Akinci, 2023). Its ability to integrate real-time data and interact conversationally with other systems made it suitable for the chatbot.

Ollama is an LLM model that allows local deployment without a server. It supports multimodal models and allows them to be executed directly on the user's machine (Gruber & Weber, 2024). The team chose Ollama because it offered a practical and efficient way to deploy the chatbot locally without having to set up any additional infrastructure.

Python is a programming language, commonly used for software development, task automation, and data analysis. Its flexibility and familiarity among developers played key reasons for its adaptation. In chatbot development, Python allows rule-based customization, such as specifying the length and tone of the response (Raj, 2019).

Streamlit is a python library, widely used in machine learning and data science. It converts data scripts into interactive web applications (Akkem et al., 2023). For the chatbot, Streamlit offered simple integration with other tools and provided chat elements.

The goal of the design and development of the chatbot was to create a conversational tool that can assist users in their daily tasks. By focusing on open-source tools and technologies, the aim was to make use of proven solutions that are publicly available. The following section describes how data was collected for this study.

7.5 Data Collection

The fourth phase of DSR is the demonstration, which according to Peffers and others (2007), is used to gather knowledge and evidence about the solution. In this study, this phase was utilized for data collection to better understand end-user needs and gather insights to shape software requirements for the Chabot.

Data was collected through a Forms Survey, which can be found in depth in Appendix 2. The survey was developed in collaboration with the case organization to ensure its relevance and comprehensiveness. On average, the survey took five minutes to complete, and it was designed to collect anonymous feedback from stakeholders involved in the software release processes.

The survey consists of three sections. The first section defined the roles of the participants in the organization, giving context to their responses. The second section had 14 statements, which participants were asked to rate from "strongly disagree" to "strongly agree" on a five-point Likert scale. A five-point Likert scale is a rating system, which allows respondents to express opinions with five options, from strong agreement to strong disagreement, including a neutral option, and enabling nuance feedback (Joshi et al., 2015). The third section was an open-ended question, allowing participants to add additional comments or suggestions.

The second section's 14 statements were designed to reflect the four aspects of the Balanced Scorecard (BSC) framework, addressing aspects of chatbot usability and its potential to optimize release processes. The first three statements were related to the customer relationship perspective, the statements four to six were based on internal processes. Statements seven and eight were about education and growth and statement number nine was about financial perspective. The last statements of the survey were additional usability and experience-related general statements. Table 6 gives an overview of these statements, describing what was asked and how they correlate with different aspects.

TABLE 6 Overview of Survey Statements

Statement	Aspect
1. If a chatbot could automate routine tasks (e.g., gathering information) it could assist you in your daily work?	Customer Relationship
2. If a chatbot could provide automatic updates or share relevant information it could improve collaboration between your team and others?	Customer Relationship
3. If a chatbot could provide updates on project status (e.g., release progress, bug tracking), do you believe this could improve your ability to stay informed?	Customer Relationship
4. A chatbot that communicates important release information to stakeholders could reduce the need for manual reports. Do you think this would improve your communication efficiency?	Internal Process
5. If a chatbot could help detect and suggest solutions to bottlenecks in the release progress, would this improve the speed and efficiency of releases?	Internal Process
6. Imagine a chatbot that can instantly retrieve the latest project information (e.g., issue statuses, release readiness). Would this help streamline your work?	Internal Process
7. A chatbot could automatically generate release notes, meeting summaries or status reports based on project data. Would this automation make your role in the release process more efficient?	Education and Growth
8. If chatbot could take over simple tasks (e.g., retrieving project data), do you believe it would help your team focus on more critical and value-adding activities?	Education and Growth
9. By offering quick answers, providing relevant information, and identifying potential issues early, the chatbot could help avoid delays in the release process. Do you think this would reduce project costs?	Financial
10. If the chatbot could handle routine tasks, such as answering to common questions, would this help reduce the overall manual workload for your team?	Usability and Experience
11. Imagine a chatbot that provides recommendations based on historical data and past release performance. Do you believe this would support better decisions-making in your role?	Usability and Experience
12. If a chatbot could take on routine queries and help with quick information retrieval, would it reduce the mental effort required for simple tasks and free up your time for more strategic work?	Usability and Experience
13. A chatbot that supports your daily tasks by automating processes and offering timely information could improve both personal and team productivity. Do you think this would positively affect your productivity?	Usability and Experience
14. If the chatbot could help track patterns or learn from previous release cycles, do you think this would improve team learning and the ability to optimize future releases?	Usability and Experience

The participants were chosen randomly. An email (Appendix 1) was sent to 10 different mailing lists inside the case organization and to six individual employees who were outside of these lists. The email was sent on Monday 7th of October 2024 and the answering time for it was 5 business days, until Friday 11th of October 2024. In total, the email was sent to around 500 employees working with the release processes inside the case organization. Altogether 51 people participated in the survey, making it around 10 percent participation rate. The roles of the participants distributed as follows:

- 21 Developer
- 1 Architect
- 16 QA/QA lead/Test Analyst
- 4 Product Manager/Product Owner/Business Analyst
- 2 Application Provider/Application Owner/
- Technology Service Lead
- 1 Release Manager/Service Manager/Infrastructure Specialist/
- DevOps Engineer
- 4 RTE/STE/ Hub Lead
- 2 Scrum Master/Agile Coach
- 0 Risk Manager/Security Specialist

The roles for the survey were grouped as above to better analyze the data. The grouping was done based on how much each role was related to each other. No risk manager nor security specialist participated in the survey leaving this group completely outside of the study. Next, how this data was analyzed and validated is opened.

7.6 Data Analysis, Validity and Reliability

Through data analysis, raw data is structured and transformed to produce meaningful results and conclusions. The software requirements for this study are based on user needs, which require a systematic approach to analyzing the collected data. Since the survey results consist mainly of quantitative data from Likert scale responses complemented by qualitative data from feedback in an open-ended question, a mixed method was used to analyze the data.

Likert-scale responses were analyzed using descriptive statistics to identify trends and patterns in stakeholder perceptions. This allows quantifying levels of agreement for differing statements related to the functionality and impact of chatbots on release processes. The results were supposed to be segmented by participant role, allowing comparison between groups and a better understanding of role-specific needs. But due to the small number of participants and the vast differences on the size of the role groups this was not possible without

having to compromise the reliability of the results, thus excluding this from the data analysis.

Although the main data set is quantitative, qualitative analysis was applied to open the survey responses to obtain information. This feedback was essential for contextualizing the quantitative results and refining the software requirements of the chatbot. For the qualitative data, inductive content analysis was chosen to identify emerging themes from the texts. This does not rely on pre-defined categories but allows patterns to emerge organically, further ensuring that the data guides the conclusions.

In research, validity refers to how well and accurately a chosen method measures what it is meant to measure. It determines the credibility and trustworthiness of the findings. Following, reliability is the consistency of a measure. It aims to provide information on whether the research will be repeated and, would it provide the same results. If a test or a tool is valued as reliable it gives consistent results over time or across different situations. (Yin, 2009).

The following steps were taken to ensure validity and reliability. The survey was designed to involve a wide range of roles involved in the release processes to capture varying perspectives. The survey invitation was distributed to over 500 stakeholders cross-organizationally, enabling participation from different functional areas. The quantitative data analysis was conducted systematically, while qualitative responses were coded and categorized in line with best practices in content analysis. The findings were contextualized with literature on chatbot functionality and release management, increasing generalizability.

By combining both quantitative and qualitative approaches, the data analysis drew strengths of both types of data. Quantitative data provided measurable insights into stakeholder preferences and needs, while qualitative feedback provided depth and context. Together, they provided information to determine the functional and non-functional requirements. With these actions taken the study aimed to maximize the generalizability and the validity of the results. Considering this, the following section discusses the evaluation process and results of the research.

7.7 Results and Evaluation

The fifth phase in the DSR process evaluation assesses the effectiveness of the solution and its relevance to the needs of the organization and user (Peffers et al., 2007). In this study, the evaluation phase involves the analysis and validation of the survey data. The Balanced Scorecard (BSC) framework presented in the previous chapter was used to classify and interpret the results. This ensures that the analysis reflects both technical and organizational dimensions, providing useful insights for formulating software requirements.

Data was collected through a survey, which included 14 Likert-scale statements and an optional open-ended question. Table 7 provides an overview of participants' responses to the statements. A detailed breakdown of the responses

by group can be found in Appendix 3. Although there were small differences between the groups, the sample size of some groups was too small to make reliable statistical comparisons. Therefore, to maintain reliability, the evaluation focuses on the pooled results.

TABLE 7 Overview of Survey Results

Statement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. If a chatbot could automate routine tasks (e.g., gathering information) it could assist you in your daily work?	1	-	12	20	18
2. If a chatbot could provide automatic updates or share relevant information it could improve collaboration between your team and others?	-	2	8	29	12
3. If a chatbot could provide updates on project status (e.g., release progress, bug tracking), do you believe this could improve your ability to stay informed?	-	1	14	22	14
4. A chatbot that communicates important release information to stakeholders could reduce the need for manual reports. Do you think this would improve your communication efficiency?	1	-	12	23	15
5. If a chatbot could automate routine tasks (e.g., gathering information) it could assist you in your daily work?	-	-	11	22	18
6. If a chatbot could provide automatic updates or share relevant information it could improve collaboration between your team and others?	-	2	13	24	12
7. If a chatbot could provide updates on project status (e.g., release progress, bug tracking), do you believe this could improve your ability to stay informed?	2	4	8	21	16
8. If a chatbot could help detect and suggest solutions to bottlenecks in the release progress, would this improve the speed and efficiency of releases?	2	2	10	24	13

TABLE 7 (Continued)

Statement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
9. A chatbot could automatically generate release notes, meeting summaries or status reports based on project data. Would this automation make your role in the release process more efficient?	1	2	17	19	12
10. By offering quick answers, providing relevant information, and identifying potential issues early, the chatbot could help avoid delays in the release process. Do you think this would reduce project costs?	1	4	8	26	12
11. Imagine a chatbot that provides recommendations based on historical data and past release performance. Do you believe this would support better decisions-making in your role?	2	1	16	23	9
12. If a chatbot could take on routine queries and help with quick information retrieval, would it reduce the mental effort required for simple tasks and free up your time for more strategic work?	1	1	13	26	10
13. A chatbot that supports your daily tasks by automating processes and offering timely information could improve both personal and team productivity. Do you think this would positively affect your productivity?	1	3	14	21	12
14. If the chatbot could help track patterns or learn from previous release cycles, do you think this would improve team learning and the ability to optimize future releases?	2	-	11	23	15

The results of the survey were generally positive, with most statements being high or a very high level of agreement. However, some areas were highlighted with a lower agreement level.

The statements related to internal processes (statements 4-6) received the most positive feedback, with over 85 percent of respondents agreeing or strongly agreeing. This suggests that stakeholders value the importance of improving the efficiency and transparency of internal processes. Similarly, the statements related to customer relations (statements 1-3) also received positive responses, highlighting the importance of improving communication and user engagement.

The statements related to education and growth (statements 7 and 8) received the most negative feedback, with 12 percent of respondents disagreeing and 8 percent strongly disagreeing. Also, the statement related to finance (statement 9) received low neutral or negative feedback of 6 percent, indicating that participants may not consider direct financial benefits as a primary objective of the chatbot at this stage.

The statements on task automation (statements 10 and 13) received less positive feedback, with around 10 percent expressing negative opinions. The responses to the open question further supported this view, as some participants stated that mere automation would not justify the use of a chatbot without thorough integration with existing tools such as Jira and Confluence. Several participants also stressed the importance of information retrieval and solution suggestions as critical functionalities for value creation. The identified functional and non-functional requirements of the chatbot are presented in Table 8.

TABLE 8 Software Requirements for an LLMs Chatbot

Category	**Description**
Functional Requirements	
Real-Time Status Updates	Provide real-time updates on release statuses as well as enhance the retrieval and access of up-to-date information.
Issue Tracking and Resolution	Track and log issues reported by users and guide them with suggestions to better resolve bottlenecks in the release processes.
Information Gathering and Sharing	Gather and share information through already existing and used data sources, such as Jira and Confluence, to better support release processes and improve cross-functional collaboration.
Non-Functional Requirements	
Performance and Response Time	The output should be fast and prompt to support smooth interaction.
Reliability	Retrieval of reliable information, with sources included, to support the perceived trust for the chatbot and not to share wrongful or made-up information.
Learning	Learn from previous release cycles and through tracking patterns to better assist in the release processes.

An effort was made to capture different perspectives by sending the survey to 500 stakeholders and by structuring the survey in a clear and meaningful way. The results were compared with the qualitative feedback from the open question

to confirm quantitative trends. Requirements were mapped against best practices in chatbot development and release management to ensure theoretical and practical relevance.

The open question provided valuable qualitative insights that complemented the quantitative results. Key themes related to the importance of integrating chatbots with existing tools such as Jira and Confluence were frequently highlighted. Several participants stressed the importance of needing the chatbot to provide workable solutions to bottlenecks. Also, participants hoped that the chatbot would facilitate smoother communication between teams.

These findings influenced the design of the functional and non-functional requirements, ensuring that the solution meets user expectations. By utilizing the BSC framework and analyzing both quantitative and qualitative data, the evaluation phase identified requirements for future chatbot iterations. In the next chapter, these findings are combined with the insights from the literature review to answer the research questions and hypotheses.

8 DISCUSSION

In this chapter the results of the study are analyzed and compared with previous literature to provide an understanding of the findings. It discussed how this study aligns with existing research, focusing on the broader role of LLM-based chatbots in optimizing software release processes.

8.1 Reflection on Research Questions

This section examines the research questions in the context of the empirical results and the literature review. By analyzing the results, their consistency or differences are discussed from previous studies. This provides an understanding of the impact of LLM chatbot development on software release processes. The first research question is reflected solely based on the literature review of the thesis while the second research question is reflected based on both literature and empirical evidence.

RQ1: How can an LLM chatbot optimize software release?

> *Hypothesis: An LLM chatbot can optimize software release by enhancing communication between teams and stakeholders.*

The literature supports the hypothesis that chatbots can serve as effective communication and knowledge management tools in DevOps and Continuous Delivery (CD) frameworks (Knaster & Leffingwell, 2020). It indicates that real-time status updates and knowledge retrieval capabilities can contribute to reducing delays and bottlenecks in software releases (Schaefer et al., 2012). However, automation can reduce repetitive tasks (Virmani, 2015), but if poorly implemented it can even disrupt workflows or increase complexity (Chang et al., 2024). In addition, some studies emphasize automation as the cornerstone of chatbot optimization (Bodea et al., 2020; Cīrule & Bērziša, 2019). These findings suggest that for

organizations with existing automation tools, the value of a chatbot is to act as a central for real-time information sharing and support.

RQ2: What are the software requirements for an LLM chatbot in software release processes?

> *Hypothesis: The software requirements for an LLM Chatbot include features especially related to project management, information sharing, and task automation.*

Based on empirical results the primary requirement for a chatbot was related to information-related functionalities. Statements regarding the ability to retrieve and share information from existing tools received strong positive feedback. Participants appreciated features such as real-time status updates and problem-solving guides. However, the requirements for project management and task automation were not as strongly emphasized and the related statements received lower positive responses or were the most neutral ones. After diving into the open-question responses, it was confirmed that participants emphasize reliability and integration over new project management features.

The role of LLMs in optimizing project management and task automation was widely discussed in the literature review. It brought up that LLMs can reduce project management costs through task automation and by providing information on resource allocation (Cīrule & Bērziša, 2019). However, the empirical study's results indicate that stakeholders involved in the release processes can perceive the information management to be greater than the introduction of new project management tools.

Furthermore, the requirements for solid integration were emphasized in the research, supporting the idea that the adoption of a chatbot depends on its ability to seamlessly integrate existing workflows and tools. This suggests that while there is potential for both project management and task automation features, their success depends on whether the chatbot meets the need for reliable and accessible information-sharing capabilities. These generalized assumptions related to project management and task automation in the literature highlight the importance of contextual factors when determining user needs.

8.2 Implications and Future Research

The results suggest that there are a wide range of potential implications for both organizations and disciplines. The wider use of LLM in software release processes is possible and there appears to be interest in this area. An LLM-based chatbot could be a potential solution for optimizing release processes, particularly for addressing bottlenecks. Overall, the results of the research indicate that a chatbot could be a valuable tool for collecting, sharing, and suggesting information from existing internal systems.

As this topic is still relatively new and under-researched, future studies could go deeper into the potential of chatbots. Future research could investigate the application of chatbots in Agile methodologies. This could provide insights into improving Agile practices and assess whether AI-based tools could contribute to the development of these methods. In addition, the technical aspect of chatbot development could be further studied, helping identification of which LLM tools are most effective in building successful chatbots for software release processes.

8.3 Evaluation of Research Quality

In the context of Design Science Research (DSR), assessing the quality of research involves evaluating both processes and results to ensure methodological accuracy and practical relevance. This study evaluated contributions such as survey design and data collection principles. Yet, the analysis can be seen going beyond these elements and focusing on the overall effectiveness and relevance of the artifact in solving the defined challenge.

In DSR reliability emphasizes consistent processes and results, although the methods are iterative and context dependent. In their article Storey and others (2017) present the Design Science Reliability Framework, highlighting key challenges such as researcher influence and reproducibility. From these considerations and the techniques to improve the reliability of DSR by Hevner and others (2004), the study followed strategies to improve reliability. It maintained transparent documentation to ensure reproducibility and methodological clarity. Through testing the artifact was initially evaluated to assess its consistency. Lastly, the study used both qualitative and quantitative evaluation methods when confirming the findings. With these techniques, the study aimed to ensure that iterative development cycles did not compromise reliability.

In DSR validity ensures that the results meet the stated objectives and represent the studied phenomena. Larsen and others (2020) view validity as crucial in DSR as it provides procedural models for justifying research claims and assessing artifacts. The iterative approach of DSR allowed for continuous refinement of both the artifact and the research processes, strengthening the validity of the study.

According to Venable and others (2016), there are four types of validity applicable to DSR: construct, internal, external, and instantiation. In this study, the construct validity was ensured by matching the characteristics of the artifact with theoretical foundations and industry standards. Internal validity was assessed by testing the components of the artifact against the needs of the case organization to ensure that the results match the objectives. External validity was limited due to the scope of a single case study, but the study addressed these by designing an artifact that can be adapted to similar organizational contexts. Instantiation validity was demonstrated through practical implementation and measurable improvements in software release processes.

The artifact developed in this research is a solution designed to optimize software release processes. This theoretical and practical focus bridges the gap between academic principles and workable real-world solutions. The artifact was evaluated for its utility and adaptability during the software development phase.

Compared to previous DSR studies, this study takes a holistic approach to the design and evaluation of artifacts. While DSR projects often emphasize either theoretical or practical aspects, this study aims to balance both. This is to contribute to the quality of the artifact and aim to ensure its applicability in a wider context. It is good to note that the artifact was tested on a limited scale, inside one case organization, which may limit its generalizability.

8.4 Limitations

There are limitations to exploring LLM-based solutions. For instance, according to Gupta and others (2023), LLM-based applications such as ChatGPT have security vulnerabilities, such as data leaks. These kinds of vulnerabilities may lead organizations to prohibit the use of such applications.

The research does not also consider the possible ethical issues related to LLM tools. As brought up by Mökander and others (2023), it is vital to ensure the technical, legal, and ethical aspects while designing and developing an AI system. This study did not emphasize the ethical side of the LLM chatbot design. This could have left out important notes and factors that should have been taken more into account while producing the software requirements.

The observations of the study are limited to one industry, the financial industry, only. In this industry, the development processes can differ when compared to the traditional software industry, which is not considered in the results of the study. Therefore, the results may not be feasible in other industries outside of the financial industry.

The limitations are also related to the scope and the setting of the research. A single case study provides insight into real-life occurrences, but it is not descriptive enough to create a conclusive overview of the chosen topic. By adding more case settings, the generalizability of the results could be improved.

It is also worth noting that the collected data can create limitations to the results. Compared to the number of stakeholders who received the survey and the number of final participants, the participation rate can be perceived as low. Also, the developer group was big compared to other role groups, where some had only one participant and some did not have any participants. Larger survey samples with more differing groups of roles could enrich the study and provide more insights, increasing the reliability of the results.

9 CONCLUSION

Software release processes are undergoing a change. Organizations leveraging on Agile methodologies face challenges in scaling Agile practices cross-organizationally. These challenges arise from factors such as employee turnover and the need to adapt to rapidly changing stakeholders' needs. Therefore, optimizing software release processes has become a much-discussed topic in various organizations.

One solution is the integration of an LLM-based chatbot. With the growing interest in AI-based solutions, organizations are increasingly looking into the possibilities of developing their own both internal and external AI applications. This trend is reflected in the increasing use of assistive chatbots, especially in customer service.

This master's thesis examines the possibilities of chatbots in software release processes. The study aimed to understand how a case organization could leverage on an LLM-based chatbot to optimize their software release processes and define the software requirements for the chatbot. The research aimed to answer two research questions: "*How can an LLM chatbot optimize software releasing?*" and "*What are the software requirements for an LLM chatbot in software release processes?*".

Both literature review and an empirical study were conducted for the thesis. The literature search was based on scientific literature from databases such as IEEE Xplore, JYKDOK, Google Scholar, ScienceDirect, and Scopus. When selecting the literature for the study the number of references and the impact on the field of information systems was considered, while aiming to correspond at least to level one of the Publication Forum's classification scales. Considering the newness of the topic, some gray literature was included to complement the understanding and definition of the topic.

The literature review focused on the first research question and investigated how an LLM chatbot could optimize software releases. Based on the literature review, an LLM chatbot could streamline release processes by automating repetitive tasks, sharing information, and improving communication between teams and stakeholders. In addition, chatbots could support DevOps and continuous

delivery (CD) practices by assisting with software management and deployment tasks.

The empirical study was conducted using the Design Science Research (DSR) method, starting with the identification of the problem and the definition of the solution goals. The development of the chatbot was described through the design and development phase. The data for their study was collected through a survey, followed by an analysis of the collected data, assessing its validity and reliability. The study was evaluated using the Balanced Scorecard (BSC), assisting in drawing conclusions from the results.

The empirical study answered the second research question. The identified software requirements for an LLM-based chatbot were categorized into functional and non-functional requirements. Functional requirements should facilitate access to and sharing of information from existing tools while ensuring reliable integration with internal systems. The non-functional requirements should emphasize fast, accurate, and reliable answers as well as learning from user interaction to improve answers over time. The results of the empirical study indicate the importance of both requirement types in ensuring the effectiveness of the chatbot in optimizing software release processes.

Both literature review and empirical study support the idea that software release processes could be optimized. An AI-based solution, such as a chatbot, could be one solution to address this need, given the advances and growing interest in AI technologies. However, the development and deployment of such solutions should consider both ethical and security aspects to avoid possible data leaks or legal violations.

From a theoretical perspective, this research contributes to the emerging field of combining LLM-based chatbot technologies and software release management. While Agile methods and AI applications have been widely studied, little attention has been paid to their intersection in this context. This study aimed to address this gap by proposing a structured framework for integrating LLM-based tools into release processes, providing a baseline for future academic research in this area.

From a practical perspective, the findings provide a useful starting point for organizations considering AI solutions in optimizing software release workflows. The software requirements specified in the study can serve as a basis for companies seeking to develop or deploy similar chatbot solutions. By identifying both functional and non-functional requirements the study highlights how to ensure smooth deployment and efficient performance in practice.

Future research is needed on this topic, as there is still a need for more in-depth research. Future research could investigate the integration of chatbots into software release processes and examine their applications in different industries. In addition, the role of chatbots in Agile methodologies could be investigated further by assessing chatbots' impact on improving Agile methodologies.

REFERENCES

Abbass, H. (2021). Editorial: What is Artificial Intelligence? *IEEE Transactions on Artificial Intelligence*, *2*(2), 94–95. https://doi.org/10.1109/tai.2021.3096243

Abdul-Kader, S. A., & Woods, J. (2015). Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, *6*(7). https://doi.org/10.14569/ijacsa.2015.060712

Agile product delivery. Scaled Agile Framework. (2024a, August 6). https://scaledagileframework.com/agile-product-delivery/

Agutter, C. (2020). *ITIL® 4 Essentials: Your essential guide for the ITIL 4 foundation exam and beyond*. IT Governance Publishing, 16-66.

Ahmad, M. O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 9-16. https://doi.org/10.1109/seaa.2013.28

Akkem, Y., Kumar, B. S., & Varanasi, A. (2023). STREAMLIT application for Advanced Ensemble Learning Methods in crop recommendation systems – a review and implementation. *Indian Journal Of Science And Technology*, *16*(48), 4688–4702. https://doi.org/10.17485/ijst/v16i48.2850

Alaidaros, H., Omar, M., & Romli, R. (2018). Identification of criteria affecting software project monitoring task of Agile Kanban method. *AIP Conference Proceedings, 2016*, 020021. https://doi.org/10.1063/1.5055423

Alnafessah, A., Gias, A. U., Wang, R., Zhu, L., Casale, G., & Filieri, A. (2021). Quality-aware DevOps research: Where do we stand*? IEEE Access, 9*, 44476–44489. https://doi.org/10.1109/access.2021.3064867

Alqudah, M., & Razali, R. (2016). A review of scaling agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology*, *6*(6), 828. https://doi.org/10.18517/ijaseit.6.6.1374

Anderson, D. J. (2010). *Kanban: successful evolutionary change for your technology business*. Blue hole press, 11-16.

Aouni, F. E., Moumane, K., Idri, A., Najib, M., & Jan, S. U. (2024). A systematic literature review on Agile, Cloud, and DevOps Integration: Challenges, benefits. *Information and Software Technology*, *177*, 107569. https://doi.org/10.1016/j.infsof.2024.107569

Bahja, M., Hammad, R., & Butt, G. (2020). A user-centric framework for Educational Chatbots Design and development. *Lecture Notes in Computer Science*, 32–43. https://doi.org/10.1007/978-3-030-60117-1_3

Banerjee, A., Banerji, R., Berry, J., Duflo, E., Kannan, H., Mukerji, S., Shotland, M., & Walton, M. (2017). From proof of concept to scalable policies:

Challenges and solutions, with an application. *Journal of Economic Perspectives*, *31*(4), 73–102. https://doi.org/10.1257/jep.31.4.73

Banica, L., Radulescu, M., Rosca, D., & Hagiu, A. (2017). Is DevOps another project management methodology? *Informatica Economica, 21(3/2017)*, 39–51. https://doi.org/10.12948/issn14531305/21.3.2017.04

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. https://agilemanifesto.org/

Berntzen, M., Moe, N. B., & Stray, V. (2019). The product owner in large-scale agile: An empirical study through the lens of relational coordination theory. *Lecture Notes in Business Information Processing*, 121–136. https://doi.org/10.1007/978-3-030-19034-7_8

Bodea, C.-N., Dascalu, M.-I., & Hang, A. (2020). Chatbot-based training for Project Management: Another way of corporate training or a must-have tool for sustainable education? *Lecture Notes in Management and Industrial Engineering,* 249–259. https://doi.org/10.1007/978-3-030-60139-3_17

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv (Cornell University).* https://doi.org/10.48550/arxiv.2005.14165

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, *15*(3), 1–45. https://doi.org/10.1145/3641289

Chavan, M. (2009). The balanced scorecard: A new challenge. *Journal of Management Development, 28*(5), 393–406. https://doi.org/10.1108/02621710910955930

Cīrule, D., & Bērziša, S. (2019). Use of chatbots in Project Management. *Communications in Computer and Information Science*, 33–43. https://doi.org/10.1007/978-3-030-30275-7_4

Cvejič, M. (2022). Evolution of agile practices during a M & A of a European and a Chinese company. *Open Journal of Business and Management, 10*(05), 2378–2388. https://doi.org/10.4236/ojbm.2022.105118

Dabbagh, M., Lee, S. P., & Parizi, R. M. (2015). Functional and non-functional requirements prioritization: Empirical evaluation of IPA, AHP-based, and

Ham-based approaches. *Soft Computing, 20*(11), 4497–4520. https://doi.org/10.1007/s00500-015-1760-z

Dennehy, D., & Conboy, K. (2017). Going with the flow: An activity theory analysis of flow techniques in software development. *Journal of Systems and Software, 133*, 160–173. https://doi.org/10.1016/j.jss.2016.10.003

Dietrich, J., Rasheed, S., & Tahir, A. (2022). Flaky test sanitisation via on-the-fly assumption inference for tests with network dependencies. *2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 264–275. https://doi.org/10.1109/scam55253.2022.00037

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87–108. https://doi.org/10.1016/j.jss.2016.06.013

Duncan, S. (2018). Safe 4.0 distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering. 2017. *Quality Management Journal*, *25*(1), 66–66. https://doi.org/10.1080/10686967.2018.1404375

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software, 33*(3), 94–100. https://doi.org/10.1109/ms.2016.68

Fagarasan, C., Popa, O., Pisla, A., & Cristea, C. (2021). Agile, waterfall and iterative approach in Information Technology Projects. *IOP Conference Series: Materials Science and Engineering*, *1169*(1), 012025. https://doi.org/10.1088/1757-899x/1169/1/012025

Faustino, J., Adriano, D., Amaro, R., Pereira, R., & da Silva, M. M. (2022). DevOps Benefits: A systematic literature review. *Software: Practice and Experience, 52*(9), 1905–1926. https://doi.org/10.1002/spe.3096

Federation of Finnish Learned Societies. (2022, November 21). *Publication Forum*. Publication forum. https://julkaisufoorumi.fi/en/publication-forum

Gandomani, T. J., & Nafchi, M. Z. (2016). Agile transition and adoption human-related challenges and issues: A grounded theory approach. *Computers in Human Behavior, 62,* 257–266. https://doi.org/10.1016/j.chb.2016.04.009

Gregor, S., & Hevner, A. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly, 37*, 337–356. https://doi.org/10.25300/MISQ/2013/37.2.01

Gruber, J. B., & Weber, M. (2024). rollama: An R package for using generative large language models through Ollama. *arXiv preprint arXiv:2404.07654*.

Gupta, M., Akiri, C., Aryal, K., Parker, E., & Praharaj, L. (2023). From ChatGPT to ThreatGPT: Impact of generative AI in cybersecurity and privacy. *IEEE Access, 11*, 80218–80245. https://doi.org/10.1109/access.2023.3300381

Hartmann, D., & Dymond, R. (2006). Appropriate Agile Measurement: Using metrics and diagnostics to deliver business value. *Agile Development Conference, AGILE 2006*, 126–134. https://doi.org/10.1109/agile.2006.17

Haugeland, I. K., Følstad, A., Taylor, C., & Bjørkli, C. A. (2022). Understanding the user experience of Customer Service Chatbots: An experimental study of chatbot interaction design. *International Journal of Human-Computer Studies*, *161*, 102788. https://doi.org/10.1016/j.ijhcs.2022.102788

Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: A case study at Ericsson. *Empirical Software Engineering, 22*(6), 2892–2936. https://doi.org/10.1007/s10664-016-9491-z

Heikkilä, V., Rautiainen, K., & Jansen, S. (2010). A revelatory case study on scaling agile release planning. *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, 289–296. https://doi.org/10.1109/seaa.2010.37

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Hill, J., Randolph Ford, W., & Farreras, I. G. (2015). Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in Human Behavior*, *49*, 245–250. https://doi.org/10.1016/j.chb.2015.02.026

Iden, J., & Eikebrokk, T. R. (2014). Using the ITIL process reference model for realizing IT governance: An empirical investigation. *Information Systems Management, 31*(1), 37–58. https://doi.org/10.1080/10580530.2014.854089

Jacobson, I., Sutherland, J., Kerr, B., & Buhnova, B. (2022). Better scrum through essence. *Software: Practice and Experience, 52*(6), 1531–1540. https://doi.org/10.1002/spe.3070

Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert scale: Explored and explained. *British Journal of Applied Science & Technology, 7*(4), 396–403. https://doi.org/10.9734/bjast/2015/14975

Kajko-Mattsson, M., & Yulong, F. (2005). Outlining a Model of a Release Management Process. *Journal of Integrated Design and Process Science, 9*(4), 13-25. doi:10.3233/JID-2005-9402

Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process, 30*(10). https://doi.org/10.1002/smr.1954

Kaplan, R. S. (2009). Conceptual Foundations of the balanced scorecard. *Handbooks of Management Accounting Research*, 1253–1269. https://doi.org/10.1016/s1751-3243(07)03003-9

Kaplan, R. S., & Norton, D. P. (1992). The balanced scorecard: measures that drive performance. *Harvard business review*, *70*, 71-79.

Knaster, R., & Leffingwell, D. (2020). *Safe 5. 0 distilled achieving business agility with the Scaled Agile Framework*. Addison-Wesley Professional, 36-50.

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum: Making the most of both.* C4Media, Inc, 1-49.

Komssi, M., Pichlis, D., Raatikainen, M., Kindstrom, K., & Jarvinen, J. (2015). What are hackathons for? *IEEE Software*, *32*(5), 60–67. https://doi.org/10.1109/ms.2014.78

Larsen, K. R., Lukyanenko, R., Mueller, R. M., Storey, V. C., VanderMeer, D., Parsons, J., & Hovorka, D. S. (2020). Validity in design science research. *Lecture Notes in Computer Science*, 272–282. https://doi.org/10.1007/978-3-030-64823-7_25

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps Concepts and challenges. *ACM Computing Surveys, 52*(6), 1–35. https://doi.org/10.1145/3359981

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems, 15(4*), 251–266. https://doi.org/10.1016/0167-9236(94)00041-2

Marchenko, A., & Abrahamsson, P. (2008). Scrum in a multiproject environment: An ethnographically-inspired case study on the adoption challenges. *Agile 2008 Conference*, 15–26. https://doi.org/10.1109/agile.2008.77

Marrone, M., & Kolbe, L. M. (2010). Uncovering ITIL claims: It executives' perception on benefits and business-IT alignment. *Information Systems and E-Business Management, 9*(3), 363–380. https://doi.org/10.1007/s10257-010-0131-7

McConnell, S. (1996). *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1-81.

Moe, N. B. (2013). Key challenges of improving agile teamwork. *Lecture Notes in Business Information Processing*, 76–90. https://doi.org/10.1007/978-3-642-38314-4_6

Mökander, J., Schuett, J., Kirk, H. R., & Floridi, L. (2023). Auditing large language models: A three-layered approach. *AI and Ethics*. https://doi.org/10.1007/s43681-023-00289-2

Nilsson Tengstrand, S., Tomaszewski, P., Borg, M., & Jabangwe, R. (2021) . Challenges of adopting safe in the banking industry – a study two years after its introduction. *Lecture Notes in Business Information Processing*, 157–171. https://doi.org/10.1007/978-3-030-78098-2_10

Nørreklit, H. (2003). The balanced scorecard: What is the score? A rhetorical analysis of the balanced scorecard. *Accounting, Organizations and Society, 28*(6), 591–619. https://doi.org/10.1016/s0361-3682(02)00097-1

Obwegeser, N., T. Nielsen, D., & M. Spandet, N. (2019). Continual process improvement for ITIL Service Operations: A lean perspective. *Information Systems Management, 36*(2), 141–167. https://doi.org/10.1080/10580530.2019.1587576

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2009). Using Scrum in distributed agile development: A multiple case study. *2009 Fourth IEEE International Conference on Global Software Engineering*, 195–204. https://doi.org/10.1109/icgse.2009.27

Paasivaara, M., Lassenius, C., Heikkila, V. T., Dikert, K., & Engblom, C. (2013). Integrating global sites into the Lean and agile transformation at Ericsson. *2013 IEEE 8th International Conference on Global Software Engineering*, 134–143. https://doi.org/10.1109/icgse.2013.25

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/mis0742-1222240302

Perkusich, M., Chaves e Silva, L., Costa, A., Ramos, F., Saraiva, R., Freire, A., Dilorenzo, E., Dantas, E., Santos, D., Gorgônio, K., Almeida, H., & Perkusich, A. (2020). Intelligent software engineering in the context of agile software development: A systematic literature review. *Information and Software Technology*, *119*, 106241. https://doi.org/10.1016/j.infsof.2019.106241

Putta, A., Paasivaara, M., & Lassenius, C. (2018). Benefits and challenges of adopting the Scaled Agile Framework (SAFE): Preliminary results from a multivocal literature review. *Lecture Notes in Computer Science*, 334–351. https://doi.org/10.1007/978-3-030-03673-7_24

Putta, A., Paasivaara, M., & Lassenius, C. (2019). How are agile release trains formed in practice? A case study in a large Financial Corporation. *Lecture Notes in Business Information Processing*, 154–170. https://doi.org/10.1007/978-3-030-19034-7_10

Raj, S. (2019). *Building Chatbots with python: Using natural language processing and machine learning*. Apress, 1-103.

Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatare, L. E., Seppänen, P., & Kuvaja, P. (2019). Advances in using agile and lean processes for software development. *Advances in Computers*, 135–224. https://doi.org/10.1016/bs.adcom.2018.03.014

Royce, W.W. (1970) *Managing the Development of Large Software Systems*. Proceedings of IEEE WESCON, 26, 328-388.

Sachdeva, S. (2016). Scrum methodology. *International Journal of Engineering And Computer Science*. https://doi.org/10.18535/ijecs/v5i6.11

Safe 6.0 framework. Scaled Agile Framework. (2024b, October 15). https://scaledagileframework.com/#overview

Samer, M. (2016). Software Release Management Evolution-Comparative Analysis across Agile and DevOps Continuous Delivery. *International Journal of Emerging Trends & Technology in Computer Science, 3,* 2349-6495.

Schaefer, A., Reichenbach, M., & Fey, D. (2012). Continuous Integration and automation for Devops. *Lecture Notes in Electrical Engineering*, 345–358. https://doi.org/10.1007/978-94-007-4786-9_28

Schmidt, C. (2016). *Agile Software Development Teams*. Springer International Publishing, 1-36.

Schwaber, K., & Sutherland, J. (2020). *The 2020 Scrum GUIDE*. Scrum Guide | Scrum Guides. https://scrumguides.org/scrum-guide.html

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *MIS Quarterly, 35*(1), 37–56. https://doi.org/10.2307/23043488

Setiaji, B., & Wibowo, F. W. (2016). Chatbot using a knowledge in database: Human-to-machine conversation modeling. *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS).* https://doi.org/10.1109/isms.2016.53

Shahin, M., Babar, M. A., & Zhu, L. (2016). The intersection of continuous deployment and architecting process. *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–10. https://doi.org/10.1145/2961111.2962587

Singh, S., & Beniwal, H. (2022). A survey on near-human conversational agents. *Journal of King Saud University - Computer and Information Sciences*, *34*(10), 8852–8866. https://doi.org/10.1016/j.jksuci.2021.10.013

Šmite, D., Moe, N. B., & Gonzalez-Huerta, J. (2021). Overcoming cultural barriers to being agile in distributed teams. *Information and Software Technology*, *138*, 106612. https://doi.org/10.1016/j.infsof.2021.106612

Sridhara, G., G, R. H., & Mazumdar, S. (2023). ChatGPT: A Study on its Utility for Ubiquitous Software Engineering Tasks. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2305.16837

Ståhl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software, 87*, 48–59. https://doi.org/10.1016/j.jss.2013.08.032

Stange, R., Schiele, H., & Henseler, J. (2022). Advancing purchasing as a design science: Publication guidelines to shift towards more relevant purchasing

research. *Journal of Purchasing and Supply Management, 28*(1), 100750. https://doi.org/10.1016/j.pursup.2022.100750

Stanica, I., Dascalu, M.-I., Bodea, C. N., & Bogdan Moldoveanu, A. D. (2018). VR job interview simulator: Where virtual reality meets artificial intelligence for education. *2018 Zooming Innovation in Consumer Technologies Conference (ZINC)*. https://doi.org/10.1109/zinc.2018.8448645

Stokel-Walker, C., & Van Noorden, R. (2023). What CHATGPT and Generative AI mean for science. *Nature*, *614*(7947), 214–216. https://doi.org/10.1038/d41586-023-00340-6

Storey, V. C., Baskerville, R. L., & Kaul, M. (2017). Reliability in design science research. *38th International Conference on Information Systems (ICIS 2017), 1*, 1688–1705.

Templier, M., & Paré, G. (2015). A framework for guiding and evaluating literature reviews. *Communications of the Association for Information Systems*, *37*. https://doi.org/10.17705/1cais.03706

Theobald, S., & Schmitt, A. (2020). Dependencies of Agile Teams – an analysis of the Scaled Agile Framework. *Lecture Notes in Business Information Processing*, 219–226. https://doi.org/10.1007/978-3-030-58858-8_22

Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile Versus Waterfall Project Management: Decision model for selecting the appropriate approach to a project. *Procedia Computer Science*, *181*, 746–756. https://doi.org/10.1016/j.procs.2021.01.227

Topsakal, O., & Akinci, T. C. (2023). Creating large language model applications utilizing LangChain: A primer on developing LLM Apps Fast. *International Conference on Applied Engineering and Natural Sciences*, *1*(1), 1050–1056. https://doi.org/10.59287/icaens.1127

Vallon, R., Strobl, S., Bernhart, M., & Grechenig, T. (2013). Inter-organizational co-development with scrum: Experiences and lessons learned from a distributed corporate development environment. *Lecture Notes in Business Information Processing*, 150–164. https://doi.org/10.1007/978-3-642-38314-4_11

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30. 31st Conference on Neural Information Processing Systems*. CA, USA: Long Beach.

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: A framework for evaluation in Design Science Research. *European Journal of Information Systems, 25*(1), 77–89. https://doi.org/10.1057/ejis.2014.36

Virmani, M. (2015). Understanding DevOps & Bridging the gap from continuous integration to continuous delivery. *Fifth International Conference*

on the Innovative Computing Technology (INTECH 2015). https://doi.org/10.1109/intech.2015.7173368

Wakode, R. B., Raut, L. P., & Talmale, P. (2015). Overview on kanban methodology and its implementation. *IJSRD-International Journal for Scientific Research & Development, 3*(02), 2321-0613.

Xu, F., Lin, Q., Han, J., Zhao, T., Liu, J., & Cambria, E. (2023). Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2306.09841

Yin, R. K. (2009). *Case study research: Design and methods*. SAGE Publications, 40-45.

Zhao, Y., Srebrenica, A., Zhou, Y., Filkov, V., & Vasilescu, B. (2017). The impact of continuous integration on other software development practices: A large-scale empirical study. *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE),* 60–71. https://doi.org/10.1109/ase.2017.8115619

APPENDIX 1 THE EMAIL FOR THE SURVEY

Subject: Survey of Identifying Requirements for a Release Chatbot

"Hi,

I hope this email finds you well.

We're conducting a study to gather insights into a new AI-powered release chatbot. The chatbot is intended to assist with various tasks, such as streamline communication, automate repetitive tasks, and to provide real-time information. Your input ensures that the chatbot can meet the needs to optimize the release processes.

The survey collects feedback on how the chatbot could enhance your daily work and interactions. No prior knowledge of AI nor chatbots are needed to participate. The survey is straightforward and quick to answer. It is part of my master's thesis, which is being carried out within the organization.

Deadline: Please submit your responses by Friday 11.10.2024

Your participation will help develop a tool that could significantly enhance productivity and efficiency within the release processes. We appreciate your time and feedback.

Please note that all responses are completely anonymous, and your data won't be stored or shared. The results are used solely for research purposes to help define the software requirements of the chatbot.

In case of any questions please do not hesitate to contact me.
Thank you in advance.

Best Regards,
Sini Annamaa"

APPENDIX 2 THE STRUCTURE OF THE SURVEY

Identifying Software Requirements for a LLM Release Chatbot in Release Processes

We are considering implementing a LLM (Large Language Model)-powered chatbot to assist with the release processes. The chatbot is intended to help automate certain tasks, provide real-time information, and improve communication. Your feedback will help us determine the most valuable features for the chatbot to enhance productivity and efficiency across teams. Please answer the following questions based on your daily tasks and experiences. No prior knowledge of chatbots is required.

Privacy and Confidentiality Statement

Your responses to the survey are anonymous and won't be linked to any personally identifiable information. The collected data will be used solely for research purposes to help define the software requirements for a potential LLM chatbot in the release process. Please note that *no data will be stored*, and your responses won't be shared with anyone outside of the research team. By proceeding, you acknowledge that you understand and agree to these terms.

* Required

1. Which of the following best describes your current role? *

- ◯ Developer
- ◯ Architect
- ◯ QA / QA lead / Test Analyst
- ◯ Product Manager / Product Owner / Business Analyst
- ◯ Application Provider / Application Owner / Technology Service Lead
- ◯ Release Manager / Service Manager / Infrastructure Specialist / DevOps Engineer
- ◯ RTE / STE / Hub Lead
- ◯ Scrum Master / Agile Coach
- ◯ Risk Manager / Security Specialist

2. Please rate each statement on how much you agree with it. *

	Strongly Disagree	Disagree	Neutral	Agree
If a chatbot could automate routine tasks (e.g., gathering information) it could assist you in your daily work?	◯	◯	◯	◯
If a chatbot could provide automatic updates or share relevant information it could improve collaboration between your team and others?	◯	◯	◯	◯
If a chatbot could provide updates on project status (e.g., release progress, bug tracking), do you believe this could improve your ability to stay informed?	◯	◯	◯	◯
A chatbot that communicates important release information to stakeholders could reduce the need for manual reports. Do you think this would improve your communication efficiency?	◯	◯	◯	◯
If a chatbot could help detect and suggest solutions to bottlenecks in the release progress, would this improve the speed and efficiency of releases?	◯	◯	◯	◯

Imagine a chatbot that can instantly retrieve the latest project information (e.g., issue statuses, release readiness). Would this help streamline your work?	○	○	○	○
A chatbot could automatically generate release notes, meeting summaries or status reports based on project data. Would this automation make your role in the release process more efficient?	○	○	○	○
If chatbot could take over simple tasks (e.g., retrieving project data), do you believe it would help your team focus on more critical and value-adding activities?	○	○	○	○
By offering quick answers, providing relevant information, and identifying potential issues early, the chatbot could help avoid delays in the release process. Do you think this would reduce project costs?	○	○	○	○

If the chatbot could handle routine tasks, such as answering to common questions, would this help reduce the overall manual workload for your team?	○	○	○	○
Imagine a chatbot that provides recommendations based on historical data and past release performance. Do you believe this would support better decisions-making in your role?	○	○	○	○
If a chatbot could take on routine queries and help with quick information retrieval, would it reduce the mental effort required for simple tasks and free up your time for more strategic work?	○	○	○	○
A chatbot that supports your daily tasks by automating processes and offering timely information could improve both personal and team productivity. Do you think this would positively affect your productivity?	○	○	○	○

If the chatbot could help track patterns or learn from previous release cycles, do you think this would improve team learning and the ability to optimize future releases?	○	○	○	○

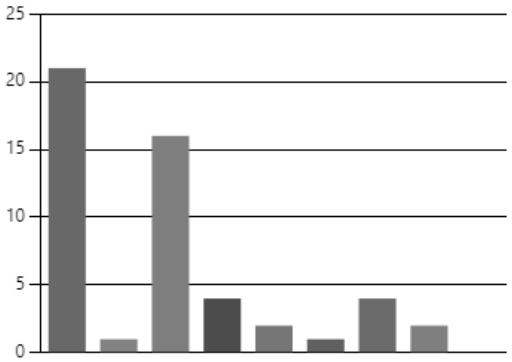
3. Please share any additional feedback on how you think a chatbot could improve your workflow, or specific features that would be valuable to you.

Enter your answer

APPENDIX 3 THE RESULTS OF THE SURVEY

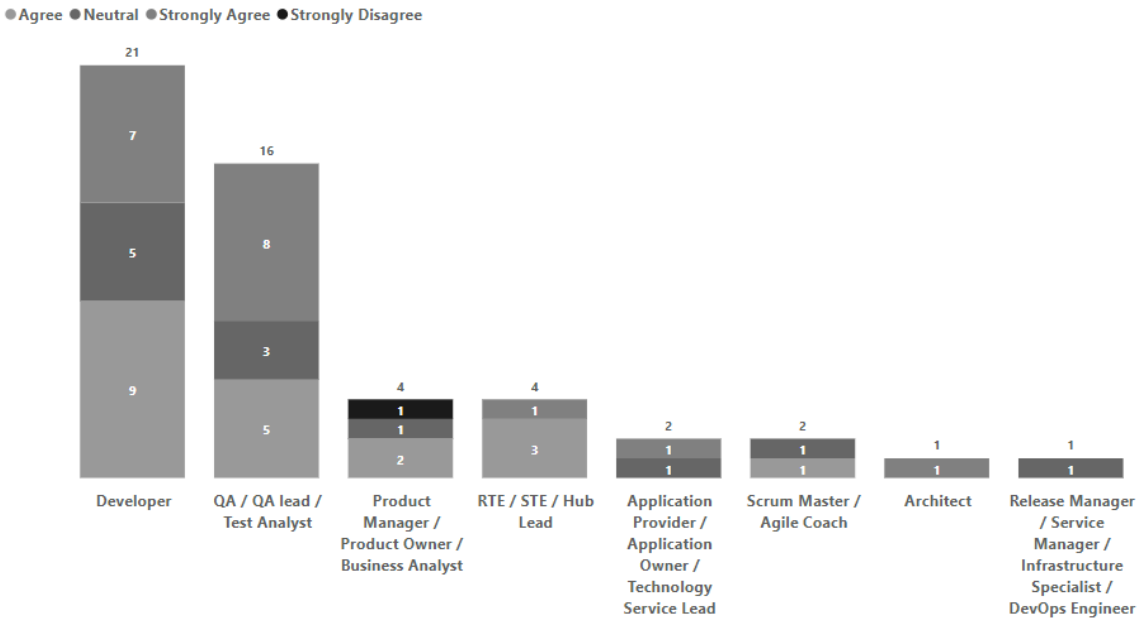
1. Which of the following best describes your current role?

Developer	21
Architect	1
QA / QA lead / Test Analyst	16
Product Manager / Product Ow...	4
Application Provider / Applicati...	2
Release Manager / Service Man...	1
RTE / STE / Hub Lead	4
Scrum Master / Agile Coach	2
Risk Manager / Security Specialist	0

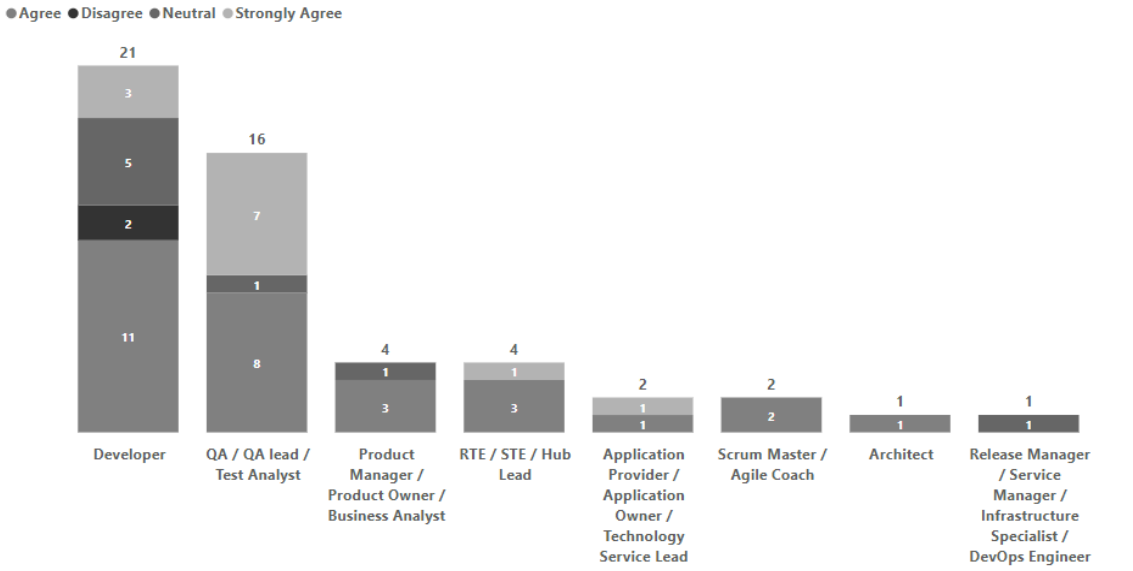


2. Please rate each statement on how much you agree with it.

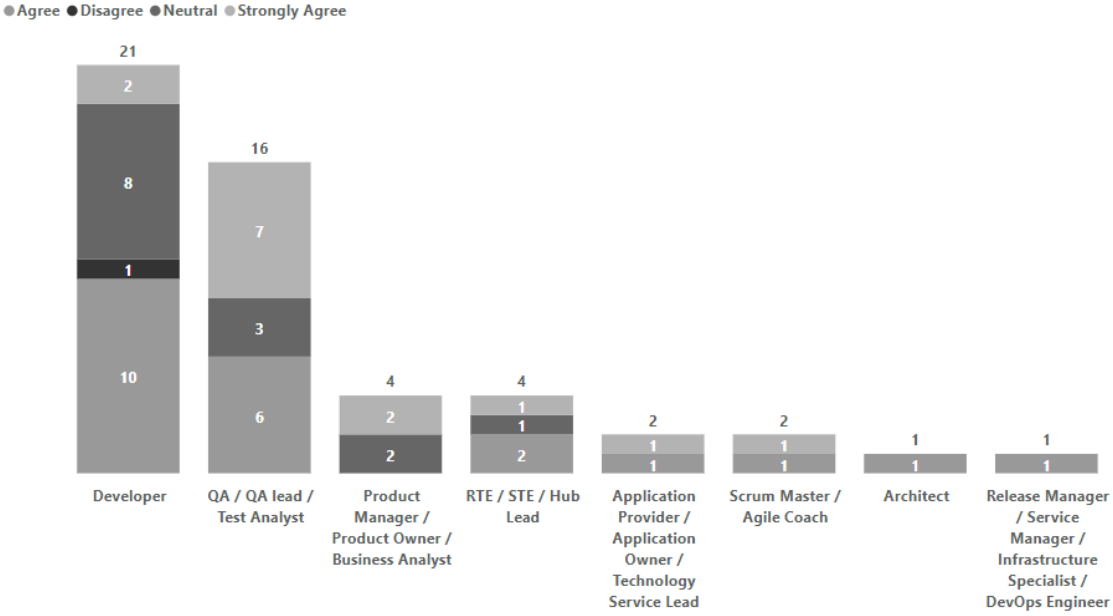
Q1: If a chatbot could automate routine tasks (e.g., gathering information) it could assist you in your daily work?



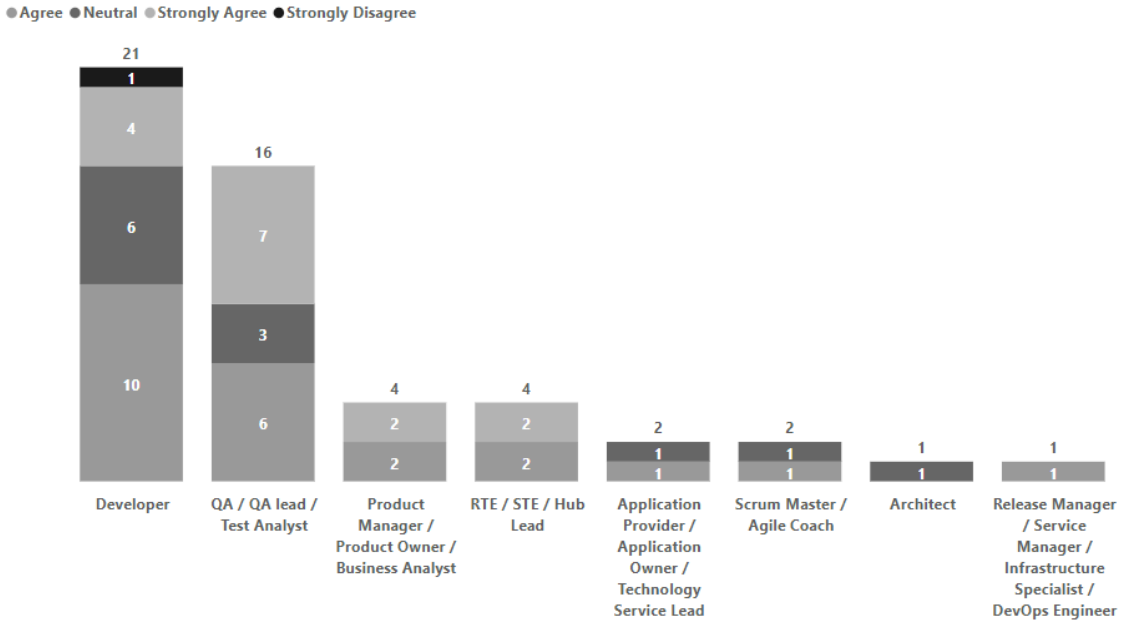
Q2: If a chatbot could provide automatic updates or share relevant information it could improve collaboration between your team and others?



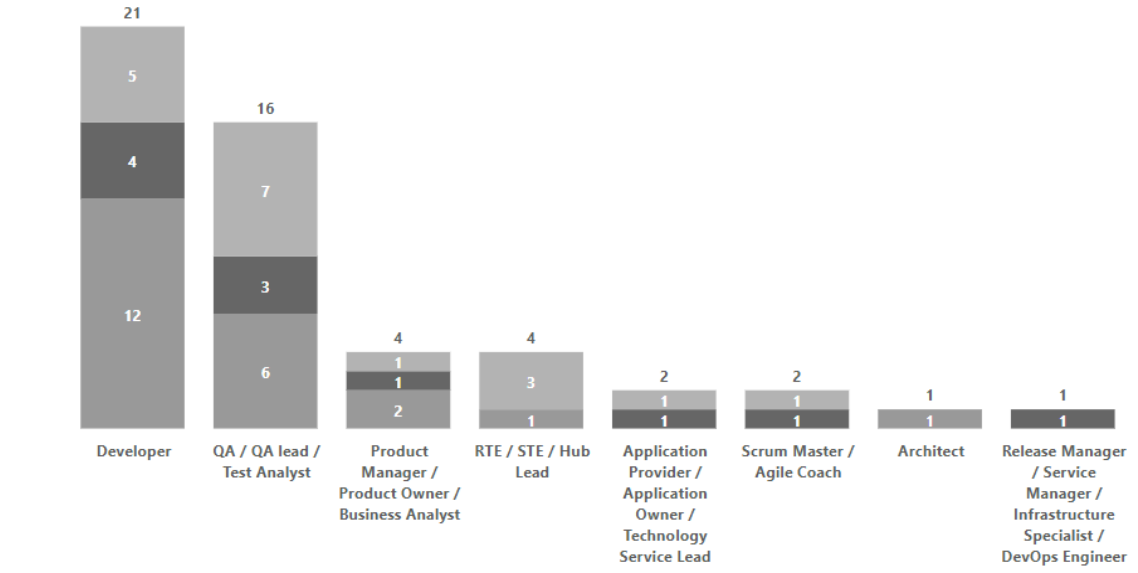
Q3: If a chatbot could provide updates on project status (e.g., release progress, bug tracking), do you believe this could improve your ability to stay informed?



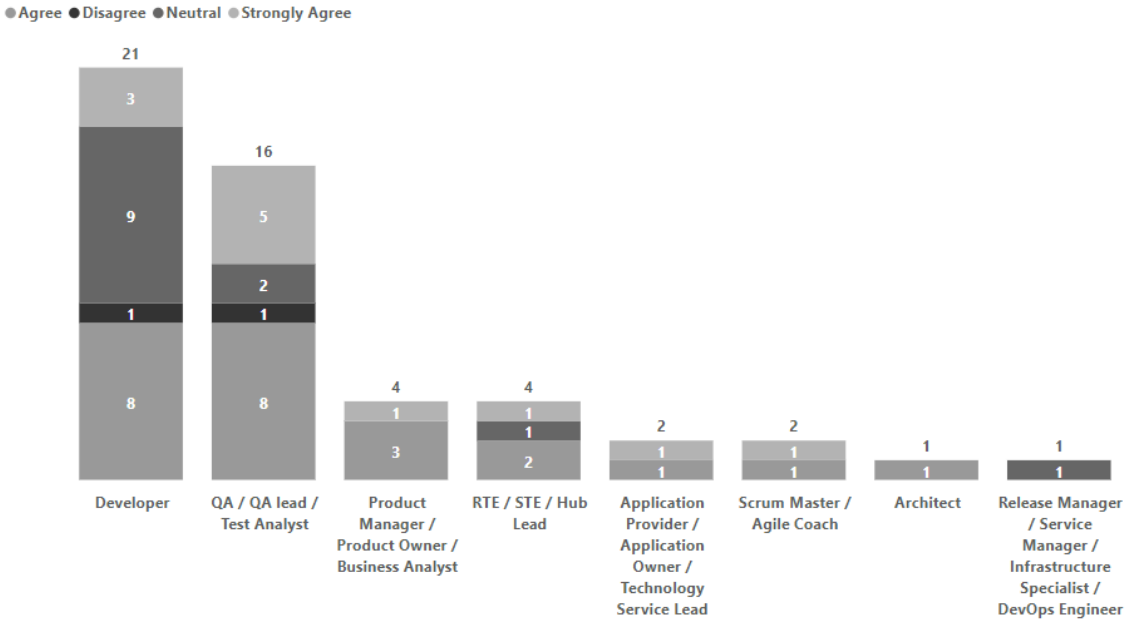
Q4: A chatbot that communicates important release information to stakeholders could reduce the need for manual reports. Do you think this would improve your communication efficiency?



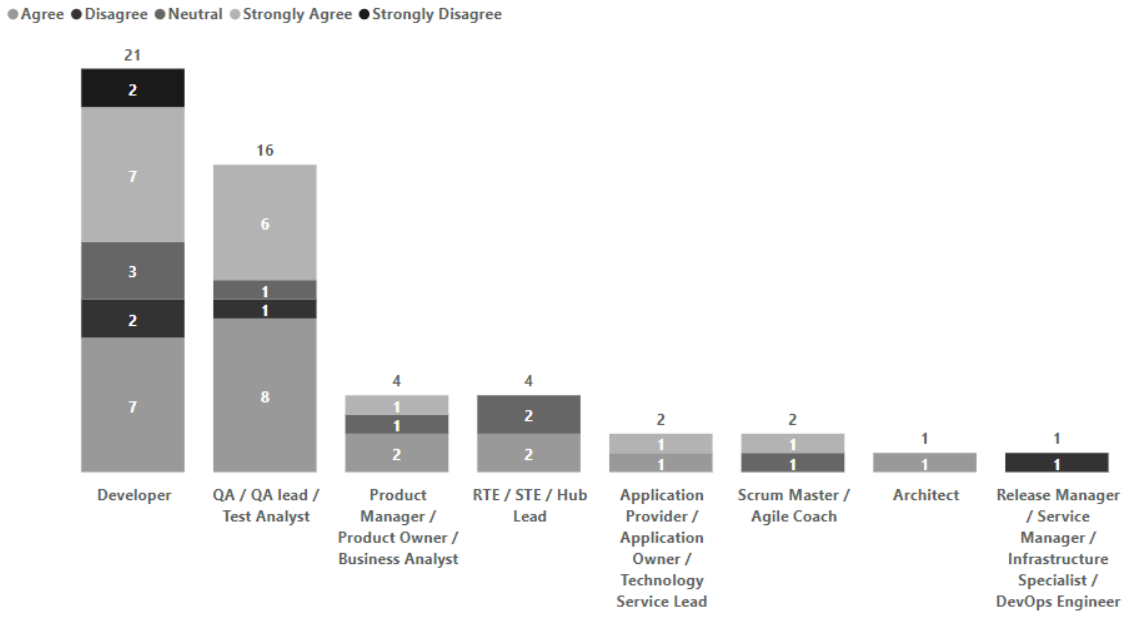
Q5: If a chatbot could help detect and suggest solutions to bottlenecks in the release progress, would this improve the speed and efficiency of releases?



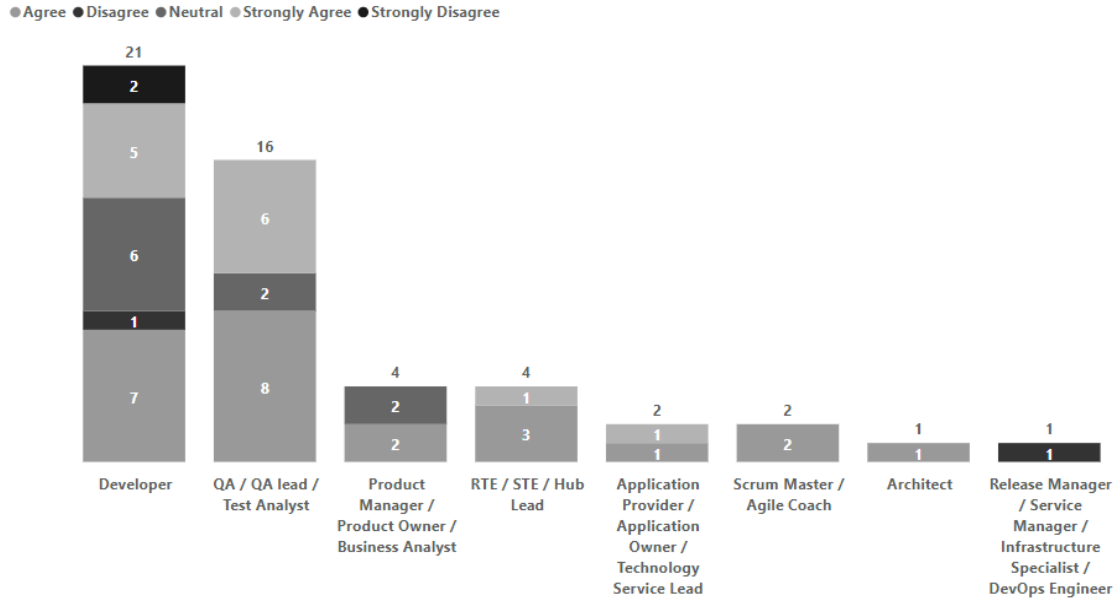
Q6: Imagine a chatbot that can instantly retrieve the latest project information (e.g., issue statuses, release readiness). Would this help streamline your work?



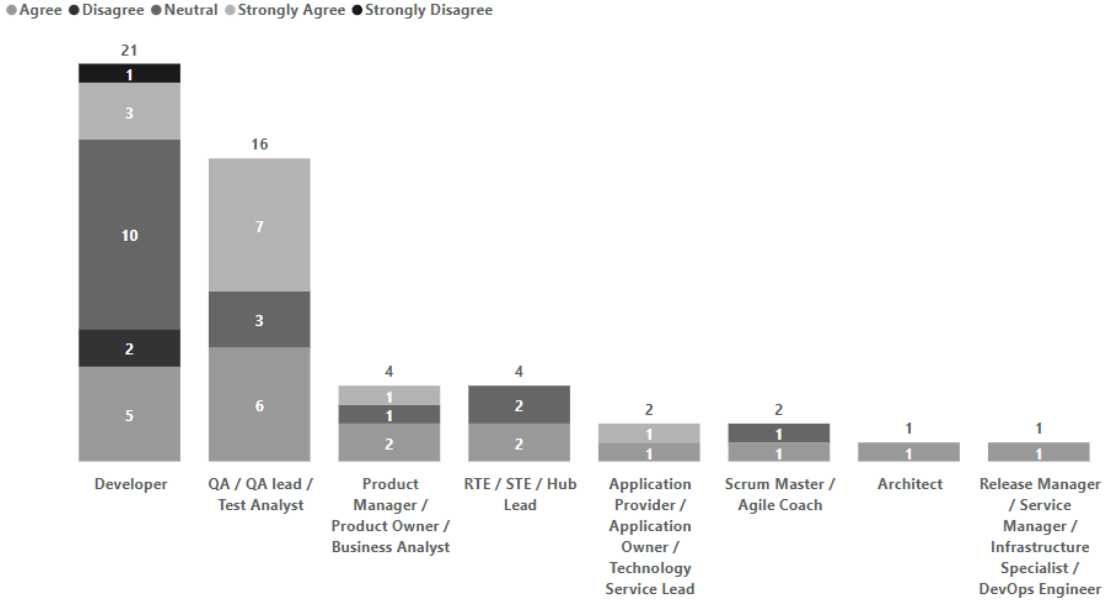
Q7: A chatbot could automatically generate release notes, meeting summaries or status reports based on project data. Would this automation make your role in the release process more efficient?



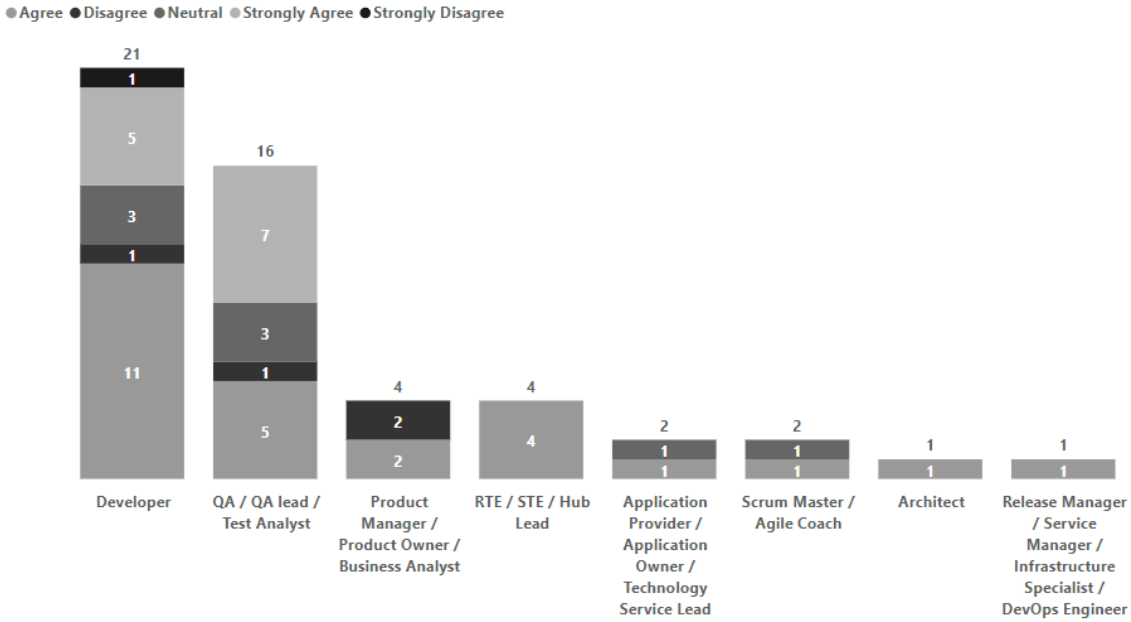
Q8: If chatbot could take over simple tasks (e.g., retrieving project data), do you believe it would help your team focus on more critical and value-adding activities?



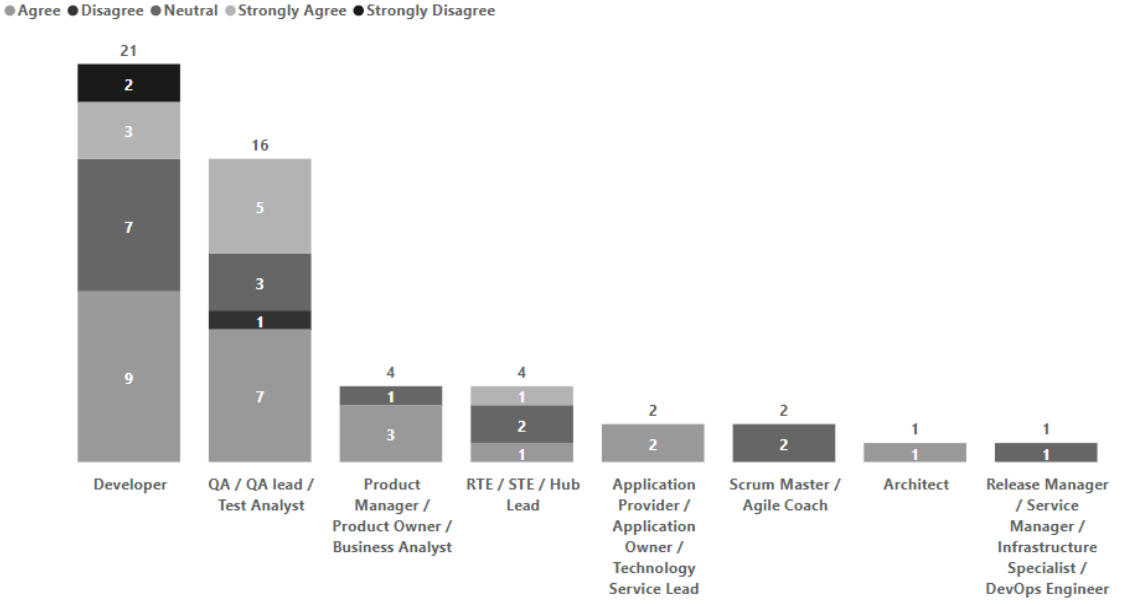
Q9: By offering quick answers, providing relevant information, and identifying potential issues early, the chatbot could help avoid delays in the release process. Do you think this would reduce project costs?



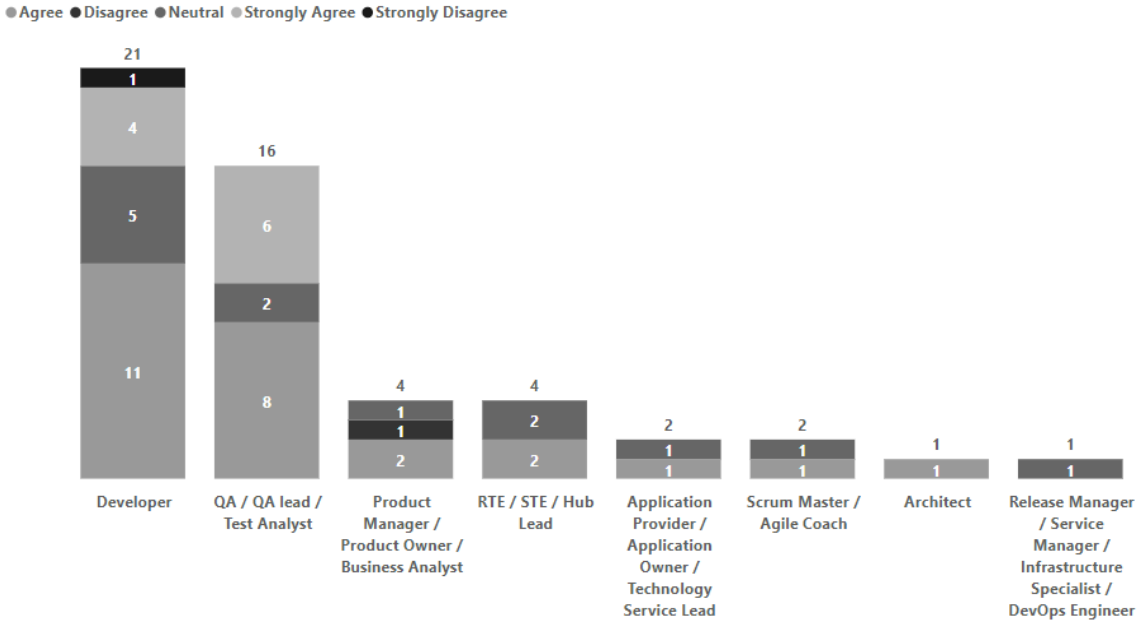
Q10: If the chatbot could handle routine tasks, such as answering to common questions, would this help reduce the overall manual workload for your team?



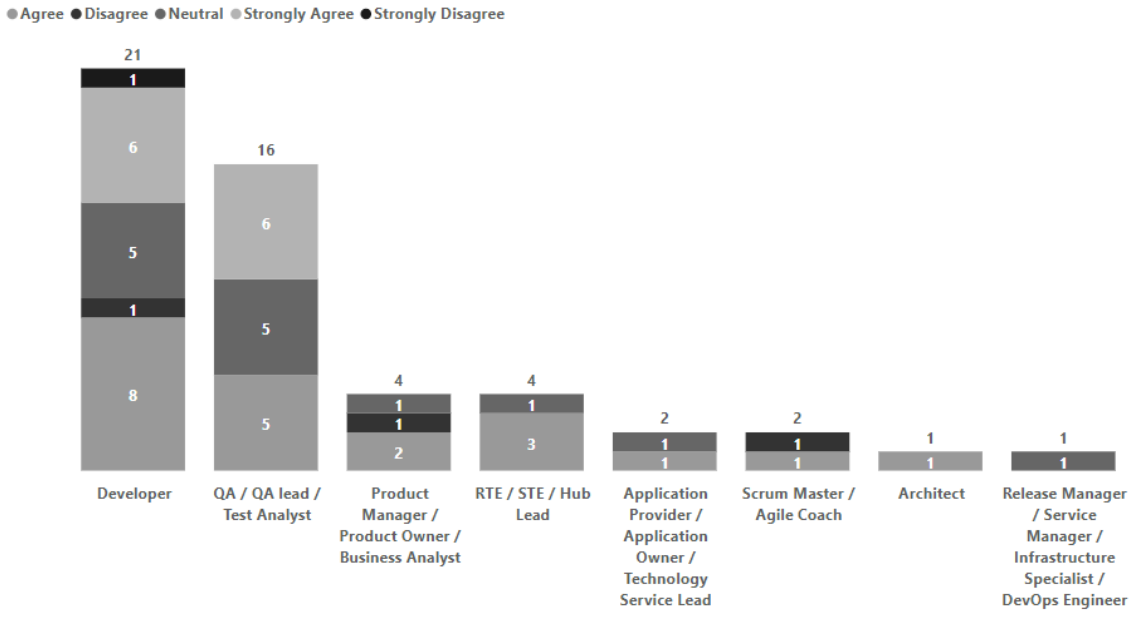
Q11: Imagine a chatbot that provides recommendations based on historical data and past release performance. Do you believe this would support better decisions-making in your role?



Q12: If a chatbot could take on routine queries and help with quick information retrieval, would it reduce the mental effort required for simple tasks and free up your time for more strategic work?



Q13: A chatbot that supports your daily tasks by automating processes and offering timely information could improve both personal and team productivity. Do you think this would positively affect your productivity?



Q14: If the chatbot could help track patterns or learn from previous release cycles, do you think this would improve team learning and the ability to optimize future releases?