CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Creating a Dependency Management DevBot to Improve Developer Workflows

A Design Science Research Study at Ericsson

Master's thesis in Computer science and engineering

Michalis Kaili, Fredrik Ullman

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

MASTER'S THESIS 2023

Creating a Dependency Management DevBot to Improve Developer Workflows

A Design Science Research Study at Ericsson

Michalis Kaili, Fredrik Ullman

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Creating a Dependency Management DevBot to Improve Developer Workflows
A Design Science Research Study at Ericsson
Michalis Kaili, Fredrik Ullman,

Supervisor: Linda Erlenhov, Computer Science and Engineering
Advisor: Johan Nicander, Ericsson
Examiner: Hans-Martin Heyn, Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and the University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Creating a Dependency Management DevBot to Improve Developer Workflows
A Design Science Research Study at Ericsson
MICHALIS KAILI
FREDRIK ULLMAN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Software Development Bots (DevBots) are automated tools that handle tasks in a software development setting, often used to simplify daily routines. Research within the field evaluates the use of DevBots but most focus on hard quantitative metrics (commits and pull requests) and only a few qualitative metrics such as how satisfied the developer is with the new DevBot and what frameworks could be used to build it. In this thesis, we use design science research to find obstacles within two teams at Ericsson. We structured the thesis into two iterations. In the first iteration we collect data and construct two initial DevBots, in iteration two we single out one DevBot, improve it, and evaluate it. The interview gave us the two main obstacles that we focused on for the DevBots: the migration of ticket data between two instances, and updating dependencies within a software project. We evaluated both DevBots through small tasks embedded in surveys sent out to potential users at Ericsson. We then chose one DevBot, Pepe, and conducted a focus group for the final validation of the tool. We had six participants in the interviews and the surveys and four in the focus group. Additionally, we investigate and list frameworks we tried to build our DevBot with, along with what benefits and drawbacks each of them has. We found that the choice of framework is heavily based on requirements, which means that there is no silver bullet. We conclude that future research should focus on expanding the list of frameworks, widening the use case for the chosen DevBots, and continuing the work to simplify the construction of DevBots as well as investigating the impact they can have on developers' lives. We argue that guidelines on building DevBots using frameworks and approaching obstacles need to be further investigated.

Keywords: Development Bot, Design Science Research, DevBot Frameworks

Acknowledgements

We owe much gratitude to our supervisors Linda Erlenhov and Francisco Gomes de Oliveira Neto. We appreciate that you patiently walked us through the world of DevBots. We would like to thank Garaget, the innovation hub at Ericsson and our industry supervisor Johan Nicander, for your input and assistance with all resources, and for making us feel very welcome at the company. We are very grateful to both teams at Ericsson that invested time in us. We would lastly like to thank Helena Eberil for organising and making the research project together with Ericsson possible.

Michalis Kaili, Fredrik Ullman, Gothenburg, June 2023

Contents

List of Figures

List of Tables

1
Introduction

Software Development is a difficult, multi-stage, time consuming, repetitive, error filled process. This is why developers need all the help they can get. This help may come in the form of years of experience, specialised courses/training, tools such as IDEs or automated testing generation etc. All of these are forms by which developers are assisted in their work environment in order to accomplish their goals and tasks as efficiently and accurately as possible. Many development tools have been augmented to act as bots, meaning that they are sophisticated scripts meant to automate or assist with processes and task but also have several human-like characteristics.

Bots for software development, or DevBots [1], are becoming increasingly popular for developers. DevBots are used to simplify software related tasks of a developer by performing a variety of tasks. To showcase different types of bots we use Erlenhov et al.'s personas [2], which indicates what type of tools different developers think about when they hear the word *bot*. The personas are named Alex, Sam and Charlie. The various personas are defined based on what they (the user) expect a DevBot to do. For instance, Alex expects *autonomy*, the Sam persona thinks of a DevBot as a *smart* type of tool and Charlie expects a type of *chatbot*. A multitude of DevBots exist i.e., sophisticated tools that fulfil specific criteria.For example Alex would most likely prefer a bot like Dependabot [1] whereas Sam might lean more towards a bot like Hubot [2] and finally Charlie might apprecieate a bot like ChatGPT [3] The main objective of all these tools is to make the development of software faster and easier for software developers. DevBots are fairly novel to the industry and as such have not been researched adequately. They are versatile and this gives them the ability to be incorporated in many types of fields and domains within software development. With that said, we believe that they can improve productivity and SPACE is a framework that can be utilised to measure exactly that. DevBots and their impact on developers have not been sufficiently evaluated using metrics from the SPACE [3] framework, nor have the frameworks used to create DevBots been covered in the literature. Additionally we collected data on the various frameworks that we used along the way, in order to share do's and dont's when getting started with DevBot construction.

We define productivity in this paragraph as an increase in quality of the workflow and development. By providing quality of life improvements to developers we hope

[1] https://github.com/dependabot

[2] https://hubot.github.com/

[3] https://chat.openai.com/

to help them be more productive and efficient which in turn translates to better products, services, systems, faster development, deployment and better maintainability. Therefore these improvements directly correlate to an increase in profits for any company that chooses to integrate them in their workplace. Improving productivity in a software team is, quite reasonably, considered a key concern [4]. With this thesis we hope to increase knowledge on development bots and to help developers in their workflow. We also explore and evaluate DevBot frameworks, as a means to assist developers and researchers getting started.

We aim to help developers as well but with the use of DevBots. As explained before, DevBots are tools that can be perceived differently by every individual, but regardless of what a DevBot is perceived to be, at the end of the day its job is to assist developers achieve their goals. We investigated what needs our developers have, how a DevBot can help them with those particular needs, and finally we developed two DevBots one of which was later refined further. To that end we developed a DevBot that was then tested, evaluated, and improved upon for several iterations before it can hopefully be fully integrated into a development teams' workflow as an integral part of their day to day process. We also found that there are many frameworks existing for building DevBots, where documentation can be scarce, communities could be idle, there could be unforeseen costs in free tools, a framework might not actually integrate with the various channels that it promises, and more. We aim to assist developers with finding the framework that suits their use case the best. We do so by evaluating a list of chosen frameworks.

Therefore, the purpose of this study is to evaluate common obstacles in software projects that could be avoided, fixed or improved using DevBots as well as to report on the DevBot framework landscape to figure what is out there, their benefits, drawbacks and potential uses in similar scenarios like ours. Furthermore, we hope to shed more light upon the world of DevBots in general because it is a novel and pioneering field that has much to offer but is relatively unknown. The thesis investigated the workflow and processes of two development teams through interviews, surveys and a focus group. We evaluated the most common tasks that developers struggle or need assistance with in the development cycle. The DevBot will then be designed as a proof of concept to complete these tasks in place or alongside of the human developer.

We pin pointed particular processes that the development team felt that they need assistance with, through interviews at Ericsson. During these, the participants received an explanation of what a DevBot is before trying to figure out together how a DevBot can provide the most amount of value and help to the developers. It should be made clear that the purpose of the DevBot and in extension, of this study, is not to overhaul the entire workflow but to provide assistance to developers in an area they feel is most needed and where other tools either cannot or do not exist to do so.

By providing a better understanding of what DevBots are, we are assisting in opening the doors for more research, studies and development to take place and elevate DevBots to their full potential. We use the word showcasing as we intend to display what DevBots could do in order to solve the various tasks.

The beneficiaries of this thesis will be the scientific community, developers as well as the software industry at large. Our findings relate to the improved work life of developers, and their desire for tools like these along with the application of our process in the context of a large company like Ericsson can be utilised not only for the development of new DevBots for a variety of large corporations but also for a deeper investigation on their effects in developers lives.

Developers will benefit directly since this thesis is meant to not only create a tool for a specific DevBot but provide a concrete and repeatable process that can be used in order to create DevBots suitable for any team in any company. In addition, this also implies that companies will indirectly benefit from this study since they will be influenced by the increased potential productivity of their developers.

We achieve the following contributions and deliverables to both researchers and practitioners working with DevBots: i)We present a list of obstacles that developers face (RQ1), ii)we construct a DevBot that solves a dependency management obstacle, one of the obstacles elicited (RQ2) and iii)we provide two sets of guidelines (RQ2) for two different DevBots Pepe and Teddy. The guidelines aim to solve two main obstacles i.e., dependency management and migrating ticket information, elicited from interviews. iv) We evaluate the DevBot (RQ3) using the SPACE framework. v) We also provide benefits and disadvantages for various frameworks that could be used when creating DevBots (RQ4).

The structure of the thesis is as follows: in Background & Related work we present definitions, acronyms and other information needed to understand DevBots, frameworks and developer workflows, as well as related research that we used and based our work on. In Methodology, we cover the method used (Design science) and motivate how we structured our work process. In the Implementation chapter, we introduce the artifact(s). The Result section covers our findings from each iteration and lastly, we analyse and conclude our work in Discussion.

2
Background and Related work

In this section of our thesis we will introduce the various definitions, concepts and terminology that are needed for understanding our research. We also investigate previous research that relates to our field of study. We will begin by conveying what is (and is not) a DevBot. Next, we provide a brief history of DevBots research. Some key concepts that we define for the context of our research are: *Use cases of DevBots*, *examples of DevBots*, *DevBot personas*, *frameworks*, *evaluating workflow* as well as *benefits and disadvantages* of using DevBots.

2.1 To be or not to be a DevBot

The general term "bot" is short for "software robot" and refers to a software program that can perform automated tasks [5]. In order to distinguish between the various types of bots, we use the term DevBots in this thesis to refer to a type of software bot that is in one way or another used in software development. Software Bots are commonly used to interact with software services. A Software bot can take the shape of a conversational user interface, an intelligent script or an agent that performs tasks [6]. It is essential to understand and define the term DevBot so that eventual hypotheses, discussions and research questions posited are consistent. In turn, a DevBot is a type of Software bot which is able to perform a plethora of tasks, these tasks are focused on simplifying procedures within software engineering (e.g., summarising stand-up meetings, creating or analysing code, scheduling or maintaining Continuous Integration pipelines). Erlenhov et al. [1] mention that the ideal DevBot is largely self-sufficient, technically knowledgeable and holds some social competence. Storey & Zagalsky define a bot as an *" [...] application that automates repetitive or predefined tasks"* [4]. A DevBot usually acts based on predefined information, using so-called static knowledge. Such a DevBot could leverage machine learning but would still not learn and change its behaviour, hence its name, "static". Erlenhov et al. [2] define the term, writing "[...] a DevBot is a tool that exhibits some human-like traits and automatically executes a task.". To conclude the definition of a DevBot, Erlenhov et al. [1] refers to a Software Bot whose purpose is to support software development.

A DevBot should not be confused with other types of utilities [1] such as Plain Old Development Tools (PODTs) [2] or mechanically devised robots. Other examples of what is not considered a DevBot are chatbots that are not used explicitly for development purposes that you may find yourself communicating with on websites.

According to Santhanam et al. [5], another example of what is not a Devbot, is a tool for continuous integration that only check for build failures. Erlenhov et al. differentiate DevBots from voice-activated assistants for non-development usage, such as Apple's Siri, Samsung's Bixby or Google Assistant [1]. The voice-activated personal bots are currently also heavily leveraging AI to assist humans, a feature of which DevBots have only briefly been introduced to.

Although DevBots have been given a definition that does not necessarily mean that they are understood the same way by everyone. Erlenhov et al. build their work on previous taxonomies as well as their own data from interviews which resulted in the creation of personas to characterise people, whereas Lebeuf and Storey [6, 7], classify bots by purpose such as *knowledge*, *generalist* or *transactional*. To properly communicate and discuss the topic of DevBots, Erlenhov et al. [2] investigated how people perceive DevBots, what they are, and what they do before dividing their findings into personas. Three personas are assigned: Sam, Charlie, and Alex. Sam is a user that expects a smart bot that learns as it goes, probably based on or built using machine learning or AI. Charlie is a user that thinks of DevBots as chatbots, with an interface that allows developers to communicate with the DevBot using natural or strict language. Alex is a user that believes that a DevBot is an autonomous bot, usually performing some type of task in the background, without a prominent visual interface [2]. We had to make sure to take these personas into account before developing our DevBot. This is because one of the challenges of constructing a good and useful DevBot is meeting the expectations and desires of the developers that will use it. To overcome these obstacles, we needed to bridge the gap between what we know about DevBots and what developers perceive as a DevBot. Taking into consideration Erlenhov et al.'s [2] findings, we used the interviews with our participants to figure out in what personas they gravitated towards and to get a better understanding of what they expected from the final product. The combination of Erlenhov et al.' s [2] work and the data from our interviews provided us with the necessary information to move forward with development with more certainty and confidence that the artifact we would create would meet our users' expectations and desires.

2.2 History and Use Cases of DevBots

Even though the notion of DevBots and Software Bots are fairly novel, the general idea of replacing manual tasks with a tool that has human-like traits has been on people's minds since around 1966 [6]. In the 1960s, researchers at MIT developed the first automated debugging system, known as MAD (Michigan Algorithm Decoder) [8]. This system used a DevBot-like program to analyse the code and identify errors, making it easier for developers to identify and fix bugs in their software. As computing power continued to increase in the 1990s and 2000s, the use of DevBots became even more widespread. One popular example is the use of DevBots in automated testing, which allows developers to quickly and efficiently test their code for bugs and other issues [4]. More recently, the rise of artificial intelligence and machine learning has led to the development of even more advanced DevBots.

They can analyse code and recommend improvements, suggest solutions to common programming problems, repository management [5], and even generate code automatically based on high-level specifications [9].

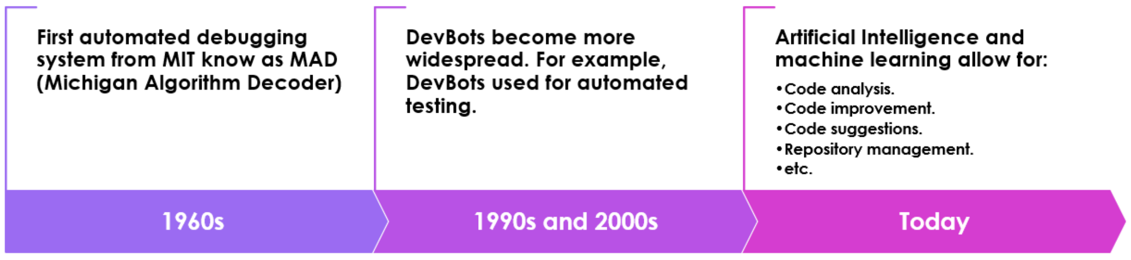

First automated debugging system from MIT know as MAD (Michigan Algorithm Decoder)
DevBots become more widespread. For example, DevBots used for automated testing.
Artificial Intelligence and machine learning allow for:
•Code analysis.
•Code improvement.
•Code suggestions.
•Repository management.
•etc.
1960s
1990s and 2000s
Today

Figure 2.1: The evolution of DevBots through time, from their beginning to today.

Overall, the history of DevBots and software bots has been one of continuous innovation and improvement. As software development continues to evolve, DevBots will likely play an increasingly important role in streamlining the development process and helping developers work more efficiently [2]. Particularly since DevBots are being used for a variety of purposes, including creating issues, updating tools, monitoring the health of systems, assisting in coding activities [4], editing documentation [10], building project images, or keeping dependencies up to date [1].

Some areas that DevBots are applied to are managing software repositories and controlling software development processes [5]. DevBots are considered a great approach to potentially solving or reducing the complexity that modern software development entails, because a DevBot interacts and accesses a system such as computer environments or software repositories [1]. One use case is to enable developers to become more productive by reducing tedious tasks [4]. DevBots in general could be used to fill knowledge and communication gaps within software development teams [4]. A DevBot could be used to traverse a systems configuration files and evaluate whether or not a dependency is in need of updating [1], a task which could become very tedious as well as error prone. We investigate similar tasks and obstacles in our initial interviews.

There are Stack Overflow[1] DevBot integrations with Slack that may help developers find answers to questions [4]. A DevBot could be utilised as pull request management assistance on version control applications such as GitHub and Gitlab [11]. A DevBot can support a developer with communication and/or making smarter decisions [4]. A DevBot may also assist in systems that employ continuous integration, automating tasks [4] such as building and testing software [12], scheduling jobs [13], agile team management [13, 14], software visualisation [15] or ensuring that software follows quality and safety standards [16]. Throughout the thesis, we had two use cases, one where we constructed a DevBot that improves productivity by reducing the manual labour required for the documentation process of planning using various tickets (e.g., a Trello board), and one where we improved a dependency management process. Storey and Zagalsky [4] use the term "gluing tools together" to explain one use case that a Software bot could accomplish. Evidently, DevBots are considered for a great

[1]Stack Overflow `https://stackoverflow.com/`

deal of tasks related to software development. To further strengthen these claims, Wessel et al. [11] elicited tasks that could be assigned to DevBots, some of these are: i) control dependencies; ii) CI failure reporting; iii) welcoming newcomers; iv) automated software builds v) code cleanup; vi) documentation.

2.3 Examples of DevBots

A DevBot makes use of many different types of technologies. Below, we will list and briefly describe a few examples of bots. First, the SapFix bot [1] utilises artificial intelligence [4], but as previously mentioned, most DevBots make use of sophisticated scripts, written in various languages. Storey and Zagalsky [4] further mention a possible use case for bots: to capture or analyse data coming from other tools and bots. An example used in the paper is the BugBot[2], which automates manual tasks and integrates developer tools. Another example is Hubot[3], a Github creation that has been called "the friendly robot sidekick". Hubot does essentially anything to improve and automate tasks, albeit not only development related. Moving on to Lita, built on the slack framework, is a highly customisable chatbot to fit users' needs[4].

More examples of chatbots are Rasa[5] and Tars[6], intelligent bots intended for developers to help scale, secure, and monitor projects and business. Storey and Zagalsky [4] write about both Hubot and Lita as bots used by developers, as well as the Freud Bot from the Atlassian ecosystem, that runs static analysis on code[7]. Storey and Zagalsky [4] also mention the Hallelujah Bot by Atlassian, which balances tests across machines. Teams are using several bots to enhance workflow, one is called DeployBot11 and is used to build and manage deployments straight from communication tools of the developer's choice[4]. Another example of a DevBot is Eurl, a Cisco-based application that assists users over their communication platform WebEx[8]. Balachandran [17] created a bot called ReviewBot, which performs code reviews of Java programs. Next, RepairNator is an autonomous bot that monitors test failures, reproduces bugs and runs repair tools [10]. We also found Brisby [4], which is a smart knowledge management bot that learns from inputs and answers questions discussed in a team. To conclude the list of DevBots and bots, PagerBot, created by Stripe, manages on-call schedules in communication channels[9].

[2]BugBot `https://www.slackreview.com/review/bugbot/`

[3]Hubot `https://hubot.github.com/`

[4]Lita `https://slack.com/apps/A0F7XDUJH-lita?tab=more_info`

[5]Rasa `https://rasa.com/`

[6]Tars `https://hellotars.com/`

[7]Frued Bot `https://mvnrepository.com/artifact/com.atlassian.qa`

[8]Eurl`https://apphub.webex.com/applications/eurl-cisco-systems-77671-27298`

[9]PagerBot `https://github.com/stripe-contrib/pagerbot`

2.4 Frameworks and Platforms for the development of DevBots

In addition to personas, we explain what a framework and a platform are in relation to DevBots. In order to build DevBots, developers can make use of various platforms and frameworks. The word “platform” refers to a piece of software where the DevBot accesses information, for example Slack, HipChat, Teams, Github, Windows, Discord, or a terminal [4]. The word “framework” refers to an ecosystem where you can build DevBots. DevBots might be difficult to construct and based on security regulations and how tailored a DevBot should be, a company might choose to code it completely from the ground up or to use pre-existing options. However, leveraging frameworks is a common and swift approach to getting up and running with a DevBot. The various frameworks provide different approaches for the creation of DevBots. For example, some may constrain us to create a DevBot for a specific platform whereas other frameworks allow us to create cross-platform DevBots. There are even frameworks that allow for the creation of DevBots without the need to write any code [6], a prime example is the framework used in this thesis, BotPress which offers the option of a simple drag and drop interface together with AI tasks and easy to understand operational nodes. If more flexibility and customisation is needed then the option to add or tweak code is available but not necessary.

Using frameworks when constructing DevBots is highly beneficial due to a number of reasons. Firstly, frameworks provide developers with pre-built templates and code libraries that can be utilised to streamline the development process. This allows developers to focus more on the unique features and functionality of the DevBot rather than spending time on building basic functionality from scratch. Additionally, frameworks often have built-in functionalities optimised for specific use cases, such as natural language processing or machine learning, which can greatly enhance the DevBot’s performance. Moreover, using a framework ensures that the DevBot’s code is standardised and structured, making it easier for other developers to understand and contribute to the project. There are evidently many benefits, so a common and swift approach to get a DevBot up and running is to leverage frameworks. The framework services could also offer different levels of complexity when creating a DevBot. In our case, we wanted a framework that allows us to connect a DevBot to Slack, Teams and our custom web applications. Preferably this option would also provide cloud solutions and boiler-plates and no-code solutions [6].

Development frameworks for Software Bots exists in a multitude of forms and shapes. Several tech companies offer tool kits to create and distribute Software Bots [6]. Microsoft provides the Microsoft Bot Framework, Bot builder, Bot connector, Power Agents and Azure Bot Service. Microsoft has good support but is not free and takes time to configure. Pandorabots have the PandoraBot[10], Slack offers the Bolt SDK, which offers great documentation but does not, in its current state, allow interconnectivity to Teams. Examples of open source options are BotKit[11], Gupshup

[10]PandoraBot `https://home.pandorabots.com/home.html`

[11]BotKit `https://github.com/howdyai/botkit`

Bot[12], OnSequel[13], BotMan[14] and BotLibre[15]. BotPress[16] is a framework that offers boilerplate and no-code solutions and we were able to utilise their service through a convenient cloud service studio, minimising the time it takes to start developing. The BotPress framework seemed to fit our project goals the best, and is the foundation that we built our prototypes on.

2.5 Workflows in the life of a developer

It is important to note that for a DevBot to be useful it needs to be well integrated into a developer's workflow, so we need to define what a workflow is. There are many different ways to define a "workflow" as seen in Unertl et al. [18]. For the thesis, when we refer to "workflow" we mean the daily process or tasks a developer has in their day-to-day life. This can include but is not limited to, daily stand-up, sprint planning, communicating with coworkers, developing code, debugging, testing, etc. In addition to this information, we were also interested in finding out what tools and technologies were being used by our various participants. All this information was very important to learn and document since it would be a defining factor when the time came to decide what sort of DevBot we would develop and what framework we would use. If a DevBot would be created that was not able to be seamlessly integrated into our participants workflow or tools then in the end they would probably not use it since the benefits that it could provide would not outweigh the inconvenience of using it.

We also wanted to focus on and define workflow obstacles. The reason for this is to be able to figure out what sort of solutions we can provide with the use of a DevBot. In order to solve a problem it is essential to understand it, but it is also necessary to determine what we do not know. There could be many things and details about a person's/team's/company's operations that we cannot know that could be potential sources of inconvenience or annoyance that a DevBot could potentially solve. These inconveniences are what we refer to as workflow obstacles. These are points in the workflow that present a problem. This can mean that a process takes longer than it should, a process tends to be error-prone, or even that a process is generally disliked by several developers. Any such problems can be workflow obstacles and are potential targets for a DevBot to either solve or minimize.

In addition to knowing what workflow obstacles are, it is also important to know that not all of them are made equal. Performing parallel tasks such as coding whilst having to update tasks on ticket boards like Trello is a challenge for developers [4]. It is however somewhat difficult to evaluate just how much of a challenge it is, since the metrics used are usually not investigating what the developer thinks or feels, but instead only look to see if there is an improvement in code production such as an increase in commits or lines of code pushed. Furthermore, developers do not always

[12]Gupshup `https://www.gupshup.io/conversational-messaging-platform/chatbot-platform`

[13]onsequel `https://tracxn.com/d/companies/onsequel/__8T5TbSETSwM7QD_gdvSlzmKLGaVZJDszh6QkRwtJSvU`

[14]Botman `https://botman.io/`

[15]Libre `https://www.botlibre.com/features.jsp`

[16]BotPress `https://botpress.com/`

share the same perception of what productivity actually is, it could be: i) having clear goals; ii) reducing interruptions or iii) holding fewer meetings [4]. Utilizing DevBots to improve developer productivity is therefore quite difficult to measure [4]. After learning about our participants' workflow it was important to figure out what problems or obstacles existed in that workflow. It is rather complicated to find the benefits of an introduced tool, in our case, the DevBot.

A recently debunked myth is that productivity's only true metrics are developer activities [3]. Knowing this we applied the SPACE framework [3] to try to solve this issue. Its aim is to capture otherwise not evaluated procedures such as pair programming. SPACE evaluates productivity by introducing dimensions such as satisfaction, well-being, performance, activity, communication & collaboration, and lastly efficiency & workflow. In each one of these dimensions, there are several metrics. These should be chosen with caution and too many metrics might instead lead to confusion. Examples of metrics belonging to the performance dimension are reliability, absence of bugs, and customer satisfaction. Other examples of metrics are from the communication & collaboration dimension: i) discoverability of documentation and expertise; ii) quality of reviews of work contributed by team members; iii) network metrics that show who is connected to who and whom [3]. It consists of several recommendations and guidelines that will assist us when evaluating if the tool constructed was indeed perceived as useful. Additionally, we wanted to evaluate whether a developer feels satisfied with an introduced tool or solution. Commonly, metrics such as the number of pull requests, code reviews, and commits are used to measure how a tool affects one or more developers. Forsgren et al. [3] reported that not only do such metrics fail to evaluate the perks of pair-programming and collaborative tasks but they have also been shown to be error-prone due to gaps in data measurements. The mentioned metrics are all in one way or another attempting to capture productivity.

Forsgren et al. [3] further emphasize that productivity can not be properly assessed using a single metric, which convinced us to attempt to measure more than merely the amounts of commits from a developer or the lines of codes pushed. An interesting type of metric, which we think is often not deemed important or simply skipped, is how the developer actually feels about the new artefact or tool. We wanted to investigate if they experienced fulfilment in the sense that they are feeling happy with the new addition to their tool box. [3]. The SPACE framework provides several recommendations that we aimed to satisfy, such that i) it is recommended that at least three dimensions should be used; ii) Several metrics for each dimension should be captured; At least one of the metrics should include perceptual measures such as survey data.

2.6 Benefits and Disadvantages of DevBots

Even though the concept of DevBots might be viewed with general excitement, there are of course, as with any other technology, advantages and disadvantages.

Software companies have been using DevBots to improve systems and communication for some time. Many major actors are using DevBots to increase team productivity and software quality, realising the importance of DevBots, as they bring value to the integration of services, users in general and communication channels [6]. DevBots are in general perceived as a partial solution to the increasingly complex systems within software engineering. We are at a stage where DevBots are successfully completing routine activities such as building images or keeping dependencies up to date [1]. Some tasks are however considered to be too delicate to allow a DevBot to conduct, but Lebeuf et al. [6] argues that we can expect a code of ethics between humans and bots in the near future. A code of ethics would likely increase trustworthiness. Meyer et al. [19], claim that DevBots that are providing awareness on commit information may keep developers in their “flow”, potentially improving productivity. Even though DevBots should be used with caution, they are usually trusted to execute simple, strictly defined tasks. Examples of such tasks could be restarting servers or re-running tests [2].

Although DevBots offer many benefits they can also have their downsides, this can be seen in Lebeuf et al.'s [6] study, where they mention the dependence that the company SendWIthUs has on DevBots. Relying on DevBots could potentially introduce issues since task knowledge could be lost and DevBots, together with important tasks might even be forgotten as they roam around autonomously in the background. Automating tasks is seldom perceived as a bad thing, but developers should consider ethical aspects and how DevBots might be misused, both intentionally and unintentionally. Lebeuf et al. [6] reports that DevBots may also have negative effects on productivity and creativity when collaboration opportunities are reduced. A similar consequence was reported by Storey et al. [4], namely that team members spend less time together and are therefore missing out on learning opportunities. By implementing a DevBot between two applications we could potentially reduce such activities within the team, possibly resulting in negative learning outcomes. Platis [12] mentions that as multiple DevBots are often cooperating in software systems, the ecosystem grows and with that, its dependencies, which increases size and complexity and could introduce bugs.

Erlenhov et al. [1] report that the ideal DevBot is largely self-sufficient, technically knowledgeable and holds some social competence. It was shown that Bots without social competence that answered to Stack overflow questions were downvoted [20], rendering the specific DevBot quite useless. A prevalent downside of DevBots is noise and interruption. A DevBot could for example overload communications, Wessel et al. [21] writes that *“maintainers also reported unexpected aspects of bot adoption, including communication noise [...]”*. Another aspect is that even if it posts important information, the DevBot post so frequently that humans stop noticing it. Another negative aspect to DevBots is the trust issue, where the problem lies at what rights the DevBot should have to take decisions [2]. Both Alex and Sam personas expressed distrust when allowing the DevBot to complete tasks. As soon as the tasks require system knowledge, the trust in DevBots seem to decrease [2]. The trust issue could also be contrasted, where developers might trust the DevBot all too much. Erlenhov et al. [2] also found that the Sam persona would approve of a DevBot only if it produced a very small number of mistakes, otherwise if it produced a greater amount of errors they would not accept it. This further complicates the integration of “smart” DevBots. Murgia et al. [20] investigated the perceived trust of two bots, one with a human-like avatar, and one with a distinct bot-like avatar. They found that the bot received a lot more negative feedback, even though it was the same code and answered the same queries. They claim that humans generally have a “low tolerance for mistakes by a bot”.

3 Methodology

This section covers the methods used to conduct the study. We split up the work into iterations one and two. First, we give an overview of both iterations and the study as a whole. After an overarching introduction, we present three subsections where we go into more detail: *Iteration one - pre-study*, *iteration one - proof of concept*, and *iteration two.* The process of our study is shown in Figure 3.1.

The research and the entirety of the thesis were written in collaboration with Ericsson, a large telecommunications company with headquarters in Sweden. We found the research questions in our thesis to be suitable for Design Science Research (DSR) as our questions aim to construct an artifact over several iterations. In this thesis we will focus on answering the following Research Questions. These were carefully chosen to fulfil a specific gap in knowledge as well as provide a useful contribution to our partner, Ericsson:

- RQ1: Which obstacles in relation to workflows do practitioners/developers face in their day-to-day life?
- RQ2: How can development bots improve an existing workflow?
- RQ3: How does the introduction of a DevBot in relation to a workflow influence the developers in their daily lives?
- RQ4: What are benefits and disadvantages of using certain DevBot frameworks for the needs of big corporations like Ericsson?

We compiled a list of obstacles in workflows to answer *(RQ1)*: these obstacles have likely been experienced before but it was necessary for us to solidify our understanding of the needs of the developers. We gathered the obstacles by eliciting them from teams at Ericsson. Data from RQ1 allowed us to investigate different approaches and construct artifacts (DevBots) for *RQ2.* We evaluated the artifacts (RQ3) and reported our findings on advantages and disadvantages *(RQ4)* of the DevBot frameworks that we tried out. In order to construct the DevBot artifact, we applied Hevner's [22] and Knauss' [23] guidelines on DSR. We conducted an observational case study [22] at Ericsson, where we applied several types of data collection methods i.e., interviews, surveys and a focus group [24] to two teams.

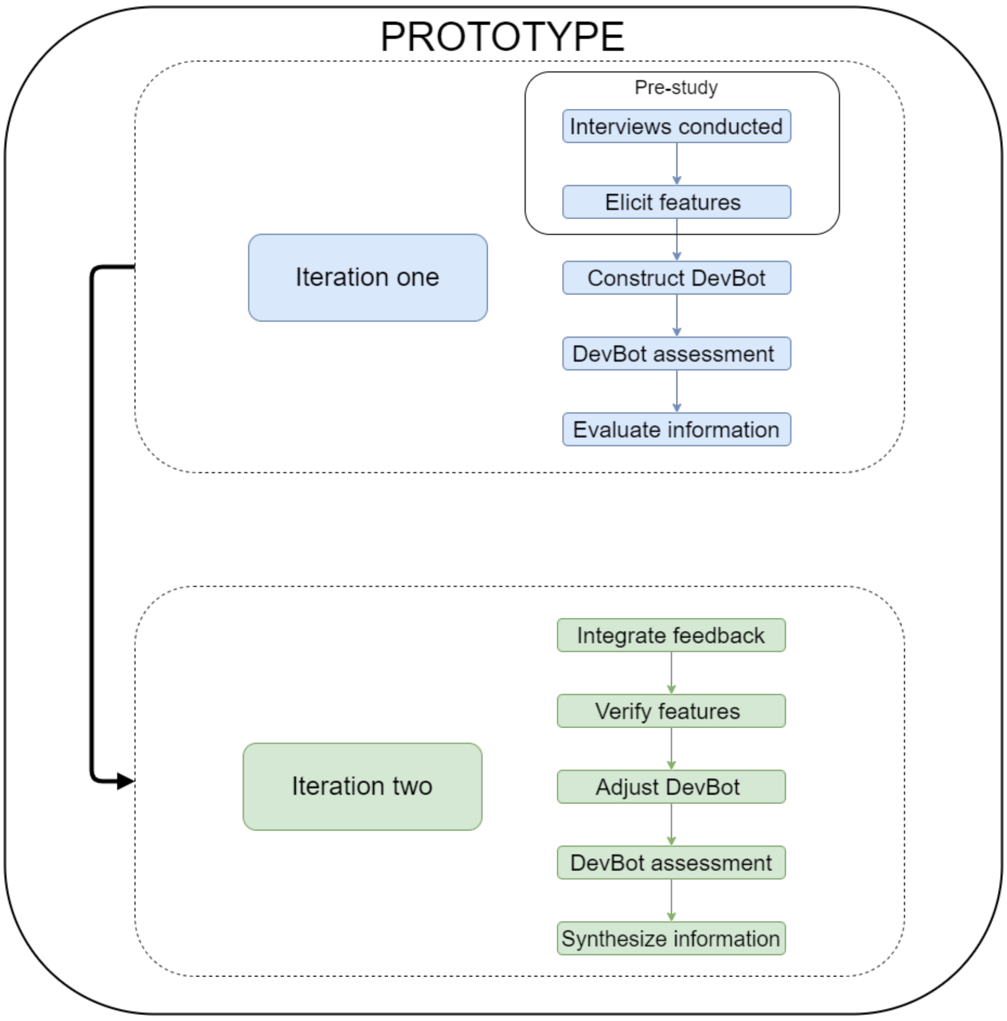


Figure 3.1: Process iterations

The first iteration focused on gathering data through interviews to inform the development process and to construct and evaluate an initial proof of concept. We conducted interviews with stakeholders and users to gain valuable insights into their needs and requirements. The information gathered was later used to construct DevBots and analyze frameworks. Iteration one was concluded by evaluating the DevBots through a survey. This approach allowed us to prioritize the tasks for the artifacts i.e., the DevBots early on in the project and ensured that our development efforts were in sync with their requirements. Iteration two consisted of integrating feedback in order to improve the selected DevBot as well as an evaluating it. The evaluation in iteration two consisted of a focus group session where participants were encouraged to discuss freely and express opinions, concerns, wants and needs.

Lastly, we present a timeline for our thesis project, Figure 3.2 shows the tasks that we have worked on during the months up until June.

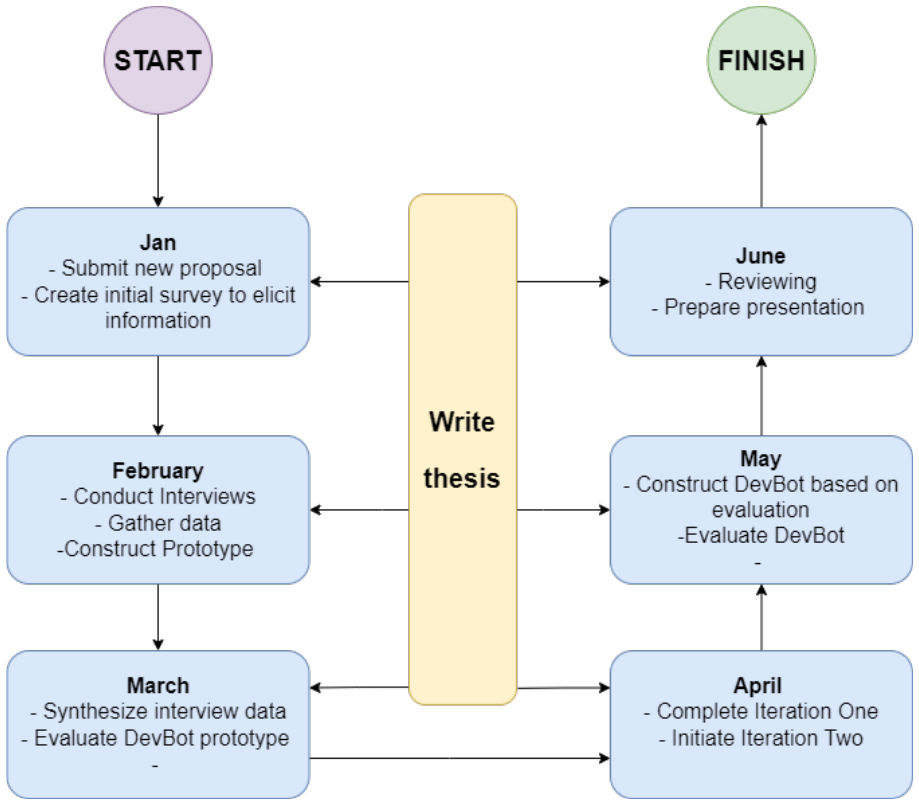


Figure 3.2: The timeline we followed for the thesis project

3.1 Iteration one

In iteration one we used Seamans's [25] guidelines on how to properly utilise interviews and surveys in order to get the most out of our participants. Both quantitative and qualitative questions were asked in the interview and the survey. We conducted one initial round of interviews, which is part of what we call the pre-study, where we elicited information to list fundamental features of the DevBot. After we extracted relevant data through thematic coding, we constructed two DevBots based on the presented needs of the participants. We evaluated the DevBots through a survey and then we collected and analysed the data.

3.1.1 Iteration one - Pre study

This section explains how the pre study in iteration one was conducted and what instruments we used. The questions from the interview guide are provided in table 3.1 and the entire guide can be found in Appendix C.

3.1.1.1 Data collection

For the initial interview, we used an interview guide(C.1) to elicit information from the participants in iteration one for the pre-study. It is advised by Easterbrook et al. [24] to pilot test data collection instruments such as interviews, which we did by first conducting a mock interview between us and then conducting the actual pilot interview with the first participant. After the pilot interview we decided that we were ready to continue conducting interviews with the rest of our participants.

We used a semi-structured approach in the interviews, where we asked open-ended questions, which meant that we were open to discussion and thoughts in between the main questions. For each candidate, we recorded the interview and transcribed the data. Participants were asked to sign a consent form to let them know what this was about, how their data would be collected and used and that they had the right to withdraw their consent at any time before the final submission of the thesis.

Before going over the collected data we made sure to located and remove or alter any names that may have been mentioned to ensure full anonymity of our participants and their colleges. In addition no defining or traceable naming-scheme was used on the saved files.

Parts of iteration one consisted of gathering data on which frameworks were useful in constructing DevBots needed for our specific use case. We extracted the data from our initial research in iteration one when we needed to create the DevBots for our surveys. We collected the data by reading articles and browsing communities.

3.1.1.2 Analysis

Thematic coding was used to extract the information from the interview in iteration one. Creswell et al. [26] introduces the strategy of Peer debriefing, which we applied for the thematic coding by individually extracting data from the interviews. We aggregated the data if we both had the same entry or if one could argue that the entry is worth keeping. After finishing up thematic coding we elicited a list of obstacles of the everyday workflows at Ericsson.

We constructed the codes inductively, i.e., starting with pieces of text from the interviews and deriving codes from those. This is referred to by Gibbs [27] as the data-driven approach. As we quickly amassed lots of information, we traversed each interview methodically, highlighting pieces of text that we found interesting or potentially useful. We highlighted the segments individually, to ensure that we did not become affected by each other, and to ensure a higher quality of the extracted data. After we highlighted all the interviews individually, we extracted the marked text into Excel sheets to be able to sort and filter effectively. Afterward, we conducted several workshops where we used a large board with digital post-its, which was constructed using the tool Miro[1]. In order to create the codes, we went through four steps (Figure 3.3):

1. conducting interviews,
2. transcribing the interviews,
3. extracting data into Excel sheets,
4. and lastly we hand-picked data which we added to a Miro board.

For the process of extracting information from the interviews we used a combination of deductive and inductive coding. By deductive we mean we had several themes in

[1]Miro tool `https://miro.com/`

Table 3.1: Questions first-round interviews

#	Question
1	How do you usually work, day to day?
1a	Do you work from home, office, hybrid?
1b	What tools do you use? (Communication, Development, Testing, Meetings, etc)
2	What are some regular or standard tasks in your work process?
2a	Where do you feel your workflow/process struggles the most?
3	How do you ... ?
3a	Push code
3b	Compile Code
3c	Test your code
3d	Fix code
3e	Debug code
4	What pipelines do you have?
5	Are there any parts that need to run simultaneously/ in parallel?
6	Do you have any tasks that are dependent on others?
7	Where do you encounter the most problems, time delays, confusion and/or errors (RQ1)
8	Please mention one to three challenges that you commonly face when developing code (RQ1)
9	If you were to be given a devbot to assist you in your work, what would you like that devbot to do? (RQ2)
10	How do you think a devbot could help you? (RQ2)
11	Would you trust the output of the devbot you have described?
11a	Why would you trust it?
11b	Why would you not trust it?
12	In what way would you prefer to trigger this DevBot invoke it whenever you want to, or have it operate automatically and in the background? (RQ2)
13	How do you think a devbot of this nature would benefit you, your team or other teams and maybe even a bigger part of the company? (RQ3)
14	How do you make sure that the 3PP and tools you use are compliant and allowed to be used?
15	In an unlikely event of your services disappearing in fire, how do you make sure to deploy the right version of everything, and how do you know where to deploy it?

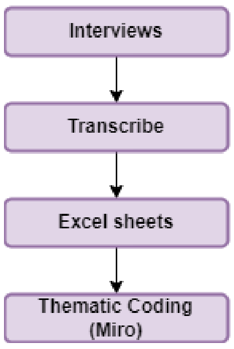


Figure 3.3: Sequence of data collection to thematic analysis

mind that we had previously agreed upon that were of interest to us i.e., information about tools used, general trust of DevBots, daily routines/obstacles/bottlenecks, what DevBots our participants would like to have and how would they like to interact with them. we also inductively selected quotes that in general we found interesting or could potentially be useful later on which we used to create more themes. These were then added to each researchers Excel sheet for that interview to be later compared and filtered. This process can also be referred to as open coding. The sequence of how we conducted the thematic coding step (the last step in Figure 3.3) is described more in detail as follows: i) pick the first entry in one of the researchers' Excel sheet for one interview; ii) discuss if it should be added to the Miro board, if not, discard the entry; iii) add the piece of text to the best-fitting code, if none fits, create a new code; iv) reiterate until the researcher's Excel sheet is out of entries; v) to assure that all entries were covered, now repeat the process with the entries from the second researchers Excel sheet with interview data; vi) repeat the process for all interviews until all entries belong to a code; vii) if a code grows too large, split up into smaller codes; viii) remove any duplicates.

We used parts from the concept of Grounded Theory (GT) [27, 28] method to apply thematic coding, where we, as previously mentioned, inductively derived codes. We used open coding, a common GT approach which means that we read pieces of text reflectively in order to identify all the codes and categories [27]. After finishing the workshop and finalising the Miro board, we compiled all the codes of importance into tables. Some of the codes were deemed unnecessary, but we had to first collect the data in order to understand that. When the thematic coding tables were completed, we constructed a list of common obstacles that helped us pinpoint the needs of the DevBot. Other important pieces of data were extracted and converted into lists, but we think the obstacles are especially noteworthy here as they were the foundation for our DevBot(s).

The data gathered was also used to construct tasks. The tasks i.e., the survey and the focus group were conducted using a web application provided through the BotPress platform. The DevBots under test are a ticket management system that needs to be updated with data from another external system, and a dependency management system.

The frameworks that we gathered during data collection were explored by searching

Google, Google scholar in addition to asking our supervisors and an internal team at Ericsson. Afterwards, each framework was entered into a table. The Excel sheet was then condensed into a table of frameworks, which were the ones that we spent extra time testing out.

3.1.1.3 Justification of methods

We used interviews because it allowed us to guide the discussion and it provided us with a platform where we could hone in on interesting topics that the participants brought up. By using interviews we could ask follow-up questions to better understand the responses which allowed us to collect much more rich data. We chose our participants through convenience sampling using similar motivations as Etikan et al. [29], namely that it is a type of nonrandom sampling in which members of the population meet certain practical criteria, such as easy accessibility, geographical proximity, or availability. We constrained the target population based on three prerequisites: i) the participants we chose worked at Ericsson and had at least one year of work experience at the company; ii) their work had to in some way be related to software development; iii) they would be open for follow-up questions and feedback after the interview for the pre-study was completed.

The aforementioned interview recordings were used both for accuracy and for convenience. Being able to go over the recording meant that we were able to take more accurate notes during the analysis, that we could clarify segments that we may have miss-heard or miss-understood and finally it was a lot faster when it came down to transcribing since we were able to use transcription software that is embedded into Microsoft Word and Microsoft Teams. To ensure that we didn't lose any of the recordings we would always record on two separate devices, the company issued laptop to ensure that any company data remained physically on company hardware and on a smartphone after which the file was moved to the cloud on our company account to again ensure the security of the data, the recording was then deleted from the smartphone.

While conducting the interviews we started noticing patterns, repeating comments and themes. This eventually led to a saturation point [30] where no new information was being produced by the interviews. With the completion of the sixth interview we made the decision that we had elicited adequate information and viewpoints to reach the final stage of our first iteration.

By using thematic coding we could systematically collect pieces of data from the interviews and even though it is a time-consuming process, we gained a much better understanding as we discussed each quote and each code before storing them.

The information gathered from the interviews provided a solid foundation for our DevBot. We were able to incorporate the insights we gained into the design and development of the software, making sure that it met the needs of our stakeholders and users. It allowed us to gain a deep understanding of the needs and requirements of our stakeholders and users, which in turn helped us to propose a solution that met the users' needs. Prioritizing interviews early in the project ensured focused and effective development efforts. This ultimately resulted in a successful proof of concept.

3.1.2 Iteration One - Proof of concept

With the pre-study done we were ready to develop our proof of concept. We used surveys to further validate our artifacts.

3.1.2.1 Data collection

As the initial interviews were completed, we constructed two DevBots as proof of concept, of which we needed to confirm the use cases and functionalities with the intended users. To accomplish this we decided to conduct two surveys (Appendix A & B) to collect data. The two surveys are almost identical and aim to evaluate each DevBot.

For the DevBot *Teddy* we sent out all surveys at the same time. The motivation was that we already had fairly concrete steps for the specific process. The survey which evaluates the DevBot *Pepe*, was first sent out to one participant, as we were not as certain about the specific tasks of the DevBot. By doing so, we could collect feedback from the survey, make edits to the tasks and update our survey. After we received feedback we decided that it was sufficient and reached out to the rest of the participants.

We leveraged parts of the SPACE [3] framework throughout our surveys where three dimensions are recommended, but for the first survey we used two dimensions and two metrics. The dimension *Satisfaction* was used to evaluate how the developers feel about the DevBots. For each dimension, several metrics exist from which it is recommended you choose one or more. We choose the metric *Employee satisfaction* where we ask how the developers feel about the potential DevBot being included in their daily workflow. We requested the participants to grade their impressions on a scale from one to five. The second dimension assessed was *Efficiency and flow*, which should capture the ability to complete work without interruptions [3]. Here, we decided to use the metric *Perceived ability to stay in flow*, which we measured by asking if they believed whether or not the DevBot would improve or disrupt their workflow. We included a URL to the DevBots in the survey, which allowed developers to test and interact with them using a chat interface. The rest of the survey consisted of questions related to the tasks and use cases of the DevBot.

3.1.2.2 Analysis

With all our participants having filled out the surveys we went over the responses in order to gather as much information as possible before making further decisions. We wanted to see if any patterns would appear, if the DevBots were well receive, if they had any big shortcomings and what suggestions and comments the participants had written.

When the data was processed we made a decision on which DevBot to focus on, as we could only work on one solution given the time and scope. The choice was based on the amount of survey responses and the quality of data in the survey. In the end the decision was made that we would focus on the DevBot Pepe as it would be more fitting both academically and business-wise for Ericsson. Based on the survey data the perceived benefits of Pepe were more prominent and direct, which is the motivation for our choice.

3.1.2.3 Justification of methods

The reason we chose to use surveys was that we had limited resources in terms of how much time we could request from Ericsson. Surveys can be made to be short and quick while still gathering useful information. The survey questions were a combination of qualitative and quantitative in order to get a more detailed understanding of our participants thoughts while also having statistics that could support our decisions.

As for the SPACE framework, it provided valid arguments as to what constitutes an improvement in a workflow from the perspective of the developers in addition to debunking several misconceptions that dominate the industry. The motivation for our choice is that we wanted to use metrics that measured the perceived feeling and happiness of the developer, not just hard metrics like lines of code or features added. The recommended amount of dimensions is three, and it is argued that using all five might cause confusion or be overwhelming. We used two dimensions, with several metrics, as it sufficed in order to extract the information we needed from participants.

3.1.3 Iteration Two

We used the second iteration (Figure 3.1) to implement the final touches of the DevBot based on the data collected from the surveys in iteration one.

3.1.3.1 Data collection

We used a focus group to gather data for iteration two. We collected the data by recording the presentation as well as taking thorough notes. We decided not to use our previous data collection options, such as thematic coding, but rather note down all information that we deemed important. We focused mainly on how to improve the DevBot Pepe and if the current version was satisfactory to the participants. We reached our last step in iteration two called *Synthesize information* where we used all data from the interviews, the survey and the focus group session to motivate our design and feature choices of the DevBot.

We used the same consent form for the surveys and the focus group in iteration two as we had used for iteration one. We started the session by presenting the latest version of our DevBot with its new functionality and changes followed by a discussion. Our aim was to validate the DevBot and to evaluate it based on the chosen dimension from the SPACE framework [3]. For the focus group we used the dimension *Efficiency and flow* with the metric *total perceived time of a process*, of which we simply ask the users if they perceive the total time to be reduced or not. We asked open-ended questions and allowed the participants to discuss among the group. The questions that we posed in the focus group were following:

1. Do you think that this DevBot would improve your workflow? If so, how? If no, why not?
2. Do you think that this would make you more productive? If so, how? If no, why not?
3. Which features would you add if you had a magic wand?
4. Do you have any other thoughts on the DevBot?

3.1.3.2 Analysis

After completing the focus group we gathered our notes and the recording, analysing them as thoroughly as we could in order to extract as much useful information as possible from them. We would listen to the recording in small chunks and write down the key takeaways relating to that particular part of the conversation and the question it related to. This way we could see how each participant felt and responded to the various questions and it gave us the opportunity to reflect on their answers in order to understand how users experience our DevBot and how to best proceed from here on.

3.1.3.3 Justification of methods

As we had to adhere to time constraints from our participants we decided to hold a focus group at Ericsson where we presented our artifact. We needed a variety of data collection methods to avoid possible threats to validity such as biased opinions from the researchers. It was also desirable because we wanted an environment where would inspire each other and add to each others ideas and comments. The focus group was conducted after completing the latest iteration of our DevBot and we felt it was time to validate our artifact. We believe that the combination of our participants' physical presence and the focus group would lead to a more open-ended discussion. We invited all the participants for the Pepe DevBot survey as well as participants with knowledge of Productification to participate. This was because we wanted to showcase the new changes made to Pepe as well as get the opinion of the intended users directly.

Another reason we chose to conduct a focus group rather than any other data collection method was to have a more open and free discussion with our participants. Additionally, we noticed that through surveys, questions may be misunderstood or misinterpreted. For that reason we wanted to ask questions in person.

The participants in the focus group were also part of the interview and the surveys. We had six participants in both the interviews and the surveys, and four in the focus group. They were all engineers at Ericsson, working as developers, DevOps engineers, telecommunications engineers and project managers. One team works with constructing tools to improve productivity within the company, which suited us very well, and the other team worked with the user plane for telecom, which was good as well, since we would be able to elicit data from a very different field with other perspectives. Each participant had to have at least one year of work experience at Ericsson, as we found it important that they had some insight in the company as a whole, and therefore would know a little more about what was needed in terms of DevBots. None had experience with DevBots before but the all seemed familiar with bots in general.

The artifact was considered sufficient at the end of this second iteration, albeit still a proof of concept. By sufficient we mean that the artifact could reliably complete the task it was created for while also conveying its potential and its future capabilities if it would be fully implemented and deployed. The reasoning for keeping it as a proof of concept was based on advice from Ericsson. The process of deploying products within Ericsson is seemingly very rigid and could take several months and require lots of resources that we did not have access to. We spoke to two senior engineers within a team working with development bots, who recommended that one of our thesis artifacts would be a compiled list of tasks that the DevBot could perform, together with a well-documented proof of concept.

4

Implementation

This section will give a detailed explanation of the DevBots that were constructed throughout the thesis. We present the initial problem, the design process, a detailed description of the DevBots and the issues we faced. Our motivation for adding a section dedicated to the artefact is based on Knauss [23] guideline (G6) on design science research, namely that a section dedicated to describing the artifact is beneficial to the reader as it allows other sections to focus on the learning outcomes.

4.1 Specification for DevBot

After conducting six interviews, meticulously looking into related work and performing thematic analysis, we extracted two ideas that we believed were DevBot material. First, we developed a minimum viable product (MVP) to collect further data that was then used to build two distinct DevBots. The specification for the MVP DevBot and the actual DevBots built (Pepe and Teddy) are listed below. Pepe intends to solve a dependency management obstacle, and Teddy solves ticket migration between two systems. The final proof of concept was implemented in a separate web-interface i.e., not using Slack or Teams.

The MVP was generic and worked as a basis for both of our potential DevBots. It did however steer more towards fulfilling features of one of the DevBots, the ticket migration DevBot. The motivation behind this decision was to be able to test the prototype as quickly as possible. Its purpose was to early on verify frameworks, features and possible channels to integrate with. We verified the prototype by conducting an experiment where candidates tested the tool and simultaneously filled in a survey. The prototype was constructed following three steps;

1. Create chatbot using a framework of choice.
2. Make it generic so you can hook it up to any chat-platform (Script, Slack or Teams).
3. Complete the tasks elicited by stakeholders for one DevBot.

The initial prototype was built using BotPress [1]. We included two main features for demo purposes: *Dependency management* and *ticket migration.* The MVP was able to complete both of the DevBot tasks. Depending on the team (Team A or Team

[1] BotPress`https://botpress.com/`

B), the DevBot would make a decision on which path to take. The path is shown in Figure 4.1.

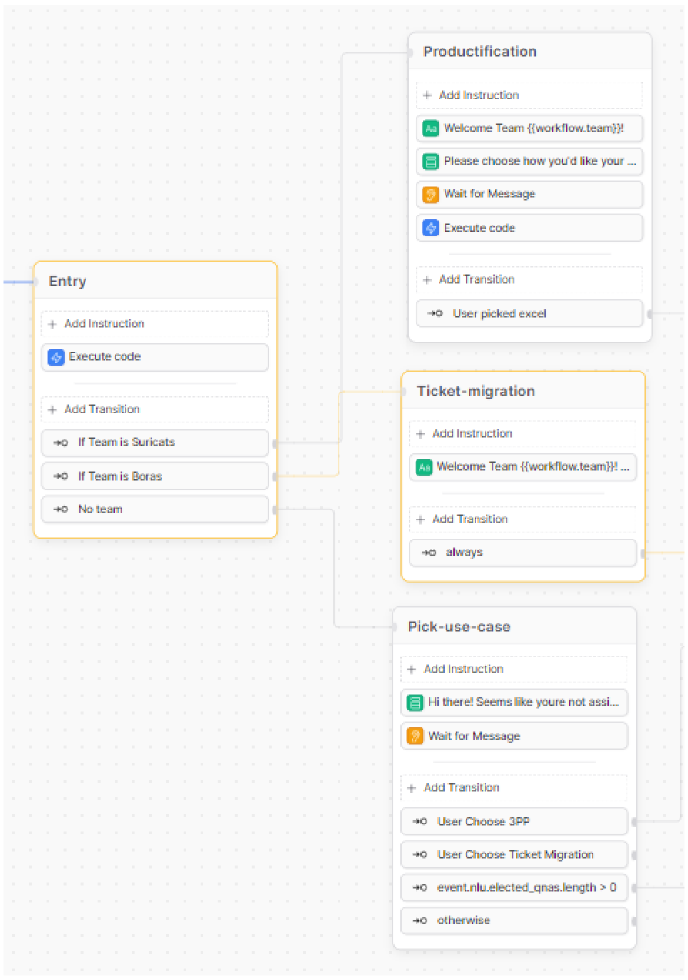
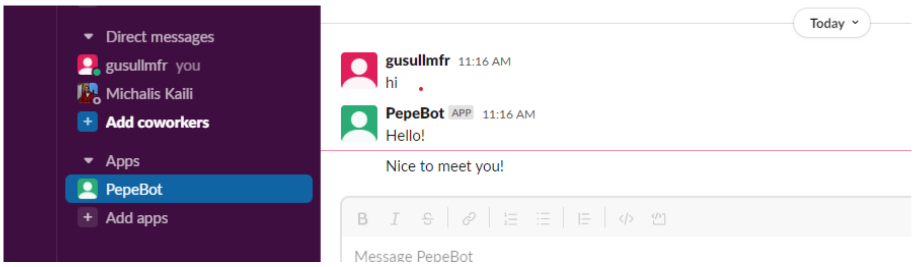


Figure 4.1: The different paths the MVP DevBot can take depending on the team to solve a particular task.

The final minimum viable product ended up being evolved into two separate DevBots, so that we could elicit more specific information on each of the DevBots, and to be able to further develop a tool suited for both use cases. The MVP DevBot turned into *Pepe the DevBot* & *Teddy the DevBot*.

To stay consistent with BotPress terminology, each square is a *node*, and we call the lines *edges*. Each node is a block of executable tasks and each line is an instruction in what sequence the blocks should be executed and how they interconnect. When you initialize a node, you get multiple options. The node can accept an instruction, for example producing text output from the DevBot, or it could listen to user input. The node can also be modified using Javascript code. A very useful feature of BotPress is that it can take natural language input instructions and convert them into business logic. When it came to the construction of the DevBot we had the option to write it from the ground up using the BotPress API or use the provided tools and framework. For our first iteration we utilised the provided BotPress features (drag and drop, nodes, edges) but it was good to know that we could use the BotPress API to code the DevBot from the ground up if needed in later iterations.

BotPress provides a great cloud based studio environment for communicating with the DevBot so we could test it at every newly created node or edge along the way. Being able to quickly test the functions was a motivation to why we chose to build the DevBot using BotPress. Several barriers prevented us from implementing the DevBot on Teams such as technical difficulties where the frameworks was expected to receive some bug fixes as well as that introducing software into the Ericsson ecosystem required us to have access to Azure cloud services which could not be provided in the context of this thesis. Instead, the first DevBot was implemented to connect to Slack and over a web-interface. The reason for this is the more open nature of the platform and its excellent documentation on how to connect the BotPress DevBot [2]. This first iteration of the DevBot was only capable of responding to simple phrases over slack (Figure 4.2). These future iterations were planned to include: i) multiple choice questions; ii) reacting to dropdowns; iii) sending files; manipulating variables; iv) and most importantly, Teams integration.



gusullmfr 11:16 AM
hi

PepeBot APP 11:16 AM
Hello!

Nice to meet you!

Figure 4.2: DevBot connected to Slack with a short conversation

The main limitation we experienced in BotPress was the lack of integration with Teams, which was the primary communication channel at Ericsson. Although the documentation was clear and easy to follow we required special permissions and access to resources that could not be provided to us at Ericsson in order to continue with the integration. Regardless, we hoped to construct a proof of concept that could be implemented over multiple communication channels, since the choice of platforms (e.g., Slack or Teams) varies depending on the needs of the users. This in turn provided its own challenge as we quickly found out. Integrating the DevBot into Slack was quick and painless thanks to the excellent documentation that was available but we had encountered issues with several functions. One of them was intended to be used to send documents like CSV files to the user so they could download it, manipulate it and use it, but every time we attempted to perform this action we would encounter crashes that we could not explain or ultimately resolve, neither by troubleshooting through logs or support services.

Additionally, we had several occurrences of our DevBot experiencing issues due to changes and updates occurring in the development studio provided by BotPress itself. Especially after major changes were made the functionality of our DevBot would sometimes be affected and several other features in the user interface would

[2]Slack-BotPress`https://botpress.com/docs/cloud/channels/slack/`

be moved around or altered slightly, resulting in a time consuming process of finding out what changed . This is not necessarily just a negative thing as it shows that the BotPress team is still very active and is continuously updating their product which adds to the benefits of this framework, but it did cause minor inconveniences and time delays.

4.1.1 Pepe: The 3PP Productification DevBot

3PP (dependency management) is time consuming and tedious. The problem itself is that the developers have to go through a process of verifying open sourced dependencies and libraries with a configuration manager. This process contains several steps, many of them which could be automated. Participants (e.g., Participant 1) of the study deem the task company-specific. However, DevBots generally assisting with dependencies in one way or another are applicable outside of Ericsson [1].

> “so yeah it is more of an Ericsson specific work there where we have to to keep track of everything that we use in terms of libraries and external tools” - Participant 1

We will explain a few terms mentioned by our participants that are relevant to understand the process. *Productification* is a process in which a list is created containing all the dependencies for one project.

> “so every time we want to make a new release of our applications we have to make this update of libraries dependencies” - Participant 1

This list needs to receive approval from a configuration manager. *FOSSing* is “FOSS” turned into a verb, where FOSS means “Free Open Source Software”. Fossing refers to using any type of free open source software. *SCAs* is where the list of dependencies from the productification process is added, it is where the configuration managers go to evaluate the list of dependencies. In essence, what the DevBot should solve is the tedious process of compiling a document of dependencies in the productification process. The initial idea for this DevBot was for it to, based on a repository link, compile a list of dependencies after which opens the default mail client of the user with a dependency CSV file already attached and allows the user to fill in the remaining fields of the email.

Both authors were responsible for a different DevBot. Nonetheless, we kept the communication and feedback loop to share experiences and mitigate eventual problems in the development process. We continuously requested feedback on each others work and we shared work-spaces so that we always had access to both artifacts. Pepe the DevBot aims to do one job, the main functionality is to provide a list of dependencies from a given project. This list has one file extension type, CSV. The latest verion of Pepe the DevBot has the following features:

1. Accept user input in a chat box,
2. Triggered when a repository is provided by a user.
3. Extract the POM.xml/package.json file from the project.
4. Traverse the dependency file and add all entries to an Excel sheet with columns: Dependency name, Version and Description.

5. When the Excel sheet is populated, automatically download the Excel file to the users' computer over the chat interface.
6. If the user wants to, send it to the configuration manager by email.
7. End of the conversation.

We provide a flow-chart of Pepe's tasks in Figure 4.3

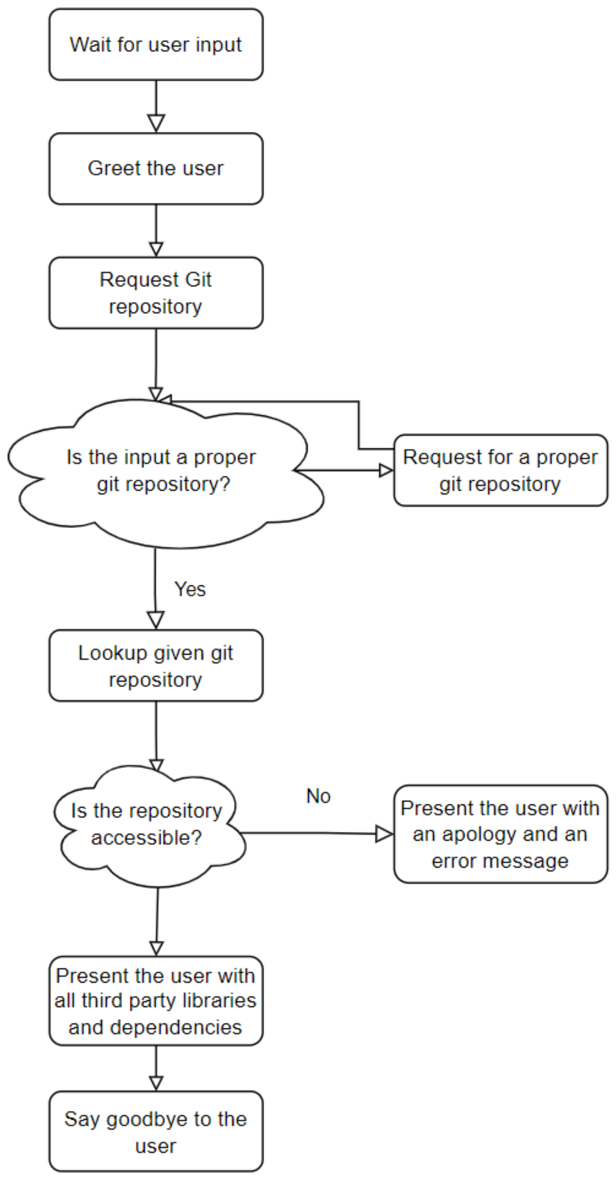

Wait for user input
Greet the user
Request Git repository
Is the input a proper git repository?
Request for a proper git repository
Yes
Lookup given git repository
Is the repository accessible?
No
Present the user with an apology and an error message
Present the user with all third party libraries and dependencies
Say goodbye to the user

Figure 4.3: Iteration One; Pepes' flow chart showing the sequence of steps taken

Pepe should be able to communicate in a friendly manner through the chat interface. Future development of new features could be beneficial, but in many cases it is better to create multiple standalone tools, for example a DevBot called Pepe Junior that works on another part of the Productification process. While this might create discord, it could at the same time be good to keep separation of concern and allow the DevBots purpose to be as clear as possible.

The DevBot is designed to reduce tedious work developers do and to make them happier or allow them to focus on more interesting tasks. As an added bonus the DevBot may also reduce the time it takes to compile and ship a product.

4.1.2 Teddy: The Ticket System Transfer DevBot

This DevBot aims to automate a tedious and manual task by migrating data from one application to another. Migrating data between different platforms or applications is often time consuming. This task has several obstacles such as the need for authorization on both platforms as well as that the data needs to be sent in a specific way and format (e.g., JSON or XML). A participant emphasized the manual labour that migrating data required, and how the entire team were forced to continuously perform this task that should, in their opinion, be automated.

> "We present the data in a transformed form so when we do that we need to verify that we actually got the correct data, that we transformed it correctly and stored it correctly and then that we present it correctly to the customer" - Participant 6

The current procedure to complete the task at the Ericsson team is that the users would receive an email about new updates in a system that we call the "TicketSystem". When the email is received, they have to access the TicketSystem and save various data entries, screenshots, descriptions, titles, ID's and more. The user would then add all this data to their own ticket system, which we here call the "Board".

Teddy had, based on the interviews, concrete steps that could feasibly be executed using BotPress, which is why we decided to start with constructing this DevBot. After being prompted, Teddy would simulate the process of checking for new tickets and moving them to where they needed to be while informing the user of each step. But despite our solid understanding of its functionality, Teddy was discontinued before iteration two was initiated. The steps required are still however validated in the survey and would thus work as guidelines when developing the production-ready DevBot. Teddy The DevBot should have following features:

1. Log in to Smart SMP
2. Open the first unassigned ticket
3. Assign Ticket to developer or to the DevBot user profile
4. Update status of the ticket to "In Progress"
5. Update status of the ticket to "Pending" and set "status reason" to "Ongoing investigation"
6. Copy description text from Smart SMP into the teams own backlog item (a board hosted on azure), including screenshots and other attachments
7. Change the title of the azure board backlog item to the ticket number(starting with INC00) + description + plus something else related service request number(starting with "REQ00")
8. Copy all comments from Smart SMP tickets into the azure board backlog item, including screenshots and other attachments

9. Tag backlogitem with “SPIS”, the affected customer and the affected part of the application(if possible). One way would be to look for keywords(predefined tags) found in the azure board and try to match against the text in the header and description(if possible).
10. Send a message to the developers that a new ticket has been migrated and added to the azure board backlog

The DevBots purpose is to fetch data from one application and move it to another. Whilst doing so, it should update the status of various tickets, assign users and set new states for variables.

The flow chart in Figure 4.4 visualizes how a user interacts with Teddy the DevBot. It should wait for user input, and as soon as the user interacts with the DevBot, it should fetch information and update the Azure cloud board. If no new data exist or if an error occurs, abort the task.

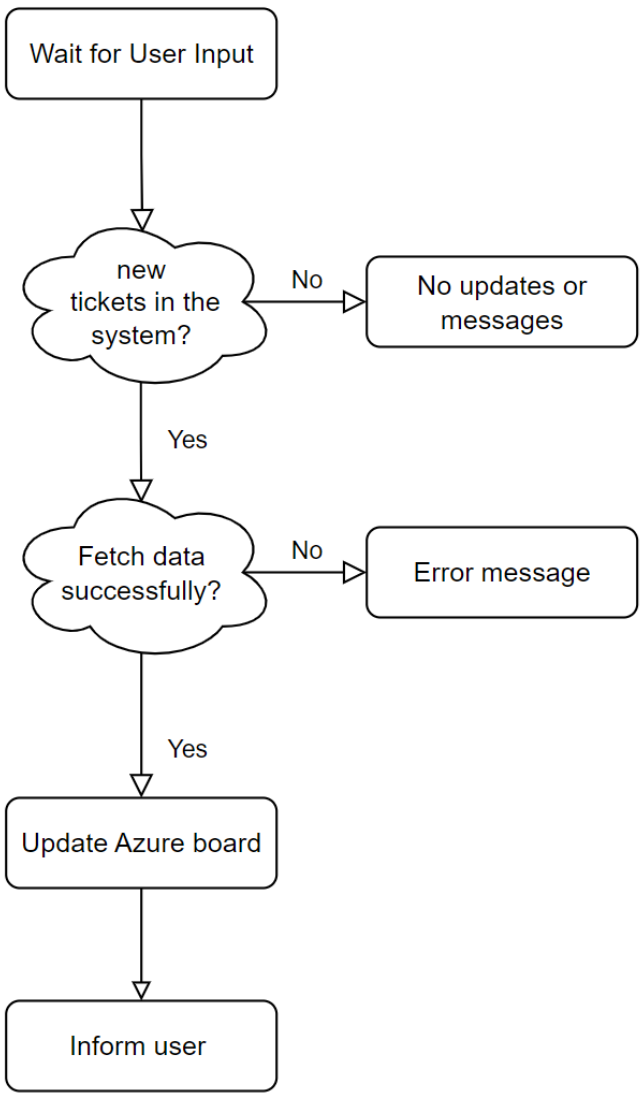


Figure 4.4: Iteration One; Pepes’ flow chart showing the sequence of steps taken

4.2 Design Decisions for the DevBots

Before and during the development process several design decisions had to be made. These extended beyond the aforementioned frameworks, platforms and general goal of the DevBot. We had to decide what the DevBot would look like, how it would interact with users, or how the users would interact with it, and finally what features would it include.

Based on the data we had gathered both from our users and from internal engineers at Ericsson it was decided that Pepe would use a chat interface but would not be attached to a platform like Teams or Slack. This is because Slack or other similar platforms are not used by the teams at Ericsson and we could not acquire the necessary permission and access needed to integrate Pepe in Teams. Instead, Pepe operates on its own URL. As mentioned before, Pepe was briefly integrated with Slack in order to showcase that it can be done and that there is plenty of flexibility when it comes to where and how a DevBot can be used. In addition, this would convey nicely what we wanted to accomplish in Microsoft Teams albeit in a different environment. Unfortunately, we had to steer away from Slack as well. This was because even though we were successful in integrating Pepe in a Slack workspace and communicating with it, we were unable to implement all the functionality we desired and were limited to simple message exchanges.

Finally, the decision was made to have Pepe operate on its own, detached from any platform for the time being. This gave us the ability to showcase features that would otherwise be left out. Pepe would also be rather simple but polite when it came to its communication with the user, greeting them and saying farewell in a friendly manner while keeping the messages short and to the point. Our hope was that having a quick but friendly interaction with Pepe would provide the user with a more pleasant experience overall, one where it didn't feel too drawn out but not rude or unfriendly.

5 Results

In this section we present findings for both iterations in our process and the information we found concerning the various DevBot frameworks that currently exist and that other researchers, companies or organisations can use for their own DevBots. Iteration one contains initial results (Section 5.1.1) and data on our initial DevBots (Section 5.1.2). Iteration two (Section 5.2) covers added features as well as focus group data. After that, we present data on the frameworks explored (Section 5.3).

5.1 Iteration one

In Iteration one we managed to gather the necessary data that we needed to conduct the thematic coding and to create the MVP. With this the ground work was laid in order for us to continue with the rest of the thesis.

5.1.1 Interviews & Thematic Coding

The initial results came from our first round of interviews. From these, we derived a set of obstacles, a tech-stack and possible features for our DevBots. We describe the thematically coded obstacles below, and their sub-categories in Table 5.1. A more detailed version of the table data can be found in Appendix E. Below, we describe the seven main obstacles that were elicited from our participants. Furthermore, Table 5.1 shows the sub-categories of these obstacles.

3PP productification is tedious and time consuming refers to the process of updating dependencies in a software project. The expressed challenge here was the manual labour that went into verifying new dependencies. The approval process of 3PP was especially emphasized:

> "And the approval process there is rather long actually, just getting your 3pp into the system" - P5

It is a tedious process of manual work, such as processing and downloading Excel sheets, accessing data, and emailing the files to a configuration manager. Participant 5 explained the hassle of going through this process step by step:

> "We have to register products and set up product structures as well, and that takes a lot of time and especially since any third-party software you use as well has to be registered in different software centers and things to be approved" - P5

Another issue expressed was having to manually compile documents:

> “you have to take that list and put it into, document and put that into a different document that you upload to a website.” - P2

Several challenges were expressed when using the technology called **Docklin** which is a tool that many participants raised as an issue,

> “our team members spent a lot of time for it and we have to spend time to understand how it works and why do we need it.” - P3

Docklin replaces an existing process which was perceived as much simpler and easier to use, hence the backlash.

Communication was one of the most brought up challenges, this theme reflects communication between developers and other peers such as product owners and configuration managers. Participants would even express that this is the main issue

> “I mean communication Is the biggest problem” - P5

Documentation was another reoccurring theme, which groups together all issues that relate to documentation tasks such as creating READMEs, writing down instructions, and reading and interpreting instructions. Participants expressed the difficulty in finding good documentation:

> “It’s kind of hard to find documentation about how to do things” - P4

Testing presents the challenges that the interview participants experienced within the field of testing. For example, an interviewee said that:

> “I mean, testing right now would be most difficult part. I mean, it’s, it’s not difficult, it’s just time consuming” - P3

TomCat Server is an obstacle that users often thought was time-consuming:

> “So it is not like huge problem, but it takes a lot some time from developers, to have, we have a rotating like monitoring, that one person is responsible for” - P2

.

Miscellaneous obstacle category is a bit of a collection of what we decided not to cluster together, for example organisational matters:

> “There are different problems when you grow to the size of ericsson” - P5

or troubleshooting through debugging:

> “So when something goes wrong, there can be quite some time of debugging and analyzing and the like troubleshooting before we actually find why the why the data is not presenting the same way as in as in in SAP” - P6

and how modules are highly coupled:

> “interconnectedness among various modules, so that you make a change in one place. And it affects something in a completely unexpected place.” - P4

Table 5.1 shows the related sub-categories, connected through the ID shown in the first column of both tables. For example, The ID *OB1* refers to the 3PP productification process that has three sub-categories (see OB1 in Table 5.1), *Delays from dependency approval*, *Having to compile documents manually* and *Awaiting results* *and replies.*

Moreover, we wanted to gather information about obstacles we also wanted to know if our participants would trust DevBots. The majority of them said that they would but some also said that trust must be earned and that it can also be lost depending on the DevBots' performance.

> "I mean, in the beginning, I would give it some chances of course. But then if I can't trust it to to do the work properly, then then I would probably want to do it myself." - P6

> "but once if I would see that it continuously uh produces a output that is that corresponds to what I would produce as a is a human then I would probably trust it more and and and use it on a regular basis to to do the work." - P1

> "I wouldn't trust bots code review of something, so important for development." - P3

Table 5.1: Sub-categories of obstacles

ID	Obstacle	Sub-categories of obstacles		
OB1	3PP productification	Delays from dependency approval	Having to compile documents manually	Awaiting results and replies
OB2	Docklin	Annyoing/Tedious/Dont like it	More complicated than previous solution	
OB3	Communication	Context Switching	Hard to find	Finding the people you need
OB3	Communication	Time wasting	Agreeing what needs to be done and how	Difficulties Communicating and Understanding other teams
OB4	Documentation	Missing, outdated, incorrect	Hard to find	Why it is needed
OB5	Testing	Time consuming	Difficulties with testing	
OB6	TomCat Server	No sub-categories		
OB7	Miscellaneous	No sub-categories		

In addition to the workflow obstacles, the interviews were also used to gather information about the various technologies, possible features of the DevBots, as well as languages and frameworks that are being used by the two teams at Ericsson. The resulting categories are found in Table 5.2. The type *communication* is used to show the tools and applications used by the different teams within Ericsson and was of great importance to us. This is because we needed to know where to integrate our DevBot, and what frameworks we should focus on. The used languages are Java, C# and Javascript, the developers are working either on frontend, backend or DevOps. Various compilers are used based on which language is needed for the project, such as GCC or an internal tool called Bob. The IDE of choice, namely Intellij, VSCODE or WebStorm is interesting for future work, in case researchers would like to look into developing a DevBot integrated with an IDE at Ericsson. The data on preferred IDEs was initially intended to explore our options for developing an integrated DevBot. The initial idea for this thesis was to construct a DevBot for testing purposes, for example, construct a DevBot that creates unit tests, displays test coverage, etc. For that reason, we also elicited the test frameworks used in the industry: Jupiter, Junit, Selenium, and Enzyme. To organize and version control their projects, the teams used Git and Gerrit [1].

[1]gerrit `https://www.gerritcodereview.com/index.html`

Table 5.2: Table of technologies used at various Ericsson Teams

Technology type	Name				
Communication	Teams	Skype	Slack	Outlook	Confluence
Language	Javascript	Java	C#		
Context	Backend	Frontend	DevOps		
Framework	Spring				
Compiler	GCC	Bob	EPG		
IDE	Intellij	VSCode	WebStorm		
Testing	Jupiter	Junit	Selenium	Enzyme	
Version Control	Git	Gerrit			

5.1.2 First DevBots

From the data gathered from our initial round of interviews, we created our first iteration of DevBots, Teddy, and Pepe, each of them designed to tackle a specific obstacle derived from our participants requests. We include the link to the DevBots Pepe [2] and Teddy [3], however, the URL's might be invalid at the time of reading this report as the BotPress framework is going through interface updates. The section below describes the information we gathered for these DevBots using the surveys found in Appendix A & B.

5.1.3 Survey results

In both surveys we saw that the majority of participants thought that the introduction of the DevBot would not interrupt their daily workflow. Out of the participants, 80% replied that they definitely disagreed with the question "Do you think this DevBot would interrupt your workflow?" for Pepe and 50% for Teddy. The remaining 50% still disagreed that it would interrupt their workflow for Teddy (Figures 5.1 & 5.2). When it came to improving the users workflow it appeared that Teddy would offer an improvement to their daily workflow whereas Pepe received a more mixed response (Figures 5.1 & 5.2). Based on the data, we made the choice to proceed and expand the DevBot Pepe. This was because we had received a greater number of responses, more clear ideas and requests in comparison to Teddy, in addition to our participants being more in consensus when it came to the operation of the DevBot with the Pepe participants mostly wanting to interact with the DevBot through a chat interface. In contrast, the Teddy participants were equally divided between a chat interface and running in the background as seen in Figures 5.3 & 5.4. The full survey responses can be seen in Appendix F

[2]Pepe: `https://mediafiles.botpress.cloud/bb3a73c9-4f06-4c44-ab18-bb27c952eea5/webchat/bot.html`

[3]Teddy: `https://mediafiles.botpress.cloud/b8d29761-d96a-415f-9da0-13e6a500274d/webchat/bot.html`

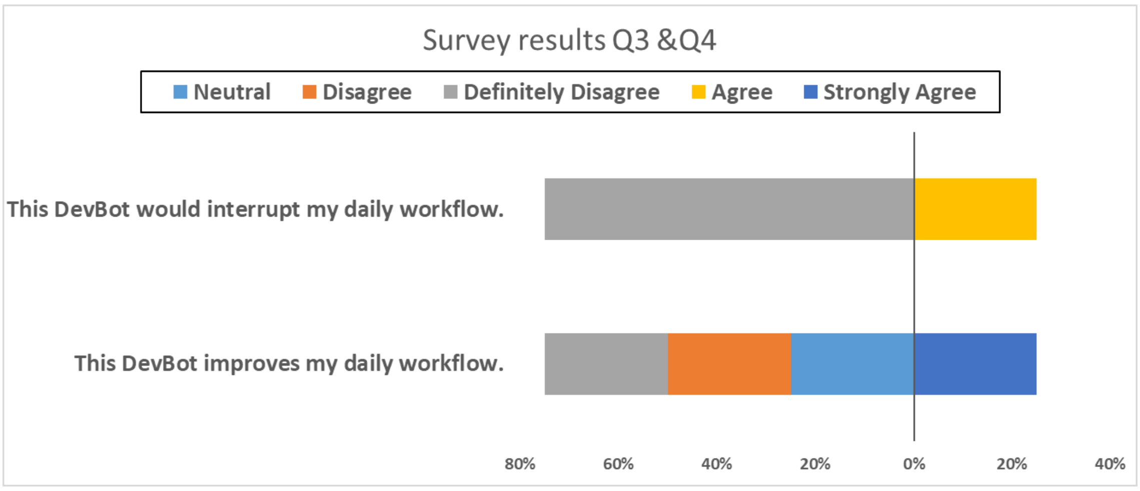


Figure 5.1: Pepe survey; Divergent chart showing the perceived interruption and improvement a DevBot would have on participants (Number of participants: 4)

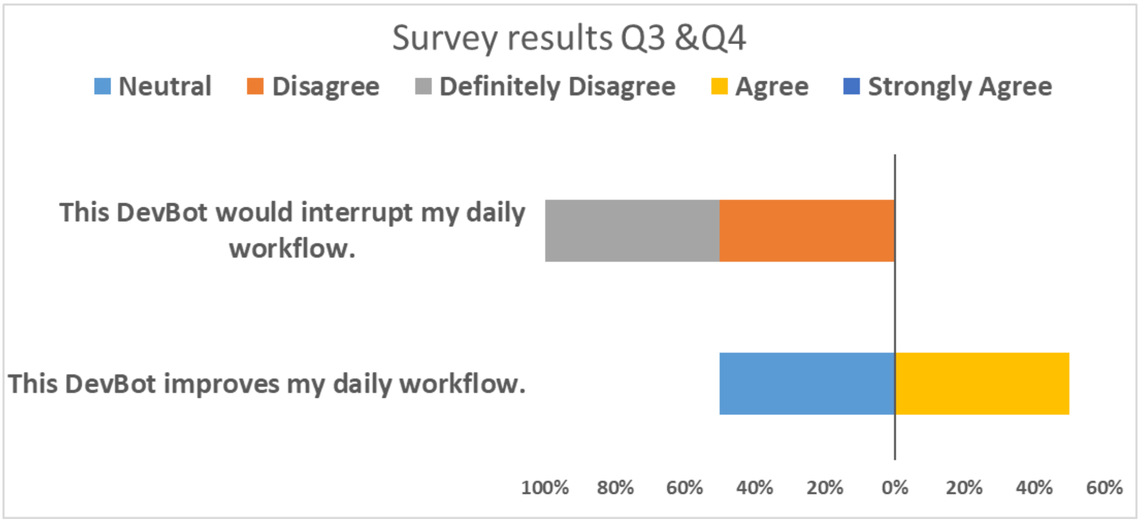
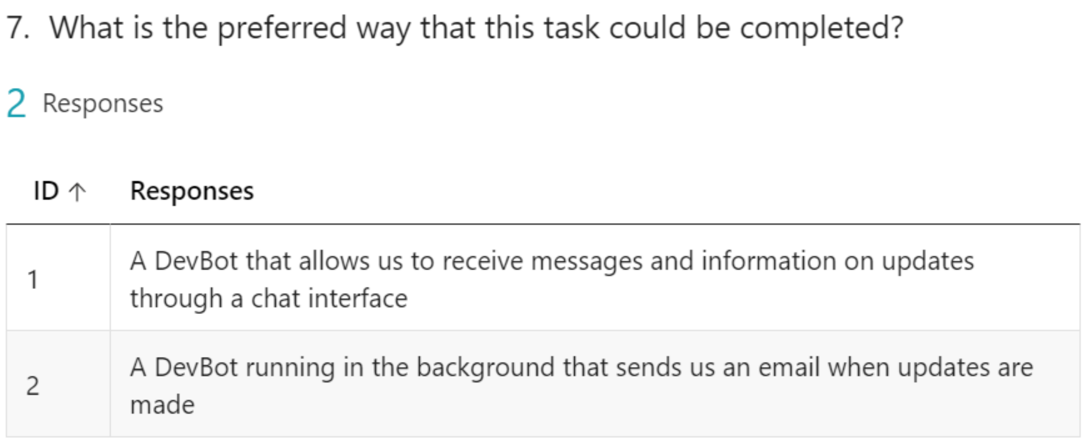


Figure 5.2: Teddy survey; Divergent chart showing the perceived interruption and improvement a DevBot would have on participants (Number of participants: 2)

5.2 Iteration two

For the second iteration, our goal was to improve the chosen DevBot from iteration one and to evaluate it again to make sure we were still headed in the right direction. With this in mind we started developing the new feature for our DevBot which was, after being given a git repository it would create an Excel sheet with all the dependencies and third party libraries included, ready for the user to inspect, modify and then send to the configuration manager. This functionality was already considered in iteration one but after it being specifically requested by our users we wanted to make sure it was added to the final prototype. Upon adding the new functionality we decided to conduct a focus group in which we would present the new version of the DevBot to see the users' response to it and to be able to have a discussion about further possible improvements, future iterations and desired features.

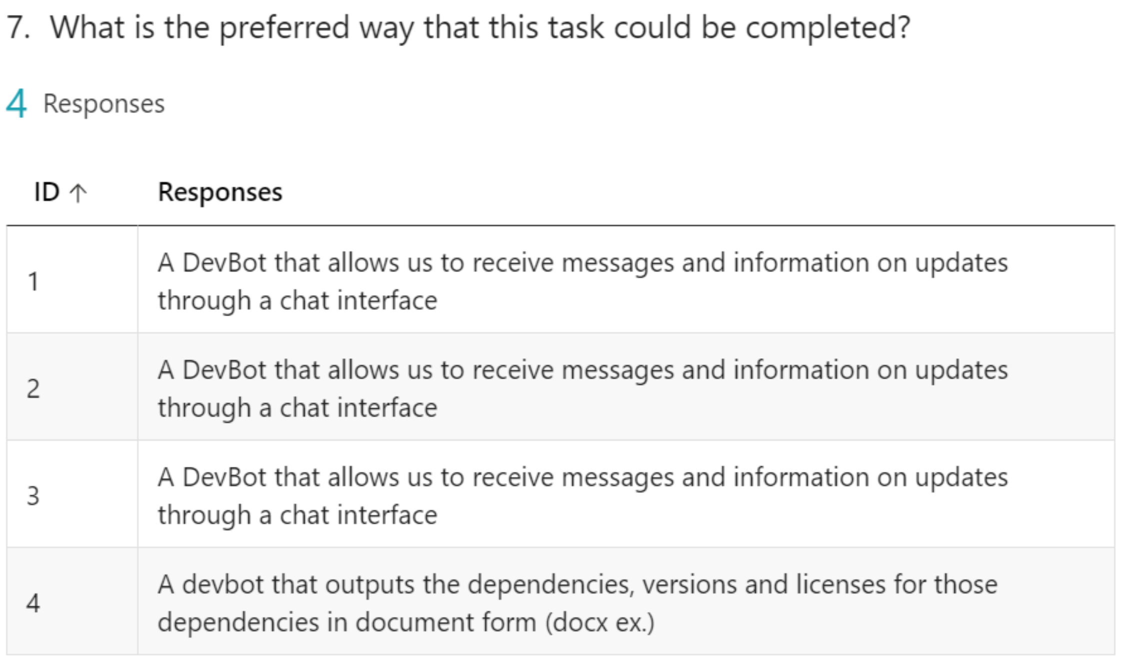


7. What is the preferred way that this task could be completed?

2 Responses

ID ↑	Responses
1	A DevBot that allows us to receive messages and information on updates through a chat interface
2	A DevBot running in the background that sends us an email when updates are made

Figure 5.3: What interaction method to participants prefer when using Teddy



7. What is the preferred way that this task could be completed?

4 Responses

ID ↑	Responses
1	A DevBot that allows us to receive messages and information on updates through a chat interface
2	A DevBot that allows us to receive messages and information on updates through a chat interface
3	A DevBot that allows us to receive messages and information on updates through a chat interface
4	A devbot that outputs the dependencies, versions and licenses for those dependencies in document form (docx ex.)

Figure 5.4: What interaction method to participants prefer when using Pepe

5.2.1 Focus Group

After completing the focus group we were delighted to see that our DevBot was well received and that our participants were excited for the possibility of having it in their arsenal in the future:

> "It has a very promising future. Even though it might not have all the all the like bells and whistles." - P2

In addition, some of our participants also provided us with useful feedback for features they would like to see in future iterations of the DevBot. For example the ability to input several repositories at the same time since most projects are not limited to one but several,

> "If you have time for like improvements, being able to put multiple like records at once, for example, would probably help out because you might be working in a context where you have fewer reports that should be regarded as one product." - P2

> "All dependencies were those projects that together make a big project would probably be very handy." - P2

or the ability to compare different dependency outputs,

> "I mean without this comparing functionality probably I would not use this bot, but if this bot would have this comparing functionality, I would use it [...]" - P3

and to include more information in the CSV file,

> "And to make it really useful for our work it would have to be a bit different. We have I think 5 or 6 columns which have some other data such as provider and I think licence and some other." - P1

but one of the most common requests was to make the DevBot easily accessible as mentioned by several participants,

> "If I will have to spend some time to find this somewhere in Ericsson internal then probably I would just not want to spend time for this" - P3

> "it would improve our workload if we had it as a contact in Teams, which we could just ping to generate this file." - P1

> "But this concept of having it in the Teams, that sounds really good." - P1

Finally, it was made clear that depending on your role in the company the DevBot would be used differently and it would also provide different benefits as well as different levels of quality of life improvements,

> "I think it depends really like on who you are." - P3

> "Well you know if you're someone who are not familiar with the projects themselves and we're just there to productify. If you're the owner of said repo, you would probably not bother, but if you are doing lots of productification over lots of repos, this would help." - P2

5.3 DevBot Frameworks

This section presents data on the frameworks that we investigated whilst trying to construct our DevBots. We provide a list of all frameworks that we explored and came across in our research in Table 5.3, and we present more detailed information below on six chosen frameworks, shown in Table 5.4.

All frameworks explored can be found in Table 5.3. The ones we looked at and tested are displayed in two columns, the first one shows the Platform or company of which the framework comes from or is currently owned by. The second column displays the various frameworks or technologies belonging to the platform/company. The list was constructed at a stage of exploration and not all might be deemed to be *frameworks* per se, but are rather technologies provided that may help the user in some way. For this reason, we provide an improved list of considered frameworks in Table 5.4.

Slack provides a Web API, the BOLT SDK, App blueprints and the RTM API.

Table 5.3: All DevBot frameworks found during exploration

Platform / Enterprise	Bot Frameworks
Slack	Web API, Bolt SDK, App blueprints, RTM API
Microsoft	Bot builder, Azure Bot Service, Microsoft Bot Framework
Other technologies	Api.ai, botmock, botman, pandorabots, chatfuel, bosify, libre, botkit, rebotify, gupshup, onsequel, bot connector, dialogflow, botpress, tars, chatterbot
Online communities	botmakers, chatbotsmagazine.org, chatbots.org
Ericsson	buddy-bot, amazon lex, IBM Watson assistant

Microsoft provides Botbuilder, Azure cloud services as well as the Microsoft Bot Frameworks which was previously owned by BotKit [4]. Other technologies holds 16 entries of frameworks or technologies, either found in research or over google. Online communities that may assist in getting started and troubleshooting are provided in this table. Lastly, Ericsson provides Buddy-bot, amazon lex and IBM Watson assistant to provide aid for a developer in need of constructing DevBot-like solutions.

Table 5.4: List of frameworks

Name of Framework	Description	Integrations	# hits on Scholar	OSS
DialogFlow	Conversational AI made by Google	10	>6000	No
BotPress	Open Source Chatbot	12	310	Yes
PowerApps	Use Power apps to create low-code apps	N/A	172 000	No
Power Virtual Agent	Design conversations on an intuitive graphical interface	15	99 000	No
Bolt SDK	It simplifies creating Slack apps.	N/A	>3500	No
Buddy-bot	Ericsson internal bot interface	2	0	No

[4]https://blogs.microsoft.com/blog/2018/11/14/microsoft-to-acquire-xoxco-bringing-together-leading-bot-development-communities-to-help-advance-conversational-ai/

Table 5.5: Frameworks checklist

Name of Framework	Open Source	Well Documented	Active Community
DialogFlow		✓	✓
BotPress	✓	✓	✓
PowerApps		✓	
Power Virtual Agent		✓	✓
Bolt SDK		✓	✓
Buddy-bot			

The frameworks listed below are displayed in Table 5.4.

DialogFlow is a natural language platform made by Google that allows you to build conversational interfaces such as DevBots. It is part of the google cloud platform and upon signing up, you receive $300 credit as of April 2023. Dialogflow provides a fairly wide range of integrations where applications such as Slack, Line, Facebook messenger, and various open source contributions (e.g., Twitter & Spark) are provided [5]. The bot framework is thoroughly discussed in research [31, 32] and it has great documentation [6] for getting up and running swiftly. There is no information on whether the framework is becoming deprecated or cancelled. It is however not open-source software. As it is such a large service, there are many blogs and tutorials.

BotPress allows for the quick creation of chatbots powered by AI. It is free and no credit card is required, but there are limitations to how much you can use the service, albeit a rather high limit. The threshold on the free limit might be an issue for larger organisations such as Ericsson, but presumably they would opt for a pricing plan if they decide to choose BotPress as a framework, which would include more requests and processing power. BotPress is able to integrate with 12 services [7] such as Line, Teams, Slack, Instagram, Messenger for Facebook, SunCo, Telegram and Viber. By entering the keyword “botpress” to google scholar we were presented with 310 results, although none were suitable to our scope or research. BotPress provides thorough documentation for getting up and running [8] with a DevBot. The community and the application seem to be growing, indicating that it is not at risk of being terminated. BotPress is open source and has a good amount of blogs and walk-throughs.

PowerApps is a Microsoft tool that enables the quick creation of applications with little to no coding. Google Scholar has over 172 000 hits for the keywords “Microsoft Power Apps”, and there are many tutorials made by the company, however not much of a community seems to exist. We report that integrations are “not available”, it

[5]https://cloud.google.com/dialogflow/es/docs/integrations

[6]Dialogflow`https://cloud.google.com/dialogflow`

[7]https://botpress.com/docs/cloud/admin-dashboard/messaging-channels/

[8]Get started BotPress:`https://botpress.com/docs/cloud/getting-started/create-and-publish-your-chatbot/`

does have possibilities for integrations but it is not a DevBot, and integrates in a different way than a DevBot would. By following Erlenhov et al.'s flowchart for determining whether or not a tool is a DevBot we found that PowerApps is likely more of a Plain Old Development Tool [2], the motivation is that it is a type of application that is commonly initiated by a human. It provides a free trial and extensive customization but it is not open source.

Power Virtual Agent is another service offered by Microsoft that fits more into the mould of being a DevBot as opposed to PowerApps and provides drag-and-drop solutions that could get you up and running in no time. It does however require you to register to Azure Cloud and MS Virtual Agent apps. It allows for integration with 15 channels such as custom websites, Facebook, Skype, Cortana, Slack, Telegram, Email, and MS Teams [9]. Power virtual agent has a notably larger community than Power Apps and there are several blog articles and tutorials to be found. The keywords "MS power virtual agent" yields 99 000 hits on Scholar. As it is a Microsoft service, it is not open source.

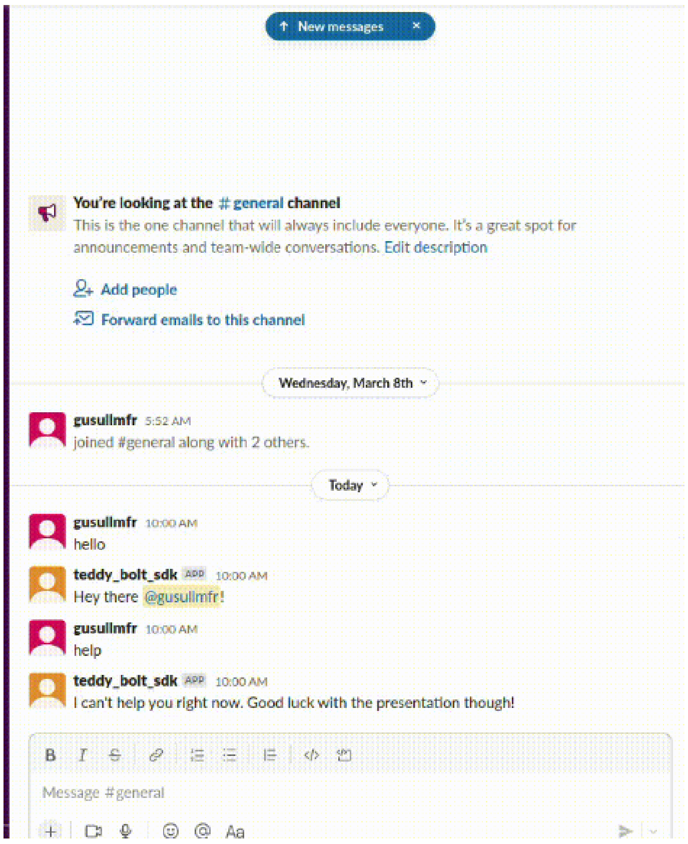
Bolt SDK is a product provided by Slack that allows a developer to create a DevBot with only a few lines of code. The amount of integrations for Bolt SDK is difficult to answer as you need to connect Slack itself to an application, not the DevBot. Slack can be integrated with more than 2400 applications, and the DevBot would be included in the Slack app integration. As the integration is not focused on the DevBot itself we wrote "Not Applicable" in our table of frameworks (Table 5.4). It seems to have a large community and many blogs, tutorials and YouTube videos in order to get started, troubleshoot, and develop. Slack also provides excellent tutorials themselves [10]. A code snippet and a minimal version of a working DevBot is shown in Figures 5.5 & 5.6.

It has more than 3 500 hits on Google Scholar. However, we could not access any papers of interest but relied on articles and Slack's own material. The software is free to use, but not open-source.

The last framework that we investigated was *buddy-bot*, an internal Ericsson bot-building framework. It seems that integration with Teams, and Ericssons internal portal, are the most common use cases. The keywords "Buddy-bot Ericsson" yields no hits on Google Scholar, and it was difficult finding information internally as well. It is free to use if you are an employee, but you need access to its ecosystem. Needless to say, buddy-bot is not open source.

[9]https://learn.microsoft.com/en-us/power-virtual-agents/publication-fundamentals-publish-channels

[10]https://slack.dev/bolt-js/tutorial/getting-started



You're looking at the # general channel
This is the one channel that will always include everyone. It's a great spot for announcements and team-wide conversations. Edit description

gusullmfr 5:52 AM
joined #general along with 2 others.

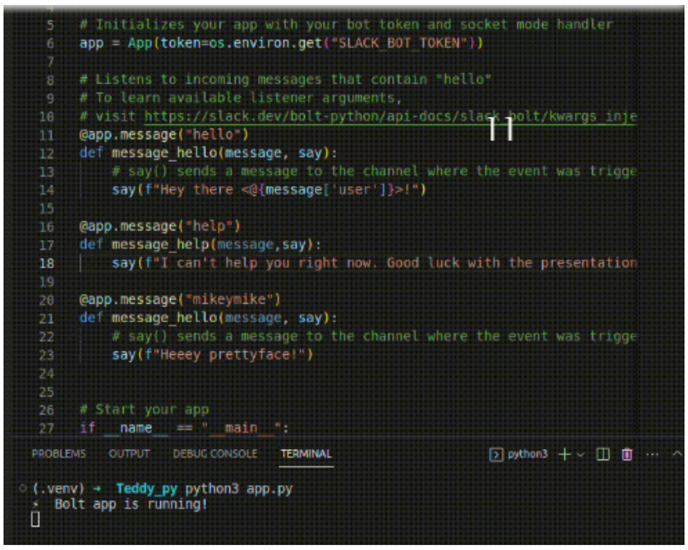
gusullmfr 10:00 AM
hello

teddy_bolt_sdk APP 10:00 AM
Hey there @gusullmfr!

gusullmfr 10:00 AM
help

teddy_bolt_sdk APP 10:00 AM
I can't help you right now. Good luck with the presentation though!

Figure 5.5: Conversation with a Slackbot created using Bolt SDK



```
# Initializes your app with your bot token and socket mode handler
app = App(token=os.environ.get("SLACK_BOT_TOKEN"))

# Listens to incoming messages that contain "hello"
# To learn available listener arguments,
# visit https://slack.dev/bolt-python/api-docs/slack_bolt/kwargs_inje
@app.message("hello")
def message_hello(message, say):
    # say() sends a message to the channel where the event was trigge
    say(f"Hey there <@{message['user']}>!")

@app.message("help")
def message_help(message,say):
    say(f"I can't help you right now. Good luck with the presentation

@app.message("mikeymike")
def message_hello(message, say):
    # say() sends a message to the channel where the event was trigge
    say(f"Heeey prettyface!")


# Start your app
if __name__ == "__main__":
```

```
(.venv) → Teddy_py python3 app.py
⚡ Bolt app is running!
```

Figure 5.6: Code snippet for Slackbot using Bolt SDK

6
Discussion

6.1 DevBot

In this section, we reflect on our results, answer the research questions and present threats to validity.

RQ1: Which obstacles do practitioners/developers face in their day-to-day life?
Practitioners and developers at Ericsson face obstacles in 3PP productification, testing, DevOps, Docklin, communication, and documentation, impacting project timelines, budgets, and quality.

In the context of software development, practitioners and developers face various obstacles in their day-to-day lives. Erlenhov et al. [2] provided us with the tools to explain what a DevBot is to the practitioners, and therefore understand what a DevBot could assist within the development process. In the confines of Ericsson and the teams that we interviewed, the challenges can be categorised into five main areas: 3PP productification, testing, docklin, communication, and documentation. The first obstacle is the tedious and time-consuming process of 3PP productification, which involves integrating third-party products into a project. This process often requires significant effort and can delay the project timeline. Secondly, testing takes time, and the development team must ensure that the software functions correctly and meets the specified requirements. Thirdly, Docklin, a tool used for container management, can be a challenging task that requires considerable technical expertise. Communication, the fourth obstacle, is vital in the development process. Clear and concise communication is necessary for effective collaboration among team members. Finally, documentation is essential for tracking progress, ensuring consistency, and maintaining project knowledge, but it can also be tedious. Overall, these obstacles can significantly impact project timelines, budgets, and the quality of the final product, and practitioners and developers must work to overcome them.

RQ2: How can development bots improve an existing workflow?
By removing tedious tasks in the developer's daily routines, such as managing tickets, verifying dependencies for FOSS, or simplifying testing tasks.

DevBots can be utilised to solve the various obstacles that were expressed by participants by automating [4], or simplifying tasks [5] with the aim of lifting some weight and responsibility off the participants shoulders. Participants eagerly expressed these obstacles and to solve a part of the 3PP obstacle using a DevBot, we created Pepe that would attempt to create the necessary documentation as quickly and as accurately as possible without encumbering the developer. As for testing [12], we are aware of the many DevBot solutions that aim to solve the plethora of obstacles within the field, such as RepairNator and the Hallelujah DevBots. The Docklin obstacles were many and there was expressed frustration over the introduction of this new tool. We nevertheless did not find any good solutions to this obstacle as it was more of a structural issue, since Docklin was already implemented, it was just very time consuming and tedious to do so. Communication was often expressed as an obstacle and a solution in the shape of a DevBot would be suitable, unfortunately we did not gather enough requirements or information to know specifically how a DevBot could be utilised to solve an existing communication issue within a teams' workflow. Storey and Zagalski do however argue that DevBots could be used to fill communication gaps [4]. Documentation, the last of the five workflow obstacles that we found, was a workflow process where we elicited enough information to be able to construct a possible solution, Teddy the DevBot. This idea was reinforced by Wessel et al. [11] who mentions documentation as a possible task for a DevBot.

RQ3: How does the introduction of a DevBot influence the developers in their daily lives?
Introducing DevBots into the daily life of a developer was perceived as positive and while trust is often an issue, many were comfortable delegating more simple tasks to a DevBot.

Our survey data indicates that Teddy would likely improve the developers' workflow and that the risk of feeling interrupted would not be very high. With that in mind, we believe that implementing this DevBot in order to attempt to improve the existing workflow is advised. In a similar fashion, data from the survey showed that the Pepe DevBot would also likely improve the developers' workflow. Based on the results from the initial survey and the interviews, we would therefore claim that if you are presented with a DevBot that solves tasks that you have explicitly requested, and the integration of such a DevBot is simple, then its introduction will be positive and would improve the developers' workflow and daily routines.

Our key takeaways were that depending on what your role is in the company or team, be that of a developer, product owner, manager, etc., the DevBot will have different benefits and certain features are more useful than others. Additionally, it is important that the DevBot is easily accessible, is compatible with existing tools like Gerrit, Microsoft Teams and can operate with existing projects and file structures. The final takeaway was that our participants have shown interest and enthusiasm for this DevBot and its future iterations, signifying to us that we have taken the first steps in the right direction.

Lastly, the developers expressed trust in DevBots[33], but only for simple tasks.

Anything complex such as validating merge requests from coworkers makes the developer more wary of the DevBot's actions. Additionally, some participants expressed that trust is earned so they would want to see the DevBot perform before putting their faith in it. Finally, it was also made clear that just as someone can come to trust a DevBot they can just as easily lose that trust if the outputs are not as expected. We believe that this is very logical as no one would want to use a tool that is not reliable.

6.2 Frameworks

The benefits and disadvantages of using certain DevBot frameworks within big corporations like Ericsson depend on various factors. These factors are discussed below, together with our advice for choosing a framework.

RQ4: What are the benefits and disadvantages of using certain DevBot frameworks when constructing DevBots within big corporations like Ericsson?
Choosing a framework to construct DevBots could reduce time and bugs. It is however difficult and could take months to introduce new tools into a corporate ecosystem. We suggest either an in-house tool or an already added third-party ecosystem such as Microsoft Power Apps.

Advantages

One significant benefit is the speed at which developers can build and deploy DevBots using frameworks. Using frameworks will save resources such as time and money and reduce the chance of bugs. Additionally, using popular frameworks can provide a larger pool of developers familiar with the technology and reduce the time and costs of training new developers. Working with templates and frameworks also allows more people other than developers to turn their use cases into a DevBot, as many frameworks provide no-code solutions to create DevBots.

Disadvantages

Even though there are many benefits to working with frameworks, there are also disadvantages, such as a lack of customisation options or being tied to a specific vendor (i.e., vendor lock-in). Additionally, certain frameworks may not be suitable for complex use cases or may need to integrate better with existing systems. Introducing new applications into an enterprise ecosystem is difficult due to strict approval processes, a time consuming endeavour unless it is already verified as an approved framework. Due to these strict processes, engineers recommended that we opt for a proof of concept of a DevBot instead of attempting to construct a finished product, as the deployment procedure alone could be stretched over several months. Another reason for larger corporations to choose their in-house or tools built from scratch is that the frameworks we tested and explored changed frequently, even in the few weeks we spent researching. This could mean that documentation and knowledge quickly becomes outdated. An external company providing framework solutions may also deal with data in a way that is non-compliant with individuals or corporations, which was the case for us as Ericsson has very secure networks and data processing regulations.

Choosing frameworks

Ultimately, the choice of DevBot framework within a big corporation like Ericsson will depend on various factors, including the organisation's goals, data storing regulations, budget, existing infrastructure, and development team's expertise.

Reinventing the wheel each time a tool is needed would undoubtedly be redundant labour and increase the chance of introducing erroneous code. Collaborating with external services such as Microsoft Power Virtual Agents or even BotPress is, therefore, much of an advantage.

Google and their service Dialogflow was our first choice, as they rarely seem to disappoint. However, although it may be a great tool, it did not fit our criteria. PowerApps is excellent for constructing tools but is not a framework for building DevBots. Power Virtual Agents was the most promising out of all of our frameworks, but we were restricted by Azure cloud permission and a paywall. Bolt SDK only works with Slack, which Ericsson employees do not use officially. BotPress was the most promising but is not included in the Ericsson ecosystem.

Lessons learned

Constructing DevBots using frameworks provided us with insights and lessons.

First, a developer should consider the purpose and requirements when choosing a framework, as each has its strengths and limitations. Second, robust documentation and supportive communities can help facilitate the development process and are helpful for troubleshooting. Third, finding a framework that allows for integrating different channels, such as Teams, Slack, or Whatsapp, would expand the reach of the DevBot. Fourth, the developer should consider the level of customisation, it would likely result in a trade-off between control, ease of use and flexibility. Lastly, the developer needs to consider the availability of resources such as free credit and access to ecosystems.

An individual building a proof of concept or smaller models would likely care most about a quick-free setup, whilst companies like Ericsson, which are in need of higher quality of security, latency and other essential metrics, might instead look for the most suitable pricing structure. Availability could impact the choice, especially for developers on a budget or who wish to hit the ground running. When we explored DevBot frameworks, we had the initial interview criteria to base our choices on. Notably, the framework should be able to integrate into the Ericsson ecosystem. We also had personal preferences, simplicity, community and amount of tutorials in mind when making our choices.

After collecting the data, we realised that choosing a framework will be a highly individual matter and too many factors play a role in order for us to provide a silver bullet. We can however motivate why we chose BotPress as our DevBot framework of choice, based on the lessons learnt above: it was fast to set up, learn and develop with. It provided the ease and simplicity of a drag and drop interface while also having the option for a more flexible and customisable code based development. Another benefit was the good documentation and healthy community that existed in addition to the variety of options for integration, where we were able to pick and choose from an extensive list of platforms to host our DevBot. Furthermore, the support team at BotPress was also quite beneficial to us. They were always available for queries and to provide assistance and guidance where needed as well as to receive feedback from us for issues and changes that they promised they will include in future updates. Lastly, as we aimed to pass on the DevBot after we finish the project, we wanted to make sure that the programming skills required were low. The last one reduced flexibility and control, but it increase ease of use and was a choice we had to make.

Our conclusion is that based on the previous factors, it is advised to choose either an in-house, custom-made tool such as Buddy-bot which is used by Ericsson as mentioned before or options already part of the ecosystem, in the case of their platforms such as Teams and Microsoft Azure using the Power Virtual Agents tool. The motivation for these choices is mainly that it would save time and effort. The downside is that great solutions might not make their way into the ecosystem, such as BotPress, which we used for both our DevBots Pepe and Teddy.

6.3 Threats to Validity

The study's validity follows Runesson's [30] four aspects of validity: internal, construct, external, and lastly reliability.

Construct validity: As we collected data from participants on three separate occasions there are plenty of threats to the thesis's construct validity. Since there is a lot of terminology, it is possible that the interview, survey and focus group questions were interpreted unexpectedly, we did however provide the participants with explanations and descriptions of all important terms before getting started, we also opened up the room for questions at any point. For example, the term *DevBot* is not yet very widely known, and many interpretations of the term bot are used. Therefore, the construct validity could be at stake, as there are many opportunities for misunderstandings and miscommunication. However, since we took into consideration the bot personas presented by Erlenhov et al., [2] when presenting DevBots to the various participants, we believe the threat was not as prominent as it could have been. We also had the option to conduct an observational study instead of the survey, which would have meant more control and possibilities to elicit richer data. We understood the trade-off between rich data and speed by choosing surveys, as we felt that it was the most time efficient and effective way to quickly gather feedback.

Internal validity is at risk if we fail to mitigate the outlying factors when experimenting. The transcriptions of the interview sessions could have been further validated, by sending them back to the participant instead of simply summing up at the end of the interview. We could have asked a third, non-biased member to join in for our thematic coding session. We did mitigate this by asking our supervisor for guidance and feedback on the thematic table and the coding. The survey also poses risks of validity, it is for example difficult to tell whether the artifact is working if all participants would easily complete the assigned task. We could argue that the task in the survey where participants interact with the DevBot, was made too simple and could have included a larger part of the productification process, and would thus bias participants into thinking positively about the DevBot. To mitigate this, we showcased the DevBot once more in a focus group, where we explained in more detail its possibilities and its flaws. Regarding the focus group study, it could have been pilot-tested more rigorously beforehand, and we could have planned it better to include more participants. The participants were requested to partake in interviews and surveys at any time, it could be argued that a participant filling in a form late in the afternoon could be more quick about awarding good scores, as they might be more tired than earlier in the day. We tried to keep the meetings and interviews scheduled mid-day to mitigate this. Lastly, we found that it is common to use students as participants due to its simplicity, for studies in Software engineering [24], we did however have the opportunity to interview and survey participants from industry, and therefore strengthening the internal validity.

External validity: Even though the initial results were relatively positive, the number of responses we received in our surveys can be considered rather small, six responses in total, two for Teddy and four for Pepe. This was not necessarily a bad turnout if we take into account the number of participants we had in our interviews which were also six, and the team sizes in Ericsson. Ideally, we would have liked to have more data to analyse but we were still able to extract enough to proceed to the second iteration of our DevBot. It is difficult to argue that the information applies to all developers as it was collected from participants from the same organisation. In an attempt to increase the generalizability, we selected two teams from different fields that were also not located in the same city. As the artifact is intended to be operating on several platforms, the research could be applied to any development team that is working with the process of productification, strengthening its external validity. Lastly, the information on how to construct a DevBot could be difficult to generalize, even though we intended to provide guidelines to anyone who wishes to understand the creation of the DevBots. We mitigated this by focusing more on the context of how to create DevBots, and less on the details of how the DevBots were constructed.

Reliability: As researchers we have some differences in our background coming from two different fields, software engineering, and computer science. This difference in background increases the reliability of this study since we view things in different ways and we can be more critical of each others work. As for member checking [34], we did not send the participants their own transcripts for validation, which could mean analysing incorrectly processed data. We mitigated this by summarizing the data collected with each participant as we ended each session and had the privilege to be able to walk into their office and ask clarifying questions at any point in time. The data collected would likely be different as it is not probable to conduct the study with the same participants, who hold expert knowledge in their fields. We did however follow guidelines and set methodology, which should in theory allow researchers to reconstruct the thesis experiments and data collection. Another risk is that the thematic coding, as well as the data extraction from the focus group might be analysed differently. Researchers can mitigate this by using best practices.

6.4 Conclusion

Software development bots are a concept which solves repetitive tasks that a developer performs. Based on elicited requirements, we created a DevBot that would solve an issue in the workflow of a developer team. In order to evaluate our DevBot we used the SPACE framework, which uses more soft metrics such as *developer satisfaction* rather than the harder metrics such as lines of code pushed or amounts of merge requests. We interviewed six participants over two teams to find the requirements and we created our DevBot in three steps, an interview, a survey and a focus group. We found that the DevBot would indeed be useful, but it needs to execute its tasks correctly and without interrupting workflows in order for the developers to want to use it and trust it. We also investigated frameworks that assist in developing DevBots, which we argue should be utilized as opposed to reinventing the wheel. There is no such thing as a one size fits all framework but a developer would have to base their decision on the requirements of the DevBot. Additionally, we propose future work, where we provide features and recommended steps for proceeding with the development of the DevBots.

6.4.1 Extending Pepe and Teddy

In this section we present features and ideas that could be used and built upon in future research.

6.4.2 Future Features

Here we briefly discuss the potential future of Pepe. Based both on feedback and the initial interviews the user should be able to send an email to a configuration manager when the list is completed, and to offer multiple extension types of the file. We also received input for aggregating multiple repositories when providing the file. Furthermore, a practitioner would also have use for being able to compare the different dependency lists from each repository, which could also include more rich information. It is also especially important for Pepe to be integrated into Microsoft Teams so that it can be easily accessible for all users or if not Teams then some other chat interface that users will not have to go out of their way to use. Additionally, after the focus group we realised that different users with different roles within a team would have varying needs and uses for Pepe, so with that in mind it would be interesting to see a version of Pepe that offers different interactions, suggestions and functionality to a user based on their assigned role in a project. Finally, we believe that Pepe could potentially be the start to a flourishing ecosystem of DevBots all working and interacting together to accomplish more sophisticated and complicated tasks.

6.4.2.1 Recommended steps DevBots

While we only created a proof of concept we also provide a set of steps that we believe anyone can follow if they want to replicated or expand our work into a working environment. The full list of execution steps for both DevBots can be found in the Implementation chapter.

6.4.2.2 Pepe

Pepe is the DevBot that received the most responses in iteration one survey, it has concrete tasks, and it would be possible to implement the new feature elicited from the survey. The team from which we got the use case for this DevBot were also available for a follow-up focus group, are close geographically and could continuously give us feedback and input on Pepe. We previously had participants express that receiving dependencies in a document was the preferred solution, which we iterated back to and asked about in the survey. A participant expressed that the best way to execute the task was to save information in a file.

> "A devbot that outputs the dependencies, versions and licenses for those dependencies in document form (docx ex.)" - P5

We decided to implement this feature, where Pepe gathers the dependencies from the repository, saves them into an Excel sheet and provides it to the user interacting with the chatbot. The list of execution steps can be seen in list 4.1.1.

6.4.2.3 Teddy

While Teddy was ultimately not fully implemented the survey answers we received offer some insight on whether Teddy could improve the workflow as seen with feedback such as:

> "What I can see it covers all needed process steps" - Participant 3

> "it looks alright, I think these are the correct steps" - Participant 2

The actions that the DevBot must take to fulfil the team's requirements were therefore considered complete. The list of execution steps can be seen in list 4.1.2.

Bibliography

[1] L. Erlenhov, F. G. de Oliveira Neto, R. Scandariato, and P. Leitner, “Current and future bots in software development,” in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE).* IEEE, 2019, pp. 7–11.

[2] L. Erlenhov, F. G. D. O. Neto, and P. Leitner, “An empirical study of bots in software development: Characteristics and challenges from a practitioner’s perspective,” in *Proceedings of the 28th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2020, pp. 445–455.

[3] N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, and J. Butler, “The space of developer productivity: There’s more to it than you think.” *Queue*, vol. 19, no. 1, pp. 20–48, 2021.

[4] M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, 2016, pp. 928–931.

[5] S. Santhanam, T. Hecking, A. Schreiber, and S. Wagner, “Bots in software engineering: a systematic mapping study,” *PeerJ Computer Science*, vol. 8, p. e866, 2022.

[6] C. Lebeuf, M.-A. Storey, and A. Zagalsky, “Software bots,” *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2017.

[7] C. Lebeuf, A. Zagalsky, M. Foucault, and M.-A. Storey, “Defining and classifying software bots: A faceted taxonomy,” in *2019 IEEE/ACM 1st international workshop on bots in software engineering (BotSE).* IEEE, 2019, pp. 1–6.

[8] B. W. Arden, B. A. Galler, and R. M. Graham, *The Michigan Algorithm Decoder.* UM Libraries, 1965.

[9] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, Z. Ming *et al.*, “Github copilot ai pair programmer: Asset or liability?” *arXiv preprint arXiv:2206.15331*, 2022.

[10] S. Urli, Z. Yu, L. Seinturier, and M. Monperrus, “How to design a program repair bot? insights from the repairnator project,” in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, pp. 95–104.

[11] M. Wessel, I. Steinmacher, I. Wiese, and M. A. Gerosa, “Should i stale or should i close? an analysis of a bot that closes abandoned issues and pull requests,” in *2019 IEEE/ACM 1st international workshop on bots in software engineering (BotSE)*. IEEE, 2019, pp. 38–42.

[12] D. Platis, “Software development bot ecosystems,” University of Chalmers, 2021.

[13] R. Ablett, E. Sharlin, F. Maurer, J. Denzinger, and C. Schock, “Buildbot: Robotic monitoring of agile software development teams,” in *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2007, pp. 931–936.

[14] C. Matthies, F. Dobrigkeit, and G. Hesse, “An additional set of (automated) eyes: chatbots for agile retrospectives,” in *2019 IEEE/ACM 1st international workshop on bots in software engineering (BotSE)*. IEEE, 2019, pp. 34–37.

[15] S. Bieliauskas and A. Schreiber, “A conversational user interface for software visualization,” in *2017 ieee working conference on software visualization (vissoft)*. IEEE, 2017, pp. 139–143.

[16] M. Wyrich and J. Bogner, “Towards an autonomous bot for automatic source code refactoring,” in *2019 IEEE/ACM 1st international workshop on bots in software engineering (BotSE)*. IEEE, 2019, pp. 24–28.

[17] V. Balachandran, “Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation,” in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 931–940.

[18] K. M. Unertl, L. L. Novak, K. B. Johnson, and N. M. Lorenzi, “Traversing the many paths of workflow research: developing a conceptual framework of workflow terminology through a systematic literature review,” *Journal of the American Medical Informatics Association*, vol. 17, no. 3, pp. 265–273, 2010.

[19] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, “Software developers’ perceptions of productivity,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 19–29.

[20] A. Murgia, D. Janssens, S. Demeyer, and B. Vasilescu, “Among the machines: Human-bot interaction on social q&a websites,” in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2016, pp. 1272–1279.

[21] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, “What to expect from code review bots on github? a survey with oss maintainers,” in *Proceedings of the 34th Brazilian symposium on software engineering*, 2020, pp. 457–462.

[22] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS quarterly*, pp. 75–105, 2004.

[23] E. Knauss, “Constructive master’s thesis work in industry: guidelines for applying design science research,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).* IEEE, 2021, pp. 110–121.

[24] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting empirical methods for software engineering research,” in *Guide to advanced empirical software engineering.* Springer, 2008, pp. 285–311.

[25] C. B. Seaman, “Qualitative methods in empirical studies of software engineering,” *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[26] J. W. Creswell *et al.*, “Qualitative, quantitative, and mixed methods approaches,” 2003.

[27] G. R. Gibbs, “Thematic coding and categorizing,” *Analyzing qualitative data*, vol. 703, pp. 38–56, 2007.

[28] K.-J. Stol, P. Ralph, and B. Fitzgerald, “Grounded theory in software engineering research: a critical review and guidelines,” in *Proceedings of the 38th International conference on software engineering*, 2016, pp. 120–131.

[29] I. Etikan, S. A. Musa, R. S. Alkassim *et al.*, “Comparison of convenience sampling and purposive sampling,” *American journal of theoretical and applied statistics*, vol. 5, no. 1, pp. 1–4, 2016.

[30] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[31] N. Sabharwal, A. Agrawal, N. Sabharwal, and A. Agrawal, “Introduction to google dialogflow,” *Cognitive Virtual Assistants Using Google Dialogflow: Develop Complex Cognitive Bots Using the Google Dialogflow Platform*, pp. 13–54, 2020.

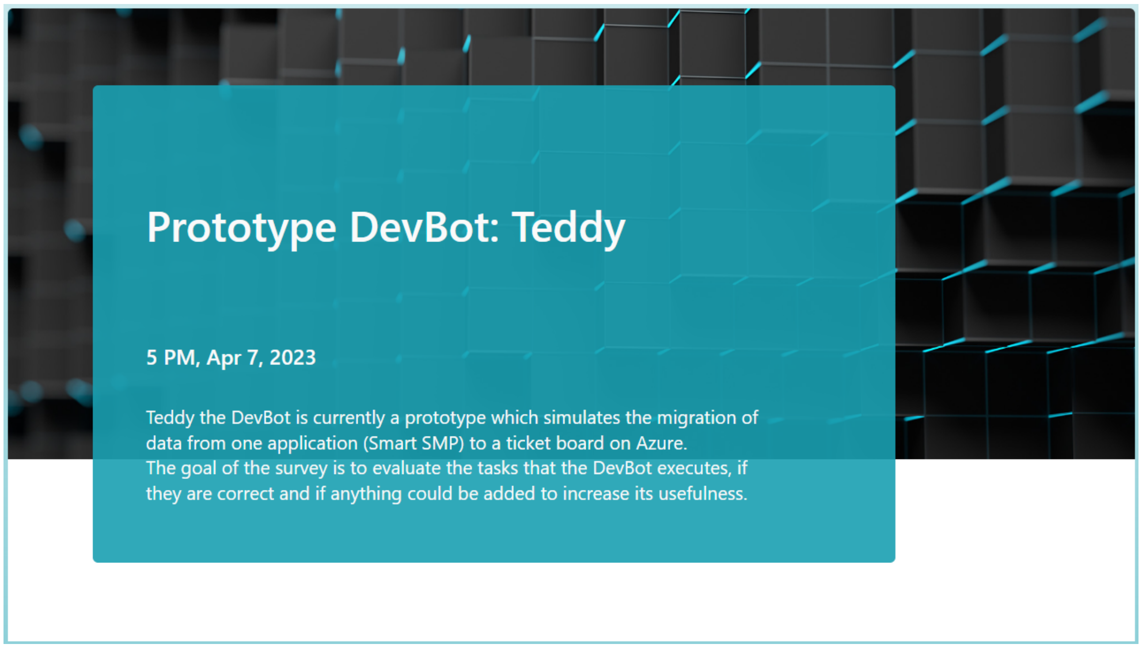
[32] R. Reyes, D. Garza, L. Garrido, V. De la Cueva, and J. Ramirez, “Methodology for the implementation of virtual assistants for education using google dialogflow,” in *Advances in Soft Computing: 18th Mexican International Conference on Artificial Intelligence, MICAI 2019, Xalapa, Mexico, October 27–November 2, 2019, Proceedings 18.* Springer, 2019, pp. 440–451.

[33] L. Erlenhov, F. G. de Oliveira Neto, and P. Leitner, “Dependency management bots in open-source systems—prevalence and adoption,” *PeerJ Computer Science*, vol. 8, p. e849, 2022.

[34] L. Birt, S. Scott, D. Cavers, C. Campbell, and F. Walter, “Member checking: a tool to enhance trustworthiness or merely a nod to validation?” *Qualitative health research*, vol. 26, no. 13, pp. 1802–1811, 2016.

A

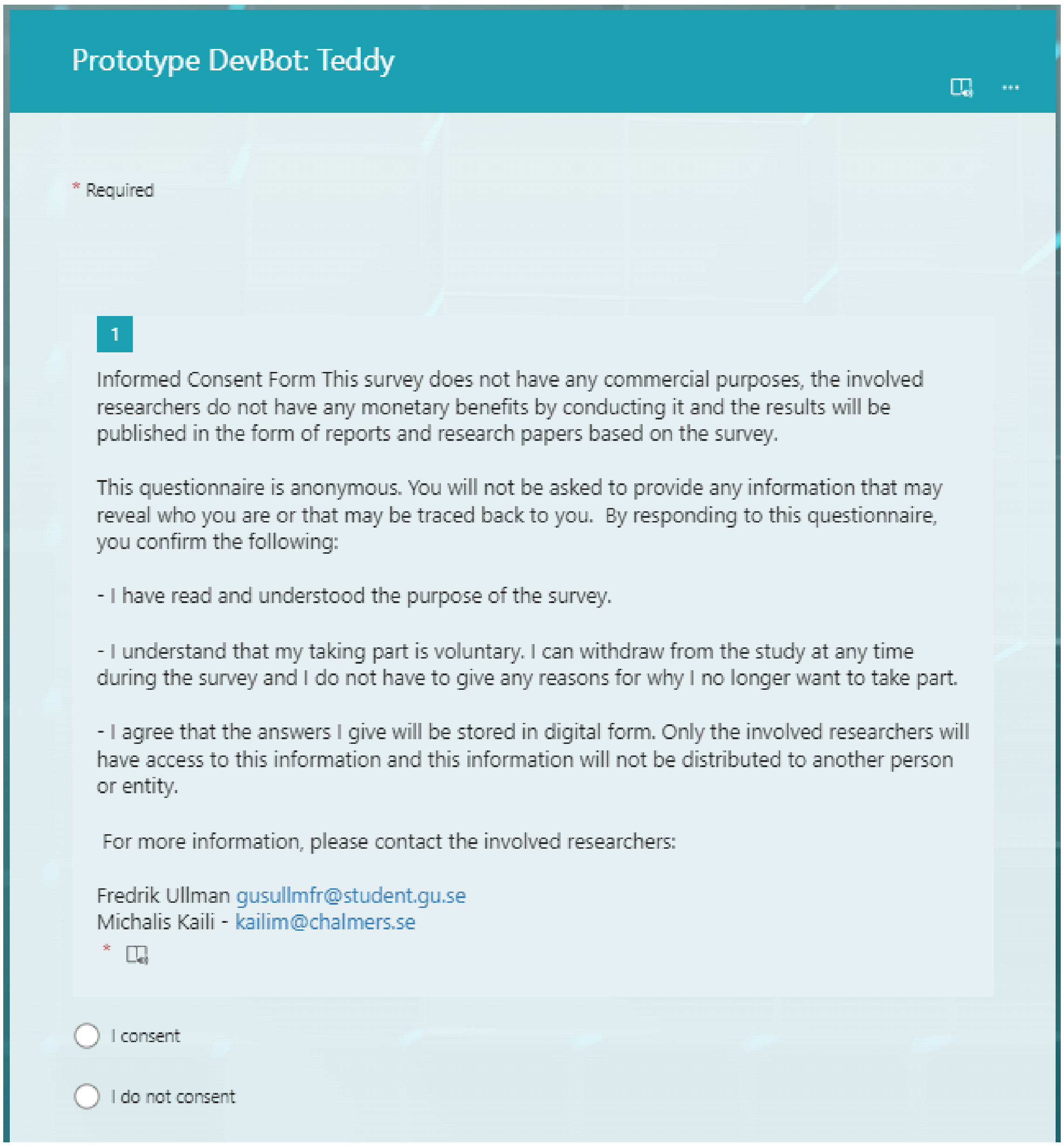
Appendix 1



Prototype DevBot: Teddy

5 PM, Apr 7, 2023

Teddy the DevBot is currently a prototype which simulates the migration of data from one application (Smart SMP) to a ticket board on Azure.
The goal of the survey is to evaluate the tasks that the DevBot executes, if they are correct and if anything could be added to increase its usefulness.



Prototype DevBot: Teddy

* Required

1

Informed Consent Form This survey does not have any commercial purposes, the involved researchers do not have any monetary benefits by conducting it and the results will be published in the form of reports and research papers based on the survey.

This questionnaire is anonymous. You will not be asked to provide any information that may reveal who you are or that may be traced back to you. By responding to this questionnaire, you confirm the following:

- I have read and understood the purpose of the survey.

- I understand that my taking part is voluntary. I can withdraw from the study at any time during the survey and I do not have to give any reasons for why I no longer want to take part.

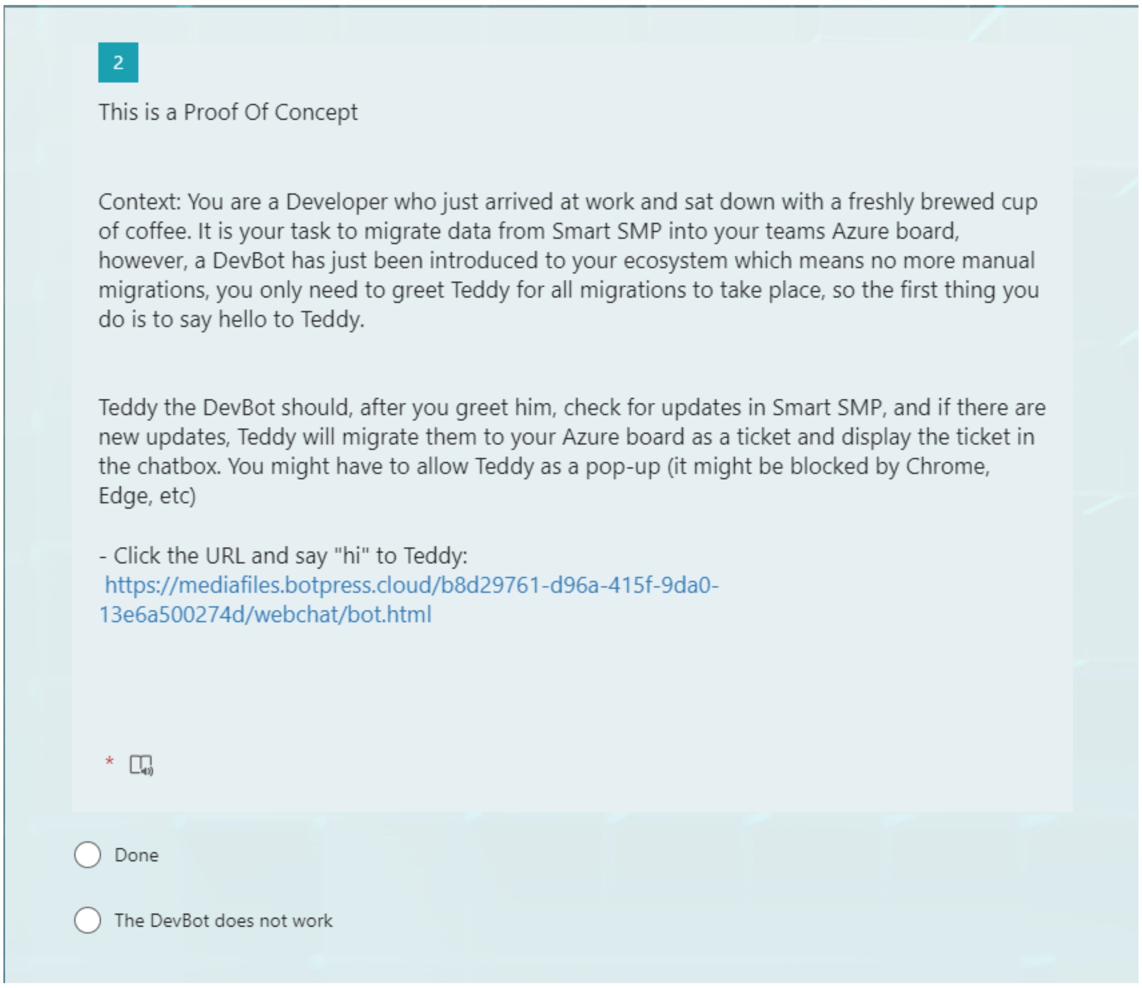
- I agree that the answers I give will be stored in digital form. Only the involved researchers will have access to this information and this information will not be distributed to another person or entity.

For more information, please contact the involved researchers:

Fredrik Ullman gusullmfr@student.gu.se
Michalis Kaili - kailim@chalmers.se
*

◯ I consent

◯ I do not consent



2

This is a Proof Of Concept

Context: You are a Developer who just arrived at work and sat down with a freshly brewed cup of coffee. It is your task to migrate data from Smart SMP into your teams Azure board, however, a DevBot has just been introduced to your ecosystem which means no more manual migrations, you only need to greet Teddy for all migrations to take place, so the first thing you do is to say hello to Teddy.

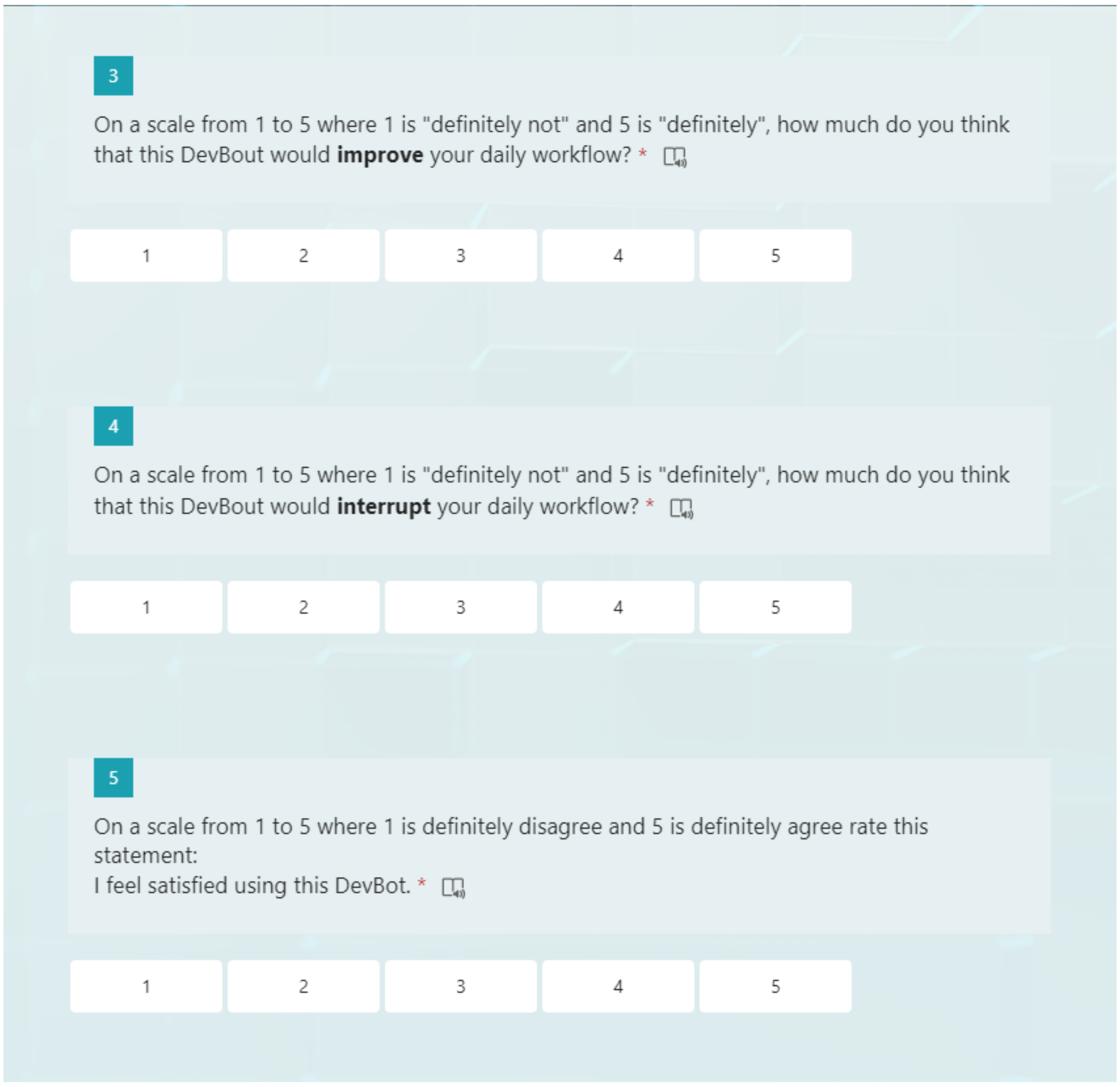
Teddy the DevBot should, after you greet him, check for updates in Smart SMP, and if there are new updates, Teddy will migrate them to your Azure board as a ticket and display the ticket in the chatbox. You might have to allow Teddy as a pop-up (it might be blocked by Chrome, Edge, etc)

- Click the URL and say "hi" to Teddy:
https://mediafiles.botpress.cloud/b8d29761-d96a-415f-9da0-13e6a500274d/webchat/bot.html

*

○ Done

○ The DevBot does not work



3

On a scale from 1 to 5 where 1 is "definitely not" and 5 is "definitely", how much do you think that this DevBout would **improve** your daily workflow? *

1	2	3	4	5

4

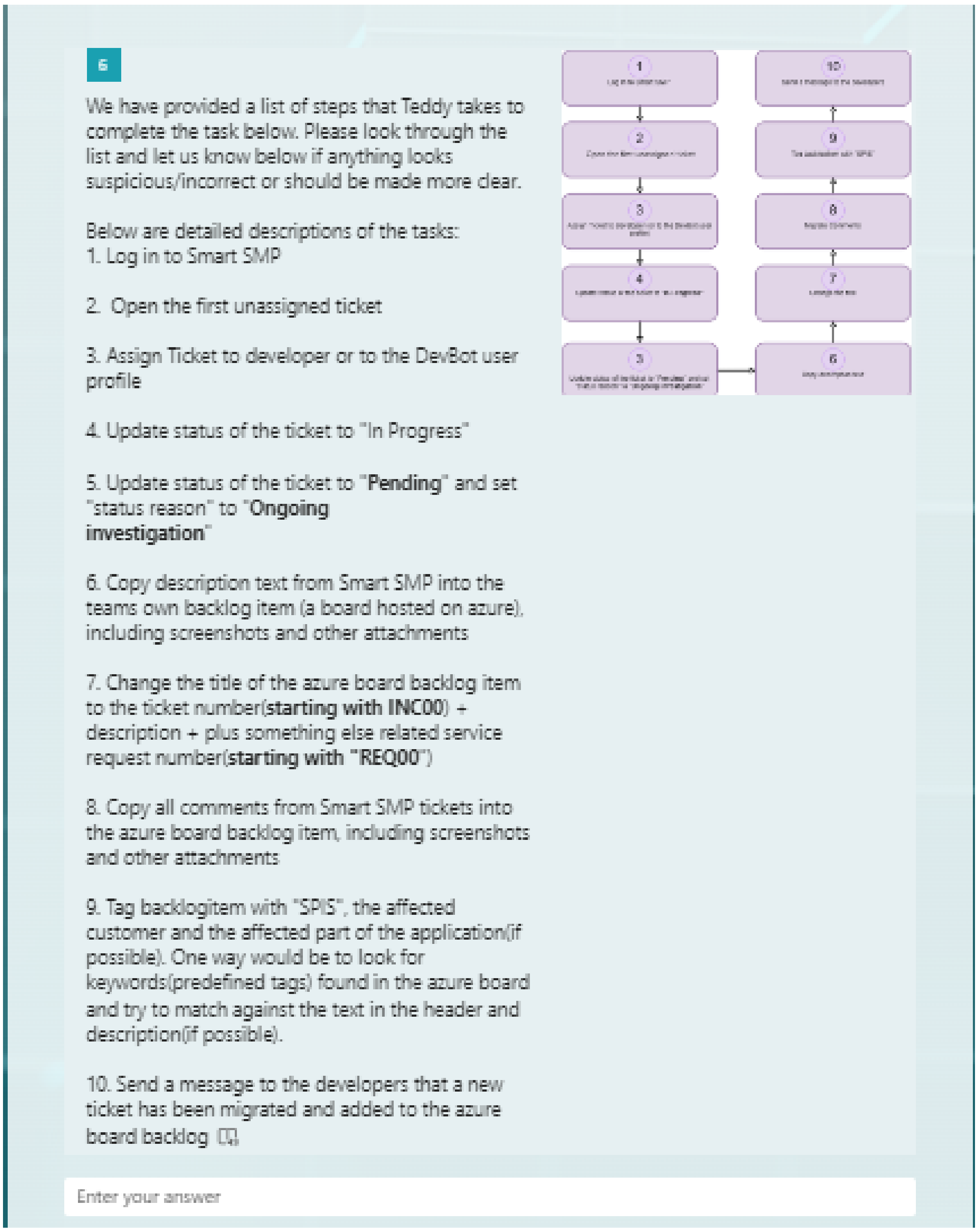
On a scale from 1 to 5 where 1 is "definitely not" and 5 is "definitely", how much do you think that this DevBout would **interrupt** your daily workflow? *

1	2	3	4	5

5

On a scale from 1 to 5 where 1 is definitely disagree and 5 is definitely agree rate this statement:
I feel satisfied using this DevBot. *

1	2	3	4	5



6

We have provided a list of steps that Teddy takes to complete the task below. Please look through the list and let us know below if anything looks suspicious/incorrect or should be made more clear.

Below are detailed descriptions of the tasks:
1. Log in to Smart SMP

2. Open the first unassigned ticket

3. Assign Ticket to developer or to the DevBot user profile

4. Update status of the ticket to "In Progress"

5. Update status of the ticket to "**Pending**" and set "status reason" to "**Ongoing investigation**"

6. Copy description text from Smart SMP into the teams own backlog item (a board hosted on azure), including screenshots and other attachments

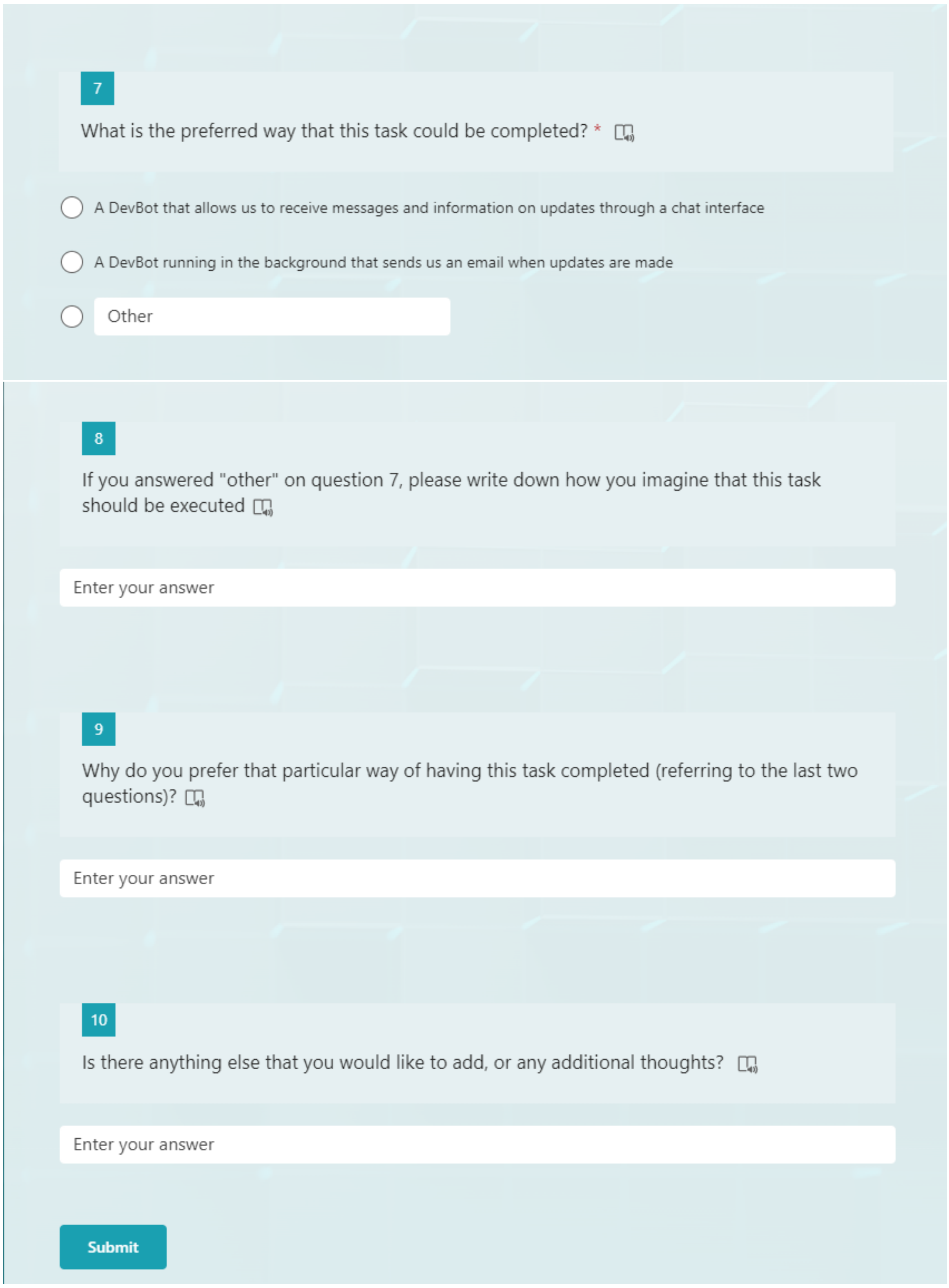
7. Change the title of the azure board backlog item to the ticket number(**starting with INC00**) + description + plus something else related service request number(**starting with "REQ00"**)

8. Copy all comments from Smart SMP tickets into the azure board backlog item, including screenshots and other attachments

9. Tag backlogitem with "SPIS", the affected customer and the affected part of the application(if possible). One way would be to look for keywords(predefined tags) found in the azure board and try to match against the text in the header and description(if possible).

10. Send a message to the developers that a new ticket has been migrated and added to the azure board backlog

Enter your answer



7

What is the preferred way that this task could be completed? *

◯ A DevBot that allows us to receive messages and information on updates through a chat interface

◯ A DevBot running in the background that sends us an email when updates are made

◯ Other

8

If you answered "other" on question 7, please write down how you imagine that this task should be executed

Enter your answer

9

Why do you prefer that particular way of having this task completed (referring to the last two questions)?

Enter your answer

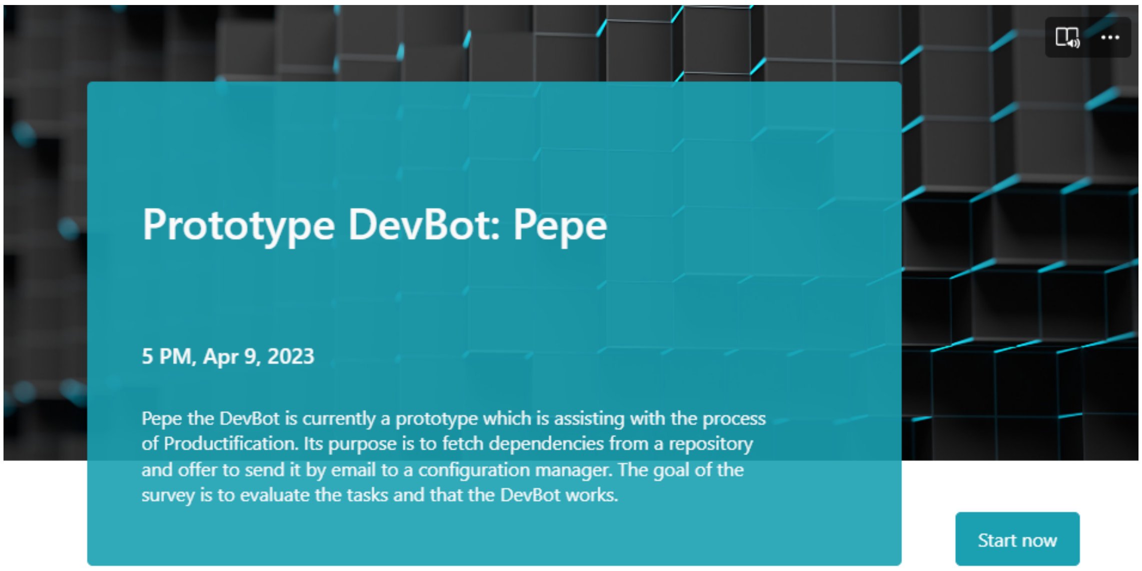
10

Is there anything else that you would like to add, or any additional thoughts?

Enter your answer

B

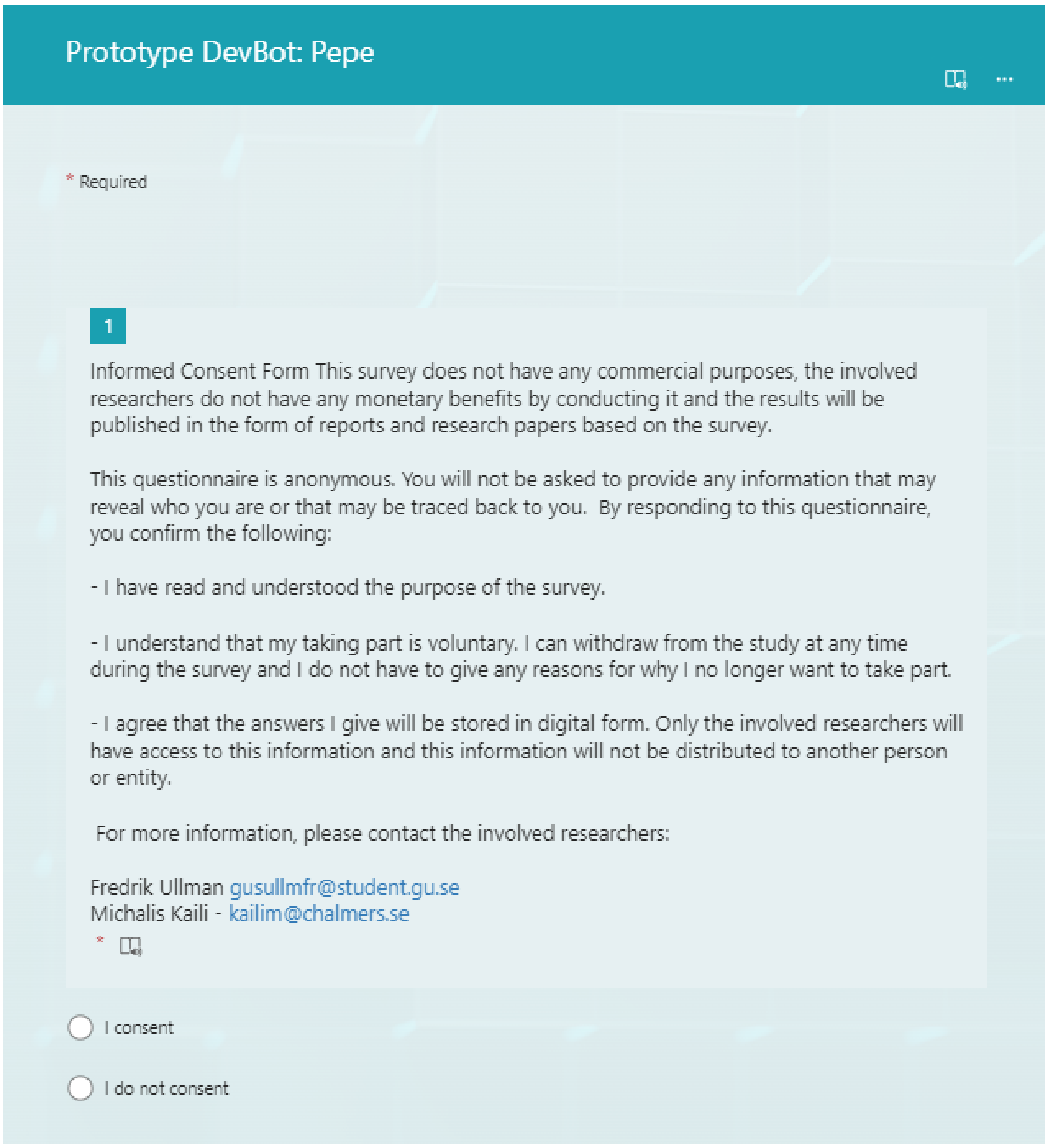
Appendix 2



Prototype DevBot: Pepe

5 PM, Apr 9, 2023

Pepe the DevBot is currently a prototype which is assisting with the process of Productification. Its purpose is to fetch dependencies from a repository and offer to send it by email to a configuration manager. The goal of the survey is to evaluate the tasks and that the DevBot works.



Prototype DevBot: Pepe

* Required

1

Informed Consent Form This survey does not have any commercial purposes, the involved researchers do not have any monetary benefits by conducting it and the results will be published in the form of reports and research papers based on the survey.

This questionnaire is anonymous. You will not be asked to provide any information that may reveal who you are or that may be traced back to you. By responding to this questionnaire, you confirm the following:

- I have read and understood the purpose of the survey.

- I understand that my taking part is voluntary. I can withdraw from the study at any time during the survey and I do not have to give any reasons for why I no longer want to take part.

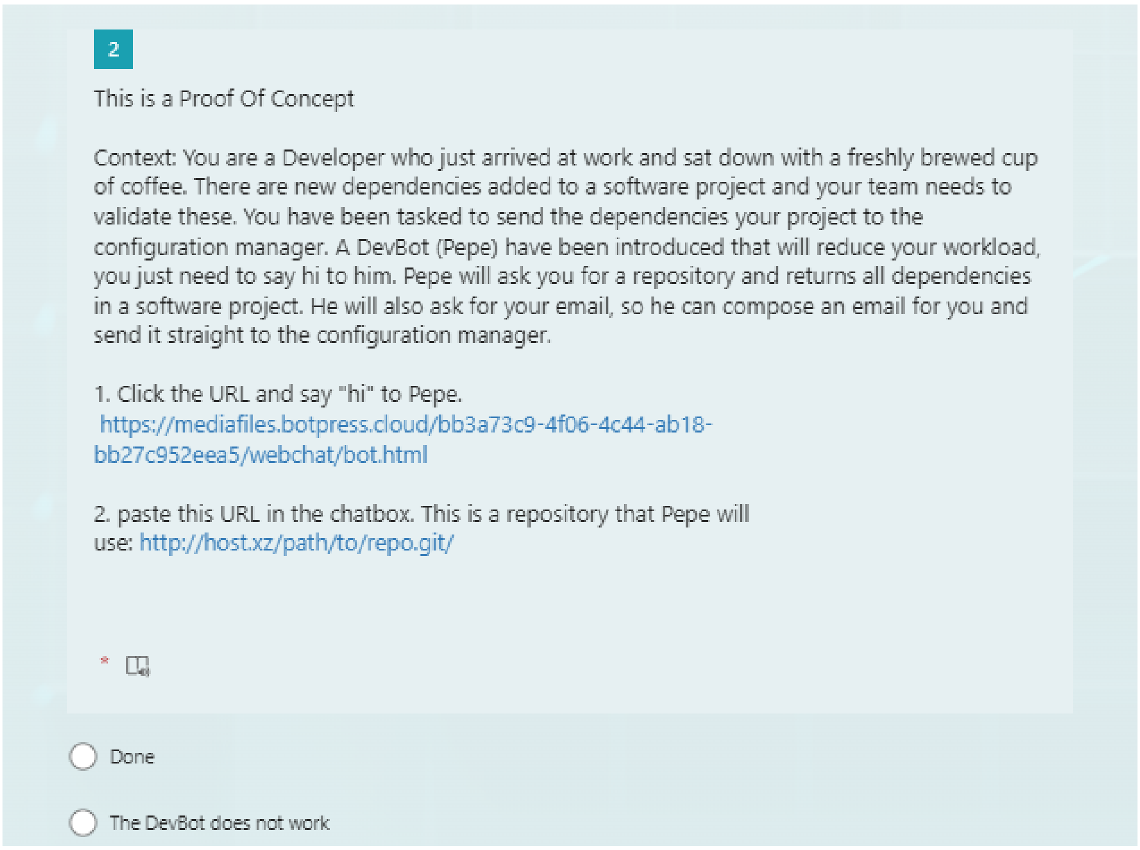
- I agree that the answers I give will be stored in digital form. Only the involved researchers will have access to this information and this information will not be distributed to another person or entity.

For more information, please contact the involved researchers:

Fredrik Ullman gusullmfr@student.gu.se
Michalis Kaili - kailim@chalmers.se
*

○ I consent

○ I do not consent



2

This is a Proof Of Concept

Context: You are a Developer who just arrived at work and sat down with a freshly brewed cup of coffee. There are new dependencies added to a software project and your team needs to validate these. You have been tasked to send the dependencies your project to the configuration manager. A DevBot (Pepe) have been introduced that will reduce your workload, you just need to say hi to him. Pepe will ask you for a repository and returns all dependencies in a software project. He will also ask for your email, so he can compose an email for you and send it straight to the configuration manager.

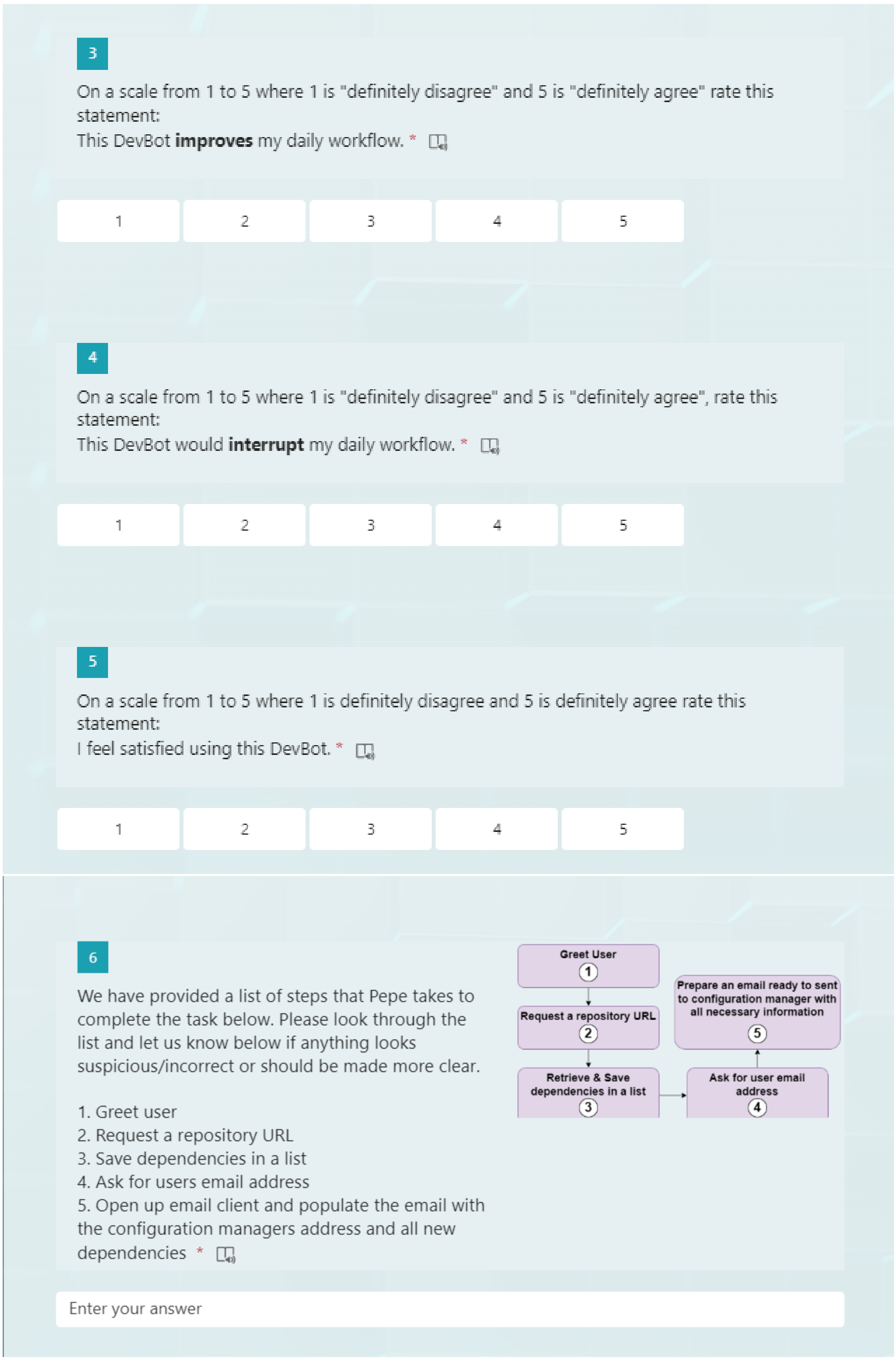
1. Click the URL and say "hi" to Pepe.
https://mediafiles.botpress.cloud/bb3a73c9-4f06-4c44-ab18-bb27c952eea5/webchat/bot.html

2. paste this URL in the chatbox. This is a repository that Pepe will use: http://host.xz/path/to/repo.git/

*

○ Done

○ The DevBot does not work



3

On a scale from 1 to 5 where 1 is "definitely disagree" and 5 is "definitely agree" rate this statement:
This DevBot **improves** my daily workflow. *

1	2	3	4	5

4

On a scale from 1 to 5 where 1 is "definitely disagree" and 5 is "definitely agree", rate this statement:
This DevBot would **interrupt** my daily workflow. *

1	2	3	4	5

5

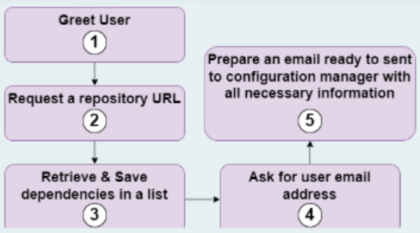
On a scale from 1 to 5 where 1 is definitely disagree and 5 is definitely agree rate this statement:
I feel satisfied using this DevBot. *

1	2	3	4	5

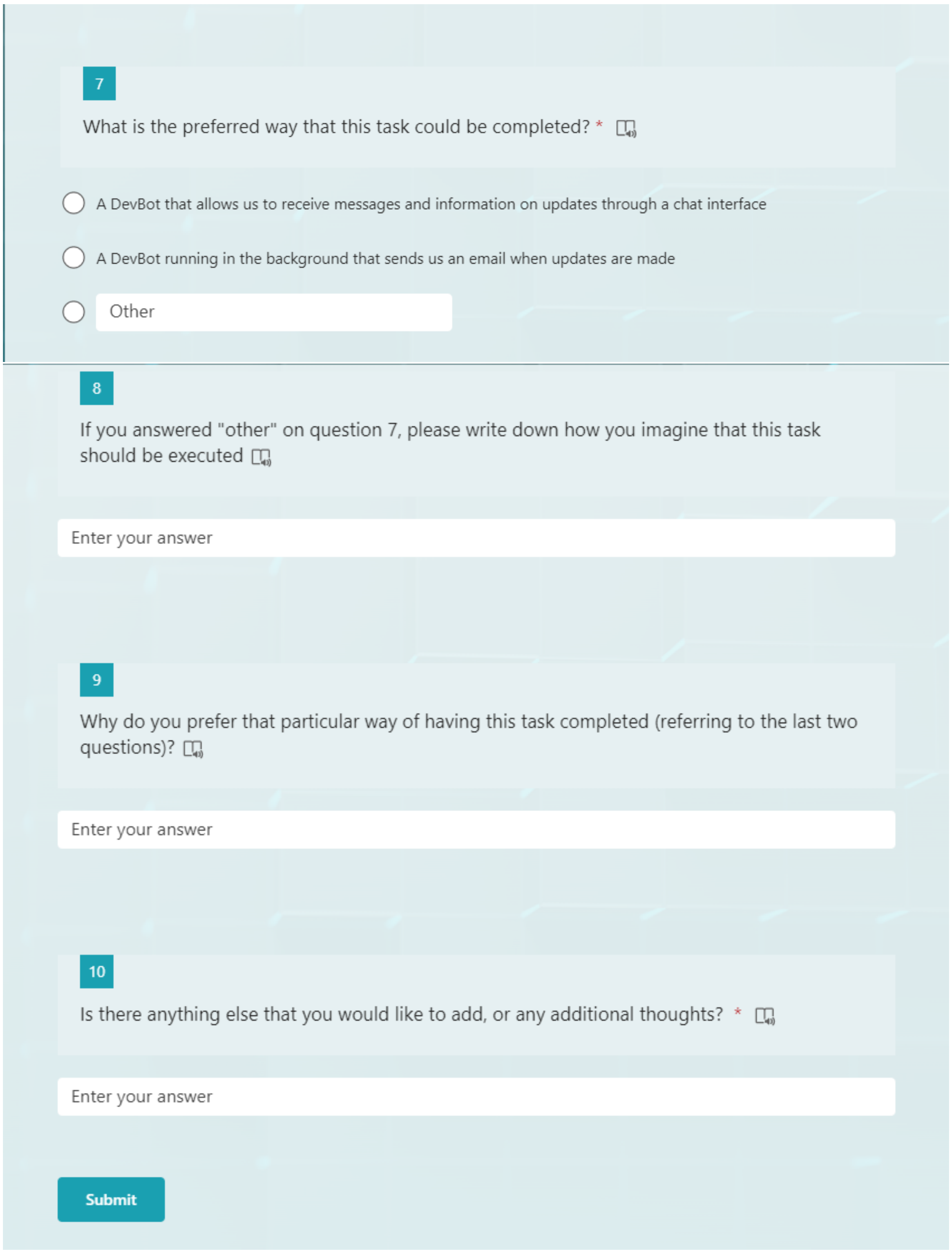
6

We have provided a list of steps that Pepe takes to complete the task below. Please look through the list and let us know below if anything looks suspicious/incorrect or should be made more clear.

1. Greet user
2. Request a repository URL
3. Save dependencies in a list
4. Ask for users email address
5. Open up email client and populate the email with the configuration managers address and all new dependencies *


Greet User
1
Request a repository URL
2
Retrieve & Save dependencies in a list
3
Ask for user email address
4
Prepare an email ready to sent to configuration manager with all necessary information
5

Enter your answer



7

What is the preferred way that this task could be completed? *

- A DevBot that allows us to receive messages and information on updates through a chat interface
- A DevBot running in the background that sends us an email when updates are made
- Other

8

If you answered "other" on question 7, please write down how you imagine that this task should be executed

Enter your answer

9

Why do you prefer that particular way of having this task completed (referring to the last two questions)?

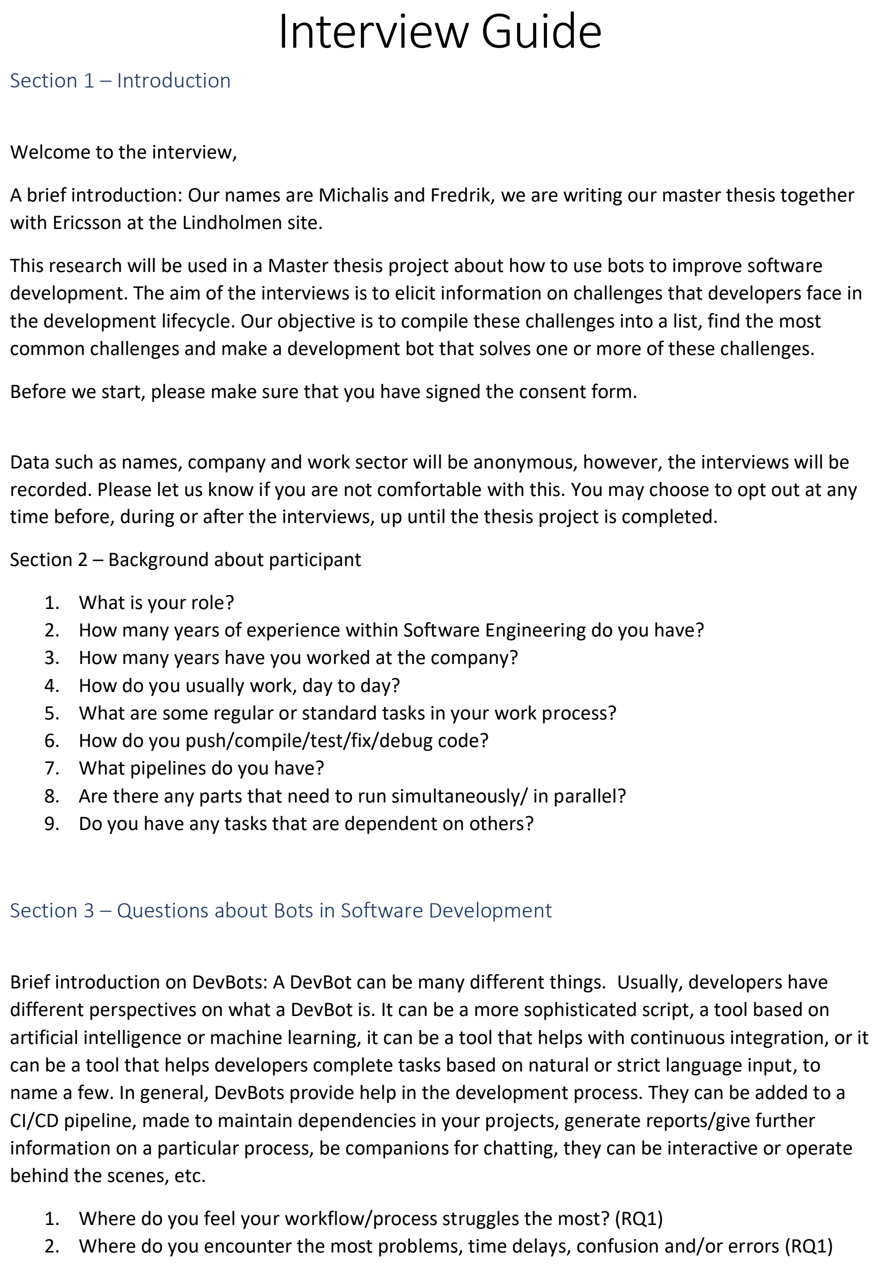
Enter your answer

10

Is there anything else that you would like to add, or any additional thoughts? *

Enter your answer

C
Appendix 3



Interview Guide

Section 1 – Introduction

Welcome to the interview,

A brief introduction: Our names are Michalis and Fredrik, we are writing our master thesis together with Ericsson at the Lindholmen site.

This research will be used in a Master thesis project about how to use bots to improve software development. The aim of the interviews is to elicit information on challenges that developers face in the development lifecycle. Our objective is to compile these challenges into a list, find the most common challenges and make a development bot that solves one or more of these challenges.

Before we start, please make sure that you have signed the consent form.

Data such as names, company and work sector will be anonymous, however, the interviews will be recorded. Please let us know if you are not comfortable with this. You may choose to opt out at any time before, during or after the interviews, up until the thesis project is completed.

Section 2 – Background about participant

1. What is your role?
2. How many years of experience within Software Engineering do you have?
3. How many years have you worked at the company?
4. How do you usually work, day to day?
5. What are some regular or standard tasks in your work process?
6. How do you push/compile/test/fix/debug code?
7. What pipelines do you have?
8. Are there any parts that need to run simultaneously/ in parallel?
9. Do you have any tasks that are dependent on others?

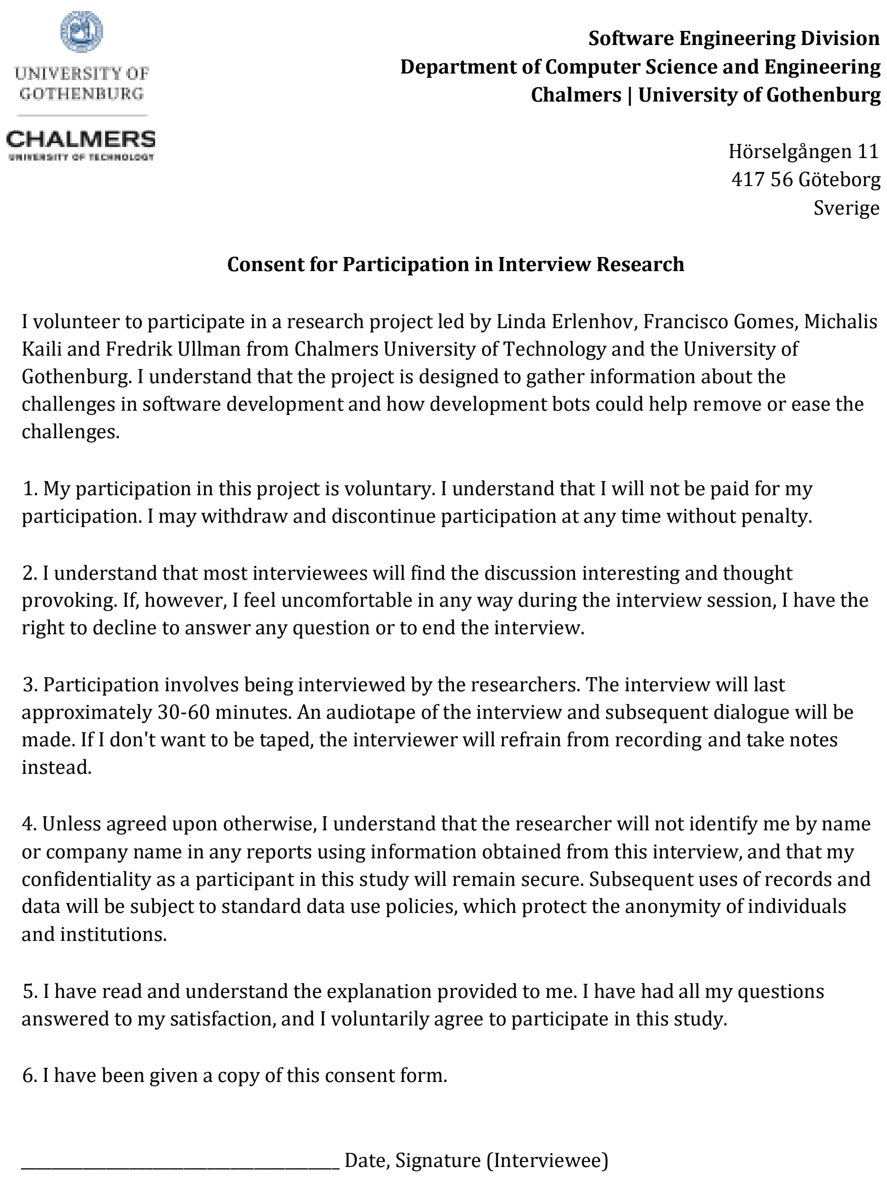
Section 3 – Questions about Bots in Software Development

Brief introduction on DevBots: A DevBot can be many different things. Usually, developers have different perspectives on what a DevBot is. It can be a more sophisticated script, a tool based on artificial intelligence or machine learning, it can be a tool that helps with continuous integration, or it can be a tool that helps developers complete tasks based on natural or strict language input, to name a few. In general, DevBots provide help in the development process. They can be added to a CI/CD pipeline, made to maintain dependencies in your projects, generate reports/give further information on a particular process, be companions for chatting, they can be interactive or operate behind the scenes, etc.

1. Where do you feel your workflow/process struggles the most? (RQ1)
2. Where do you encounter the most problems, time delays, confusion and/or errors (RQ1)

Figure C.1: Questions for requirements elicitation

D
Appendix 4



Software Engineering Division
Department of Computer Science and Engineering
Chalmers | University of Gothenburg

Hörselgången 11
417 56 Göteborg
Sverige

Consent for Participation in Interview Research

I volunteer to participate in a research project led by Linda Erlenhov, Francisco Gomes, Michalis Kaili and Fredrik Ullman from Chalmers University of Technology and the University of Gothenburg. I understand that the project is designed to gather information about the challenges in software development and how development bots could help remove or ease the challenges.

1. My participation in this project is voluntary. I understand that I will not be paid for my participation. I may withdraw and discontinue participation at any time without penalty.

2. I understand that most interviewees will find the discussion interesting and thought provoking. If, however, I feel uncomfortable in any way during the interview session, I have the right to decline to answer any question or to end the interview.

3. Participation involves being interviewed by the researchers. The interview will last approximately 30-60 minutes. An audiotape of the interview and subsequent dialogue will be made. If I don't want to be taped, the interviewer will refrain from recording and take notes instead.

4. Unless agreed upon otherwise, I understand that the researcher will not identify me by name or company name in any reports using information obtained from this interview, and that my confidentiality as a participant in this study will remain secure. Subsequent uses of records and data will be subject to standard data use policies, which protect the anonymity of individuals and institutions.

5. I have read and understand the explanation provided to me. I have had all my questions answered to my satisfaction, and I voluntarily agree to participate in this study.

6. I have been given a copy of this consent form.

______________________________________ Date, Signature (Interviewee)

Figure D.1: Interview consent form

E
Appendix 5

Theme	Theme	Code	Example
3PP productification is tedious and time consuming	Delays from dependency approval	Awaiting results and replies	"And the approval process there is rather long actually, just getting your 3pp into the system"
		Having to compile documents manually	"And then you have to take that list and put it into, document and put that into a different document that you upload to a website"
Thoughts on Testing		Time Consuming	"I mean, testing right now would be most difficult part. I mean, it's, it's not difficult, it's just time consuming"
		Difficulties with testing	"we cannot cannot run the code because it happens out at the customer site for instance"
Docklin	Annoying/Tedious/ Dont Like it	More complicated than previous solution	"our team members spent a lot of time for it and we have to spend time to understand how it works and why do we need it." "I don't like it. Okay. Because, uh, it's makes stuff more complicated and it's not a lot of benefits for me"
		Miscellaneous	"There are different problems when you grow to the size of ericsson"
		TomCat Server, Remote Operational Check	"it takes a lot some time from developers, uh, to have, we have a rotating like monitoring, uh, that one person is responsible ish, and then other people will check as well"
Communication	Context switching	Hard to find	"If you have 10s of thousands of acronyms in a In a PowerPoint and then you're supposed to relate that to some work you're doing."
		Finding the people you need	"you over communicate and then you have a situation where like we know what we're supposed to do, but it just takes a long time and now were confused" we're confused.
		Time Wasting	
		Agreeing what needs to be done and how	
		Difficulties Communicating and Understanding other teams	"each team creates their own documentation"
Documentation	Context Switching	Hard to Find	"It's hard to find an authoritative source about How are things? How do you do this?"
		Missing, Outdated, Incorrect	"And whatever documentation exists is sometimes outdated and sometimes incorrect"
		Why it is needed	

Table E.1: Thematically organised challenges

F

Appendix 6

Prototype DevBot: Teddy

2	16:34	Active
Responses	Average time to complete	Status

1. Informed Consent Form This survey does not have any commercial purposes, the involved researchers do not have any monetary benefits by conducting it and the results will be published in the form of reports and research papers based on the survey.

   This questionnaire is anonymous. You will not be asked to provide any information that may reveal who you are or that may be traced back to you. By responding to this questionnaire, you confirm the following:

   - I have read and understood the purpose of the survey.

   - I understand that my taking part is voluntary. I can withdraw from the study at any time during the survey and I do not have to give any reasons for why I no longer want to take part.

   - I agree that the answers I give will be stored in digital form. Only the involved researchers will have access to this information and this information will not be distributed to another person or entity.

   For more information, please contact the involved researchers:

   Fredrik Ullman gusullmfr@student.gu.se
   Michalis Kaili - kailim@chalmers.se

●	I consent	2
●	I do not consent	0

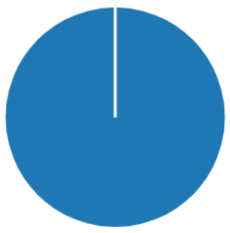
2. This is a Proof Of Concept

Context: You are a Developer who just arrived at work and sat down with a freshly brewed cup of coffee. It is your task to migrate data from Smart SMP into your teams Azure board, however, a DevBot has just been introduced to your ecosystem which means no more manual migrations, you only need to greet Teddy for all migrations to take place, so the first thing you do is to say hello to Teddy.

Teddy the DevBot should, after you greet him, check for updates in Smart SMP, and if there are new updates, Teddy will migrate them to your Azure board as a ticket and display the ticket in the chatbox. You might have to allow Teddy as a pop-up (it might be blocked by Chrome, Edge, etc)

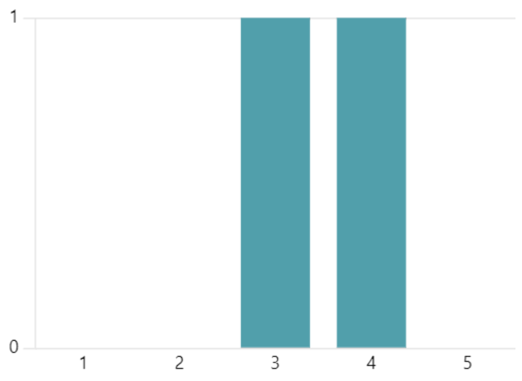
- Click the URL and say "hi" to Teddy: https://mediafiles.botpress.cloud/b8d29761-d96a-415f-9da0-13e6a500274d/webchat/bot.html

●	Done	2
●	The DevBot does not work	0



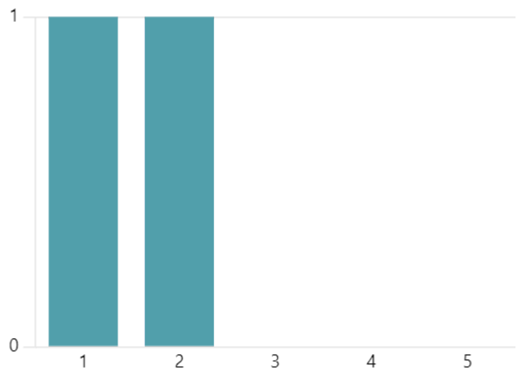
3. On a scale from 1 to 5 where 1 is "definitely not" and 5 is "definitely", how much do you think that this DevBout would **improve** your daily workflow?

3.50
Average Rating



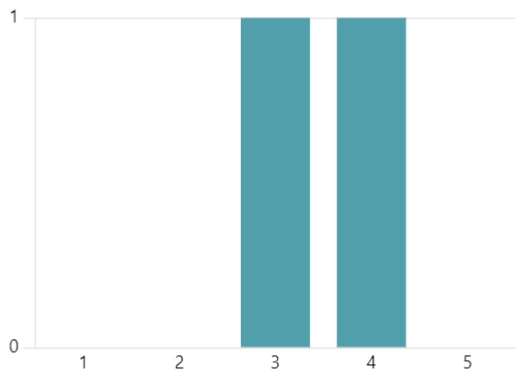
4. On a scale from 1 to 5 where 1 is "definitely not" and 5 is "definitely", how much do you think that this DevBout would **interrupt** your daily workflow?

1.50
Average Rating



5. On a scale from 1 to 5 where 1 is definitely disagree and 5 is definitely agree rate this statement:
   I feel satisfied using this DevBot.

3.50
Average Rating



6. We have provided a list of steps that Teddy takes to complete the task below. Please look through the list and let us know below if anything looks suspicious/incorrect or should be made more clear.

   Below are detailed descriptions of the tasks:
   1. Log in to Smart SMP

   2. Open the first unassigned ticket

   3. Assign Ticket to developer or to the DevBot user profile

   4. Update status of the ticket to "In Progress"

   5. Update status of the ticket to "**Pending**" and set "status reason" to "**Ongoing investigation**"

   6. Copy description text from Smart SMP into the teams own backlog item (a board hosted on azure), including screenshots and other attachments

   7. Change the title of the azure board backlog item to the ticket number(**starting with INC00**) + description + plus something else related service request number(**starting with "REQ00**")

   8. Copy all comments from Smart SMP tickets into the azure board backlog item, including screenshots and other attachments

   9. Tag backlogitem with "SPIS", the affected customer and the affected part of the application(if possible). One way would be to look for keywords(predefined tags) found in the azure board and try to match against the text in the header and description(if possible).

   10. Send a message to the developers that a new ticket has been migrated and added to the azure board backlog

2
Responses

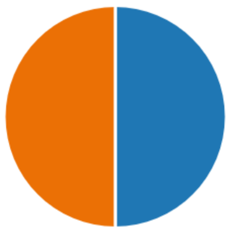
Latest Responses

"What I can see it covers all needed process steps"

"It looks alright, I think these are the correct steps"

7. What is the preferred way that this task could be completed?

●	A DevBot that allows us to recei...	1
●	A DevBot running in the backgr...	1
●	Other	0



8. If you answered "other" on question 7, please write down how you imagine that this task should be executed

0
Responses

Latest Responses

9. Why do you prefer that particular way of having this task completed (referring to the last two questions)?

2
Responses

Latest Responses

"I don't need to chat, I only need to see if there are any new ti...

"Teams feels like the better way of communicating with the e...

10. Is there anything else that you would like to add, or any additional thoughts?

2
Responses

Latest Responses

"No, not right now"

"Not at the moment"

Prototype DevBot: Pepe

4	142:42	Active
Responses	Average time to complete	Status

1. Informed Consent Form This survey does not have any commercial purposes, the involved researchers do not have any monetary benefits by conducting it and the results will be published in the form of reports and research papers based on the survey.

   This questionnaire is anonymous. You will not be asked to provide any information that may reveal who you are or that may be traced back to you. By responding to this questionnaire, you confirm the following:

   - I have read and understood the purpose of the survey.

   - I understand that my taking part is voluntary. I can withdraw from the study at any time during the survey and I do not have to give any reasons for why I no longer want to take part.

   - I agree that the answers I give will be stored in digital form. Only the involved researchers will have access to this information and this information will not be distributed to another person or entity.

   For more information, please contact the involved researchers:

   Fredrik Ullman gusullmfr@student.gu.se
   Michalis Kaili - kailim@chalmers.se

🔵	I consent	4
🟠	I do not consent	0

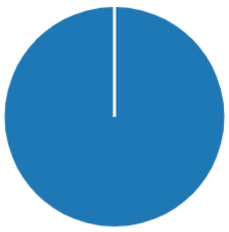
2. This is a Proof Of Concept

Context: You are a Developer who just arrived at work and sat down with a freshly brewed cup of coffee. There are new dependencies added to a software project and your team needs to validate these. You have been tasked to send the dependencies your project to the configuration manager. A DevBot (Pepe) have been introduced that will reduce your workload, you just need to say hi to him. Pepe will ask you for a repository and returns all dependencies in a software project. He will also ask for your email, so he can compose an email for you and send it straight to the configuration manager.

1. Click the URL and say "hi" to Pepe.
https://mediafiles.botpress.cloud/bb3a73c9-4f06-4c44-ab18-bb27c952eea5/webchat/bot.html

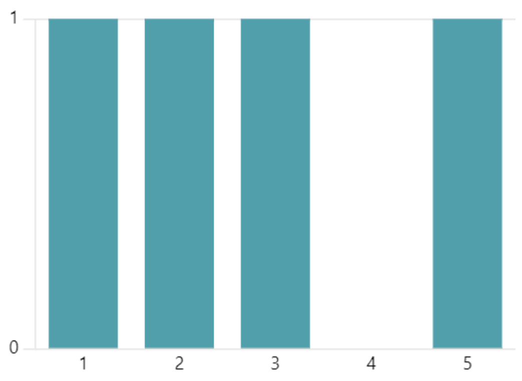
2. paste this URL in the chatbox. This is a repository that Pepe will use: http://host.xz/path/to/repo.git/

●	Done	4
●	The DevBot does not work	0



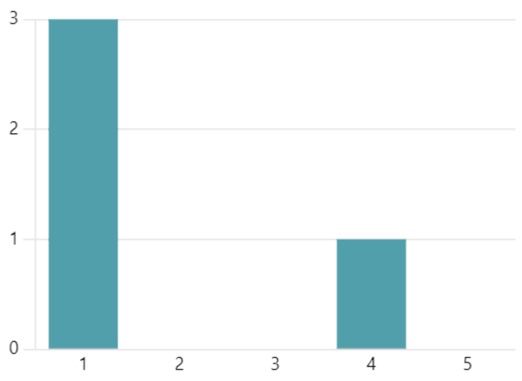
3. On a scale from 1 to 5 where 1 is "definitely disagree" and 5 is "definitely agree" rate this statement:
   This DevBot **improves** my daily workflow.

2.75
Average Rating



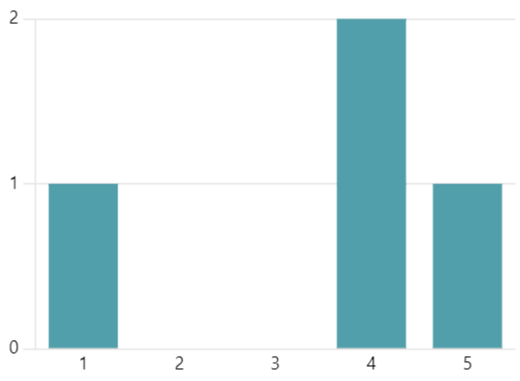
4. On a scale from 1 to 5 where 1 is "definitely disagree" and 5 is "definitely agree", rate this statement:
   This DevBot would **interrupt** my daily workflow.

1.75
Average Rating



5. On a scale from 1 to 5 where 1 is definitely disagree and 5 is definitely agree rate this statement:
   I feel satisfied using this DevBot.

3.50
Average Rating



6. We have provided a list of steps that Pepe takes to complete the task below. Please look through the list and let us know below if anything looks suspicious/incorrect or should be made more clear.

   1. Greet user
   2. Request a repository URL
   3. Save dependencies in a list
   4. Present the dependencies to the user
   5. Politely say goodbye and wait for future tasks

4
Responses

Latest Responses

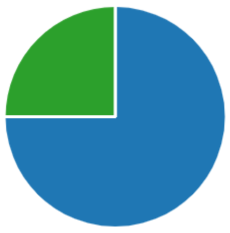
"*1.) invalid inputs causes the bot to run the rest of the comm...*

"*Would be nice if the bot could write a bit more about itself if...*

"*Looks correct*"

7. What is the preferred way that this task could be completed?

●	A DevBot that allows us to recei...	3
●	A DevBot running in the backgr...	0
●	Other	1



8. If you answered "other" on question 7, please write down how you imagine that this task should be executed

0
Responses

Latest Responses

9. Why do you prefer that particular way of having this task completed (referring to the last two questions)?

4
Responses

Latest Responses

"currently the bot adds a step to a process that is already sim...

"I thinks that chat interface is enough for this task."

"Batching the updates will bring less work for the configurati...

10. Is there anything else that you would like to add, or any additional thoughts?

3
Responses

Latest Responses

" "

"no"

"Looks good."