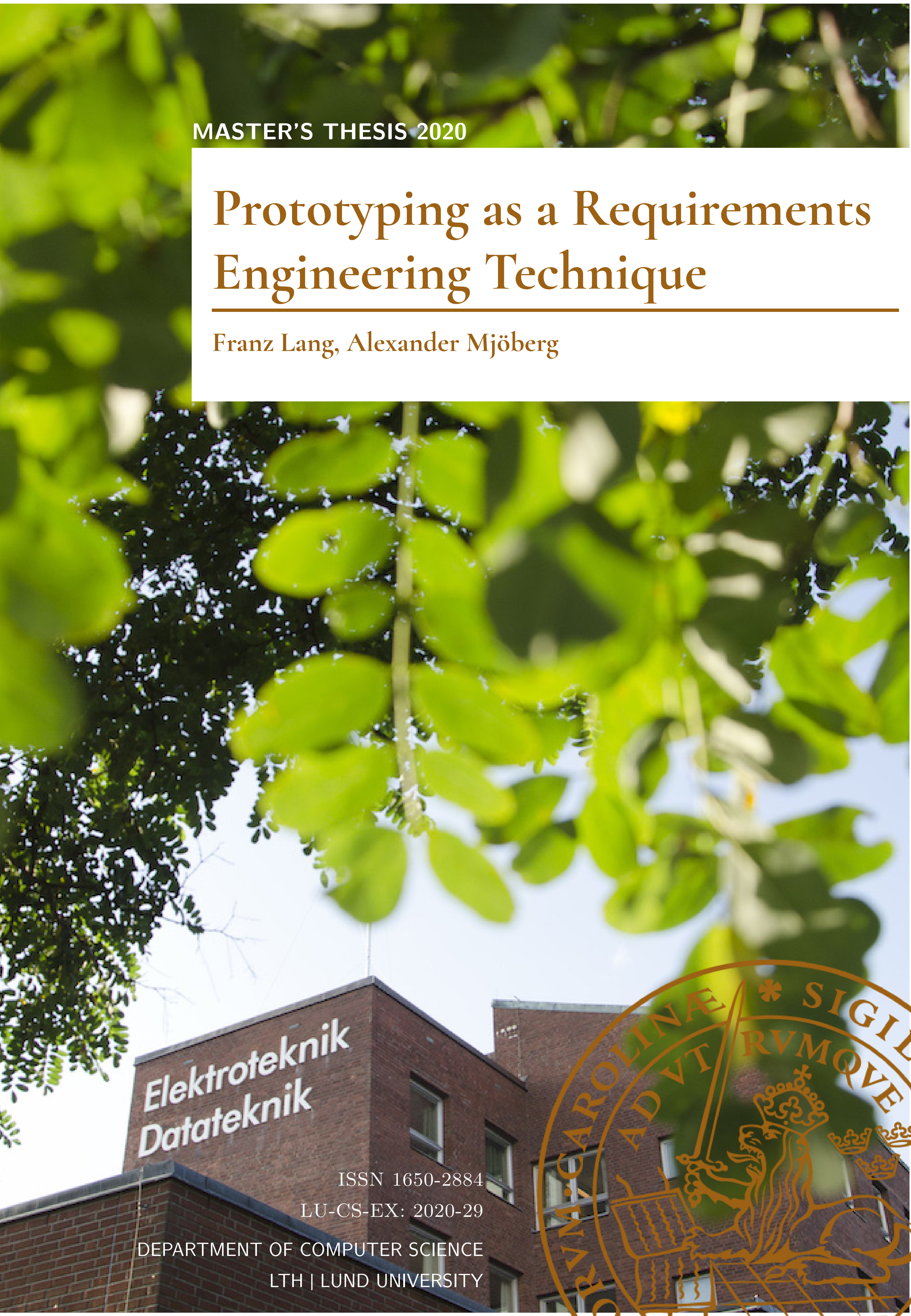
MASTER'S THESIS 2020

Prototyping as a Requirements Engineering Technique

Franz Lang, Alexander Mjöberg



ISSN 1650-2884
LU-CS-EX: 2020-29

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-29

Prototyping as a Requirements Engineering Technique

Franz Lang, Alexander Mjöberg

Prototyping as a Requirements Engineering Technique

Franz Lang
tfy13fla@student.lu.se

Alexander Mjöberg
dat15amj@student.lu.se

June 22, 2020

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Elizabeth Bjarnason, elizabeth.bjarnason@cs.lth.se
Maria Blomberg, maria.blomberg@telavox.com

Examiner: Björn Regnell, bjorn.regnell@cs.lth.se

Abstract

CONTEXT: Requirements Engineering is an important part of Software development. Software development has largely moved towards agile practices and the field of Requirements Engineering is no exception. However, within Agile Requirements Engineering much research remains to be done [1]. Prototyping has been identified by one study as a technique that can solve challenges surrounding Agile Requirements Engineering [2]. As the practical use of prototypes is not self-evident, prototyping methodology is useful to understand before being used in an Agile Requirements Engineering context. This begs the question, what is a meaningful model of prototyping and how can it be applied in an Agile Requirements Engineering context?

METHOD: We perform a case-study at Telavox, a company seeking to develop a new product with agile software development methodology. A literature study is conducted to gather knowledge on prototyping methodology and assemble a model. An exploratory case study with prototyping activities is conducted to better understand fidelity in the context of prototyping. Finally we attempt to provide guidelines for prototyping in an Agile development project and validate our approach using a focus group.

OUTCOME: We present the Prototyping Aspects Model which breaks down prototyping into four aspects: Purpose, Strategy, Scope and Review method. Findings in the exploratory case study indicate that altering these aspects has an effect on the perceived values of the prototype. We produce a set of guidelines that use prototyping as a means of coordinating the requirements engineering process in agile development projects.

CONTRIBUTION: The Prototyping Aspects Model can be used to conduct further research within the field of prototyping. It can also help companies understand the potential applications of prototyping. The agile prototyping guidelines can help researchers and business alike establish valuable use of prototyping within agile development contexts.

Keywords: MSc, prototyping, requirements engineering

Acknowledgements

We would like to thank our supervisor at Telavox, Maria Blomberg, for her unwavering support throughout this endeavour.

We would also like to thank Professor Elizabeth Bjarnason, without whom this thesis would be much less interesting and less pleasant to read. Apart from structure and grammar, Elizabeth has provided us with insight in the academics of requirements engineering, allowing us to explore an area of great interest to us.

Thanks also go to Telavox and its employees. Many of whom have assisted us with unrelenting feedback, energy, and advice over the past months.

Contents

Chapter 1

Introduction

Software prototyping is a consolidated term for the creation and evaluation of a prototype that depicts a simplified version of a software product [3]. Prototypes are used to explore certain aspects of a product throughout the product development process, such as testing the usability of a mobile application before release [4]. There is a large number of different types of prototypes as each type is suited for exploring certain aspects. For example, evaluating physical properties of a flood barrier may necessitate the use of physical materials in a physical environment while user interaction in a mobile application may only require digital means [5].

Within requirements engineering, prototyping is used as a technique for understanding what a product shall accomplish and whether a suggested solution is viable [4]. This begs the question: What are the fundamental differences among prototypes that are built for these purposes? Research suggests that the attributes of a prototype, such as its resemblance to the final product, may have an effect on the type of feedback one receives [3]. Studies have also found that the method of evaluation can have an impact on what type of requirements are perceived [6].

Prototyping is well situated to provide value in the modern software development process, particularly as this process has grown to become ever more agile [7]. The agile manifesto values *individuals and interactions* over *processes and tools* as well as *responding to change* over *following a plan* [8]. Prototypes have been proven to be valuable in improving communication [9] and are as per the definition simplified versions of a product that are therefore easily disposable. These properties make prototyping a valuable asset in agile requirements engineering.

We have set out to shed light onto the area of prototyping. Our goal is to understand what aspects there are to software prototyping, how these aspects relate to the use of prototyping in software projects, and how such projects can be supported in their requirements engineering process. To achieve the goal we conduct three different studies. First a literature study that maps the area of prototyping. Then an exploratory case study of a software development project wherein we construct prototype variants and study their perceived values. The last study is a focus group where individuals with experience in prototyping evaluate a set of guidelines that are to support them in their requirements engineering process.

The thesis is conducted in collaboration with Telavox, a supplier of cloud based private branch exchange solutions and collaborative tools. Their ambition is to expand their product offering with Unified Communication as a Service. A product that we use as a basis for our study. By allowing the use of this material the company receives insight on their use of prototyping and potential applications of such techniques in their future projects. The experienced individuals that assisted in the focus group are part of their user experience team.

1.1 Research Questions

This thesis covers three questions in the domain of prototyping and requirements engineering. The first question *What is a meaningful model for aspects of software prototyping?* synthesizes existing literature to understand the topic of prototyping. Material for this synthesize is collected with a literature study. The second question *What effect does the scope of a prototype have on the output and learning in software development?* studies whether altering the scope of a prototype has an effect on its perceived values. We explore this behaviour with an exploratory case study wherein advisors at Telavox are presented with three prototype variants. The third question *How can agile development projects be supported in requirements engineering processes concerning prototyping aspects?* explores the use of prototyping to support the requirements engineering process in software projects. The exploration and result are evaluated with a focus group.

RQ1 What is a meaningful model for aspects of software prototyping?
RQ2 What effect does the scope of a prototype have on the output and learning in software development?
RQ3 How can agile development projects be supported in requirements engineering processes concerning prototyping aspects?

1.2 Contribution

The thesis has been carried out by two authors; Franz Lang and Alexander Mjöberg. Both have been involved to some extent in each chapter. However, the larger components required a driver who prepared, executed, and finalised the chapters. The driver of a chapter was deemed to have contributed more to their area of responsibility and are therefore noted below.

Abstract - Alexander
Introduction - Franz
Method - Joint effort
Prototyping Aspects Model - Alexander

Guidelines - Alexander
Results - Joint effort
Discussion - Franz
Conclusion - Franz

1.3 Structure

This section aims to provide an understanding of how the vast number of pages in this thesis work in unison, from the introduction that sets the tone to the conclusion which presents the outcome.

There are eight major components to this thesis. The Introduction (1) introduces the domain of prototyping and what advancements are pursued with this thesis. Background (2) provides the reader with a basic level of knowledge surrounding the domain at hand. The third chapter, Research method (3), specifies the details, execution, and analysis of the literature study, case study, and focus group. The Prototyping Aspects Model (4) is a product of the aforementioned literature study and it depicts the various aspects of prototyping. The fifth chapter, Guidelines for Establishing Software Requirements with Prototypes (5), was evaluated with the focus group and it presents a series of steps with which one can apply prototyping in the requirements engineering process. The Results (6) presents the information from the literature study, case study, and focus group. The seventh chapter, Discussion (7), puts the results in context with related literature and explores what conclusions can be drawn. The last chapter, Conclusions (8), presents a consolidation concerning the advancements that were pursued with this thesis and suggestions on areas wherein one can pursue further advancements.

Chapter 2

Background

Requirements Engineering is an integral process of software engineering that ensures alignment between the needs and expectations of customers and users for a product. Prototyping assists in the alignment by enabling users and other stakeholders to explore, or elicit, requirements through a simplified version of the product to be built. Our thesis examines prototyping and its use in the context of product development at a case company.

2.1 Requirements Engineering

Requirements Engineering joined the modern era in the 1990s, as part of the IEEE standard and the International Requirements Engineering Conference. It provides a systematic approach to determine what a product should achieve. Eloquently put, it is the art of balancing **elicitation**, **specification**, **prioritisation**, and **validation** of requirements that through various techniques represent the scope of a product [4].

The four parts of requirements engineering can be likened to the steps that product development encompasses, from idea to product. Elicitation is the first stage and assists in establishing a vision of what the product shall accomplish by finding necessary functionality and determining valuable stakeholders. Specification provides a framework with which to properly specify functionality, ensuring testability and implementability. Prioritisation is the act of determining what functionality shall be built first and what can be delayed. Validation acts as a final step to confirm that the established material and models are both consistent and fulfill the needs of the stakeholders. Requirements that proceed through the stages of product development evolve and mature over time while some disappear altogether. The coordination of this process is known as Requirements Management.

Representation of requirements engineering in a product development process varies depending on the model that the process adheres to. Two common models are waterfall and agile development, of which the former is considered to be a more conservative approach where requirements are treated in linear fashion [10]. Agile development is an alternative to the waterfall model and provides an iterative approach wherein requirements are continuously evaluated as they change and mature [10]. The fusion of agile development with requirements engineering has seen an increase in attention from academics in recent years [11], especially in Asia, North America, and Europe.

2.2 Prototyping

Prototyping is the art of creating and evaluating a simple version of a product whose purpose it is to bring clarity on a specific topic. While being a popular technique for elicitation and validation in requirements engineering [4], prototypes are prevalent outside this terminology and are used in many industries by both designers and sales teams to support their cases. The challenge of creating a prototype remains the same nevermind the terminology; minimise effort and maximise benefit. To optimise the results one must understand the balance between practical use and evaluation.

Be it in the form of an interactive application or a drawing on a paper napkin. A prototype can support a wide range of activities such as communicating ideas, exploring form, and validating requirements. A prototype can be a small step towards the final product and still answer many questions without the cost of developing a full system. Prototypes are often used to involve stakeholders and guide the development effort.

2.2.1 Practical use

Requirements engineering suggests the use of prototyping as a technique for elicitation, testing, and validation of usability [4]. Depending on the phase for which a prototype is created, it will produce two different categories of requirements when used in an experiment; product-level requirements or design-level requirements. Product-level requirements are used to highlight the need of a required functionality while design-level requirements present the full solution of how a given functionality is resolved.

While creating a prototype one must balance numerous parameters to optimise its performance for a given phase. Houde et al. made an attempt to consolidate these parameters into categories that were then linked to a suitable purpose [3]. The resulting model suggested that all prototypes could fit into one of three categories; look and feel, implementation, and role. Role implied that a given prototype may be more suitable for investigating what it could do for a user while prototypes in implementation were more suitable for analysing technical feasibility.

Proof-of-concept is a type of prototype that is created with a focus on implementation and provides design-level requirements. Sigmund A. Tronvoll et al. studied an academic-industrial collaboration, whose goal it was to improve water tightness of flood barriers, from the perspective of prototyping [5]. They found that confining a real world problem into a prototype lead to a trade-off between six attributes; iteration time, iteration cost, approximation level, user level, result presentation, and experiment flexibility. Focusing on these factors provided better material for creating design-level requirements.

Tools lend a creator the ability to adjust the level of resemblance of a prototype to the real product. One such tool is *Sketch*, a digital tool that simplifies the creation of interactive prototypes by supplying a set of features that allow you to design views and link them with transitions. A transition can be initiated with the click of a button, mimicking the type of interaction produced from a website or mobile application. Prototypes that have been created with *Sketch* can be shared as hyperlinks with the use of *InVision*. While digital tools are popular in software development there are also benefits with using physical tools such as pen and paper. Such benefits include the ability to make quick adaptations and communicating ideas while not requiring a pixel perfect illustration.

2.2.2 Evaluation

Evaluation of a prototype differs based on the material with which it is created and the media in which it is consumed. While a car manufacturer may produce a car in clay that is placed in an air tunnel to analyse aerodynamics one could also create a digital model of the same car and expose it to virtual airflow. The approach may initially seem similar but it does have a significant effect on the type of feedback one receives.

Zahra et al. compared different evaluation methodologies and found that they have an impact on the type of feedback received. They found that Silent Paper Prototyping, a category of prototype where the illusion of functionality is achieved by having a human supply said functionality, is more efficient than No Paper Prototyping at capturing non-functional requirements [6]. They also found that Loud Paper Prototyping, a type of evaluation where participants are allowed to communicate, is superior to Silent Paper Prototyping at capturing and influencing functional requirements.

Software development uses the term *fidelity* to describe the resemblance of a prototype to the final product. Fidelity is commonly split into two categories, low and high. Low-fidelity scenarios address the layout and terminology of applications while high-fidelity prototypes address the issues of navigation and flow and of matching the design and user models of a system [12]. One interpretation of low-fidelity prototyping is *wireframing*, a technique that focuses on simple illustrations of layouts and placement of information. Research suggests that prototypes with a high degree of visual refinement increase credibility from potential customers when used in a sales pitch [13]. As a side effect, it found preliminary results suggesting that different variations of prototypes might lead to more requirements being elicited, but increased resistance to suggesting substantial change.

2.3 Case company description

Telavox offers cloudbased Private Branch Exchange (PBX) solutions to simplify the use and management of telephony for their customers. The company was founded in 2002 as a startup in telephony and communication with an ambition to create a product that provided a wow-experience and was easy to use. Their journey has been a successful one, expanding throughout Scandinavia and Europe, culminating in 250 employees and more than 250 000 active users in 2019.

2.3.1 Product

Realising the potential of new markets is what sets a surviving tech company apart from what was once their equals and brings forth the ability to shape the future of a trade. Software giants such as Microsoft or Cisco were early in discovering the necessity of communication and the time has come for Telavox to shape the future by creating a product for Unified Communication as a Service (UCaaS). Leading research and advisory company Gartner defines UCaaS as a cloud-delivered unified service that supports five communication functions; Enterprise Telephony, Unified messaging, Instant messaging and presence (personal and team), Mobility, and Communications-enabled business processes[14].

Telavox consider their solid foundation in telephony and modern approach to software development as a strong and unique advantage in their journey to create a product for UCaaS. The exact definition of what the product shall become is yet to be determined but bits and pieces are coming together by discussing expectations with customers and analysing competitors. While the company does not adhere to an official requirements engineering process, it consolidates and communicates functional requirements with prototypes and general statements. The study is undertaken in conjuncture to this endeavour and the prototypes that we are to create and experiment with will support Telavox in shaping their vision.

Product support is coordinated and provided by their advisors in Malmö. Advisors work in teams of five that are split according to the customer segments that Telavox focuses on; Small/home office (SOHO), Small/medium (SME), Enterprise, and Core. The responsibility of an advisor can be likened to that of a Key Account Manager (KAM) wherein they are assigned to a number of customers to whom they are the first point of contact. Thanks to this assignment the advisor and customer can build a mutually beneficial relationship that allows for a closer collaboration and optimal use of the product offering. The day of an advisor is primarily spent as a first-line operator, responding to emails and calls, and managing back-office tasks such as invoicing.

2.3.2 Product development

Innovation and improvement of their product is coordinated by their DevOps department at the office in Malmö. The 90 employees are split into roughly 15 teams that are responsible for different areas of their offering. These areas include, but are not limited to, app development for Android and iOS, user experience, and web development. The user experience team is cross functional and focuses on ensuring a coherent appearance and providing visual material.

Telavox adheres to an agile mindset, following scrum methodology. The creation of a new product is coordinated by a product owner who prioritises and assigns responsibility to appropriate teams. When the product has matured and become ready for development the appropriate teams receive assignments. Assignments are defined in broad terms and are picked up by team leaders. At this point the strategies differ slightly but the most common approach is that the team splits the stories into tasks. Tasks are picked up by developers and implemented together with appropriate tests. Thanks to the separation of environments, teams and their members can release their contributions continuously. Before a product is released it is validated with their advisors and customers to ensure that requirements are met. When the product has become a part of the product portfolio, the product proceeds into a maintenance phase where product ownership is transferred from the product owner to a team.

Chapter 3

Research Method

The research was conducted in three phases where each phase relates to one of the research questions, see Figure 3.1. Phase one consisted of a literature study that provided the material necessary to produce a model that depicts the aspects of software prototyping. During phase two we performed an exploratory case study where subjects worked with three prototypes. Each prototype was treated with specific attributes based on an aspect from the aforementioned model to measure the effect of said aspect on the output and learning. In phase three we created a set of guidelines that were to support agile development projects in their requirements engineering efforts. The guidelines were created by applying the prototyping aspects model on the requirements engineering process and validating the result with a focus group consisting of participants experienced in software prototyping and user experience.

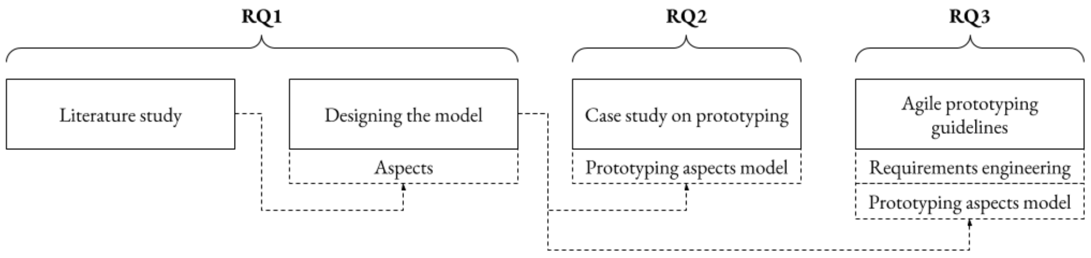

RQ1
RQ2
RQ3
Literature study
Designing the model
Aspects
Case study on prototyping
Prototyping aspects model
Agile prototyping guidelines
Requirements engineering
Prototyping aspects model

Figure 3.1: Overview of research method where the literature study feeds aspects into prototyping aspects model. The model is then used as a foundation for the exploratory case study. Finally the prototyping aspects model is applied with requirements engineering to create the Agile Prototyping Guidelines.

3.1 Literature study

A literature study was determined to be a suitable approach for exploring the existing knowledge base and collecting the information that is necessary to create a model that depicts the aspects of software prototyping. The type of literature study conducted was a systematic mapping study with thematic coding. A systematic mapping study provides categorization and a summary of research area in a structured approach [15]. Our study adhered to the four stages of a literature study as defined by the Swedish Agency for Health Technology Assessment and Assessment of Social Services (SBU) [16] as depicted in Figure 3.2. These four stages are Goals and boundaries, Validity and relevance, Literature search, and Synthesis and result.

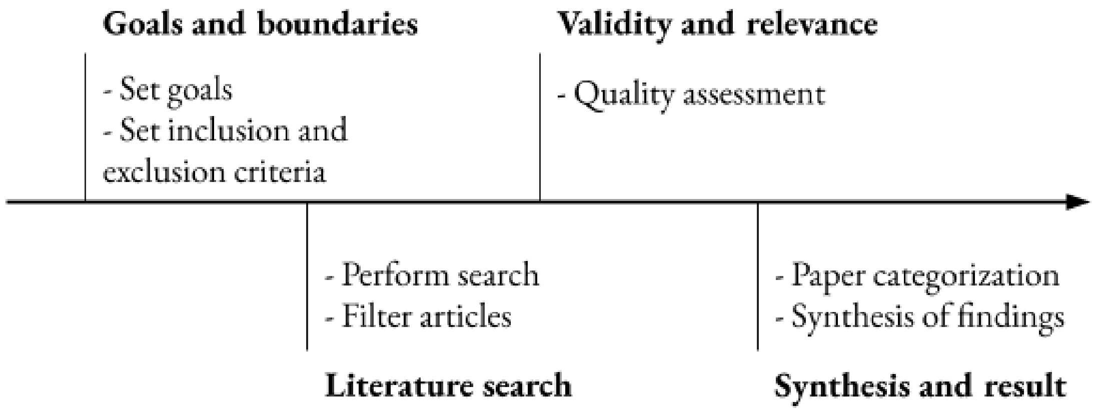


Figure 3.2: The process of performing a literature study.

3.1.1 Goals and boundaries

Establishing a premise for the literature study was an iterative process through which we determined a set of goals and boundaries. A process that had two primary parts, experimental searches and interviews with subject matter experts.

The experimental search highlighted important pieces of literature on the subject of prototyping which we subsequently consolidated into one page essays. Searching for prototype or prototyping yielded slightly different material of which both sides were determined to be relevant to our study. The consolidations showed that while articles may contain the relevant keywords they often used prototyping as a means to and end, not discussing the methodology nor its dimensions.

We conducted interviews with subject matter experts from Telavox and Lunds Faculty of Engineering (LTH). The interviews provided insight on the topic at hand but also showed the value of widening the scope, from focusing solely on software development to considering any field of expertise. While the increase in scope brought with it an abundance of empirical studies that provided little value on theory concerning prototyping, the change did substantially increase the amount of articles.

Include prototyping or prototype.
Exclude articles that do not discuss prototyping methodology or prototype dimensions.
Include all fields, not only software engineering.

The experimental searches and interviews shed light on the topic of prototyping but there was a distinct absence of material discussing the aspects of prototyping. Therefore we decided that our goal with the literature study would be to map the topic of prototyping and discern the most significant aspects in the form of a model. Boundaries were established in the form of abstract inclusion and exclusion criteria. The goal and boundaries are presented below.

Goal map the topic of prototyping.
Goal produce a model depicting significant aspects of prototyping.

3.1.2 Literature search

A literature as per SBU is to be conducted in three phases: test search, main search, and updated search [16]. Test search and main search were implemented in this literature study and the time spent on the literature search was split equally between the two. During the test search we experimented with a variety of search parameters and formulated a search strategy that was then executed in the main search.

The search results were exported and the data set was pruned during two iterations to reduce the number of articles that were irrelevant to our literature study. At the end of the main search the material was collected and stored in a shared area.

Test search

The test search required repeated tweaking and experimentation concerning what and where to perform the search. **What** was determined by the keywords and **where** was defined by the search engine on which to issue the search. Combining these two parameters yielded a search query that was optimised for our specific needs of goals and boundaries.

Where was limited by comparing the results of searching for prototype and prototyping. Results were judged on the amount of entries and the uniqueness of the resulting set of papers. Two search engines remained after this stage: Lund University Library (LUBSearch) and Association for Computing Machinery (ACM). The former was selected as it included results from other search engines such as IEEE Explore and ScienceDirect. The latter was chosen due to presenting a set of papers that suited the goals and boundaries extremely well.

What was specified with an exploratory and evaluating process that differed slightly between the two search engines. For LUBSearch, a small set of papers matching the inclusion and exclusion criteria were selected. Keywords, as defined by LUBSearch, were then extracted from each paper and listed in order of the rate of their occurrence. The most common keywords (Prototyping, Fidelity, Software Prototype, and Agile) were then used as a basis for the search query. ACM did not provide keywords for papers and therefore we determined it would be best to issue the search on the existence of prototype or prototyping in the title of papers. The final combination of search queries was as follows.

ACM [Publication Title: prototyping] OR [Publication Title: prototype]

LUB Prototyping AND (Fidelity OR Software Prototype OR Agile)

Main search

The search queries from the test search were performed on the respective search engines and the resulting material was combined and subjected to multiple iterations of pruning. LUBSearch provided an additional set of filters which were used to limit the results to being peer reviewed and in English before exporting and consolidating the material with ACM. This produced a total of 4671 search results of which 2713 originated from LUB while the remaining 1958 entries were from ACM.

Two iterations of pruning, Title review and Abstract review, were performed to produce the final literature selection as depicted in Figure 3.3. The Title review removed any material that did not appear to focus on prototyping methodology or the exploration of such. After the Abstract review only articles that focused on prototyping methodology remained. Studies that simply used prototyping in a case study were rejected.

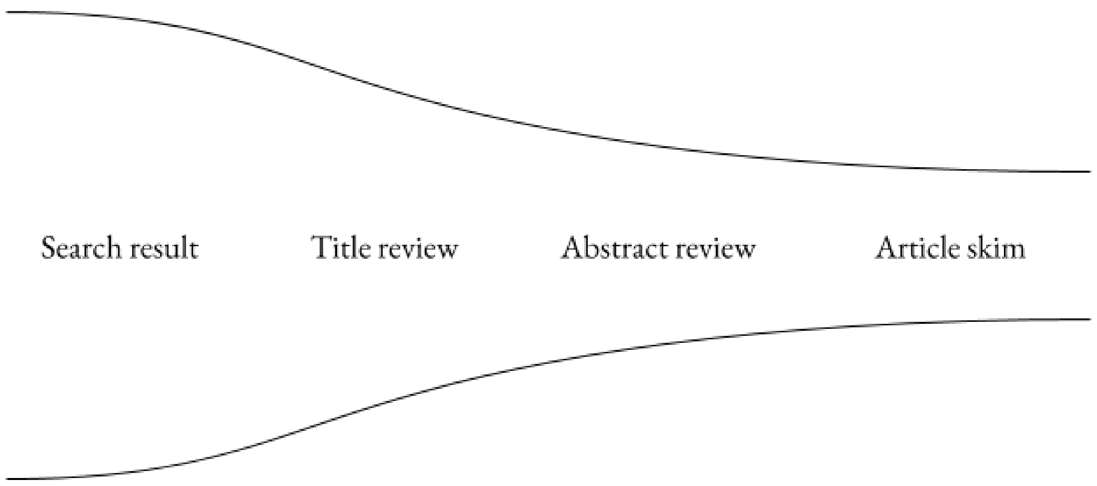


Figure 3.3: A funnel detailing the progress from obtaining search results to pruning with a title review, abstract review and article skim.

The abstract review could only be partially executed due to multiple articles missing an abstract and therefore we executed a final step of validation in the form of an article skim that further ensured that result suited our goals and boundaries. Articles were skimmed and checked against our inclusion and exclusion criteria. This produced a final result of 34 papers. The result for each stage is shown in Table 3.1.

Table 3.1: Results from each step of the literature search and its validation.

Search Engine	Search result	Title review	Abstract review	Article skim
LUB	2713	84	30	20
ACM	1958	62	24	14
Total	4671	146	54	34

Articles were indexed with a number from 1 to 34 where the first 14 were from ACM and the remainder from LUBSearch. The final article selection along with actual numbering can be found in Appendix A.

3.1.3 Validity and relevance

While a certain level of validity and relevance was achieved by filtering for peer reviewed literature and performing an article skim during the Main search, no formal quality assessment was conducted in the literature study. Kitchenham states that for a systematic mapping study a quality evaluation is not essential, as the goal of mapping study is to classify and aggregate research rather than investigate research [17].

3.1.4 Synthesis and result

Analysis of the articles consisted of a preliminary overview of the material followed by a systematic analysis of the articles by applying thematic coding. The preliminary overview covered the year in which the material was published and the country from which the article originated while the thematic coding provided a systematic approach for analysing the acquired material and produce a meaningful model for aspects of software prototyping.

Articles were categorised through thematic coding, a process that was split into three stages: read, identify categories and categorize. The first stage was to read each article in depth and write a short summary how it was relevant to our study. In the second stage of synthesizing, the categories were gradually established. By analyzing papers and using our summaries we established four categories; Purpose, Strategy, Scope, and Review method. In the final stage of synthesis all articles were assigned a strong link to one or more of these categories and a handful of articles were also attributed with weak links in the case that a consolidation had weak ties to one of the categories.

3.2 Designing the model

The Prototyping Aspects Model was created by building on the categories established in the literature study thematic coding. The model is presented Chapter 4. Each literature study category was made into an aspect and was developed by using the articles in the literature study. The explicit design choices that were made when establishing categories and developing the framework are explained in Model design choices (see Section 4.5).

3.3 Case study on prototyping

To investigate the effects that prototyping aspects in Chapter 4 have on output and learning in software projects from our second research question we conducted an exploratory case study. The case study was established in four phases as shown in figure 3.4. During the first phase we established the goal and what variables shall be treated to bring clarity to the topic. In the second phase we constructed a standardised process that implemented the aforementioned variables. The third phase consisted of executing the case study and collect the information. Phase number four consisted of analysing the gathered information in accordance with the goal.

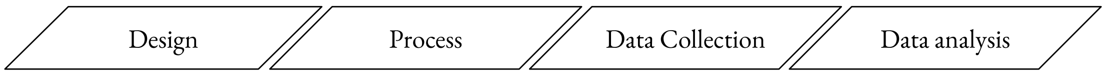


Figure 3.4: The phases of a case study.

Preceding the data collection was a pilot study that served two purposes. First and foremost, it validated the overall approach of the design and secondly it tested the material that was going to be used in the sessions. This ensured that instructions were clear and maintained a sufficient level of detail. The pilot study consisted of three session. Resulting surveys and recordings were not included in the main experiment.

3.3.1 Design

The case study consisted of sessions where advisors from Telavox explored three prototypes that were said to depict new functionality of their UCaaS product, a product which is described in section 2.3.1. The prototypes were treated with a specific set of prototyping aspects in mind. Advisors were only made aware of one prototype in order to be able to compare what the effect of an aspect had on the output and learning. Meetings were held in a virtual environment as physical meetings were not possible.

Chapter 4 depicts four aspects that are to be considered in prototyping. We determined that the aspect that was to be adjusted in the prototypes had to be easy to differentiate, easy to produce, and yet lead to different results. The aspect that was deemed to fulfill this criteria was *scope*. A study by Melin et al. validated our choice of aspect as they found that a change in visual or functional refinement may render a prototype more suitable for certain occasions [13]. The procedure for creating the prototypes is further elaborated on in Section 3.3.1.

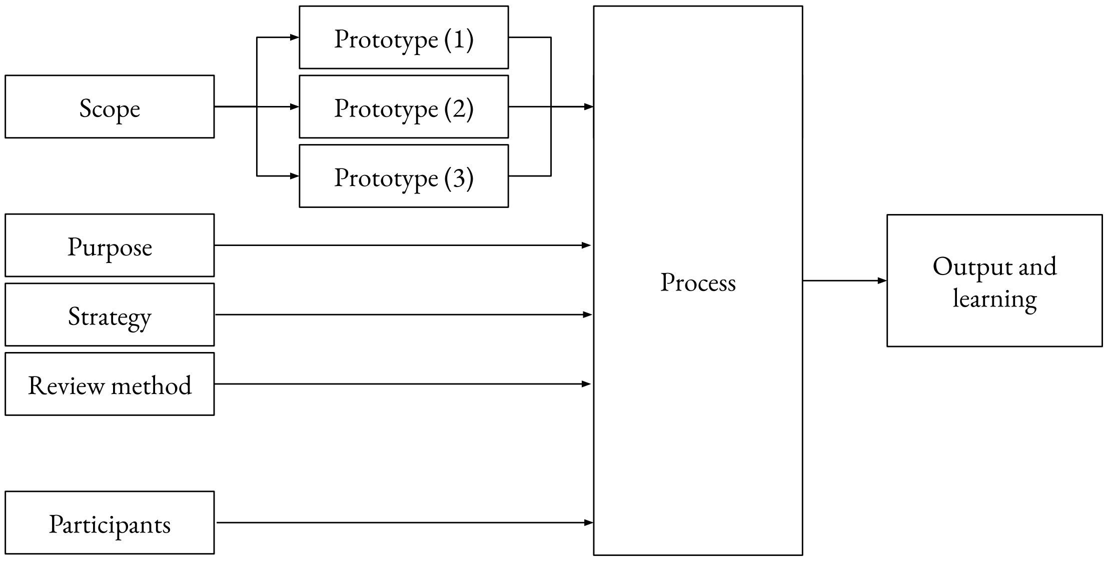


Figure 3.5: The four aspects of prototyping. Scope, purpose, strategy, review method, and participants are independent variables. All but the first are fixed. These are applied in a process to study the effect on the dependent variable output and learning.

Humans are rather unique and in order to gather correct conclusions it is important to minimise the effect of such differences. While Telavox has a wide variety of positions and employees we chose their advisors as subjects for this study. They have direct customer contact and make frequent use of their software. Subjects were asked to enter their experience in their field of expertise and the amount of years they had worked at Telavox to control for confounding variables.

Prototypes

Prototype (1), Prototype (2), and Prototype (3) were treated with a set of parameters shown in figure 4.2 and the definition of *scope* is further detailed in Section 4.3. These parameters were (1) high visual - high functional, (2) high visual - low functional, and (3) low visual - high functional as depicted in Figure 3.6. All prototypes originated from Sketch but required different adaptations to become suitable for virtual meetings. An overview of each prototype is shown in Appendices B and C.

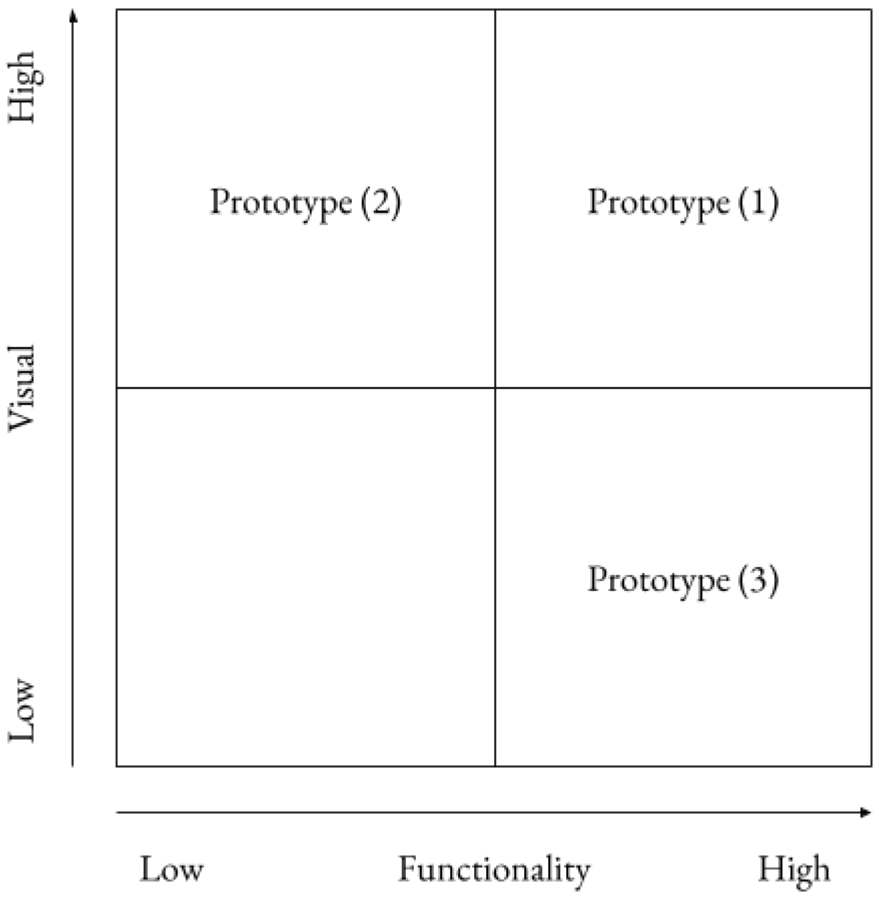


Figure 3.6: Visual and functional aspects of prototypes and the categories of the prototypes that were created.

Prototype (1) was built with the existing material from the pilot study and focused on providing a high degree of visual quality while enabling a high level of functionality for both features and interactability with the use of connectors in Sketch. While participants of the pilot study had Sketch installed and could use the raw file of the prototype, future participants would not have the software. Therefore we used InVision to create a temporary link that allowed users to interact with the prototype with their web browser.

Prototype (2) maintained a high degree of visual quality but required a low level of functionality. Prototype (1) was used as a base for Prototype (2) but instead of allowing users to interact with the prototype through a link, each view was exported as an image and added to a presentation in Google Slides. Users interacted with the prototype by receiving a link to the presentation and proceeding through the slides, one by one.

Prototype (3) was the opposite of Prototype (2). While functionality likened that of Prototype (1), the degree of visual refinement was at a much lower level. A level that can be likened to that of wireframes. Interaction with the prototype resembled that of Prototype (1) wherein users received a link and explored the prototype with their web browser.

3.3.2 Process

The sessions were standardised with the assistance of four documents. The first three documents are Opening Statement, Scenarios and the Survey, the last of which is presented Appendix D. Opening Statement was read aloud by the authors (us) to explain the structure of the session for the participant. Scenarios guided the user in using the Prototype while highlighting new functionality. The Survey consolidated information from the session.

The fourth and last piece of standardised material was a framework of how a meetings were to be conducted. It received slight adjustments after the pilot study. Most of the changes related to improving the participants ability to become familiar with the prototype. The final version of the framework can be seen below.

1. Authors schedule meeting.
2. Authors and Participant join the meeting.
3. Authors read Opening Statement.
4. Participant receives Prototype and Scenarios.
5. Participant initiates screen sharing.
6. Authors initiate screen and audio recording.
7. Participant performs the scenarios one by one while communicating verbally.
8. Upon the completion of the scenarios, the participant receives an additional three minutes to familiarise themselves with prototype as they deem necessary.
9. Participant fills out survey.
10. Participant is given the opportunity to ask complementary questions.
11. Authors close meeting.

3.3.3 Data collection

All 15 sessions were executed in the span of 11 days. Each session produced one complete survey answer and one recording that was stored and uploaded to a shared folder. These are first and second degree techniques of data collection and unison they provided the data upon which the analysis was built. The recordings were primarily used for a continuous validation and evaluation of the approach.

The routine by which a session was conducted adhered to the process depicted in Section 3.3.2. Sessions took about 30 minutes, evenly split between the process leading up to the survey and the following interview. Subjects were invited to a virtual meeting with three members, one being the subject themselves and the other two being the authors. The latter began the experiment by reading the Opening Statement. The subject received the Prototype and the Scenarios after which they shared their screen. Upon seeing the subjects screen, the authors initiated the screen recording and informed the subject that they could start. The subject traversed the scenarios, assisted by the authors who confirmed when a scenario had been fulfilled, which in turn allowed the subject to proceed to the next scenario. Once each scenario had been completed the subject received the Survey. After completing the survey the session was completed.

3.3.4 Data analysis

Subjects entered their feedback in a survey created with *Google Form* which fed information into a *Google Spreadsheet*, a tool that resembles *Excel* by Microsoft. The survey was split into four segments with different focus which laid the foundation for the qualitative analysis. Answers from the various segments were treated in different ways depending on the type of response that was obtained.

The first segment focused on gathering their insights from working with the prototype with a series of questions that allowed for text based answers. Responses were analysed by creating a thematic encoding based on the text from each answer. The answers were then connected to one or more of the resulting categories to analyse and compare their frequency between different prototypes.

In the second segment we measured the participant's agreement with a selection of statements by allowing a rating of one to five, more commonly known as a *Likert Scale* [18]. A value of one indicated disagreement with the statement while a five indicated complete agreement. Thanks to the responses being of numerical nature, the process of analysis was easier than that of the text based answers. Responses were analysed based on their average and median score and then compared between different prototypes.

Segment three was of different nature as it was only used to support the analysis of the first two segments. The primary cause for its creation was to assist Telavox in understanding the behaviour of their users. While the information did provide additional insight for each individuals response we conducted no analysis with the acquired information.

The last segment was created to validate the demographic of the case study. There were two questions in the fourth segment to assist in this process. One regarding the role of the subject and one surrounding their experience with the area at hand. 14 of the participants were advisors while one participant had recently been promoted to manager of advisors. An overview of the years of experience is presented in Figure 3.7.

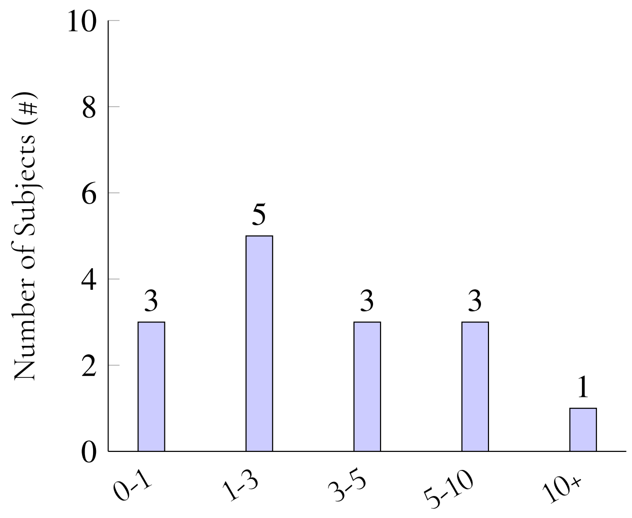


Figure 3.7: Number of subjects split by their years of experience.

3.4 Agile Prototyping Guidelines

We set out to provide recommendations for working with prototypes in agile software projects to answer our third research question (RQ3). The recommendations was in the form of guidelines that were created by applying our findings in the context of requirements engineering and validating the result with a focus group.

3.4.1 Guideline construction

Guidelines were created with the goal of demonstrating how the main activities of Requirements Engineering could be conducted using prototyping. Construction was done by taking activities from Requirements Engineering and prescribing suitable prototyping activities based on the knowledge in the Prototyping Aspects Model (see Chapter 4). Of the main activities that are common in Requirements Engineering (see Section 2.1 for our description), we proceeded with elicitation, specification, and validation. Each of these activities were then matched with relevant purposes from the purpose part of the prototyping aspects model. The matching was accomplished by clarifying the goal of the Requirements Engineering activity and then judging which prototyping purposes were suitable in fulfilling the goal. Several suitable purposes were identified for each Requirement Engineering activity. Each Requirement Engineering activity was then split into logical steps based on when it made sense to use several prototyping purposes in parallel. The aspects of Strategy, Scope and Review Method were then considered and selected for each of the steps. The step were then explained in text in a manner that made them accessible without having prior knowledge of our model nor requirements engineering. Each step was iteratively designed and improved. In the final iterations the results from the case study and focus group were also incorporated into the guidelines and a small section about incorporating these practices were then also added. The decision to make the guidelines accessible to any reader was based in the research question three goal of supporting software development projects in general and not just requirement engineers.

3.4.2 Focus group

A focus group was conducted to evaluate the guidelines. A focus group is a qualitative method that is essentially a group interview. Its power lies in establishing a peer-setting where participants are more likely to share experiences and perceptions, allowing deep insight into themes [19].

Participants

Five designers from the user experience team at the company participated in the focus group. The individuals had slightly different backgrounds and levels of experience in the area. The participants either had degrees in industrial economics or interaction design, three participants held master degrees and two held bachelor degrees. Apart from the individuals, both authors joined the meeting, bringing the total number of participants to seven. Both authors acted as moderators. One of the participating developers had some limited knowledge about the prototyping aspects model prior to the focus group. The others had no knowledge concerning the prototyping aspects model and were hence not biased by our own design.

Session

The focus group was conducted as a virtual meeting, due to external circumstances a physical meeting was not viable. The moderators opened the meeting and presented the prototyping guidelines (see Chapter 5). The group was then asked to review and discuss each step of the guidelines. Material was offered to participants in terms of potential talking points which can be found in Appendix E. The talking points consisted of questions relating to the guidelines for the group to answer. The moderators asked probing questions and suggested when it was a good time to move on in the discussion, thereby ensuring that the conversations could be focused and steered in good directions. To provide everyone with an equal opportunity to talk a sort of speaker list was established. As one talking point was concluded, the next person on the speaker list got to choose the next talking point and be the first to provide their thoughts on that talking point. The meeting was recorded.

Analysis

The recorded meeting was transcribed and its results are available in appendix F. The transcription was subjected to qualitative analysis to identify important points regarding the model. A thematic coding of the points was conducted to allow for an easier presentation of the results.

Chapter 4

Prototyping Aspects Model

This chapter presents a novel approach to prototyping based on four levels of aspects as depicted in Table 4.1. These aspects represent the underlying methodology of prototyping activities and supports the reader in understanding and making use of prototypes. The model provides guidance to traversing through the levels by elaborating on the choices within each aspect. Each level offers a number of decisions that may, or may not, impose restrictions on succeeding levels. This model was synthesised using the literature study (see Section 3.2) and its design choices are explained in Design of Framework (see Section 4.5).

Table 4.1: Depicts the four aspects and their hierarchy.

Prototyping aspect	Choices
Purpose *What is the reason for prototyping? Pick one or more.*	Exploration, Desirability testing Viability testing, Feasibility testing Communication, Usability testing Partial product, Validation Performance improvement
Strategy *How should we handle uncertainty? Pick one.*	Point-based design, Set-based solution array Performance set investigation, Flexible design
Scope *What level of refinement should be used? Decide for each dimension.*	Visual refinement dimension Functional refinement dimension
Review method *How should the prototype be evaluated? Decide for each dimension.*	User presentation Proximity to actual conditions Feedback collection

In the following sections we use terminology that is similar but different in order to stay true to the source material. *Designer* and *Developer* are used interchangeably here as both perform essentially the same prototyping activities with the distinction that developers might have additional responsibility for implementing the solution. Similarly *users* and *customers* are somewhat interchangeable with the difference a customer pays for the product, either by buying the product or financing the development project. A customer therefore has more influence than a typical user.

4.1 Purpose

The prototyping purpose is the reason why the prototype is needed. A large variety of reasons exist to construct a prototype and a single prototype can serve multiple purposes. Serving multiple purposes corresponds to selecting several purposes below to simultaneously be served. The purpose of a prototype is determined by external parameters such as the state and nature of a project. For example, given a project that is in an early stage where resources are plentiful one may wish to test new concepts using prototyping. At a later stage where resources are scarce, fine-tuning the current concept with prototyping might be emphasized. The following chapters present areas of prototyping that were found in the literature.

4.1.1 Exploration

While in the act of creating an exploratory prototype the designer can learn what the product may become and which design choices are essential when shaping the product. This type of prototype is commonly built for the designer themself and allows for experimentation with different emergent ideas. Building a prototype for oneself allows you to focus on ideas and possibilities as it removes the expectations that come with knowing there is an upcoming presentation of the prototype. An exploration prototype can be shared with others but this should not be the focus. In summary, exploration prototyping is essentially brainstorming with prototypes, allowing the designer to generate and organize ideas in a concrete manner.

Lim et al. suggested viewing prototypes as considered filters that can be used to traverse a design space [20]. It posits that in a design or development process a prototype can discover problems and explore new solution directions, hence being a generative process. When a designer tests ideas using a prototype, that individual discovers the important factors of the design, which leads to new design ideas and design spaces to explore.

Schneider expresses concern with exploratory prototyping [21]. He states in his paper that many prototypes turn into private toys of developers that are not properly shared with users. Suddenly only the developers learn from the prototyping activity while users are left out of the process. As an alternative approach he suggests that prototypes should be used by developers to prove, falsify, or demonstrate a point.

Dow et al. describe exploration as the process of exploring design spaces and their relative merits[22]. They say that exploration is necessary as otherwise people often interpret design problems narrowly and focus on refining those ideas. Furthermore it notes that exploration can act as a counterbalance to a focus on presenting and validating, activities that may otherwise lead to overinvesting in a single concept. It finds that in terms of exploration, there are several benefits to creating multiple designs.

Lichter et al. highlight the use of exploratory prototyping for requirement clarification [23]. The use of such prototyping lets developers learn about the utilisation area of the future system. In this paper the "exploratory" purpose is a combination of exploration, desirability testing and communication purposes.

4.1.2 Desirability testing

Desirability is the customer perspective on the product. Desirability testing can be done to investigate what the customer wants and a prototype can help support a conversation about desirability, gathering ideas and feedback from users. A perspective of *Role*, *Look and feel* and *implementation* can be used to further break down desirability into constituent parts. "*Role*" represents the need and functionality the product serves, *Look and feel* represents what the user experience should be, and *implementation* represents how the product solves the problem.

Zink et al. present Desirability as one out of three constituent parts in a design solution space model, the other two being viability and feasibility [24]. It states that desirability is the value to the customer and likeliness of purchase. Prototyping can be undertaken to test desirability of the future system.

Various papers briefly mention or allude to desirability prototyping. Some papers state that Desirability prototyping is done to support decision making by clarifying customer requirements and tasks [9] [25]. Lichter et al. Suggests a geographical difference in approaches to prototyping, stating that the European approach is primarily user-oriented while the North American approach focuses on getting a working product quickly [23].

Yasar discusses that when deciding what to prototype a perspective can be taken of "Role", "Look and feel" and "implementation" [26]. The corresponding questions to ask are *What role will the device/product being developed play in a users life?*, *How the device/product should look and feel?*, and *How should the device/product be implemented?*. This can help designers establish frames and purpose for their prototyping.

4.1.3 Viability testing

Viability is the business perspective on the product. Viability testing with prototypes can explore if the product is profitable and what costs are attached to it.

Zink et al. present Viability is another part of the three sided solution space model [24]. Viability is the ability of the design to fit within time and budget constraints. The paper finds that viability is highly complex and is affected by many factors, while viability as system characteristic is more independent of prototyping goals than other parts of the model. Ciriello et al. describe Viability testing as a practice to support decision making in projects [9]. The other articles in the selection did not discuss viability testing.

4.1.4 Feasibility testing

Feasibility is the technical perspective on the product. Feasibility testing with prototypes explores whether capabilities exist to support the product. If the envisioned product is potentially technically challenging or impossible to construct, feasibility testing allows the possibilities to be assessed. The product could also have requirements such as needing to operate at scale or fulfill security requirements, adding additional feasibility considerations.

Zink et al. present Describes feasibility as the final part of the three part solution space model [24]. It states that feasibility is a measurement of the technical functionality. The paper finds that feasibility testing had the highest priority in development process experiments and maintained a high share of invested time regardless of prototyping goals.

A common type of prototype to test feasibility is the *Breadboard prototype* [20] [23] [9] [25]. Such prototypes are built to investigate technical aspects and are normally not evaluated by users. This kind of prototype can support system specification and coding during the development project. In addition to the aforementioned kind, Lichter et al. also states that a *presentation prototype* can be built to demonstrate feasibility and help with project acquisition [23].

Tronvoll et al. state that prototypes built to investigate the suitability of a concept can be referred to as *proof-of-concept* prototypes [5]. These prototypes are built to resolve structural uncertainties and are usually the first prototypes constructed in projects.

Fern et al. mention feasibility testing as something used to evaluate novel technical approaches via exploratory programming [27]. The authors also state that rapid prototyping emphasises proof of concept and feasibility studies.

4.1.5 Communication

A prototype can communicate possible forms of a product. Designers can use it to make their vision of the product tangible, allowing them to show instead of tell. This can then be used for discussion, evaluation, and agreement surrounding the product. Prototypes can be beneficial to communication with stakeholders but it can also be a means of aligning within a design team.

Lichter et al. states that a *presentation prototype* can be used to support the initiation of a software project [23]. It shows that a future application is feasible and in line with customer expectations. The authors find that while the *presentation prototype* can be built in a "quick and dirty" sense to help with acquisition of projects, there is a risk that customers misunderstand the meaning of a prototype and enforce that the early version evolves into the final system.

Ciriello et al. claims that a key problem in software innovating firms is communicating ideas purposefully to different audiences and that skilled use of prototypes is required to persuade and collaborate with stakeholders [9]. They propose a set of practices to combine prototyping with storytelling to improve communication, increase customer involvement, and provide better customer satisfaction. They note that prototypes can clarify problems early and provide basis for discussion and development. For instance, they conclude that prototypes are useful for requirement elicitation and decision making. They also provide a concrete basis for communication and enrich coordination between developers and stakeholders. Finally, they also point out a risk of setting unrealistic expectations when communicating with prototypes.

Ratcliff claims that users often have fuzzy, incomplete, or inconsistent ideas about their requirements [28]. Prototyping can be used to clarify the objectives and communicate different paths to the solution. It also states that prototyping fosters an increased sense of cooperation between developers and users. It suggests a difference between early and middle prototyping, in that middle prototyping moves towards verification.

Multiple articles make shorter mentions of communication as a prototype purpose. Dow et al. mentions briefly that prototypes anchor group communication [22]. Zainuddin et al. attributes popularity of low-fidelity prototypes for communication to the fact that they can be realised without programming [29]. Budde et al. says prototypes supports discussions surrounding problems, the clarification of particular questions, or preparation of a particular decision [25]. The authors also states that prototypes can be supplemented with written system specifications for additional communication needs. Fern et al. says that prototypes can act as a lingua franca, proving a common point of reference for developers and stakeholders [21].

4.1.6 Usability testing

A prototype can support the evaluation of ideas and improvement of usability. Numerous frameworks for such evaluation exist and among the usability aspects to be evaluated are learnability, effort to operate, and understandability. Usability is different from desirability testing as it is not concerned with what the user should be able to do but instead focuses on how to make the experience efficient and enjoyable.

Lim et al. states that prototypes are rarely used for evaluations such as usability testing but are rather a tool for the designer to explore and learn [20]. They carefully note that evidence exists that usability testing provides similar results whether the prototype is of low or high fidelity. Using a case study, the paper finds that different usability issues were discovered when using different kinds of prototypes such as paper or computer-based. However, as long as the function was sufficiently simulated in the manifestation, the same usability issues were discovered.

Hendry et al. performs an experiment focused on usability testing prototyping [30]. The authors brings paper prototypes to the streets and recruits people for testing. This allowed the authors to reach user segments otherwise inaccessible for usability testing. It concludes that evaluating paper prototypes on the street is both possible and useful.

Sefelin et al. explores usability testing in different mediums [31]. It finds that the amount of suggestions and critiques does not significantly differ between computer and paper based prototypes. Users preferred testing with a computer based prototype as it gave them more freedom to explore the system themselves. Whereas in the paper prototype they needed to ask the test operator to simulate certain behavior. This leads to the paper reflecting whether this is significant reason enough for designers to prefer computer-based prototypes and how large a role the user satisfaction with testing matters.

McCurdy et al. refer to usability testing as collecting usability data via user testing [32]. They suggest that some usability issues cannot be discovered unless fidelity is sufficiently high. It approaches prototyping from a perspective of Human-Computer-Interaction (HCI) and suggests that the traditional definition of prototype fidelity has some shortcomings in terms of representing the artefact. It suggests *mixed fidelity*, a combination of low and high fidelity, as an approach that can allow for usability testing in a more economical manner.

Hakim et al. describe usability testing as finding a user interface that optimises the usefulness and usability of a solution [33]. The interface should allow the user to carry out the use cases in a good manner. The prototype could possibly incorporate the ability to capture various metrics. Usability data to examine could include task completion time and overall user satisfaction. The paper also notes that the associated instructions, task descriptions, and data in the prototype can affect the results of the testing. Therefore these are also a significant part of the usability testing.

Zink et al. state usability as one of four prototyping purposes [24]. In this model usability prototypes create an interaction between the customer and the product. By using such a prototype the developer generates data about the usage of the product. In a practical case study, usability was the least common prototype purpose. Furthermore prototypes were allowed to have more than one purpose and usability prototypes were usually combined with another purpose. Prototypes with a singular purpose of usability were extremely rare. The authors suggests that the reason could be the ease with which usability can be combined with another purpose. Usability as a purpose saw an uptick during the last iterations of prototyping.

4.1.7 Partial product

A software prototype can be assembled with the goal of evolving into the actual product. Development can then be performed incrementally and the prototype iterations can have purposes that differ from common prototyping. This implies that the prototype is developed in the programming language that the final product will use. Three common types of partial product prototypes are alpha, beta, and minimum viable product (mvp).

Schneider notes that some prototypes are built with the purpose of evolving into a final product [21]. Such a prototype is called a pilot system and should, in terms of characteristics, have no real distinction to the final system other than its completeness. Associated documentation should focus on judging system parts and the priorities for improvement.

Lichter et al. also refers to a prototype intended to become the final product as a "pilot system" [23]. The authors describes using the pilot system in the application area, starting with a core that becomes sophisticated with the addition of more features. As time progresses, the divide between prototype and final system slowly disappears. The work suggests that in such a development method the pilot system increments should focus on the user priorities.

Bellomo et al. discusses a continuous delivery approach to partial product prototyping [34]. They address two kinds of agile prototyping activities, the first being spikes and tracer bullets. Spikes are timeboxed and typically thrown away after having served their purpose while tracer bullets are exploratory activities that are kept. Using these activities, the authors describe a workflow where prototyping gradually implements functionality, using the prototypes to reduce uncertainty rather than having a separate and extensive requirement specifications. The focus of the paper is to optimise the quality of a prototype before the merge that occurs at the end of an iteration by comparing the effect of different activities.

Fairley et al. discusses several ways of prototyping a partial product under the term iterative development [35]. Iterative prototyping can be used when evolving a user interface. Agile development closely involves the user in prototyping activities that can happen several times in a single day. Incremental build sees developers creating weekly builds of evolving software. Finally, a spiral model can be used to mitigate risk in evolving software. The paper provides a "Plan-do-check-act" model for iterations which focuses on the "act". A term described as the rework conducted after prototyping reviews. The authors categorize different kinds of rework in terms of corrective, evolutionary, and retrospective and analyse these categories. The paper also provides guidelines for a health check on projects in terms of recurring rework to ensure a smooth execution of the continuous partial product prototyping.

Toffolon et al. proposes a framework for iterative development that incorporates prototyping based on software crisis [36]. The software crisis is defined by the complexity and uncertainty of software engineering and organisational activities around it. It states that these issues lead to exorbitant maintenance costs, backlogs of up to eight years, project costs and time overruns, and perhaps most significantly bad quality in the final software product. The paper defines two kinds of prototypes, informative or operational. Informative prototypes support exploration, clarification, and overall reduced uncertainty. Operational software are versions of the future system that have implemented well-documented requirements from stakeholders and are iteratively evolved. The framework specifies a Problem-Architecture-Construction-Operation (PACO) model that states how actors should interact with one another and which activities should take place during the different phases.

Goldman et al. describe a prototype as an incomplete approximation of the final system that, through iterative evolution, gradually moves towards completeness [37]. It defines three dimensions of completeness: system performance, system functionality, and user interface. The paper then proceeds to discuss the different toolkits useful fro developing these aspects, for instance Virtual Database Programming (VDP) that allows the simulation of data relationships.

Tronvoll et al. describes product development as a qualified guess for a solution [5]. This guess will sometimes miss the target due to the complexity and assumptions that have been made, therefore one should expect to redesign the solution and test it again. These cycles are the centerpieces of design methodologies design thinking and agile development and are avoided in methodologies like quality function deployment or total quality management. The paper proceeds to discuss a number of design approaches that can support a variety of prototyping purposes, these approaches are covered in the strategy chapter of this paper.

Fern et al. propose a three part prototyping methodology called Tri-Cycle [27]. They state that the process culminates with an operational (but possibly incomplete) prototyped system that can either evolve into the deliverable system or be considered a learning exercises (i.e throw-away prototyping). In the former case the prototype should be evaluated in a "Customer acceptance review" and further evolved.

4.1.8 Validation

A prototype can allow a design to be validated. It allows examining whether the requirements are adequately satisfied, which might affect how the project proceeds. Validation is essentially asking if we “built the right thing”, and can be done with either internal or external stakeholders. When the prototyping purpose is validation it implies that assumptions have been made and some form of artefact has been built.

Yasar states that low-fidelity methods are able to validate designs and predict large problems at a low cost, garnering them popularity [26]. The low cost is attributed to using a whiteboard drawing or paper prototyping.

Raja discusses requirements prototyping [38]. Requirement prototyping helps requirement engineers to discover and understand the customer requirements, allowing for the customer to be involved with requirements validation. They suggest two kinds of prototyping to support such validation. Throwaway prototyping can be used to resolve unclear requirements while evolutionary prototyping is based on implementation of agreed requirements. These processes help with visualizing the requirements, possibly providing customers with a clearer view of requirements than a requirement specification document can. Finally, Evolutionary prototypes can also be used during system testing to provide consistency into the final product.

Bellomo et al. described the use of prototyping in one company [34]. Minimal prototyping was encouraged, only prototyping with sufficient depth and breadth to validate the desired concept. This is in line with the thought that unjustified early planning and precision in requirements is counter-productive.

Ciriello et al. Says prototyping is a way to find out what works or not [9]. The existence of prototypes is justified by their ability create knowledge about solutions and problems. The authors state that the outcome of such a process can be an minimum viable product (MVP) containing just the essential features along with validated learning. They also refer to a quote stating "The whole idea is only sketch until you build a prototype and validate with someone who has business knowledge. Unless you create something tangible, you never get to the next level." The authors also state that while a prototype can support the activity, it can not alone validate the requirements.

Fairley et al. advocate a more continuous approach [26]. Development is done iteratively and as capabilities are added they also validate those capabilities, reworking if necessary. Iterative development can take different shapes, for instance iterative prototyping or agile development. In agile development the customer is closely involved and prototypical reviews might occur several times daily. The authors suggest that validation should can lead to either too much or too little rework, both of which can be indicative of issues in the process. They provide some reasoning that a suitable amount of rework could lie between 10-20 % of the development effort.

4.1.9 Performance improvement

Performance can be of varying importance depending on the type of system. If performance is good it can be a competitive advantage for the product, and if performance is poor it might create issues. The system might have long response-times or become unstable when handling a certain quantity of data. Prototyping can then be undertaken to keep all other aspects of the product the same, while making changes to try and improve the performance.

Kordon states rapid prototyping is used to improve efficiency of the code [39]. The code is run on hardware with very limited amounts of resources, hence the code must be as efficient as possible. Prototyping is done in either a throw-away approach focused on a subset of requirements or in an evolutionary approach. In this type of prototyping, it is also important to keep overhead from measuring execution to a minimum, as it can interfere with providing an accurate measurement.

4.2 Strategy

The prototyping strategy determines how resources and design decisions are organised in order to manage uncertainty. There are four strategies to consider when undertaking prototyping [5]. The approach to design is dictated by the strategy and in some strategies multiple designs are pursued. The strategies are described in further detail below.

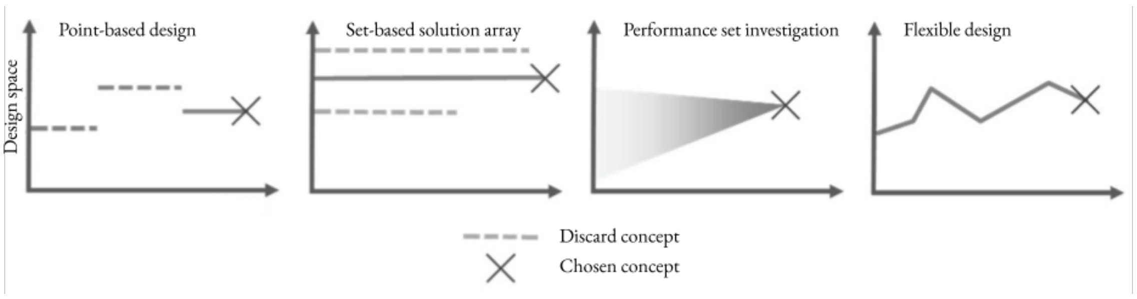


Figure 4.1: Visual representation of the four strategies and their design approach.

4.2.1 Point-based design

Resources are focused on achieving a high degree of confidence in one single design. Design choices are considered carefully before being cemented in the design. If the design appears to be missing its target it might be necessary to discard the design and redo the process.

4.2.2 Set-based solution array

In a set-based solution array several potential designs are pursued simultaneously. Design decisions are managed by diverging into two separate design tracks. Potential designs can be merged either continuously or during stage-wise iterations.

4.2.3 Performance set investigation

Performance set investigation converges on one design by applying a systematic performance evaluation. Decisions surrounding design are postponed until they can be validated and when encountering several alternatives only the most promising one is pursued. The number of design decisions to be investigated simultaneously is kept to a minimum in order to minimise cross-contamination.

Design decisions are judged with performance evaluation which slightly changes the nature in which questions are asked but provides a more complete understanding of the design. This form of evaluation treats performance as a range rather than a binary *can* or *cannot*. Due to the nature of having a range as an answer one must inquire on additional performance factors. For instance, instead of asking whether a design for a word processor allows saving one can investigate the average amount of time it takes to perform a save. Additional performance factors could be that a certain design is more efficient at saving but drastically increases the size of the save file. The design gradually converges as the number of additional performance factors increases.

4.2.4 Flexible design

In a flexible design strategy the solution is based on making best-guesses and making it easy to perform changes as the work progresses. The design is iterated and evaluated, changing direction when required. Thanks to its modular approach it is suitable for iterative product development.

4.3 Scope

Scope is the extent to which the prototype is made to look and act like the real product. The prototyping scope can be described in terms of visual and functional refinement. These two dimensions represent the fundamental essence of a prototype. Though related to some degree as one can not represent functionality without visuals, these dimensions are two different sides of the prototypes. The dimensions can be refined to varying degrees and can be represented as a 2D-space as depicted in Figure 4.2.

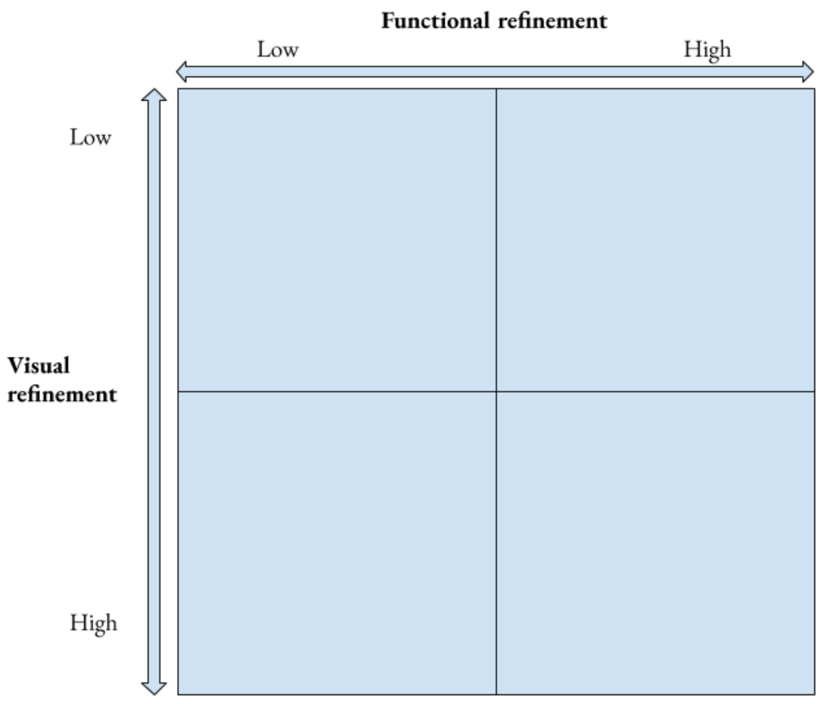


Figure 4.2: Depicts the functional and visual fidelity as prototype scope.

There are many different ways to assemble a software prototype, which lend themselves differently well to achieving these degrees of refinement. These dimensions possess some complexity and can be further explored.

4.3.1 Visual refinement

Comparing the looks of a prototype with ones perception of what the final product is to achieve yields an estimation of similarity called visual refinement. It encompasses aesthetics such as colours, layout, and fonts but also user interface elements such as icons and buttons.

4.3.2 Functional refinement

Functional refinement states the same comparison as in visual refinement but with a focus on the behaviour of the system. There are four different aspects to functional refinement. These aspects can be likened to properties with some degree of hierarchy.

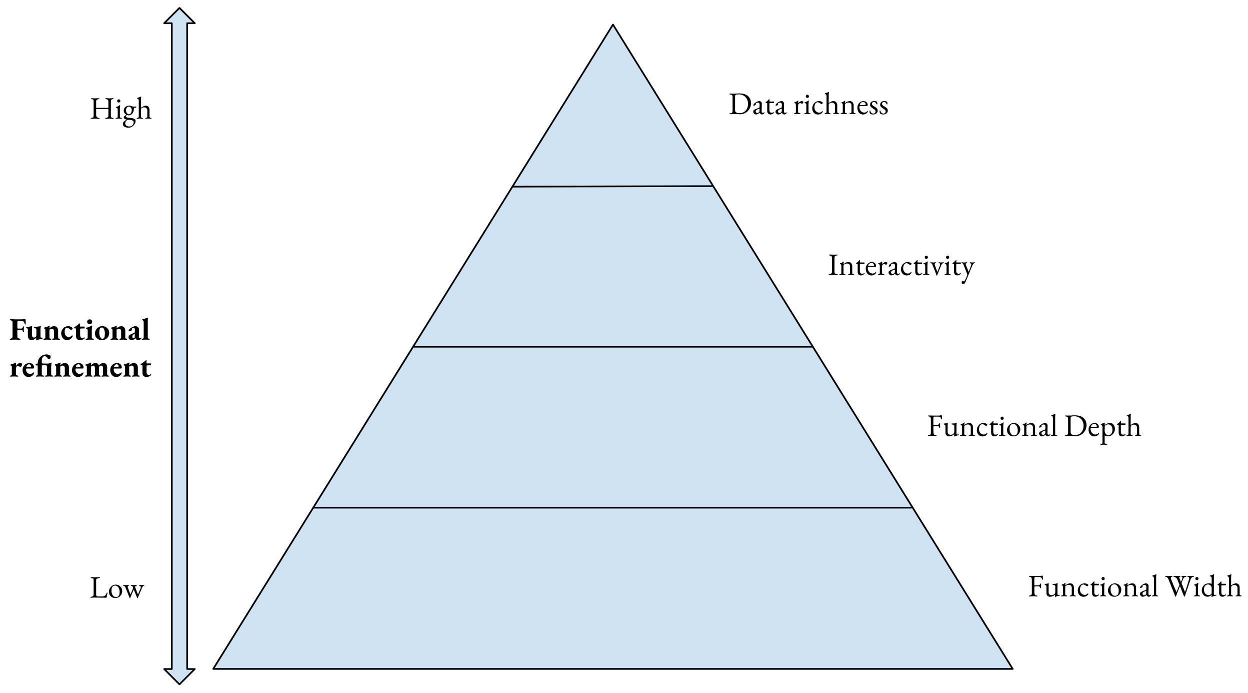


Figure 4.3: A breakdown of the four stages in functional refinement.

Specifying functionality can be described as a process that traverses four stages, from the idea of what a product shall achieve to what data is necessary to reflect real world usage. The first stage is functional width where one decides what needs should be addressed by the system and what functions are required to address the needs. Each functionality is then divided into its constituent parts in the form of functional depth. The third step is to establish the user's connection to the system, also known as interactivity. This is best accomplished by answering *how do users access a function?* and *what feedback does the system provide?*. Data richness is the final step of functional refinement and concerns the data richness of the prototype. The prototype might require a certain level of data richness in order to accurately simulate normal use. Additionally, certain issues might only be discovered if the data sufficiently diverse. A low-functional refinement prototype has limited functional breadth and depth, perhaps only hinting at some functionality by showing a non-working buttons. A mid-functional refinement has a higher degree of functional width, depth and possibly some degree of interactivity. A high-functional refinement prototype has a high degree of functional width and depth and some degree of interactivity and data richness.

4.3.3 Contemporary prototype dimensions

Traditional literature discusses the dimensions of software prototypes in terms of fidelity. A term used to describe how much resemblance there is between the prototype and the final product. Low-fidelity indicates significant difference while high-fidelity can be 9/10'ths of a complete product.

Advances in software that aid in the creation of prototypes has lead to a gap concerning terminology. Modern prototypes can have a high degree of visual refinement and low degree of functional refinement while still being referred to as a high-fidelity prototype. It is evident that for some low-fidelity is synonymous with paper-based prototyping and high-fidelity is synonymous with any computer-based prototyping.

In recent literature the term mixed-fidelity has emerged as an alternative to low and high fidelity. It bridges the aforementioned gap and is used to describe a prototype that is low in one and high in another dimension of refinement, be it visual or functional. The introduction of this term acknowledges that low and high fidelity are insufficient at describing the complexity of a prototype but ambiguity remains on what kind of refinement exists.

4.4 Review method

Review method is the manner in which the prototype is evaluated. It covers three areas; the way in which a presentation is conducted, the environment in which the prototype is used, and finally the potential division of the prototyping process into iterations.

4.4.1 Presentation and use

There are four different ways in which a presentation can be conducted.

- Demonstration *A presenter shows the prototype, either by operating it live or using prepared media such as video or photos.*
- Scenario testing *The user operates the prototype and is given instructions by the presenter on what to do. The steps required to perform the instructions may be implicit or explicit.*
- Free testing *The user is allowed to explore the prototype and is given a minimal amount of instructions.*
- No user presentation *The review might be entirely a technical evaluation where performance is measured and no user feedback is required.*

4.4.2 Setting

A setting is judged based on its similarity to the conditions in which the final product will be used. This reduces the risk of false pretenses and has a potential of highlighting requirements that are specific to the setting. A few things should be considered

- Environment *A lab environment offers a controlled setting and operational convenience. However, it might mean that the prototype is tested in an environment that is different from where it will be used. For instance, a prototype for a system intended to be used in a loud environment produce other feedback in a quiet lab environment than its natural habitat.*
- Representation *Depending on how reviews are conducted some future users might become underrepresented in design. Feedback about desirable properties of the system might not be captured if stakeholder participation is lacking. Conversely, a user experience engineer might be better trained to discover usability.*

4.4.3 Iterations

Development of a prototype can be split over several iterations. Such a process can feature cyclical repetition of activities such as plan, build, test, and evaluate. When deciding about iterations there are some facets to consider.

- Length of iterations *The length of an iteration can have an impact on design. Jaskiewicz et al. found that multiple short iterations lead to a diverse but superficial set of design solutions [40]. In contrast longer and fewer iterations promoted larger focus on certain aspects.*
- Iteration purposes *Each iteration can come with a different set of objectives and purposes. Focus and development can then be directed towards certain purposes, possibly simplifying the process.*
- Concurrency *Several design paths can be pursued simultaneously in an iteration. This can lead to more diverse functionality in the final design but can also result in contradictions between prototypes that make the merger of design paths difficult or even impossible [40].*

4.5 Model design choices

The design of the framework was based on the literature selection and the categories that were found with thematic coding. **Purpose** and **scope** were the most common dimensions throughout all articles and were therefore easily identified. Scope is typically, but not always, referred to as *fidelity*. The term scope is however more encompassing and was therefore deemed to be more appropriate than fidelity.

Two additional aspects were discovered and added to the framework. The first was **strategy** which describes how resources and uncertainty is handled during prototyping [5]. The second aspect was **review method** which describes how the prototype is tested and how knowledge is extracted from it. The latter received much attention as multiple papers discussed the practical application of various methodologies.

Aspects were developed individually. We decided to use [24] as our starting point for prototyping purposes as the paper offered the most comprehensive view of prototyping purposes. The rest of the literature was then scanned for additional purposes that did not overlap with the already defined ones. The aspect of prototyping strategy was only described thoroughly by a single paper, thus it was used for describing this aspect [5]. The aspect of scope was discussed in several papers. The most sophisticated model was offered by McCurdy et al. which described five dimensions of prototypes[32]. A compromise was made, the common functional/visual fidelity model was used but with the five dimensions integrated as parts. As for the final aspect review method, a combination of different papers were used to characterise this aspect.

Chapter 5

Agile Prototyping Guidelines

5.1 Guidelines

These guidelines recommend how prototyping can be used in a software development project to conduct Requirement Engineering Activities. The guidelines consist of five steps that should be undertaken in the development process. The five steps demonstrate how prototyping can support requirements engineering in software development by establishing a blueprint for the system that should be built and agreement around that blueprint. During these steps prototypes serve as a system specification. The three main activities of Requirements engineering that are addressed in this step is elicitation, specification, and validation. The intended user of these guidelines is the specification architect. By *specification architect* we refer to someone driving the development, this could be a requirements engineer, developer, manager, product owner, consultant, or designer and could be more than one person. An overview of these guidelines is presented in figure 5.1

The pre-requisites for applying the guidelines is that there exists a primitive outlining of that project that depicts what type of software system should be conceived and who the stakeholders are. The intended type of system can be vague but should at least state whether it is an invoice handler, fitness tracker, or something else that is being developed. As for stakeholders we presume some future users are identified and available as well as a customer responsible representative referred to as customer for brevity.

Steps can be undertaken in a different order than presented depending on what the the current project needs are. As with many agile development projects the steps may be performed in cycles. We also add that project context can make certain steps more or less necessary. For instance proximity to a deciding stakeholder during specification might reduce the need for a formal validation as prescribed in the fifth step. Stakeholders without proximity might need to be included during various steps of the development.

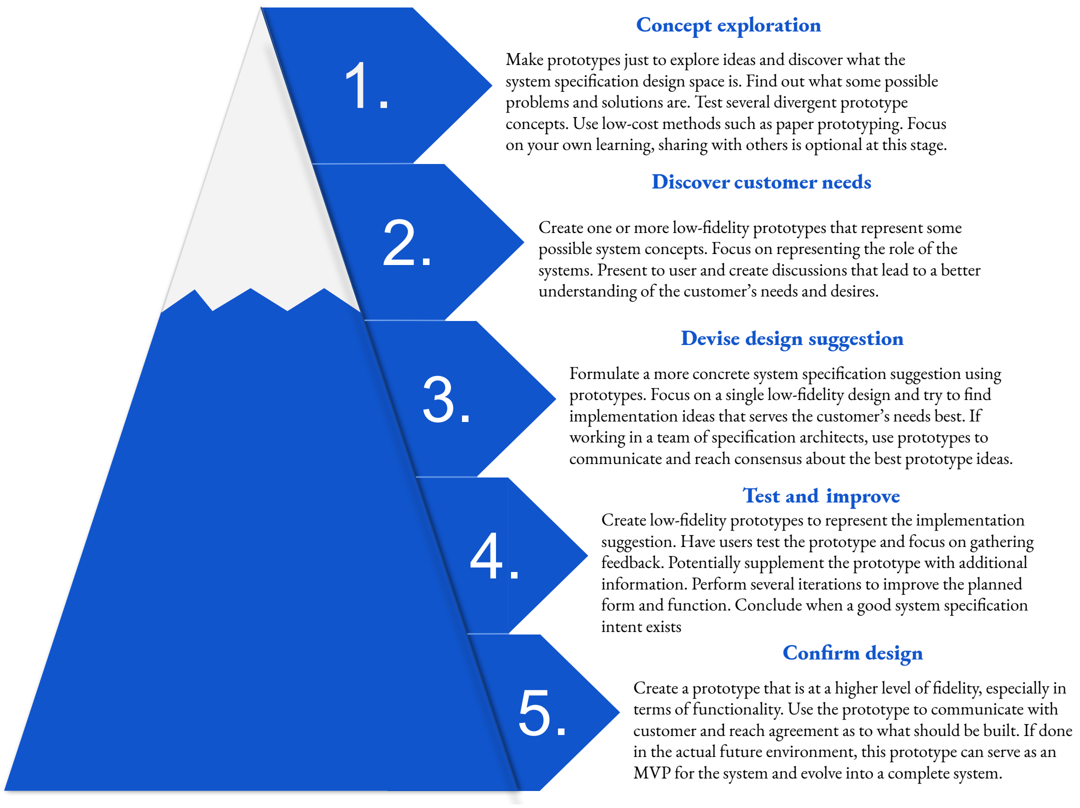

1.
Concept exploration
Make prototypes just to explore ideas and discover what the system specification design space is. Find out what some possible problems and solutions are. Test several divergent prototype concepts. Use low-cost methods such as paper prototyping. Focus on your own learning, sharing with others is optional at this stage.
2.
Discover customer needs
Create one or more low-fidelity prototypes that represent some possible system concepts. Focus on representing the role of the systems. Present to user and create discussions that lead to a better understanding of the customer's needs and desires.
3.
Devise design suggestion
Formulate a more concrete system specification suggestion using prototypes. Focus on a single low-fidelity design and try to find implementation ideas that serves the customer's needs best. If working in a team of specification architects, use prototypes to communicate and reach consensus about the best prototype ideas.
4.
Test and improve
Create low-fidelity prototypes to represent the implementation suggestion. Have users test the prototype and focus on gathering feedback. Potentially supplement the prototype with additional information. Perform several iterations to improve the planned form and function. Conclude when a good system specification intent exists
5.
Confirm design
Create a prototype that is at a higher level of fidelity, especially in terms of functionality. Use the prototype to communicate with customer and reach agreement as to what should be built. If done in the actual future environment, this prototype can serve as an MVP for the system and evolve into a complete system.

Figure 5.1: Summary of the 5 guideline steps

Step 1: Concept exploration

In this step the specification architects should use exploratory prototyping to take the first steps in the development process. This kind of prototyping lets the specification architects explore a design space and in this parallel discover both problems and possible solution directions [20]. By doing so the specification architects learn what the product can become and what the essential design choices could be. The specification architects do not need to share the prototypes with the users at this point but can rather focus on their own learning as each prototype that is tested can lead to new system specification ideas and design spaces to explore [20].

Low fidelity methods, such as paper prototyping, are recommended as they can be used to explore specification aspects at a low cost while being able to predict large problems [40]. In essence, even a few quick sketches on paper can allow for experimenting with ideas. Several prototypes with different concepts should be created to explore a variety of ideas as this can lead to more a divergent set of ideas and a better end-product [22]. Concepts can then be integrated and merged as the work progresses [24]. This step can be concluded once specification architects have some ideas about design space and possible rudimentary concepts for the product.

Purpose Exploration

Strategy Set-based solution array
Scope Low-visual, Low-functional
Review method Any
Requirements Engineering activity Elicitation

Step 2: Discover customer needs

Specification architects can use prototypes to support discussions about the future system. In this step, prototyping is done with a focus of what role the product will play in the users life, thereby establishing the concept frames [26]. Low-fidelity prototyping is viable as the prototyping scope only needs to be high enough to represent the role and concept of the system. As the conversations about *desirability* progresses the user tasks are elucidated [9][25]. *Desirability* will become clearer and the value to the customer is also better understood by the specification architects [24].

Multiple paths should be examined if possible, either by creating multiple kinds of prototypes or taking a flexible and modular approach. This allows several design spaces and their relative merits to be explored [22]. By sharing prototypes with users the specification architects can prove, disprove, and demonstrate various points [21]. Particularly points from the previous step of these guidelines can be important to review. This step can be concluded once the customer needs and the functionality required has been established.

Purpose Desirability (Role), Exploration
Strategy Set-based solution array, Flexible design
Scope Low-visual, Low-functional
Review method Any
Requirements Engineering activity Elicitation

Step 3: Devise design suggestion

In this step specification architects should use prototyping to carve out a design for the future system. Specification architects can expand the *desirability* perspective to include the *implementation* of the system, i.e how the system should be designed to best provide value to users [26]. The design is formulated by creating a prototype that fits with the discoveries in the previous step.

To provide a clear direction the focus should shift towards a single design, converging previous design options. Formulating the design is also a useful way of communicating and aligning within the specification architect team. Group communication is anchored through prototypes supporting discussions surrounding problems and clarification of particular questions [25]. This step can be concluded once the specification architects agree on a specification suggestion.

Purpose Desirability (Role, Implementation), Communication
Strategy Point-based
Scope Low-visual, Low-functional
Review method Any
Requirements Engineering activity Specification

Step 4: Test and improve

Specification architects should use prototypes to test their suggested concepts. In this step, users test the prototype and provide feedback on the *Implementation* of the system and whether it is well suited to the system's intended *role*. Simultaneously the prototype is iteratively improved with the help of the feedback. The prototype is used as a specification as well as a communication tool, providing a common reference point for them and the users while gathering useful feedback [21]. Prototyping at this point can also be used to clarify objectives and different paths to solutions, allowing a shared vision to emerge between the groups. [28] Low fidelity methods can be used as the concept is in focus, however the prototype can be supplemented with additional information like a written system specification if required for further communication needs [25]. A flexible design approach is preferable as many changes might be required while user feedback is collected. This step can be concluded when specification architects are content with the amount of feedback collected.

Purpose Desirability (Implementation), Communication
Strategy Flexible design
Scope Low-visual, Low-functional
Review method Any
Requirements Engineering activity Specification

Step 5: Confirm design

In the final step the specification architects should use prototypes to formally confirm specification with the customer. Customers can validate either a throw-away prototype or an evolutionary prototype. [38] The first type is used as reference when building the actual system while the other one is gradually evolved into the final system. If the latter approach is chosen, the prototype produced is typically called a Minimum Viable Product (MVP) or Pilot System. The goal of this step is to achieve high confidence in a single prototype and reach agreement on what should be built. To properly simulate the final system a mid or high level of functionality must be implemented, in the process of doing so issues and inconsistencies might be revealed. The step is concluded when the customer accepts the prototype as specification and implementation commences.

Purpose Communication, Partial product, Validation
Strategy Point-based
Scope Low/mid-visual, Mid/high-functional
Review method Any
Requirements Engineering activity Validation

5.2 Implementation of Guidelines

Companies wishing to implement these guidelines should start by considering who should be represented as stakeholders. The focus-group indicated that the composition of stakeholders has an impact on the execution of the different steps. One should consider if all the right people are involved to successfully execute all the steps. Next one should consider if certain steps should be merged or possibly iteratively repeated depending on the nature of the project and company. In the final step of implementation the processes and tools to be used should be decided on. We recommend that practitioners start of simple and then gradually adjust the process so that it becomes a good fit.

Chapter 6

Result

Our work has produced two theoretical frameworks to advance the area of prototyping. The Prototyping Aspects Model presented in Chapter 4 and the Agile Prototyping Guidelines presented in Chapter 5. The former provides an insight on the topic of prototyping and the latter presents guidelines for applying prototyping in the requirements engineering process to support agile software development.

In this chapter we present three intermediate results used in designing and validating the aforementioned model and guidelines. The first result originates from the literature study and concerns a demographic overview of the underlying systematic literature study. Findings from the exploratory case study represent the second result and provide an insight on the effect of altering the aspect of prototyping scope and the impact on the output and learning in software development. The focus group was the last result and was conducted in order to validate a set guidelines that are to support agile development projects and its results are presented here.

6.1 Literature study

The literature study provided a structural approach for identifying a set of articles that laid the foundation for the Prototyping Aspects Model presented in Chapter 4. This section provides an overview of how the articles relate to the aspects of the model and presents the demographics of the set of articles regarding publication date and country of origin. The literature study is further described in Section 3.1 while the model can be found in Chapter 4.

The four aspects of the Prototyping Aspects Model were identified by analysing the articles of the literature study. Table 6.1 specifies how each article contributed to the aspects. Some articles covered multiple aspects and therefore occur more than once while a handful have been marked with parentheses due to only

Table 6.1: Overview of the articles that were used to create the aspects of the prototyping aspects model.

Category	Articles
Purpose	1, 3, 4, 6, 11, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34 (18)
Strategy	8, 12, 22, 23, 30
Scope	2, 4, 7, 9, 13, 14, 16, 18, 19, 21, 28 (1,30)
Review method	4, 5, 10, 15, 17, 20, 21, 23, 25, 28 (16, 30, 31)

The articles in our final set were published in the years from 1988 to 2018. Figure 6.1 presents this by clustering the articles in sections of four years to balance the level of detail with the legibility of the graph. Except for a dip in *1998-2002* for which we cannot find a reasonable explanation, the results are spread out evenly with a slight increase in recent years.

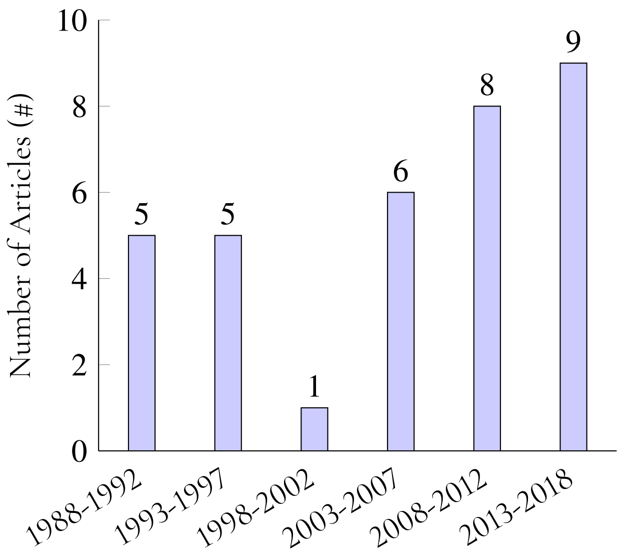


Figure 6.1: Number of articles split by group of years.

The country of origin was determined by looking at the primary author of each article and grouping the countries by continent. The result of this is shown in Figure 6.2.

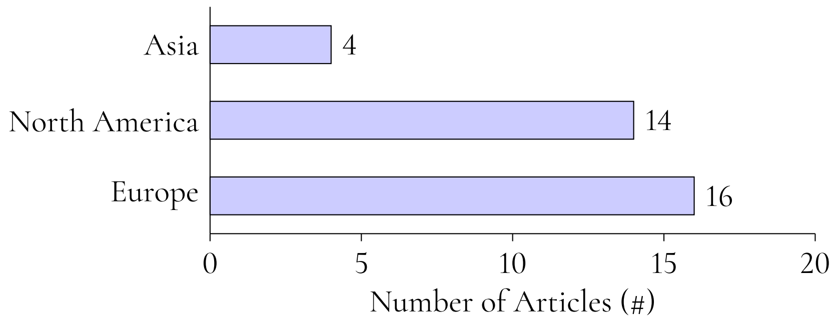


Figure 6.2: Number of articles split by origin.

6.2 Case study

We performed an exploratory case study to investigate the aspects of the Prototyping Aspects Model (see Chapter 4) and its relation to the outcome and learning within a software project. The study consisted of 15 sessions during which subjects were exposed to one of three prototypes after which they filled out a survey. All 15 employees were employed as advisors, a role which is described in section 2.3.1. A reminder of the differences between the prototypes is provided below and a detailed description of each prototype can be found in Section 3.3.1.

Prototype (1)	High visual - high functional
Prototype (2)	High visual - low functional
Prototype (3)	Low visual - high functional

6.2.1 Experiences of Prototype Variants

To procure an image of what areas subjects discussed during their session we applied thematic coding on the first segment of the survey and on notes from the interviews. The thematic coding highlighted three areas that could be of interest when optimising output and learning with prototyping in a software project; *visual*, *functional*, *conceptual*. A consolidation of the result is presented in table 6.2 and a detailed insight is provided thereafter.

Table 6.2: Overview of participants perception of the different prototype variants.

Question	Prototype (1)	Prototype (2)	Prototype (3)
What was good?	visual, conceptual	visual, conceptual	conceptual, functional
What was bad?	visual	visual	visual
How could you improve the system?	visual, functional	visual, functional	-

Subjects that experienced Prototype (1) and Prototype (2) discussed the visual and conceptual attributes when being posed with the question *What was good?*. Visual comments often referred to the interface with statements such as *Clean interface* and *Nice interface*. Comments that referred to conceptual attributes hinted at the overall look and feel of the prototype and how well this would fit with the current product offering. Examples of responses include *The overall design was great*, *Easy to find functionality since it is logically divided*, and *This feels like the logical next step for the product.*

Subjects that had worked with Prototype (3) were prone to discuss conceptual and functional attributes when being asked what they perceived to be good. The former resembled the feedback that subjects had given with Prototype (1) and Prototype (2) but the amount of feedback on functional attributes was higher than with those prototypes. This feedback consisted of comments that discussed in detail how something should work, such as *It is good to see which channels you are active in* and *GOOD to be able to edit the interface to my liking as with the dots in that slide.*

The length of responses varied significantly when subjects were asked to highlight what they perceived to be bad with the prototype they had been shown. A majority of the answers were short statements, resembling *Not much at all* or *I did not think anything was bad*, while a handful were elaborate with multiple paragraphs that covered all three areas. The number of comments that could be linked to visual attributes were slightly higher than for conceptual and functional but a vast majority of the comments were variations of *The icon for issue management was unclear* or *A lifebuoy is not a good icon for issue management.*

When asked how one may improve the system, subjects that worked with Prototype (1) and Prototype (2) provided feedback on visual and functional attributes. Comments included *Change the icon for issue management* and *See emails from customers in the system.* While subjects that had worked with Prototype (3) did provide a handful of comments regarding visual attributes there was such a large gap between these and those of Prototype (1) and Prototype (2) that we felt it unfair to highlight any area at all.

6.2.2 Quantitative Performance Evaluation of Prototypes

The second segment of the survey provided quantitative data to evaluate and compare the performance of each prototype. To obtain this data, we posed five questions to which subjects responded with a Likert scale.

Table 6.3 contains the complete collection of responses. Values are displayed in ascending order meaning that they do not necessarily stem from one and the same session. Each combination of question and prototype has received five responses leading to a total number of 75 values.

Table 6.3: Results from the statements in the second segment of the survey. The values are shown for each combination of question and prototype.

Question	Prototype (1)	Prototype (2)	Prototype (3)
(A) It was easy to grasp what functionality the system will have.	3, 4, 4, 5, 5	4, 4, 4, 4, 4	3, 4, 5, 5, 5
(B) It was easy to grasp how the functionality will work.	2, 4, 4, 5, 5	3, 3, 4, 5, 5	2, 4, 4, 5, 5
(C) It was easy to grasp how the system will look.	3, 4, 4, 5, 5	4, 4, 5, 5, 5	2, 2, 3, 4, 5
(D) It would feel difficult to suggest large changes in the system.	1, 1, 1, 2, 3	1, 1, 2, 3, 3	2, 2, 2, 4, 4
(E) It was easy to imagine alternative ways the system could be designed.	2, 3, 3, 3, 5	2, 3, 4, 4, 4	1, 1, 3, 3, 4

Substance for the evaluation and comparison was created by calculating the average, median, and variance of each combination of question and prototype. The results of these calculations are presented in Table 6.4.

Table 6.4: Average, median, and variance of each prototype in the second segment of the survey. The values are based on the figures presented in Table 6.3 and are shown for each combination of question and prototype.

	Prototype (1)			Prototype (2)			Prototype (3)		
	Avg.	Med.	Var.	Avg.	Med.	Var.	Avg.	Med.	Var.
A	4.2	4	0.7	4.0	4	0.0	4.4	5	0.8
B	4.0	4	1.5	4.0	4	1.0	4.0	4	1.5
C	4.2	4	0.7	4.6	5	0.3	3.2	3	1.7
D	1.6	1	0.8	2.0	2	1.0	2.8	2	1.2
E	3.2	3	1.2	3.4	4	0.8	2.4	3	1.8

6.3 Focus group

The focus group acted as an evaluation and validation of the Agile Prototyping Guidelines (see Chapter 5). Participants expressed their opinions surrounding the guidelines in their current form, how it suited their work style, and elaborated on possible ideas for how to improve the guidelines. This section presents the qualitative results of the transcribed material from the recording of the interview.

6.3.1 Agreement with guidelines

Several statements were identified which agreed with the guidelines presented. The participants said these guidelines are a pretty good match with their method of working and confirmed that it is a good way of working. When stating this one of these participants also added that she believes the guidelines match a typical design process rather well and did not add much new. Another of these participants added that a good thing about the model is that goals are made clear for each step. Particularly step 1 and 2 had support expressed during the focus group. Regarding step 3 and 4 a similar but slightly different was advocated by the company which is elaborated on in Section 6.3.2. Step 5 was not seen as less applicable for Telavox and is discussed in Section 6.3.6. In this section we present some statements that were found to be in agreement with various specific parts of the guidelines.

One participant stated the importance of creating several prototyping variants to not get stuck in a single solution early, agreeing with step one and two that multiple concepts should be tested in an early stage. Telavox uses low fidelity methods to create the variants at this stage, typically either paper sketches or a tool called Invision.

When asked if she usually shows her first drawings of a potential system, a participant responded that she tends to not do so. However, she finds that having a discussion with users can be difficult without having anything to show. She added that she believes it is important to create several prototype variants to consider a variety of directions. She said that presenting these prototyping variants to users makes it easier for them to give feedback about which direction for the system is best. In step two of our guidelines it is recommended that several different low-scope prototyping variants are presented to users, which can be supported by the comment of the participant that this helps user express which direction for the system that they believe in.

Whilst discussing step two of the guidelines and specifically elicitation of requirements a participant said that it is generally easy to get feedback on what is good but hard to get feedback on what is bad when presenting prototypes to others. He added that it is important to structure the test in order to get feedback on the right things when a prototype is being presented to a user. Another participant added that at this stage of project they usually work with wire-frames and avoid pixel-perfect prototypes. He said that this can be a helpful approach in acquiring the particular feedback they are seeking, implying that prototype feedback on functionality is desirable over feedback on visual style at this point. This statement is in line with the guideline recommendation of working in low-visual prototyping scope during step 2.

One participant first said that he does not think prototypes should come into play during a typical step 2, i.e. the first meetings with a user. However after some consideration he said that prototypes are beneficial at this point in loosely defined projects. He stated that in these kinds of projects the prototypes can help create discussion surrounding the desires of the future system. In a recent project they had shown sketches which helped both designers and users understand what they were discussing and assisted in framing what the role of the system was. In the guidelines we recommend that during step 2 prototypes are used to support conversation about the role of the future system (*desirability testing*) with users.

A participant brought up the topic of what questions you are asking a user or customer proxy. He said that if you ask things like *how should we design this page?* then you get faulty answers. Instead he recommends trying to figure out what the problem that the future system shall solve is through asking *solution-agnostic* questions. He implied that a prototype can be employed for this purpose. This is in accordance with the second step of our guidelines where we recommend prototyping to support a discussion focused on the problem to solve the (role) of the future system rather than how it is solved.

6.3.2 Questioning linear order of steps

Some of the focus group discussions concerned that carrying out the steps in linear order might not be feasible nor desirable. As a result of this the guidelines were adjusted to state that steps can be repeated and the order of steps can be rearranged to suit current project needs. Relevant points to this are presented below.

One participant asked whether moving backwards in the model was possible. She presented a theoretical scenario in which a system specification fails during the validation step (step 5) for not presenting a sufficiently strong business case. She then inquired whether it was possible to move back to a previous step or one would have to redo the entire guidelines process. This demonstrated a need for moving backwards and repeating steps.

The participants explained that they often return to step 3. The team use each other as sounding boards to discuss ideas. They will then explore these ideas on their own and might proceed to develop or update a prototype that a user or other designer then tests (step 4). Once they have gathered that feedback they return to step 3, discussing the next iteration with the design team. This sequence is repeated several times over a specification process. They added that often the same people are involved several times throughout the process when trying to come up with a concrete system design, meaning for them step 3 and 4 is more intertwined than the guidelines.

A participant also informed us that they have implemented a process of weekly feedback meetings. In the meetings the team shares prototypes that are between 10-70 percent complete. During these meetings they bounce ideas of each other and review each other's work (Step 3 and 4). Between meetings they work on improving their prototype. This demonstrates a cyclical repetition of steps that can be useful.

6.3.3 Terminology discussed by participants

Some discussion was held surrounding terminology in the guidelines. It was considered whether some terminology should be changed surrounding the guidelines or if a terminology chapter should be introduced, but ultimately it was not deemed necessary. Points are presented below.

One participant said he does not consider paper sketches a form of prototype and added that for him a prototype is something interactive that you can test out and really use to get a feel for the final system. Most literature on the topic does however consider paper sketches to be a form of prototyping (paper prototyping) including this paper, hence no changes were made.

Another participant built on this by stating that *prototyping* can mean different things and that the same is true for *testing* and *customer needs*. She did not offer her own definition on these things but rather pointed out that they can be difficult things to interpret and that it is a potential area of improvement for the guidelines. The definition of *prototyping* offered in this paper (see Section 2.2) allows for a very broad interpretation of what *prototyping* is in order to cover these different types of prototypes. In regards to *testing* and *customer needs* these changes would require the guidelines to take a step become much more of a concrete method rather than guidelines, hence were not addressed.

6.3.4 Unclear results

In some instances participants expressed conflicting views on parts of the guidelines. These points are presented below.

One point of discussion during the focus group were if any prototypes should be used at all during step 1 and 2. One participant said they rarely make prototypes during early phases such as step one or two with the reasoning that it is time consuming and is difficult. He advocated spending the time on understanding the task instead through interviews of stakeholders and other types of analysis. Another participant disagreed and stated that prototyping at this time can be useful. She stated, that for her, sketching is a way of understanding the problem. Yet another participant agreed with the sketching approach and said he often produces a small requirement list for himself before starting to sketch.

While discussing step 3, one participant said they do not think prototypes can be used with the purpose of specifying but should instead be used to communicate. Another participant disagreed and stated he sometimes creates prototypes for himself to get an overview of what the system should contain, essentially working as specification. In step three it is advisable that prototypes start being used as specifications.

During discussion of step 4 it was discussed whether users that test the prototype could provide solution to problems. When participants were asked whether they ever use prototypes to throw problems out in front of other people in an attempt to solve an issue they gave two different answers. One participant said they sometimes involve users to solve problems while another participant preferred to solve them privately. Step 4 of the guidelines advocates the approach of involving users and suggests an iterative approach to testing and improving the system by having users test the prototype.

6.3.5 New ideas

Some new ideas concerning prototyping surfaced during the focus group. While not a good fit for the structure guidelines, they are still interesting to present. The points are presented below.

When asked for other ways to brainstorm and generate ideas about potential system design, one participant provided an interesting answer. In the case that the prototyping task requires one to find a good structure for an information hierarchy, such as a menu, the participant sometimes uses bullet lists to start exploring different structures. He added that bullet lists can be good for a lot of things but did not provide additional examples. This demonstrates that bullet lists can sometime work as a prototype for organising information.

When discussing the elicitation of needs from a user, such as during step 2, one participant explained how she typically goes about it. She stated that even though she might have an interactive prototype, she first discusses their expectation on how something should work and what it should look like visually. After she has gathered a good understanding about the problem and the user desires she then presents her prototype to discuss whether it ticks all the boxes that they previously discussed. The participant said she finds this useful to understand the current solution and problems surrounding it. This suggests that as part of the review method, one can consider showing the prototype after an interview has been conducted to avoid priming the user towards the new system concept before understanding problems with the current one.

When asked about how new system designs are tested, one participant shared that while they sometimes have the CEO approve new system concepts, they often test functionality by gradually rolling it out in beta test with users. This demonstrates that instead of getting formal approval from a stakeholder, validation (step 5) can be conducted through beta-testing. Doing so could potentially be a beneficial alternative.

When asked what people should be involved during step 4, one participant talked about involving developers. She said that as she does not write code but merely produces the specification for what should be implemented she finds it important to include developers during the process. Her goal is to find out which design solution is easiest to implement. She explained that she might have three different options of how something can be implemented where one option might require two weeks of development effort while another can let you reuse an already existing component. This shows that in steps 3-5 one could potentially add *Viability testing* as an additional purpose to find out how much time and resources a certain design path will require.

6.3.6 Disagreement

Some disagreement was expressed with the guidelines. We report on these points below.

When discussing the third step, one participant disagreed with creating a prototype at this point. He stated that once you start constructing a prototype you might find it difficult to go backwards and consider other design option. He postpones prototyping as long as possible as otherwise several requests might show up to add additional things to the system which are challenging to fit into the emerging system specification concept. He said that to fit those things in you need to compromise and sometimes reset, which is a difficult process. This can be viewed as in disagreement with all steps.

When asked about step 5 and how they go about getting approval for a suggested system specification, one participant stated that they typically do not need to conduct this step in the suggested manner. He explained that the lead designer in the design team holds a fair bit of influence at the company. It is therefore rare that anyone would disagree if he has approved a design and his approval is usually given implicitly as he is involved throughout the design process. This demonstrates that depending on proximity from the specification architect to important stakeholders the steps might look different and step five might be unnecessary altogether.

Chapter 7

Discussion

We will now discuss prototyping in the context software development, what effects it may have, and how prototyping can be used to support the requirements engineering process. The foundation for this discussion stems from three research activities. A literature study that maps the area of prototyping. An exploratory case study that investigates the effect of prototyping on output and learning in software development. A focus group that evaluates prototyping as an aid in the requirements engineering process. We draw conclusions by discussing the results from these activities and how they relate to the existing literature on prototyping and requirements engineering.

7.1 Relevant Aspects of Prototyping (RQ1)

The Prototyping Aspects Model (see Chapter 4) depicts four aspects of prototyping; *purpose*, *strategy*, *scope*, and *review method*. This section discusses the relevance and completeness of each aspect, highlights an alternative aspect, and provides an evaluation of the model in its entirety.

Purpose is the first aspect of the Prototyping Aspects Model and it consists of nine categories of reasons for creating a prototype. The aspect was mentioned in 18 out of the 34 articles of the literature study (see Table 6.1) but it was often a byproduct from discussing other areas of prototyping. This is a strong indication that it is an important aspect of prototyping with many facets to consider. In comparison, Lausen introduces three reasons for the use of prototypes in requirements engineering [4]. All of these are covered in the categories of the aspect (see Sections 4.1.1, 4.1.6, and 4.1.8). We assess that the model provides a high level of completeness of the aspect as it covers the elements of requirements engineering and has strong ties to related literature.

Strategy determines the organisation of resources and design decisions to manage uncertainty. It was the least common aspect in the literature study and was only covered in detail in a study by Sigmund A. et al. [5]. Our assessment of *strategy* is that it has a low to moderate level of completeness, primarily based on the work by Sigmund A. et al. While it is the aspect that has the lowest representation in our literature study we believe the number of results indicate that it is a valid concern in prototyping.

Scope is the level of completeness in a prototyping activity. The foundation of this aspect stems from the literature study and an additional layer was added with the exploratory case study. Among the articles from the literature study, some describe *scope* with the term fidelity (see Section 4.3.3) while others highlight that fidelity provides an insufficient level of detail. Michael McCurdy et al. present a comprehensive overview of a proposed continuation of fidelity that would manage the insufficient level of detail [32]. We use this as an inspiration for Sections 4.3.1 and 4.3.2. We depict *scope* as a balance of visual and functional refinement to describe the entire process of a prototyping activity. This definition was evaluated in the exploratory case study. It is our belief that *scope* has as high level of completeness in the model, the only point of contention being the level of detail of visual refinement in Section 4.3.1. We see the amount of literature on the topic and the results from the exploratory case study as a confirmation that *scope* is an integral part of prototyping.

Review method covers the manner in which a prototype is evaluated. The model presents the aspect as having three subcategories of which all were identified through the literature study. Jaskiewicz et al. discuss the effect of iterations on the output and learning [34]. Our assessment is that the model presents the aspect with an acceptable level of completeness, primarily due to the solid foundation in the literature study. We are unable to assign a higher level since we found literature outside the literature study that turned out to be of high relevance to the topic. Among those is an article by Zahra et al. who compared the output and learning from evaluating paper prototypes in different manner [6]. While the study by Zahra et al. puts the completeness in question we believe it is a validation of *review method* being an integral part of prototyping.

What to include and exclude in the Prototyping Aspects Model (see Section 4) has been a continuous topic of debate, both among the authors but also during the focus group. We determined that the model must have a clear focus but also remain within certain boundaries. One cause for debate was whether to include an aspect concerning the choice of media such as paper based or software based prototypes. We determined that the academic material on the topic was not sufficient to draw any conclusions and that one should choose whatever is easier to use. Another cause for debate was the inclusion of a business perspective. The aspect would discuss areas such as how to balance budget constraints or how to evaluate the commercial viability of a product. This was also brought up during the focus group where participants were more inclined to discuss the business perspective of their products. We incorporated some of these elements in *strategy* (see Section 4.2) and otherwise remain steadfast that the boundaries of the model are clear with its current selection of aspects.

The literature study is the foundation of the Prototyping Aspects Model (see Chapter 4) and a valid demographic is therefore an indication of the model's completeness. Our analysis of the demographic of the literature study covers two areas; the year in which an article was published and the country from which its authors originated. It is evident that the distribution over the past 20 years has remained steadfast with a slight uptick in recent years. This matches the general notion that there has been an increasing interest in agile requirements engineering. The geographical distribution has clear signs of bias towards North America and Europe. However this result is aligned with other research on the topic of agile requirements engineering. It is our belief that this model is an accurate representation of the aspects of prototyping and provides a solid foundation for understanding the area of prototyping.

7.2 Output and Learning of Prototyping (RQ2)

In the exploratory case study we used the scope of the Prototyping Aspects Model (see Chapter 4) to create three prototype variants. These were shown to subjects and measured on their output and learning. This section discusses and compares the effects of altering the scope of a prototype and the effect that such alterations have on the output and learning in software development.

Prototype (1) had the same level of visual refinement as Prototype (2) and the same level of functional refinement as Prototype (3). While the balance in refinement was expected to produce balanced feedback the results of Prototype (1) were almost identical to those of Prototype (2), both having a strong inclination towards visual attributes. Zahra et al. conducted a similar experiment where they compared different evaluation techniques using one prototype resulting in the capture of different requirements [6]. This contradicts our findings since we used prototypes that were visually the same and only differed in how they were evaluated but captured the same requirements. We believe that there are two possible explanations for the unexpected result. Either the added functionality did not achieve enough difference or the evaluation of Prototype (2) accidentally gave it a higher level of functionality than intended. Based on the inconsistencies highlighted in this paragraph we have chosen to exclude Prototype (1) from the remainder of this section.

Prototype (2) and Prototype (3) were polar opposites as the former was treated with a high level of visual refinement and the latter with a high level of functional refinement. These differences appear to have had an effect on the perceived attributes of the prototypes. Subjects that experienced Prototype (2) perceived visual and conceptual attributes (see Table 6.2) and stated they had a good understanding of what the software would look like (see Table 6.4). Prototype (3) received less feedback on visual attributes, and was worse at communicating what the system would look like. However, subjects that had experienced Prototype (3) were more inclined to comment on functional attributes than those that had experienced Prototype (2). Prototype (3) was also better than Prototype (2) at communicating what functionality the software would have and how said functionality would be implemented. We assess that an increased level of visual or functional refinement will produce more feedback on whatever attribute is in focus as evident by the perceived attributes of Prototype (2) and Prototype (3). We also believe that a variation in scope has an effect on the prototype's ability to communicate certain areas of the final product.

Prototype variants were evaluated with the same set of three questions and thematic coding of responses produced three categories; *visual*, *functional*, and *conceptual*. The first two categories were anticipated as they were essential for altering the prototypes but conceptual was an unexpected addition. Conceptual comments were common across all prototype variants whether the level of visual or functional refinement was low or high. We believe that this stems from the fact that subjects were overall familiar with the software that was illustrated in the prototypes. This familiarity may have lead to subjects judging the prototypes based on their suitability in the software as a whole and not on the individual solutions they presented. Therefore producing less comments on functional attributes. We believe that all three categories of feedback are important for evaluating a prototype and that further research is needed to understand conceptual feedback and how it relates to the other categories.

The exploratory case study shows that the visual and functional refinement of a prototype has an effect on its output and learning. This means that there exists a combination of attributes that will produce the optimal feedback for a given topic of interest. Among the feedback for a topic of interest are comments surrounding visual, functional, and conceptual attributes. The process for finding a combination is undoubtedly complicated, as is evident from Prototype (1), but we believe that Prototype (2) and Prototype (3) show that it can be done.

7.3 Supporting Agile Development (RQ3)

The Agile Prototyping Guidelines (see Chapter 5) presents a five step process for establishing software requirements with prototypes. The steps have strong ties to the Prototyping Aspects Model (see Chapter 4) and the area of requirements engineering as described by Lausen [4]. This section discusses each step in separate by relating our findings with existing literature followed by an evaluation of the guidelines in their entirety.

Step 1 is an exploratory phase in which to understand what the product shall encompass. In many ways step 1 resembles the phase of elicitation in requirements engineering[4]. A notable difference between the approach of Lausen and ours is that we present step 1 with a set of boundaries to encompass the act of prototyping itself. The evaluation of our guidelines highlight that concept exploration is an essential part of product development, a statement that is reiterated by the focus group. Exploration is a category of purpose and was a thoroughly discussed topic in the Prototyping Aspects Model. The weakest link among the boundaries for this step is the review method. Participants of the focus group highlighted that they evaluated exploratory prototypes in very different manner. We believe that step 1 is a valuable asset in the guidelines but that further research is needed to understand the implications and boundaries for review method.

Step 2 concerns the use of prototyping to understand the relation between product and customer needs. Lausen includes this in his definition of elicitation in requirements engineering [4]. We suggest splitting elicitation into step 1 and step 2 to tackle the differentiation between exploratory and desirability prototyping, a difference that is described in the Prototyping Aspects Model. The focus group suggested that a split was appropriate as the difficulties of finding a solution lay not the solutions themselves but in understanding the needs of the customer. They also supported the choice of strategy and scope but again presented a large variety of alternatives for review methods. Our assessment is that the introduction step 2 in the guidelines adds value to the process but that, as in the previous step, review method requires additional research.

Step 3 presents a first step towards specifying the boundaries within which the product shall exist. Lausen suggests the use of requirements to accomplish this and states that requirements can have different styles such as text, diagrams, or tables [4]. This step also bears some resemblance to the use of prototyping in user-testing as suggested by Lausen [4]. We believe that the primary concern in this phase is not to specify perfect requirements but to create a uniform vision of the product. Prototyping allows for a vague definition of requirements and the prototype can act as a basis for internal and external discussions in group environments. The focus group confirmed that the goal of this step should be communication and not specification as communication is a daunting task in business where work is conducted in groups. Furthermore they highlighted that this is a step that is often revisited during the product development process and that it therefore must be quick and easy to grasp. We believe that the overall feedback on this step shows that communication is an integral part of product development which therefore solidifies the position of step 3 as part of the guidelines.

Step 4 aims to solidify the boundaries of a product in the form of a specification and is a continuation of step 3. Lausen presents a selection of eight criteria that a good specification should adhere to [4]. While a prototype may be easy to modify and present traceable changes it cannot encompass all eight criteria. We believe that this selection of criteria is too strict and requires adaption to fit in agile requirements engineering. To accomplish this, we propose the use of prototyping within the boundaries that step 4 establishes. Participants of the focus group had experience from enterprise business and highlighted that such a specification would be unsuitable in that environment. Telavox is not considered to be an enterprise business and does not have a continuous documentation of their requirements other than prototypes. The results from the focus group are therefore inconclusive as they argue for both sides. We believe there is a niche where prototyping can be used as a means of specification but that it requires further research to identify the niche.

Step 5 confirms the design of the product and should be used to formally confirm the specification with the customer. The use of prototyping as a means for validation is an element of requirements engineering [4]. Whereas Lausen suggests the use of prototypes as a means for validating individual requirements, we propose that prototyping can encompass a specification. Our approach puts more emphasis on the importance of the prototype as it can also act as a basis for communication or a partial product. Participants of the focus group stated that the confirmation of designs was a rather simple process for them, the root cause being that they involved lead designers throughout the product development process. They did however have experience difficulties in communicating with the development team who implement the designs that are confirmed. This highlights a weakness that we believe to be typical for any model as contextual circumstances can lead to significant changes in the process. It is debatable whether a model should conform to all contextual circumstances or present a generalised approach with which users can determine by themselves what adjustments they want to make. We believe that a general approach is better for the guidelines since the contextual circumstances are vast. We further conclude that the strong ties to Lausen's work and experiences from the focus group validate the existence of step 5 in the guidelines.

The use of prototyping is not an unforeseen occurrence in requirements engineering. Steps 1, 2, and 5 of the guidelines resemble the use of prototyping as described by Lausen [4]. Prioritisation was excluded as we did not see a clear fit with the guidelines. We present these guidelines as a means of assisting requirements engineering in software development and have received an overall positive response. The focus group that evaluated the guidelines stated that the guidelines shared many similarities with their own process, a process that has been proven to be effective. Participants questioned whether specifications could properly specify requirements as is necessary in enterprise businesses. We therefore believe that the guidelines in their current form are more suitable for companies with simple products, such as software applications, instead of complex products, such as cars that must follow rules and regulations. The focus group highlighted the use of many review methods but provided no clear answer as to what review method is suitable at what point. We believe that this is a topic that requires further assessment. Our conclusive assessment of the guidelines is that they pose can be a possible addition in the requirements engineering process to assist agile software development.

7.4 Validity threats

The use of three research methods throughout this study implies a wide variety of threats to the validity of each individual method but also the result as a whole. Throughout the research, these threats have been noted and minimised to procure the best possible information for the result and discussion.

The transferablity of results concerns the extent to which results are accurate in other contexts. The prototyping aspects model was constructed using the combined set of published papers and should therefore have good transferability as a whole. However some statements in the prototyping aspect model refer to a single paper in which transferability might be limited, hence some statements might have limited transferability. In regards to the exploratory case study results, a higher risk of low transferability exist due to the nature of the study. All subjects worked at the company in the same role and tested the same product which both reduces external variation as well as transferability. The guidelines were described in a more general and purpose oriented way to allow them to be implemented in a variety of contexts. Whether they are a beneficial way of working is harder to judge as we can only state that a similar process is beneficial at Telavox. Therefore guidelines are believed to have good transferability in terms of being used, but unknown transferability in terms of the results from using them.

7.4.1 Literature study

Literature studies must cover the demographic of the topic that they portray, a concern that is tackled in the previous section, but also balance a wide variety of parameters during the search and provide a streamlined process for pruning irrelevant information. Considerable effort was put into producing a good process for the search itself and it is possible that there were further tweaks which could have produced a better search query or yield more relevant articles.

An initial search on the keywords prototype and prototyping returned 10 to 500 000 results. Such a large number of articles was impossible to manage and required a careful selection of parameters to retain important material while removing that which was irrelevant. The process and reasoning behind the choice of parameters is elaborated on in Section 3.1.2.

The search filter for ACM had been set to only include peer-reviewed papers, offering a degree quality assurance outside of the project. However, our initial process was not adequate for the number of articles without abstracts and the difference in search method between LUB and ACM yielded slightly different results. To resolve both these issues we skimmed every article, resolving the absence of abstracts and aligning our approach one final time.

Title and abstract review work was evenly split between authors and is a potential source of bias. Maintaining a consistent evaluation of titles and abstracts throughout each review was of utmost importance for the quality of the study. To ensure a continuous level we established two separate guidelines that were to be followed for the reviews. We evaluated whether or not we followed these guidelines by reviewing each others results of the first 100 titles and 10 abstracts.

The addition of snowballing after producing the final set of articles was considered in order to obtain more material. A brief test of the strategy produced good results and as the Prototyping Aspects Model (see Chapter 4) provides a foundation for the entire thesis, more information or a lack thereof could have a cascading effect of validity. We ultimately chose not to pursue this addition as time was of essence.

7.4.2 Exploratory case study

The design of a case study establishes a set of independent variables, of which a limited amount is treated, to study the effect on the dependent variable. Such a setup reduces validity threats but by no means eliminates them entirely. In our study the major points of concern were how we communicated with subjects during sessions, how we created prototypes, and whether the demographic would be sufficient to portray a valid image.

To reduce the risk of communication having an effect during sessions we adhered created a welcoming letter that introduced the topic and created a structure for the overall meeting. These instructions were followed by point. Recordings of the interviews also assisted in a continuous validation and evaluation of our technique.

Creating prototypes on your own leads to an inherent risk of the creator having an effect on the outcome of the study. A risk that can be significantly reduced by performing multiple iterations of a given experiment, providing an added benefit of additional data points. Additional data points enabled us to eliminate (or question) potential variance.

The average session with subjects that viewed Prototype (3) was about 20 percent longer than of those who viewed Prototype (1) or Prototype (2). Subjects experienced difficulties with performing the scenarios that were too guide them through the prototype and we therefore occasionally had to assist them in proceeding.

Chapter 8

Conclusions

We have studied the topic of prototyping, its effect on software development, and its ability to assist in the requirements engineering process. This chapter presents the context of this study, what advances we set out to achieve, our fulfilment of these achievements, and their implication on the topic of the study.

Requirements Engineering (RE) provides a systematic approach to determine what a product should achieve. The software industry found that the systematic approach of RE was incompatible with their agile development strategies, leading to the creation of an agile version of RE known as Agile Requirements Engineering (ARE). Prototyping is a popular technique used in both RE and ARE to understand what a product shall accomplish and whether a suggested solution is viable [4]. The description of prototyping in RE is brief at best and fails to describe how a prototype can suit both the elicitation and validation of requirements. Considering these are different stages of an RE process it is reasonable to wonder, what are the aspects of prototyping?

We identified three areas in the domain of prototyping and requirements engineering to address. *What is a meaningful model for aspects in software prototyping?* addresses the issue of literature on the topic of prototyping being vast and covering anything from the development of bookcases to mobile applications (RQ1). *What effect does the scope of a prototype have on the output and learning in software development?* addresses the issue that prototyping is for different contexts but it is unclear how a prototype is adapted to suit these contexts (RQ2). *How can agile development projects be supported in requirements engineering processes concerning prototyping aspects?* addresses how prototyping could be used to assist software development projects in their requirements engineering process.

The Prototyping Aspects Model (see chapter 4) portrays prototyping as having four aspects; *purpose*, *strategy*, *scope*, and *review method*. We conclude that the model portrays the four most important aspects of software prototyping and that each aspect is at a respectable level of completeness (RQ1).

The scope in prototyping encompasses among other things *visual refinement* and *functional refinement*. We conclude that an alteration of the scope in prototyping can both steer feedback towards visual or functional attributes and alter a prototype's ability to communicate some of its areas (RQ2).

The Agile Prototyping Guidelines (see Chapter 5) presents five steps; *concept exploration*, *discover customer needs*, *devise design suggestion*, *test and improve*, and *Confirm design*. We conclude that the guidelines encompass the necessities of a requirements engineering process using aspects of prototyping and that they present a possible approach for supporting agile development projects in non-enterprise-businesses (RQ3).

Requirements engineering and prototyping are a common occurrence across industries. We believe that the advancements of this study can pose a valuable addition in both industry and science. Industry is concerned with optimisation and the ability to alter the scope in prototyping to steer feedback towards certain attributes can reduce prototyping costs and improve prototyping feedback. For non-enterprise-businesses with agile development projects the guidelines can assist in establishing a requirements engineering process which could improve requirements management and documentation. Science may find the consolidation of the prototyping domain and guidelines to be of interest as it could prove to be a valuable asset in agile requirements engineering.

Prototyping is by no means uncharted territory and the advances of this study are but a small step towards understanding its capabilities in product development. The Prototyping Aspects Model is a step towards depicting the aspects of prototyping but its aspects and the model as a whole require further research to confirm their validity. Strategy was the least complete aspect and should be further studied to understand its full implications. Scope was the most complete aspect but the exploratory case study highlighted that we do not fully understand the relation between refinement and feedback. The same relation is relevant for the review method where we received results that contradicted a previous study but could not identify the cause. Review method requires further attention to raise its level of completeness as we found numerous studies that were not caught in the literature study but definitely concerned the aspect. We found that the guidelines might be a valid approach to implement a requirements engineering process in agile development projects but our method of evaluation was limited to a focus group. The guidelines and their relation to the model can be further investigated, especially the review method for steps 1 and 2. In consolidation, an empirical study that focuses on any of the parts of this study would shed light on its strengths and weaknesses while drastically increasing the reliability of each part.

References

[1] Karina Curcio, Tiago Navarro, Andreia Malucelli, and Sheila Reinehr. Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, 139:32 – 50, 2018.

[2] Marja Kapyaho and Marjo Kauppinen. Agile requirements engineering with prototyping: A case study. *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, page 334, 2015.

[3] Stephanie Houde and Charles Hill. What do prototypes prototype? 2, 12 1997.

[4] Soren Lauesen. *Software requirements : styles and techniques.* Addison-Wesley, 2002.

[5] Sigmund A. Tronvoll, Christer W. Elverum, and Torgeir Welo. Prototype experiments: Strategies and trade-offs. *Procedia CIRP*, 60(Complex Systems Engineering and Development Proceedings of the 27th CIRP Design Conference Cranfield University, UK 10th - 12th May 2017):554 – 559, 2017.

[6] C. Lo T. Lan E. Frroku Z. Shakeri Hossein Abad, S. Moazzam and H. Kim. Loud and interactive paper prototyping in requirements elicitation: What is it good for? *2018 IEEE 7th International Workshop on Empirical Requirements Engineering (EmpiRE), Empirical Requirements Engineering (EmpiRE), 2018 IEEE 7th International Workshop on, EMPIRE*, 2018. `http://ieeexplore.ieee.org.ludwig.lub.lu.se/stamp/stamp.jsp?tp=&arnumber=8501347&isnumber=8501338`.

[7] Rashina Hoda, Norsaremah Salleh, and John Grundy. The rise and evolution of agile software development. *IEEE Software, Software, IEEE, IEEE Softw*, 35(5):58 – 63, 2018.

[8] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.

[9] R.F. Ciriello, A. Richter, and G. Schwabe. When prototyping meets storytelling: Practices and malpractices in innovating software firms. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2017 IEEE/ACM 39th International Conference on, ICSE-SEIP*, pages 163 – 172, 2017.

[10] Maria Alexandra Maassen. *Product development models in the IT sector-From Waterfall to Agile Project Management Models in the case of AVIRA SOFT SRL.* Sciendo, 5 2018.

[11] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 12 2014.

[12] Jim Rudd, Ken Stern, and Scott Isensee. Low vs. high-fidelity prototyping debate. *Interactions*, 3(1):76–85, January 1996.

[13] Melin Wenström P. Abrahamsson L. Användning av prototyper som verktyg för kravhantering i agil mjukvaruutveckling, 2018. `http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-150528`.

[14] Gartner. Gartner glossary - u - unified communications as a service (ucaas).

[15] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, page 68–77. BCS Learning & Development Ltd., 2008.

[16] SBU. *Utvärdering av metoder i hälso- och sjukvården och insatser i socialtjänsten: en handbok.* Stockholm: Statens Beredning för Medicinsk Utvärdering (SBU), 3 2017.

[17] Barbara Kitchenham, David Budgen, and Pearl Brereton. The value of mapping studies–a participant-observer case study. *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*, 01 2010.

[18] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[19] Bella Martin and Bruce M. Hanington. *Universal methods of design : 100 ways to research complex problems, develop innovative ideas, and design effective solutions.* Rockport Publishers, 2012.

[20] Youn-Kyung Lim, Erik Stolterman, and Josh Tenenberg. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15(2), July 2008.

[21] Kurt Schneider. Prototypes as assets, not toys: Why and how to extract knowledge from prototypes. In *Proceedings of the 18th International Conference on Software Engineering*, ICSE '96, page 522–531, USA, 1996. IEEE Computer Society.

[22] Steven Dow, Julie Fortuna, Dan Schwartz, Beth Altringer, Daniel Schwartz, and Scott Klemmer. Prototyping dynamics: Sharing multiple designs improves exploration, group rapport, and results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2807–2816, New York, NY, USA, 2011. Association for Computing Machinery.

[23] Horst Lichter, Matthias Schneider-Hufschmidt, and Heinz Züllighoven. Prototyping in industrial software projects—bridging the gap between theory and practice. In *Proceedings of the 15th International Conference on Software Engineering*, ICSE '93, page 221–229, Washington, DC, USA, 1993. IEEE Computer Society Press.

[24] Lukas Zink, Rafael Hostetter, Annette Isabel Bohmer, Udo Lindemann, and Alois Knoll. The use of prototypes within agile product development explorative case study of a makeathon. *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Engineering, Technology and Innovation (ICE/ITMC), 2017 International Conference on*, pages 68 – 77, 2017.

[25] R. Budde and H. Zullighoven. Prototyping revisited. *COMPEURO'90: Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering @ Systems Engineering Aspects of Complex Computerized Systems, CompEuro '90. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering*, pages 418 – 427, 1990.

[26] Ansar-Ul-Haque Yasar. Enhancing experience prototyping by the help of mixed-fidelity prototypes. In *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, Mobility '07, page 468–473, New York, NY, USA, 2007. Association for Computing Machinery.

[27] D.A. Fern and S.E. Donaldson. Tri-cycle: a prototype methodology for advanced software development. *[1989] Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track, System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, 2:377, 1989.

[28] B. Ratcliff. Early and not-so-early prototyping-rationale and tool support. *Proceedings COMPSAC 88: The Twelfth Annual International Computer Software & Applications Conference, Computer Software and Applications Conference, 1988. COMPSAC 88. Proceedings., Twelfth International*, pages 127 – 134, 1988.

[29] F.B. Zainuddin and Liu Shaoying. An approach to low-fidelity prototyping based on sofl informal specification. *2012 19th Asia-Pacific Software Engineering Conference, Software Engineering Conference (APSEC), 2012 19th Asia-Pacific, Asia-Pacific Software Engineering Conference*, 1:654 – 663, 2012.

[30] David G. Hendry, Sara Mackenzie, Ann Kurth, Freya Spielberg, and Jim Larkin. Evaluating paper prototypes on the street. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, page 1447–1450, New York, NY, USA, 2005. Association for Computing Machinery.

[31] Reinhard Sefelin, Manfred Tscheligi, and Verena Giller. Paper prototyping - what is it good for? a comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, page 778–779, New York, NY, USA, 2003. Association for Computing Machinery.

[32] Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, and Alonso Vera. Breaking the fidelity barrier: An examination of our current characterization of proto-

types and an example of a mixed-fidelity success. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 1233–1242, New York, NY, USA, 2006. Association for Computing Machinery.

[33] J. Hakim and T. Spitzer. Effective prototyping for usability. *18th Annual Conference on Computer Documentation. ipcc sigdoc 2000. Technology and Teamwork. Proceedings. IEEE Professional Communication Society International Professional Communication Conference an, Professional Communication Conference, 2000. Proceedings of 2000 Joint IEEE International and 18th Annual Conference on Computer Documentation (IPCC/SIGDOC 2000)*, pages 47 – 54, 2000.

[34] S. Bellomo, R.L. Nord, and I. Ozkaya. Elaboration on an integrated architecture and requirement practice: Prototyping with quality attribute focus. *2013 2nd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks), Twin Peaks of Requirements and Architecture (TwinPeaks), 2013 2nd International Workshop on the*, pages 8 – 13, 2013.

[35] R.E. Fairley and M.J. Willshire. Iterative rework: the good, the bad, and the ugly. *Computer*, 38(9):34 – 41, 2005.

[36] C. Toffolon and S. Dakhli. An iterative meta-lifecycle for software development, evolution and maintenance. *2008 The Third International Conference on Software Engineering Advances, Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on*, pages 284 – 289, 2008.

[37] N. Goldman and K. Narayanaswamy. Software evolution through iterative prototyping. *International Conference on Software Engineering, Software Engineering, 1992. International Conference on*, pages 158 – 172, 1992.

[38] U.A. Raja. Empirical studies of requirements validation techniques. *2009 2nd International Conference on Computer, Control and Communication, Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, pages 1 – 9, 2009.

[39] F. Kordon. Proposal for a generic prototyping approach. *ETFA '94. 1994 IEEE Symposium on Emerging Technologies and Factory Automation. (SEIKEN) Symposium) -Novel Disciplines for the Next Century- Proceedings, Emerging Technologies and Factory Automation, 1994. ETFA '94., IEEE Symposium on*, pages 396 – 403, 1994.

[40] Tomasz Jaskiewicz and Aadjan van der Helm. Unlocking the interactive office: Concurrent prototyping approach. In *Proceedings of the 2018 Designing Interactive Systems Conference*, DIS '18, page 547–558, New York, NY, USA, 2018. Association for Computing Machinery.

Appendices

Appendix A

Literature selection

Literature selection references

[1] T. Arano, C.K. Chang, P. Mongkolwat, Y. Liu, and X. Shu. An object-oriented prototyping approach to system development. *Proceedings of 1993 IEEE 17th International Computer Software and Applications Conference COMPSAC '93, Computer Software and Applications Conference, 1993. COMPSAC 93. Proceedings., Seventeenth Annual International*, pages 56 – 62, 1993.

[2] S. Bellomo, R.L. Nord, and I. Ozkaya. Elaboration on an integrated architecture and requirement practice: Prototyping with quality attribute focus. *2013 2nd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks), Twin Peaks of Requirements and Architecture (TwinPeaks), 2013 2nd International Workshop on the*, pages 8 – 13, 2013.

[3] Pascal Bruegger, Denis Lalanne, Agnes Lisowska, and Béat Hirsbrunner. Tools for designing and prototyping activity-based pervasive applications. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '09, page 129–136, New York, NY, USA, 2009. Association for Computing Machinery.

[4] R. Budde and H. Zullighoven. Prototyping revisited. *COMPEURO'90: Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering @ Systems Engineering Aspects of Complex Computerized Systems, CompEuro '90. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering*, pages 418 – 427, 1990.

[5] F. Cafer and S. Misra. A cognitive requirement specification model. *2009 24th International Symposium on Computer and Information Sciences, Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, pages 518 – 521, 2009.

[6] Chen Chen, A. Porter, and J. Purtilo. Tool support for tailored software prototyping. *Proceedings of 3rd Symposium on Assessments of Quality Software Development Tools, Assessment of Quality Software Development Tools, 1994, Proceedings., Third Symposium on*, pages 171 – 181, 1994.

[7] R.F. Ciriello, A. Richter, and G. Schwabe. When prototyping meets storytelling: Practices and malpractices in innovating software firms. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2017 IEEE/ACM 39th International Conference on, ICSE-SEIP*, pages 163 – 172, 2017.

[8] Jan Derboven, Dries De Roeck, Mathijs Verstraete, David Geerts, Jan Schneider-Barnes, and Kris Luyten. Comparing user interaction with low and high fidelity prototypes of tabletop surfaces. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, page 148–157, New York, NY, USA, 2010. Association for Computing Machinery.

[9] Steven Dow, Julie Fortuna, Dan Schwartz, Beth Altringer, Daniel Schwartz, and Scott Klemmer. Prototyping dynamics: Sharing multiple designs improves exploration, group rapport, and results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2807–2816, New York, NY, USA, 2011. Association for Computing Machinery.

[10] R.E. Fairley and M.J. Willshire. Iterative rework: the good, the bad, and the ugly. *Computer*, 38(9):34 – 41, 2005.

[11] D.A. Fern and S.E. Donaldson. Tri-cycle: a prototype methodology for advanced software development. *[1989] Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track, System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, 2:377, 1989.

[12] N. Goldman and K. Narayanaswamy. Software evolution through iterative prototyping. *International Conference on Software Engineering, Software Engineering, 1992. International Conference on*, pages 158 – 172, 1992.

[13] Catherine Grevet and Eric Gilbert. Piggyback prototyping: Using existing, large-scale social computing systems to prototype new ones. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 4047–4056, New York, NY, USA, 2015. Association for Computing Machinery.

[14] J. Hakim and T. Spitzer. Effective prototyping for usability. *18th Annual Conference on Computer Documentation. ipcc sigdoc 2000. Technology and Teamwork. Proceedings. IEEE Professional Communication Society International Professional Communication Conference an, Professional Communication Conference, 2000. Proceedings of 2000 Joint IEEE International and 18th Annual Conference on Computer Documentation (IPCC/SIGDOC 2000)*, pages 47 – 54, 2000.

[15] K.G. Heisler, W.T. Tsai, and C.V. Ramamoorthy. Integrating the role of requirements specification into the process of prototyping: the protospec. *[1989] Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track, System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, 2:348, 1989.

[16] David G. Hendry, Sara Mackenzie, Ann Kurth, Freya Spielberg, and Jim Larkin. Evaluating paper prototypes on the street. In *CHI '05 Extended Abstracts on Human Factors in*

Computing Systems, CHI EA '05, page 1447–1450, New York, NY, USA, 2005. Association for Computing Machinery.

[17] Rong Huigui, Zhou Ning, Jin Min, and Wu Jiaxin. Research on service-oriented framework of interface prototype driven development. *2008 International Conference on Computer Science and Software Engineering, Computer Science and Software Engineering, 2008 International Conference on*, 2:552 – 557, 2008.

[18] Tomasz Jaskiewicz and Aadjan van der Helm. Unlocking the interactive office: Concurrent prototyping approach. In *Proceedings of the 2018 Designing Interactive Systems Conference*, DIS '18, page 547–558, New York, NY, USA, 2018. Association for Computing Machinery.

[19] F. Kordon. Proposal for a generic prototyping approach. *ETFA '94. 1994 IEEE Symposium on Emerging Technologies and Factory Automation. (SEIKEN) Symposium) -Novel Disciplines for the Next Century- Proceedings, Emerging Technologies and Factory Automation, 1994. ETFA '94., IEEE Symposium on*, pages 396 – 403, 1994.

[20] Horst Lichter, Matthias Schneider-Hufschmidt, and Heinz Züllighoven. Prototyping in industrial software projects—bridging the gap between theory and practice. In *Proceedings of the 15th International Conference on Software Engineering*, ICSE '93, page 221–229, Washington, DC, USA, 1993. IEEE Computer Society Press.

[21] Youn-Kyung Lim, Erik Stolterman, and Josh Tenenberg. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15(2), July 2008.

[22] Linchuan Liu and Peter Khooshabeh. Paper or interactive? a study of prototyping techniques for ubiquitous computing environments. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, page 1030–1031, New York, NY, USA, 2003. Association for Computing Machinery.

[23] Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, and Alonso Vera. Breaking the fidelity barrier: An examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 1233–1242, New York, NY, USA, 2006. Association for Computing Machinery.

[24] Abdul Rahman, Abdel Razek, and Christian van Husen. Innovation by service prototyping design dimensions and attributes, key design aspects, and toolbox. *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Engineering, Technology and Innovation (ICE/ITMC), 2017 International Conference on*, pages 571 – 576, 2017.

[25] U.A. Raja. Empirical studies of requirements validation techniques. *2009 2nd International Conference on Computer, Control and Communication, Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, pages 1 – 9, 2009.

[26] B. Ratcliff. Early and not-so-early prototyping-rationale and tool support. *Proceedings COMPSAC 88: The Twelfth Annual International Computer Software & Applications Con-*

ference, Computer Software and Applications Conference, 1988. COMPSAC 88. Proceedings., Twelfth International, pages 127 – 134, 1988.

[27] Kurt Schneider. Prototypes as assets, not toys: Why and how to extract knowledge from prototypes. In *Proceedings of the 18th International Conference on Software Engineering*, ICSE '96, page 522–531, USA, 1996. IEEE Computer Society.

[28] Reinhard Sefelin, Manfred Tscheligi, and Verena Giller. Paper prototyping - what is it good for? a comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, page 778–779, New York, NY, USA, 2003. Association for Computing Machinery.

[29] C. Toffolon and S. Dakhli. An iterative meta-lifecycle for software development, evolution and maintenance. *2008 The Third International Conference on Software Engineering Advances, Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on*, pages 284 – 289, 2008.

[30] Sigmund A. Tronvoll, Christer W. Elverum, and Torgeir Welo. Prototype experiments: Strategies and trade-offs. *Procedia CIRP*, 60(Complex Systems Engineering and Development Proceedings of the 27th CIRP Design Conference Cranfield University, UK 10th - 12th May 2017):554 – 559, 2017.

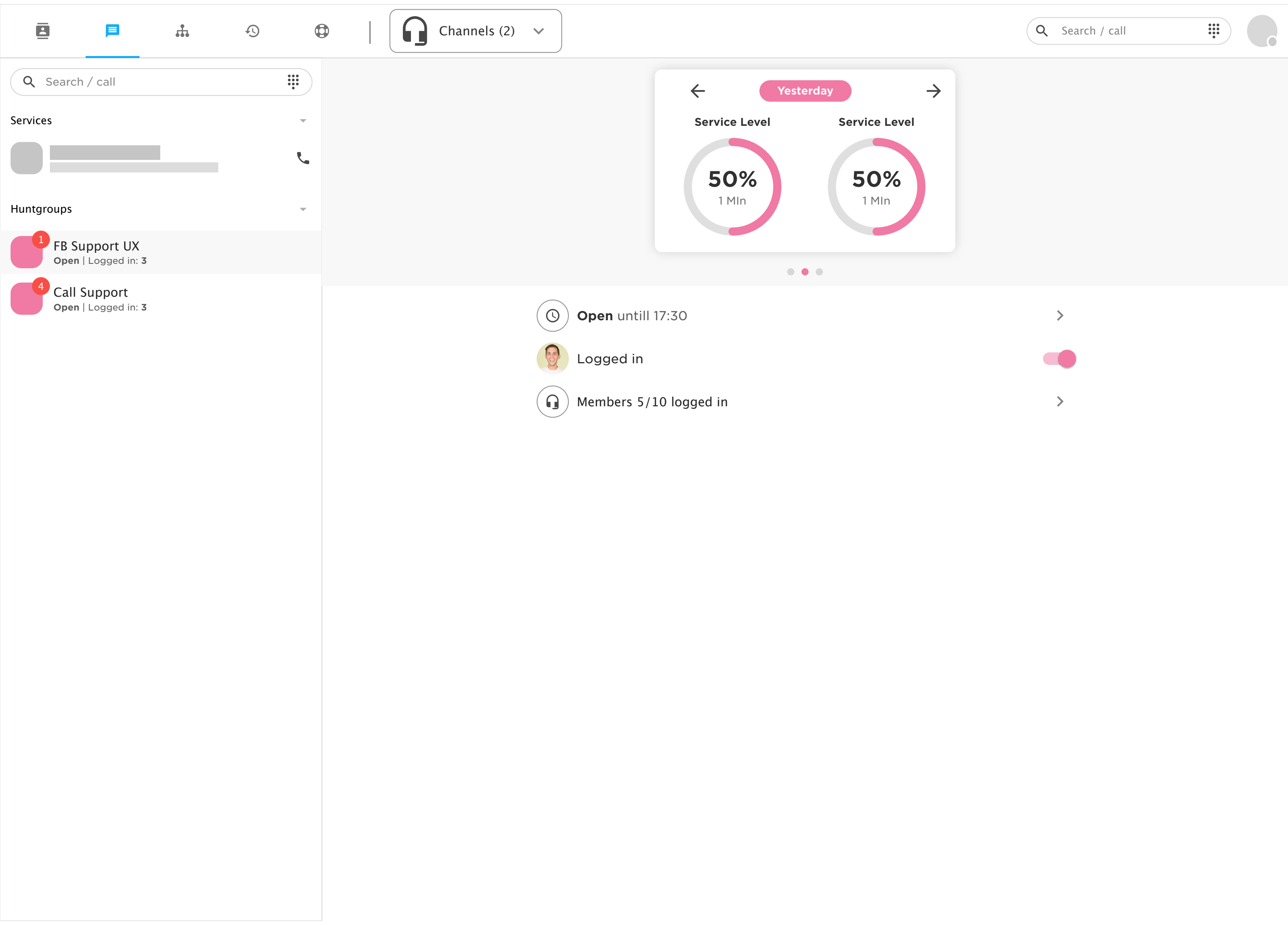
[31] Mikael Wiberg and Erik Stolterman. What makes a prototype novel? a knowledge contribution concern for interaction design research. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, NordiCHI '14, page 531–540, New York, NY, USA, 2014. Association for Computing Machinery.

[32] Ansar-Ul-Haque Yasar. Enhancing experience prototyping by the help of mixed-fidelity prototypes. In *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, Mobility '07, page 468–473, New York, NY, USA, 2007. Association for Computing Machinery.

[33] F.B. Zainuddin and Liu Shaoying. An approach to low-fidelity prototyping based on sofl informal specification. *2012 19th Asia-Pacific Software Engineering Conference, Software Engineering Conference (APSEC), 2012 19th Asia-Pacific, Asia-Pacific Software Engineering Conference*, 1:654 – 663, 2012.

Appendix B

Prototype 1 and 2



Channels (2)

Search / call

Search / call

Services

Huntgroups

1 FB Support UX
Open | Logged in: **3**

4 Call Support
Open | Logged in: **3**

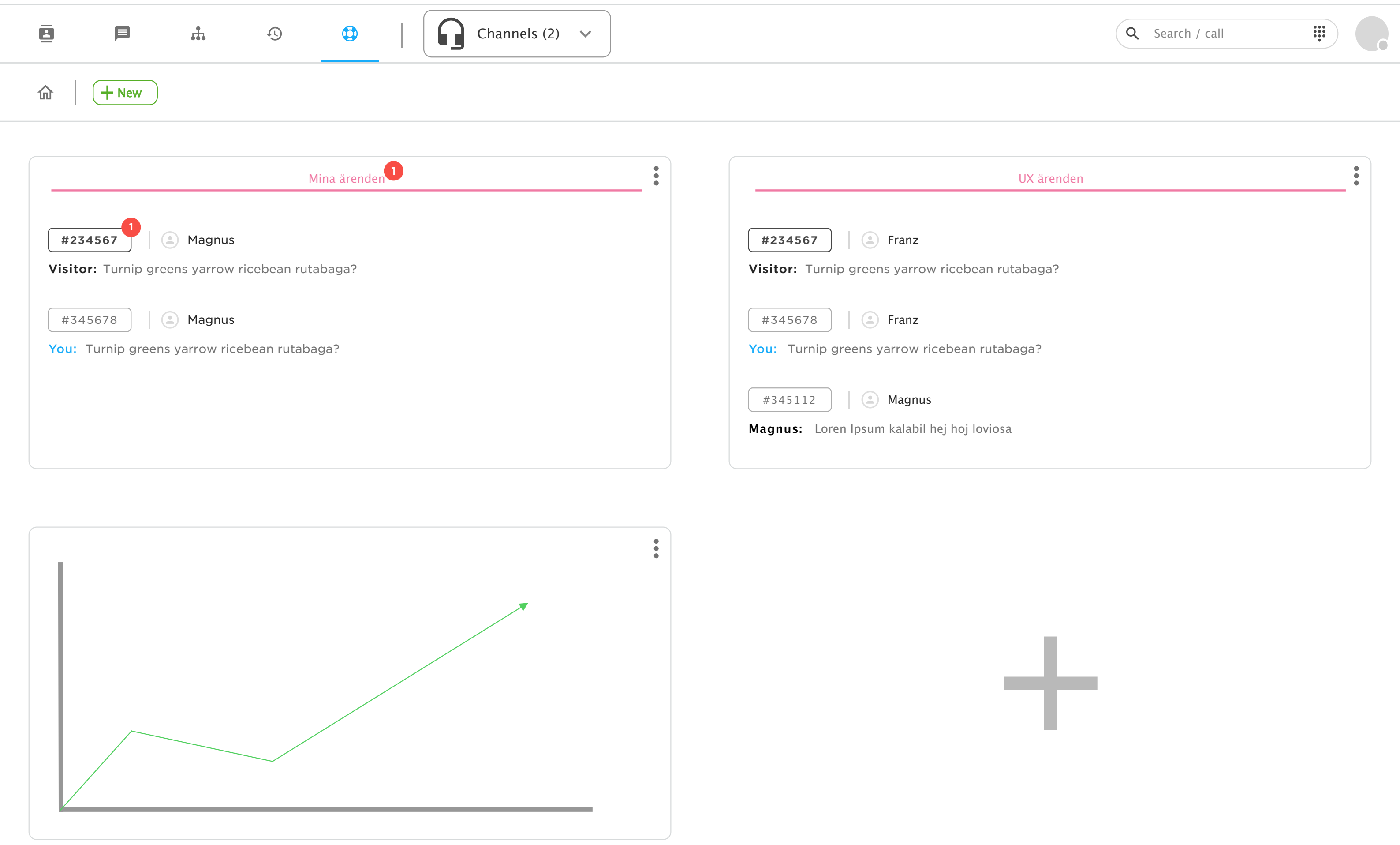
Yesterday

Service Level	Service Level
50%	50%
1 Min	1 Min

Open untill 17:30

Logged in

Members 5/10 logged in



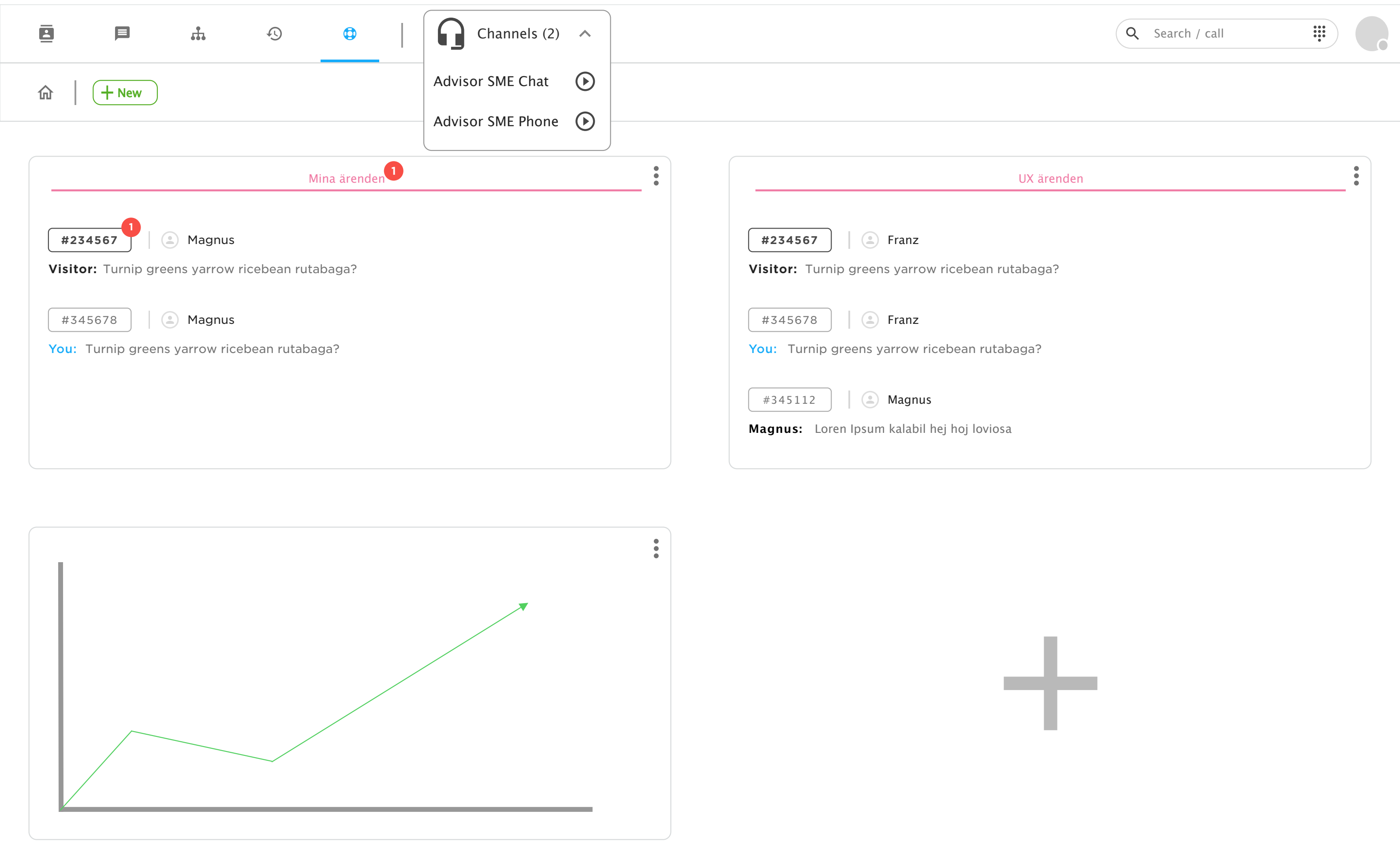
#234567 | Magnus
Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Magnus
You: Turnip greens yarrow ricebean rutabaga?

#234567 | Franz
Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Franz
You: Turnip greens yarrow ricebean rutabaga?

#345112 | Magnus
Magnus: Loren Ipsum kalabil hej hoj loviosa



Mina ärenden 1

#234567 1 | Magnus
Visitor: Turnip greens yarrow ricebean rutabaga?

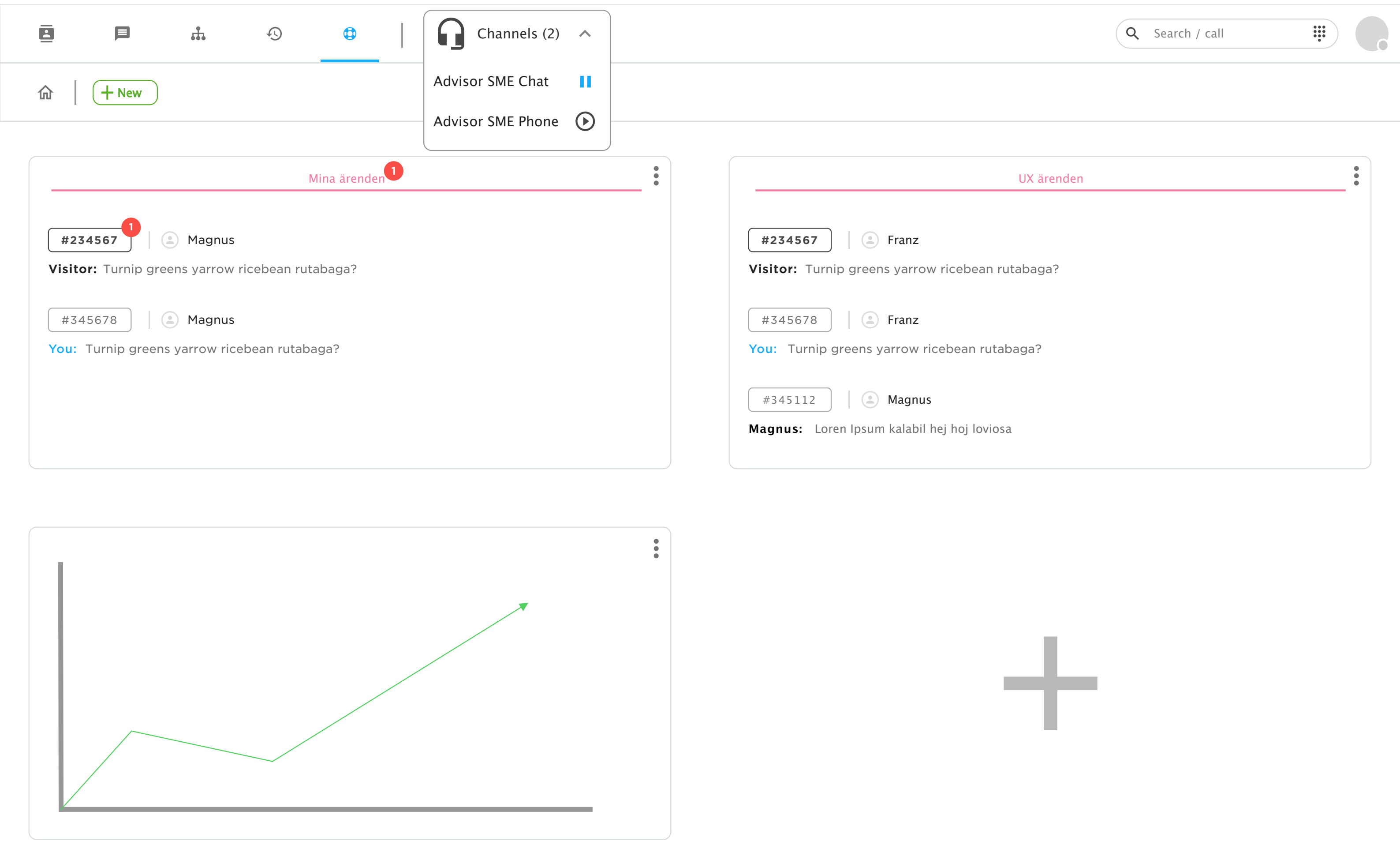
#345678 | Magnus
You: Turnip greens yarrow ricebean rutabaga?

UX ärenden

#234567 | Franz
Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Franz
You: Turnip greens yarrow ricebean rutabaga?

#345112 | Magnus
Magnus: Loren Ipsum kalabil hej hoj loviosa



Mina ärenden 1

#234567 1 | Magnus

Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Magnus

You: Turnip greens yarrow ricebean rutabaga?

UX ärenden

#234567 | Franz

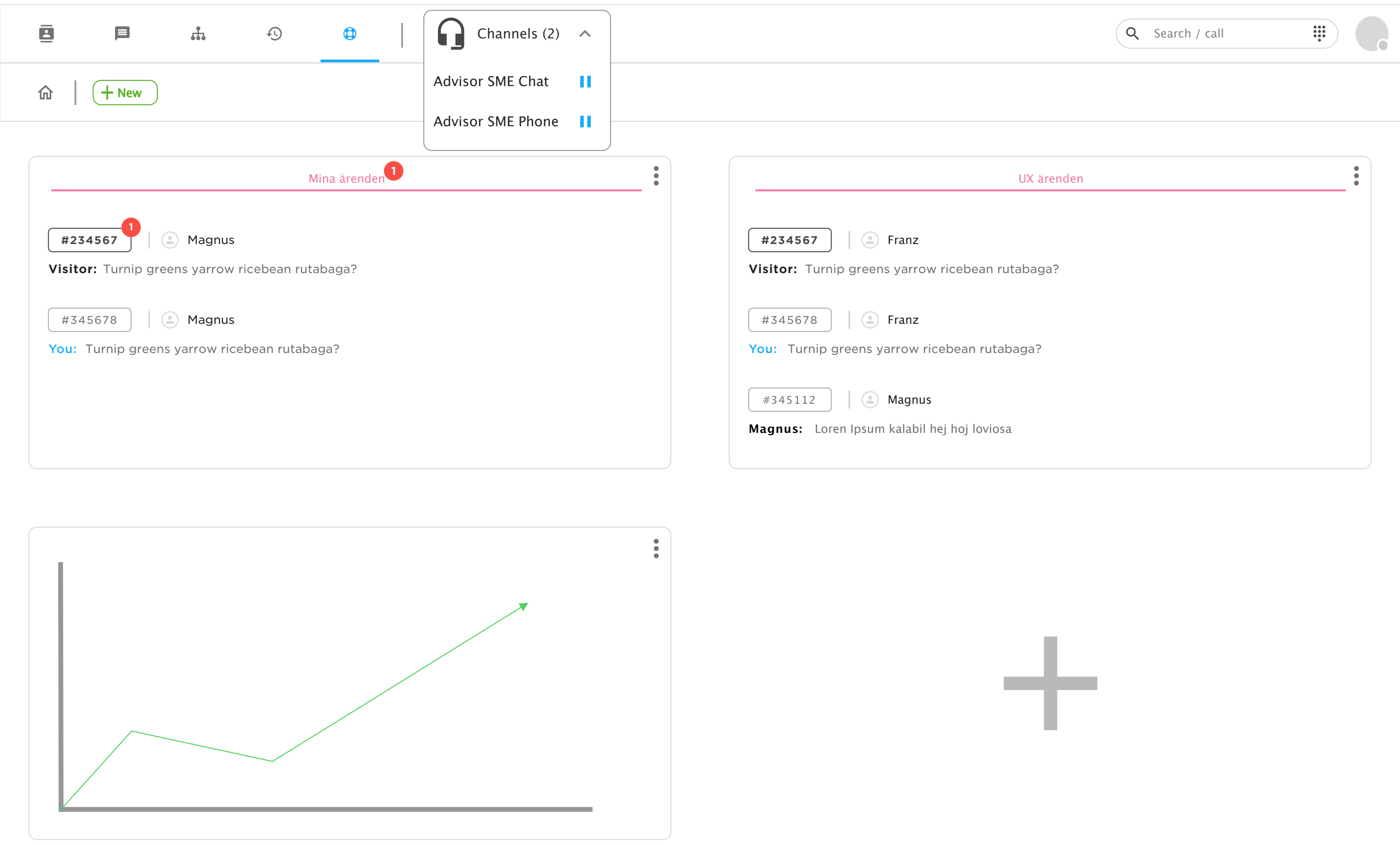
Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Franz

You: Turnip greens yarrow ricebean rutabaga?

#345112 | Magnus

Magnus: Loren Ipsum kalabil hej hoj loviosa



Channels (2)

Advisor SME Chat

Advisor SME Phone

Search / call

New

Mina ärenden 1

#234567 1 | Magnus

Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Magnus

You: Turnip greens yarrow ricebean rutabaga?

UX ärenden

#234567 | Franz

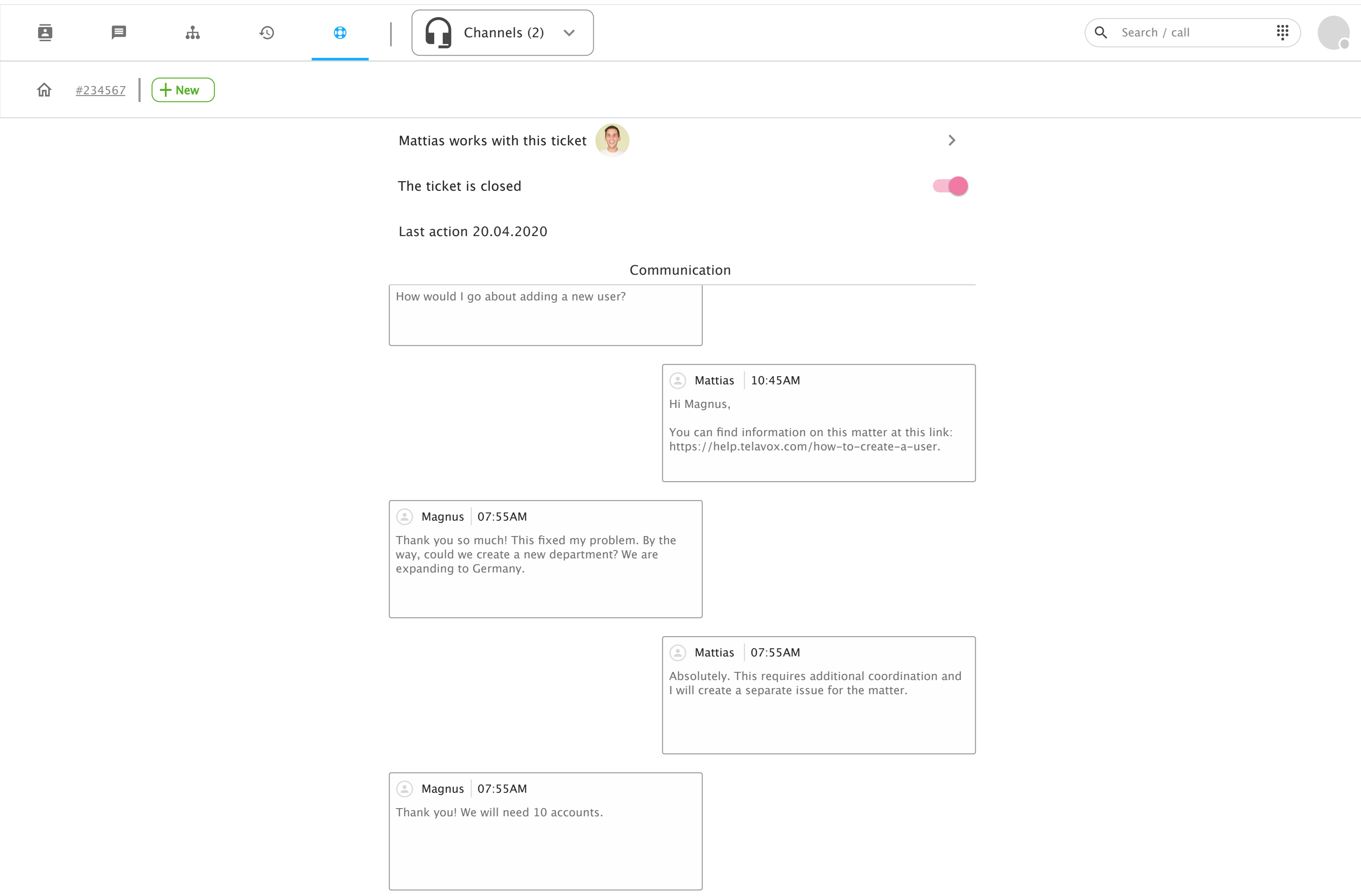
Visitor: Turnip greens yarrow ricebean rutabaga?

#345678 | Franz

You: Turnip greens yarrow ricebean rutabaga?

#345112 | Magnus

Magnus: Loren Ipsum kalabil hej hoj loviosa



Mattias works with this ticket

The ticket is closed

Last action 20.04.2020

Communication

How would I go about adding a new user?

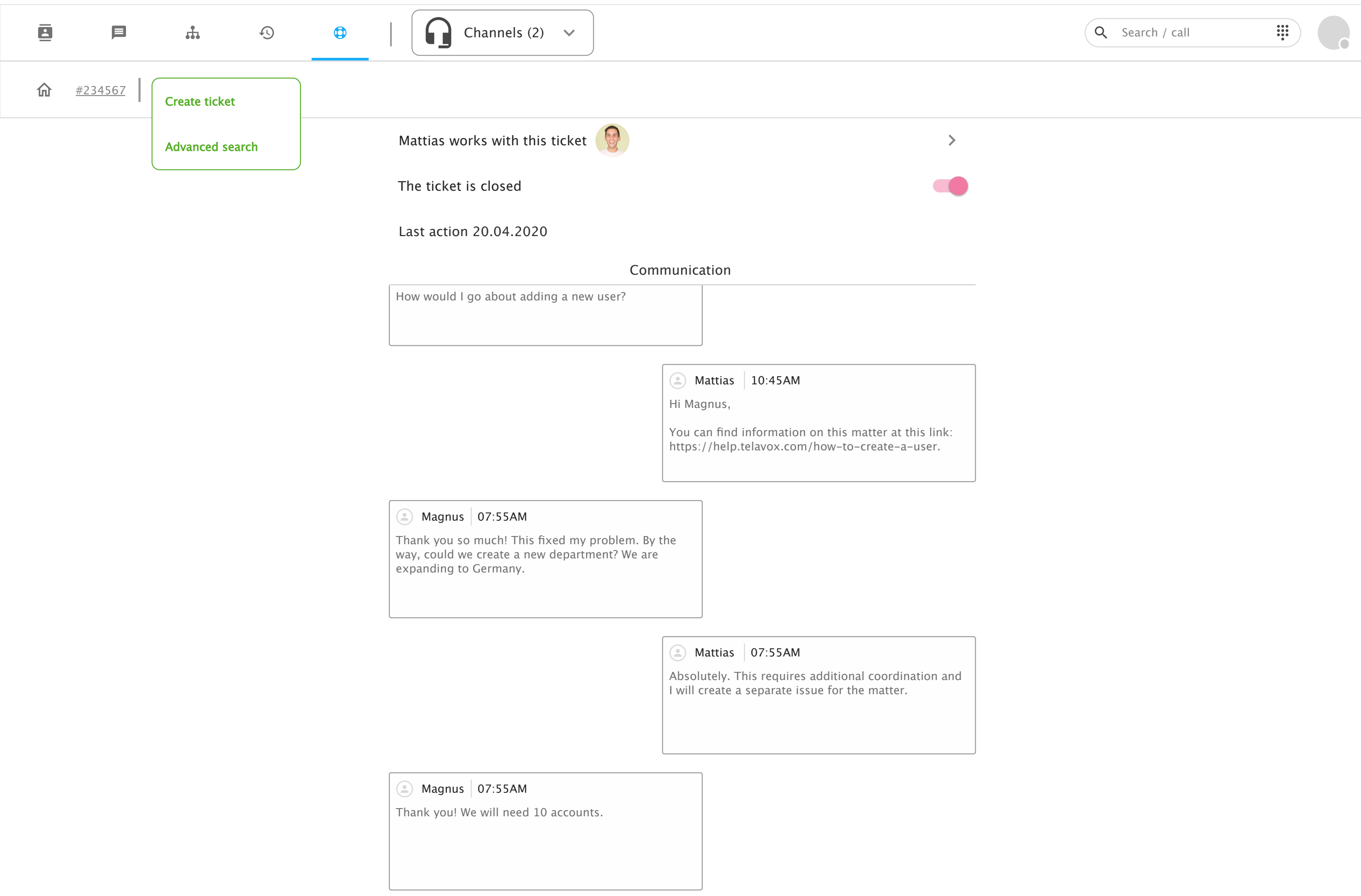
Mattias | 10:45AM
Hi Magnus,

You can find information on this matter at this link: https://help.telavox.com/how-to-create-a-user.

Magnus | 07:55AM
Thank you so much! This fixed my problem. By the way, could we create a new department? We are expanding to Germany.

Mattias | 07:55AM
Absolutely. This requires additional coordination and I will create a separate issue for the matter.

Magnus | 07:55AM
Thank you! We will need 10 accounts.



Mattias works with this ticket

The ticket is closed

Last action 20.04.2020

Communication

How would I go about adding a new user?

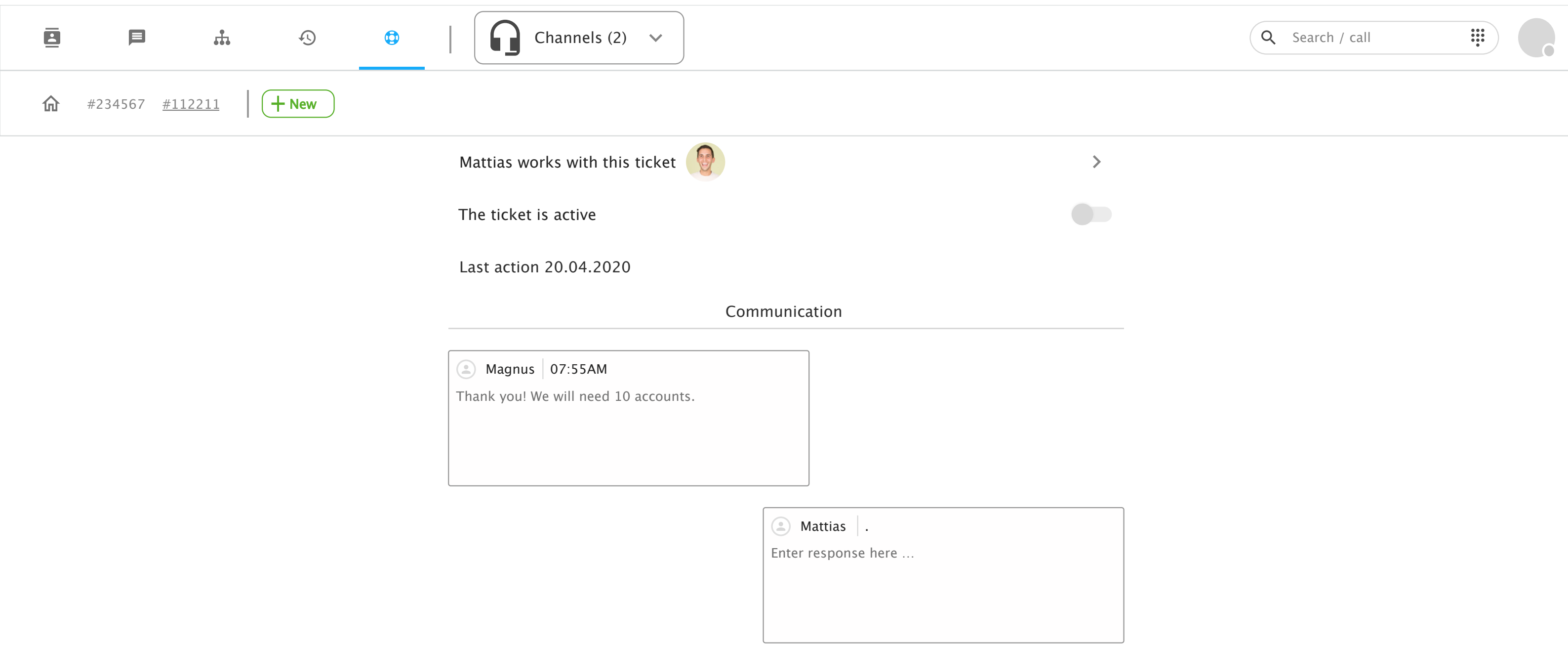
Mattias | 10:45AM
Hi Magnus,

You can find information on this matter at this link: https://help.telavox.com/how-to-create-a-user.

Magnus | 07:55AM
Thank you so much! This fixed my problem. By the way, could we create a new department? We are expanding to Germany.

Mattias | 07:55AM
Absolutely. This requires additional coordination and I will create a separate issue for the matter.

Magnus | 07:55AM
Thank you! We will need 10 accounts.



Mattias works with this ticket

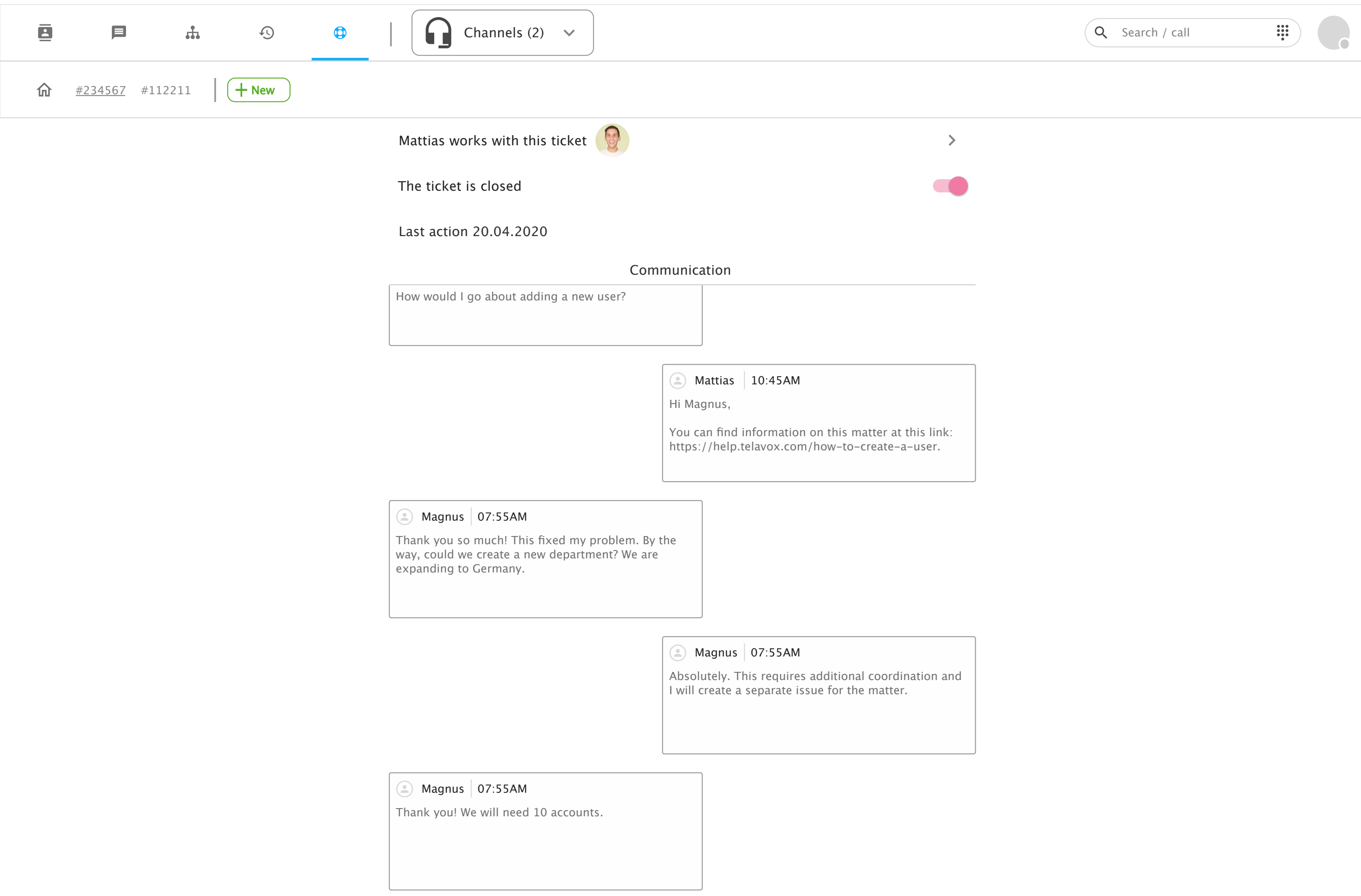
The ticket is active

Last action 20.04.2020

Communication

Magnus | 07:55AM
Thank you! We will need 10 accounts.

Mattias | .
Enter response here ...



Mattias works with this ticket

The ticket is closed

Last action 20.04.2020

Communication

How would I go about adding a new user?

Mattias | 10:45AM

Hi Magnus,

You can find information on this matter at this link: https://help.telavox.com/how-to-create-a-user.

Magnus | 07:55AM

Thank you so much! This fixed my problem. By the way, could we create a new department? We are expanding to Germany.

Magnus | 07:55AM

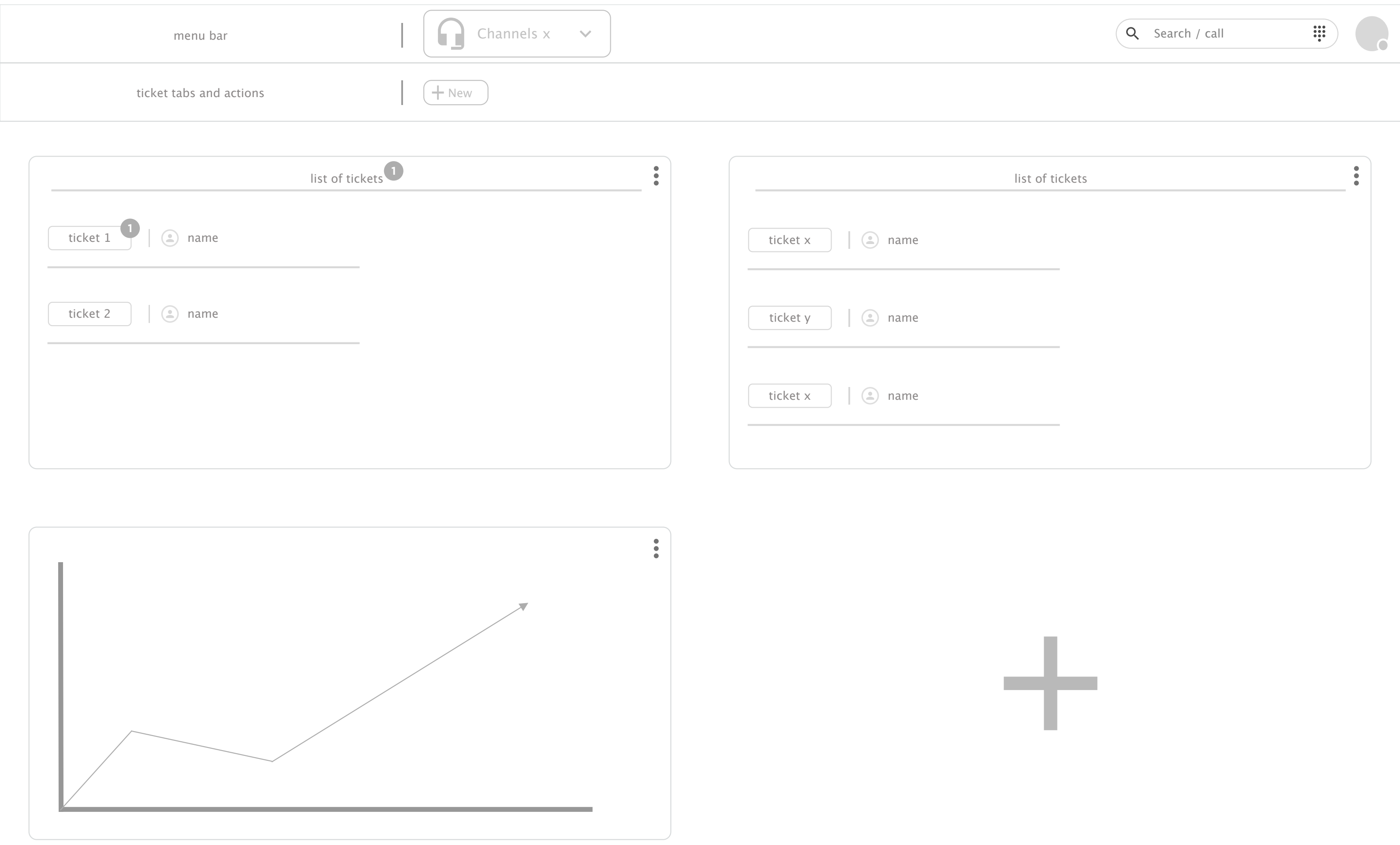
Absolutely. This requires additional coordination and I will create a separate issue for the matter.

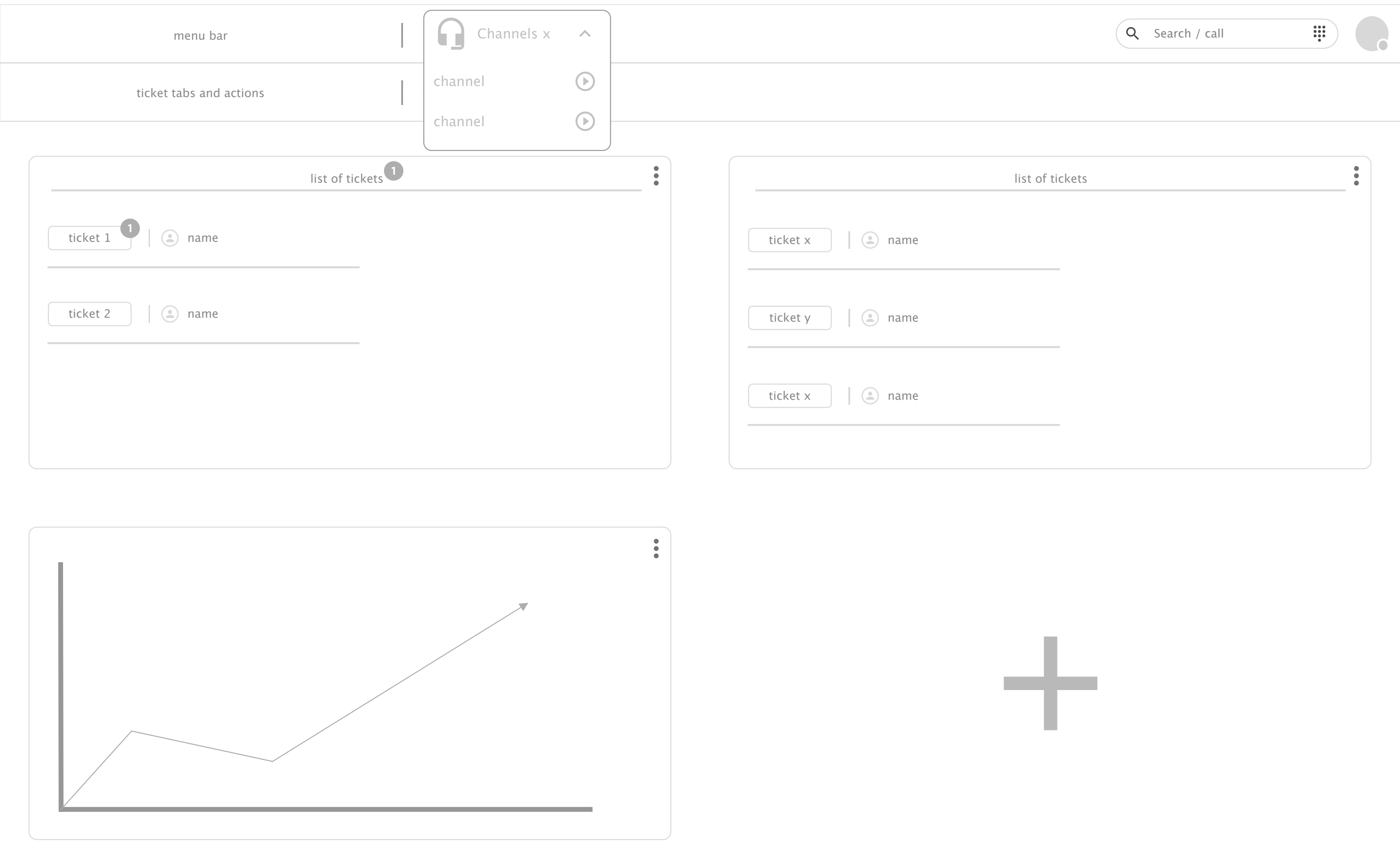
Magnus | 07:55AM

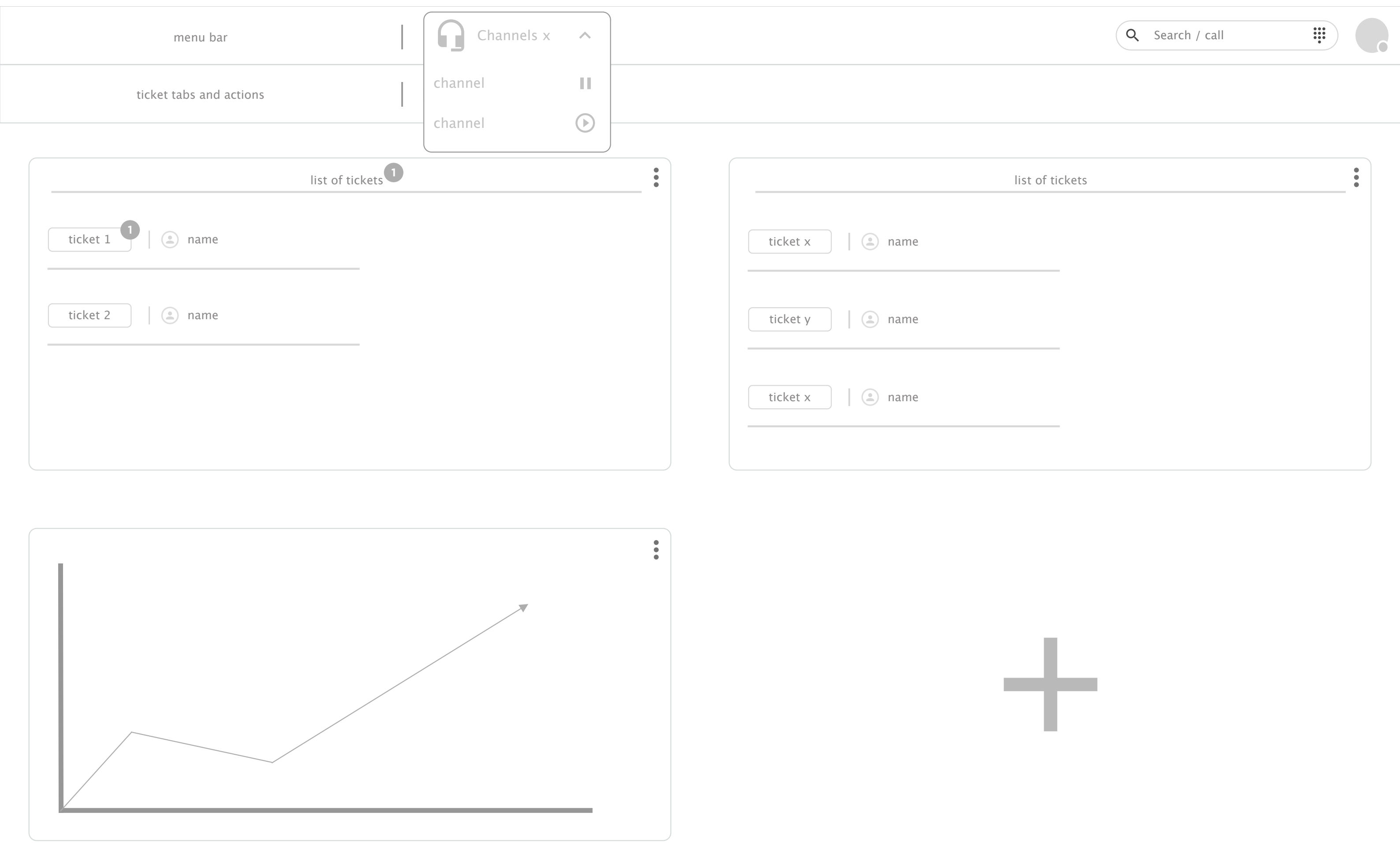
Thank you! We will need 10 accounts.

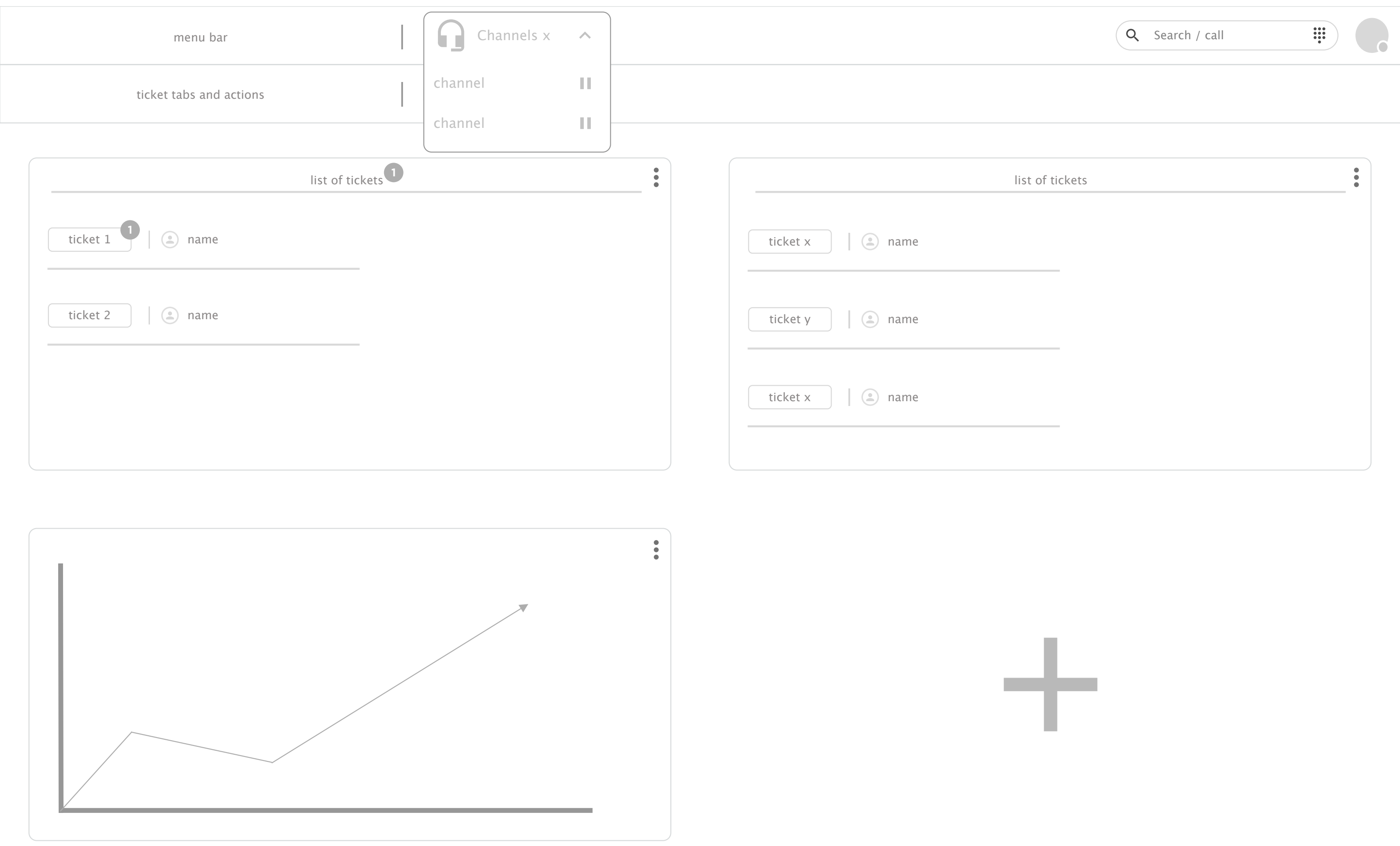
Appendix C

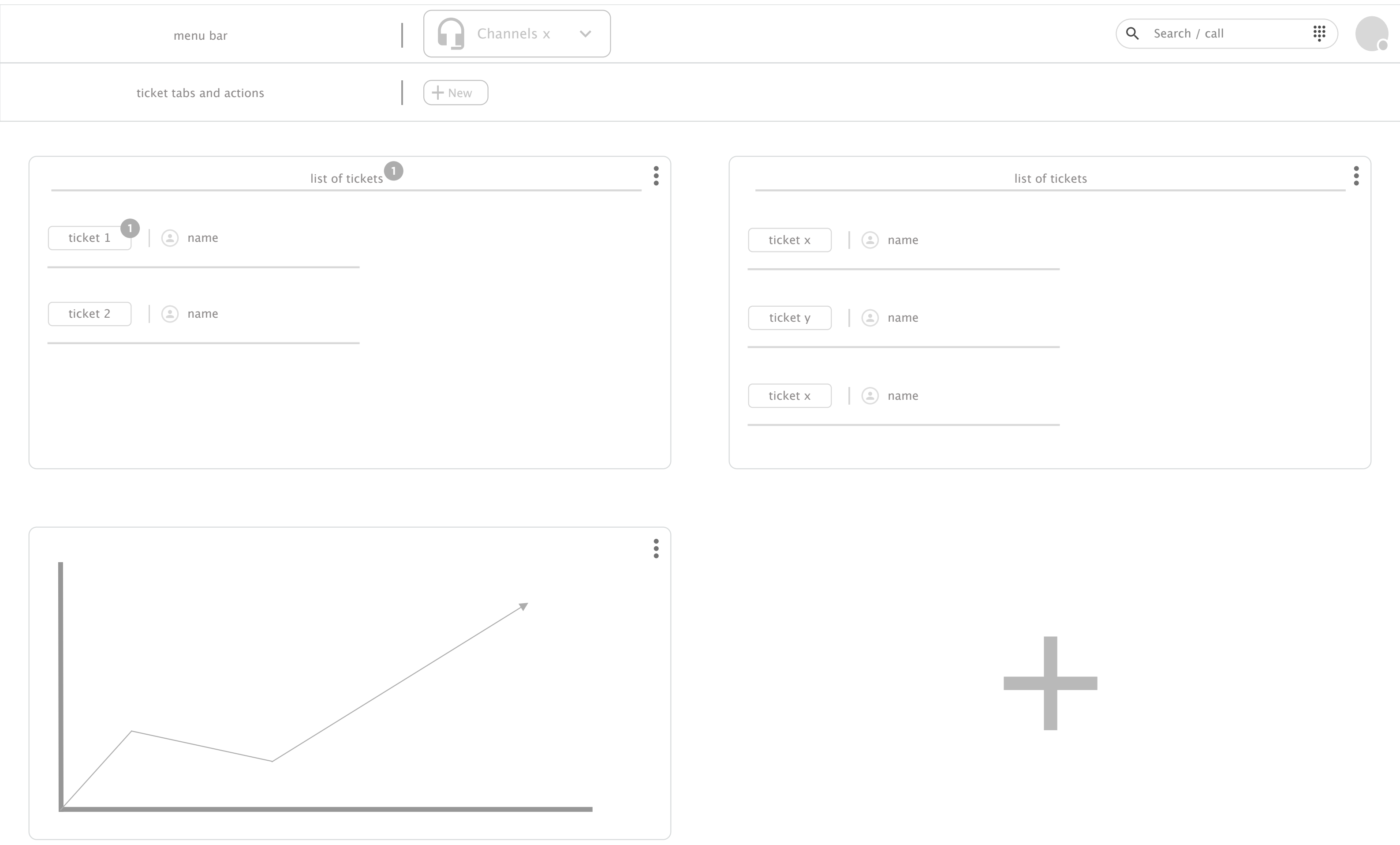
Prototype 3

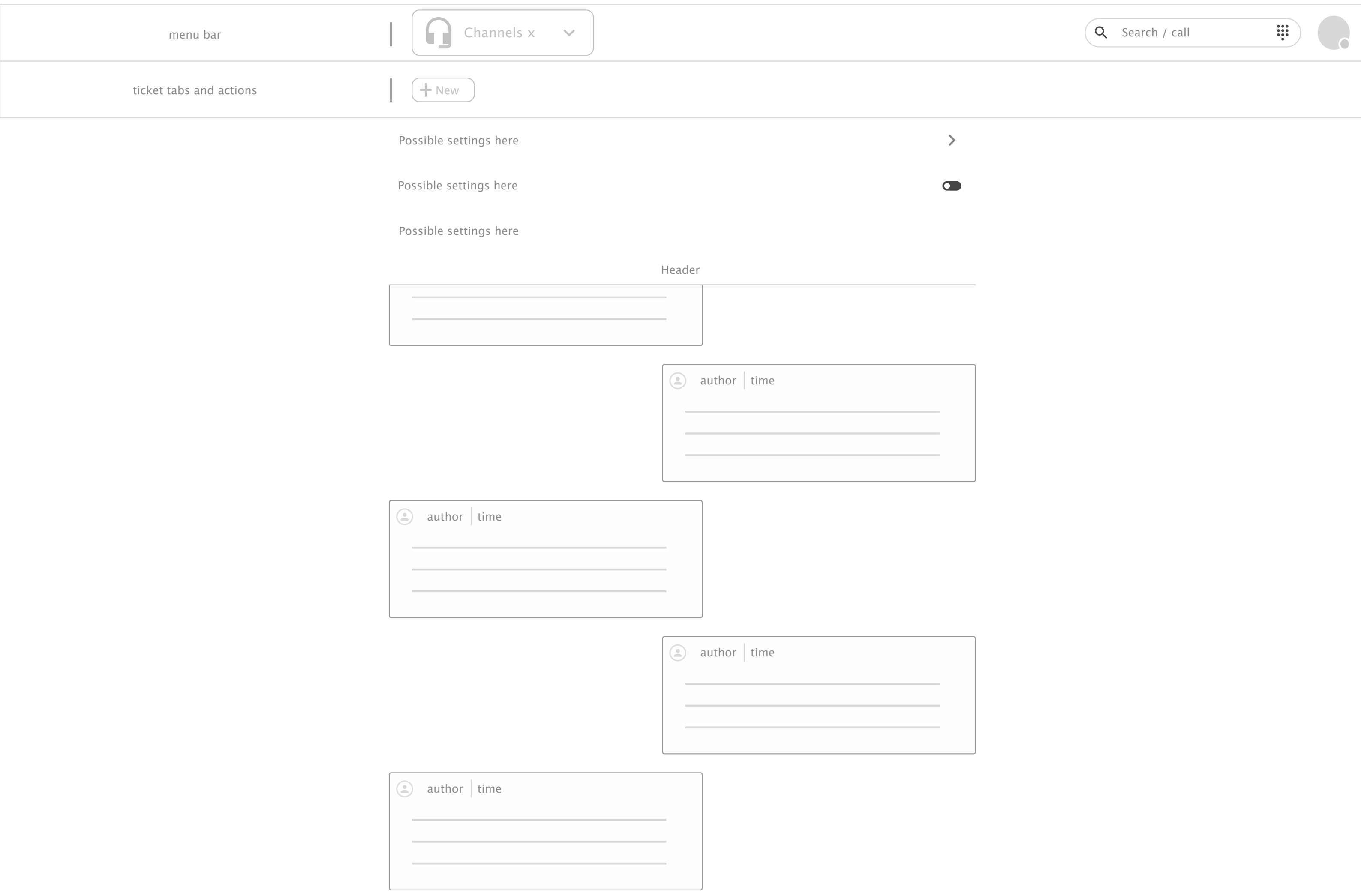

menu bar
Channels x
Search / call
ticket tabs and actions
New
list of tickets
1
ticket 1
1
name
ticket 2
name
list of tickets
ticket x
name
ticket y
name
ticket x
name

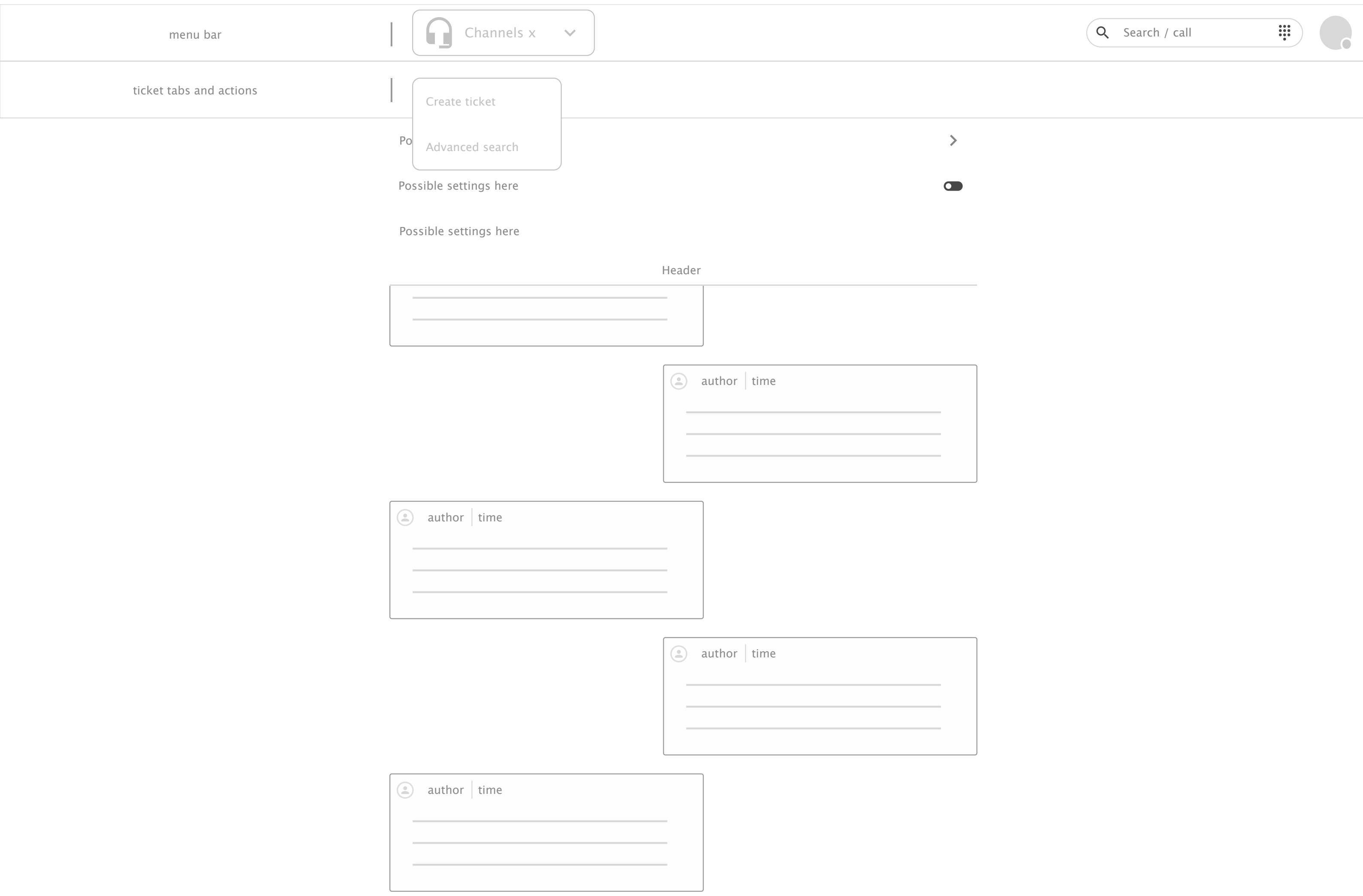

menu bar
Channels x
channel
channel
Search / call
ticket tabs and actions
list of tickets
1
ticket 1
1
name
ticket 2
name
list of tickets
ticket x
name
ticket y
name
ticket x
name

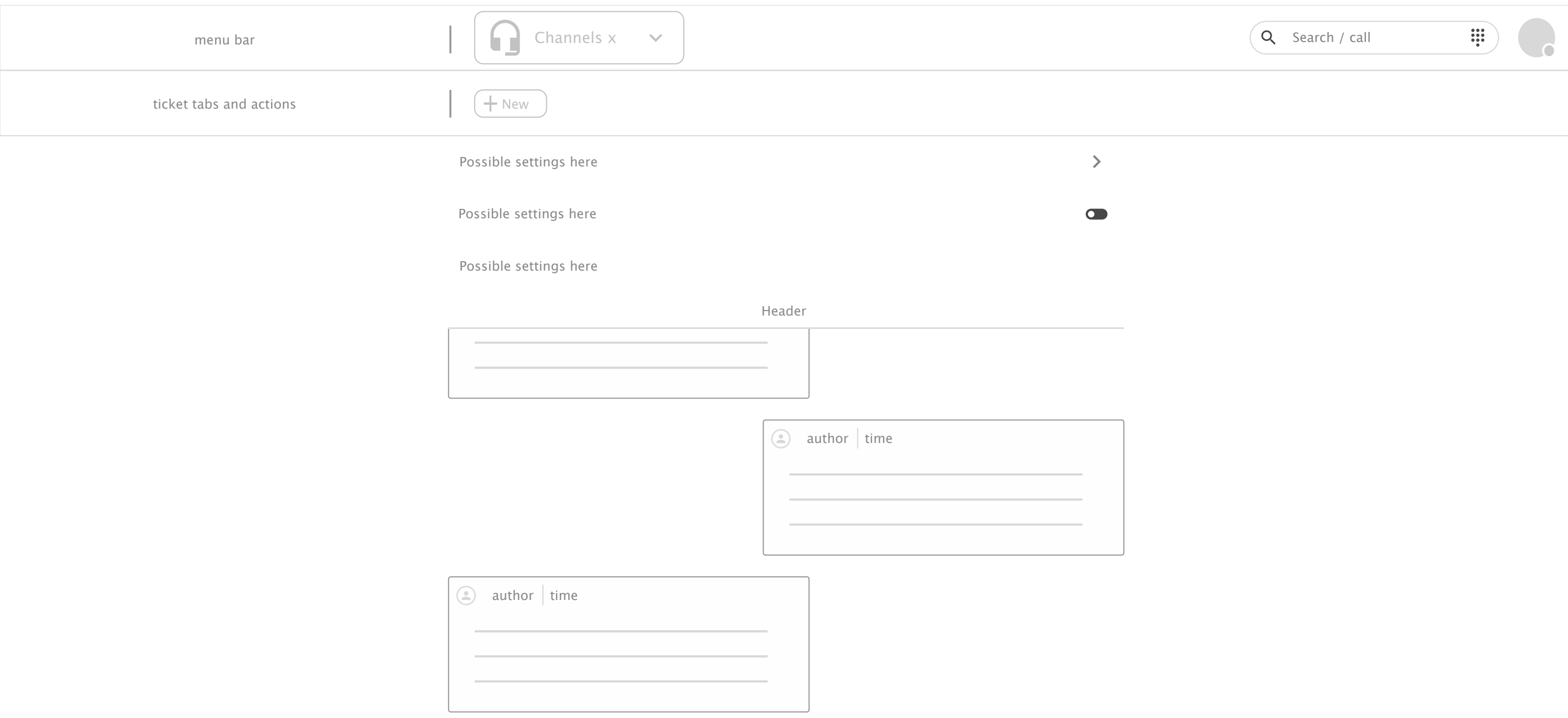

menu bar
ticket tabs and actions
Channels x
channel
channel
Search / call
list of tickets
1
ticket 1
1
name
ticket 2
name
list of tickets
ticket x
name
ticket y
name
ticket x
name


menu bar
ticket tabs and actions
Channels x
channel
channel
Search / call
list of tickets
1
ticket 1
1
name
ticket 2
name
list of tickets
ticket x
name
ticket y
name
ticket x
name


menu bar
Channels x
Search / call
ticket tabs and actions
New
list of tickets
1
ticket 1
1
name
ticket 2
name
list of tickets
ticket x
name
ticket y
name
ticket x
name


menu bar
Channels x
Search / call
ticket tabs and actions
New
Possible settings here
Possible settings here
Possible settings here
Header
author
time
author
time
author
time
author
time


menu bar
Channels x
Search / call
ticket tabs and actions
Create ticket
Advanced search
Po
Possible settings here
Possible settings here
Header
author
time
author
time
author
time
author
time


menu bar
Channels x
Search / call
ticket tabs and actions
New
Possible settings here
Possible settings here
Possible settings here
Header
author
time
author
time

Appendix D

Survey

Prototypenkät

Den här enkäten består av 14 frågor, varav 6 är textsvar och 8 är kryssfrågor. På början av varje sida står instruktioner. Ställ gärna frågor närsomhelst om något är oklart. Tack för din medverkan!

*Required

Skip to question 1Skip to question 1

Frågor om prototypen

Nedan följer några frågor om prototypen. Skriv ner det du kommer att tänka på, stort som litet. Du kan skriva ditt svar som löptext eller punktlista.

1. Vad var bra?

2. Vad var dåligt?

3. Hur kan man förbättra systemet?

Påståenden om prototypen

Kryssa för det alternativet som bäst stämmer överens med din upplevelse

4. Det var enkelt att bilda en uppfattning om vilka funktioner systemet kommer att ha *

Mark only one oval.

	1	2	3	4	5	
Instämmer inte alls	⬭	⬭	⬭	⬭	⬭	Instämmer helt

5. Det var enkelt att bilda en uppfattning hur funktionerna kommer att fungera *

Mark only one oval.

	1	2	3	4	5	
Instämmer inte alls	⬭	⬭	⬭	⬭	⬭	Instämmer helt

6. Det var enkelt att bilda en uppfattning om hur system kommer att se ut *

Mark only one oval.

	1	2	3	4	5	
Instämmer inte alls	⬭	⬭	⬭	⬭	⬭	Instämmer helt

7. Det hade känts jobbigt att föreslå stora ändringar på systemet *

Mark only one oval.

	1	2	3	4	5	
Instämmer inte alls	⬭	⬭	⬭	⬭	⬭	Instämmer helt

8. Det var enkelt att föreställa sig andra sätt systemet hade kunnat vara designat *

Mark only one oval.

	1	2	3	4	5	
Instämmer inte alls	⬭	⬭	⬭	⬭	⬭	Instämmer helt

Upplevelse som Telavoxkund

Föreställ dig nu att du vore att du är en kund till Telavox och vill använda det här systemet för att hantera din kundtjänst.

9. Finns det några behov som inte täcks av systemet? Vilka?

10. Vilka är dom viktigaste behoven du har i ett sånt här system? Uppfylls dom?

11. Finns det saker i systemet du hade varit tveksam till? Vad?

12. Hur väl tycker du produkten passar i Telavoxs profil?

Mark only one oval.

	1	2	3	4	5	
Passar inte in	◯	◯	◯	◯	◯	Passar in

Din bakgrund

Vi är nyfikna på din bakgrund och erfarenhet inom ditt område.

13. Hur många år har du arbetat inom ditt område?

Mark only one oval.

- ◯ 0-1 år
- ◯ 1-3 år
- ◯ 3-5 år
- ◯ 5-10 år
- ◯ 10+ år

14. Jag är anställd som

Mark only one oval.

- ◯ Advisor
- ◯ Developer
- ◯ Sales
- ◯ Other: ____________________

Appendix E

Talking points focus group

Generella diskussionfrågor för alla steg

- Vad är det ett bra/dåligt resultat i detta steget?
- Hur når man bra resultat i detta steg?
- Vad är tydligt/otydligt med det här steget?
- Vad behöver man ha för kunskap för att detta steget ska bli bra?
- Vilka personer borde vara inblandade i detta steget för att det ska bli bra?

Diskussionsfrågor specifika för olika steg

Steg 1

- Vilka är dom bästa sätten att brainstorma ideér till ett nytt system? För/nackdelar ?
- Vilka för/nackdelar finns det med att designern inte funderat på designmöjligheter innan dom pratar med en kund om ett framtida system?

Steg 2

- I det här steget kan en enkel papperskiss vara ett hjälpmedel. Vad kan man lära sig från diskussioner om sådana skisser? Vad kan man inte lära sig med dom?
- Några artiklar påstår att det ofta upplevs som jobbigt att presentera prototyper tidigt i en designprocess för diskussion, håller ni med och vad ser ni för nackdelar/fördelar med det?

Steg 3

- Krav finns i både traditionella och agila mjukvaruprojekt, men hanteras olika. I traditionell kravhantering dokumenteras krav i en Software Requirement Specification (SRS), ofta en excel-lista. Upplever ni att det ibland finns behovet av ett sånt dokument i agila projekt? Hur löser man det?
- När man som designer försöker skapa sig själv en bild av hur implementationen ska se ut, vad är bästa sättet att dokumentera sina tankar?

Steg 4

- Hur tidigt bör man testa sin prototyp?

Steg 5

- När behöver man genomföra en mer formell utvärdering av sin design?
- Vad skiljer sig på att utföra en mer formell utvärdering och på att ha en kollega till att testa?

Appendix F

Focus group transcribed meeting notes

Transcription

Participants
P1 - Participant 1 (Anonymized)
P2 - Participant 2 (Anonymized)
P3 - Participant 3 (Anonymized)
P4 - Participant 4 (Anonymized)
P5 - Participant 5 (Anonymized)
A - Alexander (Moderator)
F - Franz (Moderator)

INTRODUCTORY TALK

1. A: Before we start we'd like to ask for your permission to record this session. We will then transcribe that recording, we will not use your names or anything and we could possibly anonymize as required to not point towards any specific company or so. Is that okay with everyone?
2. Participants: yes that's okay
3. P3 yes, as long as I can ramble as much as possible
4. A haha, yes that's totally okay. F can you tell me when the recording is up and running?
5. F it is now started!
6. P4 Is the thought that we should all answer these questions individually?
7. A Ah, no my thought was that we could carry a group discussion surrounding the questions, but obviously one by one. You do not need to answer all the questions. Look at the suggestion questions as a buffet from which you can pick the most interesting topics. If you come up with something more interesting to bring up that's not covered by the questions feel free to bring that up. My thought is that we will use a speaker list to help. I have P1 at the top of the speaker list. My thought is that she can pick one of the questions to discuss for the first step. She can also choose to speak freely and then everybody can chime in and add their thoughts
8. P3 I have a question surrounding the steps first. Is the thought behind this that you can go from step five back to step one, like if you realise the idea is not good enough from a business perspective, do you need to go all the way back or can you iterate between the steps?
9. A we imagine that you can go backwards , maybe something wasn't sufficiently identified or so in step 2. Great question.
10. P1 Perhaps I missed something at the start, but for whom is the model intended ?
11. A Like the user of this model?
12. P1 yeah who would be the person using these steps?
13. A we imagine that somebody driving development can be the user of the steps. It could be a developer. It could be someone working within a design team. It could also be someone working as a requirements engineer. It could even be a manager. Essentially someone who is driving the development.

14. P1 Ah okey, and this is intended as a method for them to use?
15. A yes exactly, it's a suggestion for how to use prototypes and follow (the steps).
16. A what else should we tell you about it… another thing is that projects might have varying complexity, in some projects, for instance your own application, you probably know a lot about the product already. But we've tried to make this model general. But I was definitely thinking that, we are very curious to hear your opinions and have you be able to build on eachother's answers', so the suggested talking point list is only intended to help you get started, for when we are discussing a certain question, and Franz and I will act as moderators. But if there's not any question that's particularly appealing for you P1, what are like your general thoughts on this?
    STEP 1
17. P1 I was thinking about something, like to me, this is a weird process because it goes from top to bottom. I usually rather think of it as the design gets narrower and more specific, the funnel gets smaller. Ok so concept exploration. You haven't written anything here about research, like analyzing competitors, that's something I imagine maybe should be a part of this. In that step I usually also have some sort of inspirational phase.
18. A what could such a phase look like?
19. P1 I can look at what we have previously, and look at what competitors have done. I collect this material and then I can use the best parts. That's material I collect at the start and go back to look at later, and during this point I can get a pretty good picture of what I wanna do and what direction I want the design to go. But then obviously you move onto the next step pretty quick. Okay let me have another look at the questions. What's a good or a bad result during this step.. Hmm.. What kind of knowledge you need, I'm not sure you need any particular knowledge. The more you work with design the more you know what works for you. It's difficult to define a method that you say you have to follow and that should work for everyone.
20. A Ah, yes very good reflection
21. P1 Like for us within the team, we work with having a personal sound board with each other, and we have some people involved throughout the process
22. A very interesting
23. P3 In a lot of ways, the JIRA ticket reflects this step. Like the design space exploration is often defined in the JIRA
24. A Can you tell us a bit about that?
25. P3 I'm probably the worst person to talk about that because I work the least with JIRA, but, the goal is to formulate the problem and define a background
26. P2 In the best of worlds that's what we have, I have a couple of JIRA tickets at the moment that are completely blank. I agree with Elizabeth that before you start exploring ideas and design space, you need to do some sort of exercise where you get to know the current product and use of it, so you start building understanding before you start thinking about solutions. I also think it's interesting, like what is a prototype? What defines a prototype? You mention lo-fi prototypes and paper prototyping. We could maybe discuss, what is a prototype? Does it need to be simulated and do you need to be able to click and navigate between different environments?

27. A it’s a very good question. We use a wide definition and consider that also a simple sketch on paper is a prototype. An artboard could also be a prototype. But to ask a related question. The first prototype you make, is that usually a sketch or an actual system?
28. P1 We rarely make any prototypes at that stage, it takes a lot of time and is difficult usually. At this point it’s more about understanding the task and create a common view of what you are meant to do.
29. P2 In practice, if you take theoretical guidelines of how a design process is supposed to look you need to keep some distance from starting to design too early. You shouldn’t start designing until you have gotten a good grasp of the problem. If you start designing too early you quickly accumulate ideas about what you concretely wanna do, and you put those ideas on paper. And at this stage you need to make sure you don’t settle into a specific design too much, you need to be able to kill your darlings. At this point I wouldn’t call it a prototype, I would call it a sketch
30. P5 I agree with the previous speaker. in this stage, what’s good and bad in this step. It’s good if you consider all directions, a bad result is getting locked into a certain path and not considering all options.
31. P3 For me, sketching is a way to understand the problem. I’ll start thinking about things like commissions
32. P2 I do the same thing, I think visually and I use sketches to start sorting ideas in my head and mapping the problem, it’s a good help and it sort of fits in with your suggested step. You kind of use it to organize and explore the design space.
33. P4 Yeah I agree, and I think a lot has to do with what you define as a prototype. I usually start by using the JIRA to produce some sort of requirement specification. Then I start sketching. I try to throw a wide net with many different options. You had one question in this document about when to start testing. I usually start showing it within the team early but also show it to some users that we are closely connected to, you could view that as some sort of early testing and viewing
34. P1 Exactly, that is a kind of feedback
35. A Do you usually show the first sketches you produce to anyone?
36. P4 Well sometimes, but usually within the team and not with users
37. P1 Yes likewise, but it can be difficult to have discussions without something to show. Like P4 said you try to throw a wide net in the beginning which makes it easier for others to give feedback about what they think are valid directions
38. P3 It’s a very difficult thing though to show an incomplete sketch to someone who doesn’t understand that it’s an incomplete sketch, that’s why it is so beneficial to show it within our own design team. Even if you ask someone else to not look at certain aspects, they still might give you feedback about font size and that kind of things
39. A very interesting. And as a later question, what are the options to get inspired and brainstorm? What do you do to generate ideas?
40. P2 it kind of depends, sometimes it’s about finding a good structure for an information hierarchy. This is a typical case for menus. Then you can create bullet lists in order to

start finding some sort of structure, bullet lists are really good to use for a lot of different things.

41. A Interesting
42. P4 You also at this point should gather some information from the different stakeholders, get some idea of what they want and conduct some sort of startup. Collect their thoughts and ideas during that startup.
43. P3 I was thinking about this morning, I was talking about if you click this then that happens.. But when don't have something to show and you have to describe with words like, if you do this then that happens, it can't get very complicated because it gets too complicated to follow.
44. A Okay very good. I feel like we have gotten great feedback for this first step, and propose we move on unless someone has something they are very eager to add or discuss?
45. P2 sounds good

STEP 2

46. A So to recap quickly on this step, this step is about understanding the customer needs. I have P5 as the next speaker on the speaker list. If you have any thoughts on this P5 you are welcome to share
47. P5 we talked a bit about eliciting the requirements. Like how you can distill the requirements and find what sticks. An important thing is that it's usually very easy to get feedback on what's good in a system. And things can also be shot down even though they are good, because the tester didn't understand the purpose of the thing. P3 was discussing earlier about thinking about what feedback you want, and I think you need to structure the test so you get feedback on the right. You don't wanna just get feedback on whether the system looks good or ugly. It's good to frame the test in order to get the right feedback and be clear about what kind of feedback you are trying to get. Does anyone wanna add on this?
48. P2 I'm thinking about one of the questions in the talking, "is it difficult to present prototypes early?" it can be, and sometimes we work with wireframes and avoid having pixel perfect prototypes. This can be a good solution and help you get the right feedback you are looking for.
49. P3 I'd like to add that this depends a lot on the problem. It's easy to model and say that you will follow a textbook on how to approach a problem, if you need to design a button then maybe none of these steps apply. Then you hastily assemble some suggestion and throw them out there
50. P2 I don't totally agree, I think you could follow this step in that scenario too
51. P3 That's a good point, but I'm not sure, maybe you don't need to use paper sketches to understand the problem
52. A What's the more difficult thing to understand, is it the user problem or what kind of look the user wants the system to have?
53. P3 We don't really discuss with the users very often unfortunately
54. P2 Yepp we kind of mess it up there right away!
55. *Laughter*

56. P1 I do talk to some customer proxies sometime
57. P3 that can happen, but often you have to run into the best possible customer proxy to get the right feedback. And often they have a biased view towards their own situation. Talking with just one person isn't enough
58. P1 Yeah. But we don't really show them bad or good design, it becomes some sort of interview. We ask them how they use a certain feature today. And we say, if we were to add this thing, where would you place it? And you fish for their ideas. And if you do have an interactive prototype, then I don't show them until the end of the interview. I don't really expect them to comment on the way the design is
59. A Did I understand you correctly, you save the prototype for the end of the meeting?
60. P1 That's correct, I never show it at the start
61. A How come?
62. P1 well I wanna be able to understand how they use the current solution, and they can describe what they do. And then I say, this is the problem you presented to me and here is a solution I have in mind. And then we can have a discussion about whether we ticked all the boxes and properly solved the problem. And then we talk about if it's a good match.
63. P2 This is usually two separate meetings or checkups. Like we were discussing, our customer proxies are usually the ones we talk to. And the first meeting is used to investigate what the problem is. What is the problem that needs to be solved, what do users experience as the problem? Why do they not use a certain feature? And you find out questions about whatever whatever, And you ask questions to find out about the problem. And then you get back to them maybe a week later and do some sort of update. And then you go back to your corner and try to solve it. But this is something that would happen before step 1
64. A Ah I see, you would throw the steps around a bit
65. P2 Yes exactly
66. A And do you others also recognize that you wait a bit before prototyping. That you do some research first before anything?
67. P5 yes exactly, but maybe that is something that happens as a step zero. And you in step zero do some research
68. P2 absolutely. The step we are discussing now, "discover customer needs". But never use prototypes to discuss customer needs, prototypes don't come into play this early. It depends though now that I think if it… if it is a very loosely defined project, then it can ofcourse help to have some prototype to create discussion about their desires, that's the way we did it in the terminator project. We showed some sketches and that helped us understand what we were discussing and we framed what we should do. But in the more common and simpler projects, we don't really use prototypes that early
69. P3 In general, if you ask someone what they think about a certain functionality or how at the look at something, then there's a chance you get a answer that they think you want, but if you wanna get an answer that's more accurate, then a prototype can help you see what they actually do. And then you can talk about what was done. That can be a better foundation to base a discussion on.

70. A Yes I understand what you mean
71. P2 I'm not following at all
72. *laughter*
73. P2 but yeah, you can use a prototype and observe what they do, but you can use the current system and see what they do. How is that different using the prototype instead of the current system?
74. P3 no that's true you can use the current system and that's a different research method. But if you just talk to people that can lead to some "dirty" data
75. P2 absolutely
76. P3 it's not always totally kosher to use that from a research perspective. Then a prototype can help you get better data.
77. P2 Ah okey. And you were talking a bit about what the questions are that you're asking. If you ask a customer proxy, how should we design this page, then you get faulty answers. But if you just try and figure out what the problem is, then it's easier to get good answers. The more solution-agnostic kind of questions you ask, the better data you can get
78. P3 yeah
79. P1 Did you say solution-agnostic?
80. P2 Yeah
81. P1 that is not a word I've heard recently..
82. P2 well yeah…. Do you wanna hear it one more time ?
83. *laughter*
84. P3 throw that straight into the thesis report!
85. P2 yes straight into the report
86. A haha yes that was the first time I've heard that. I thought you provided some very interesting answers for step 2. I'm thinking we can move on to step 3.

STEP 3

87. A so in this step I imagine that we start formulating a suggestion for a solution. And I have P2 on top of the speaker list for this step.
88. P2 So if I look back at some of my most recent work, I've done more things related to this. I often work with invision to build something. I build a prototype, sometimes several variants of it, and bring that into a meeting with stakeholders and collect ideas about how to proceed and what path to choose. So in that sense I feel like my work very much matches the text presented in this step
89. A Ah okey. And is the first prototype the most difficult or the last one more difficult?
90. P2 It's not always a given that you build a final prototype. It depends on if the prototype is a deliverable or if it's just intended to support the discussion. I usually imagine sketches as more like, just pictures. And the hardest thing.. Hmm.. I'm actually not sure, it depends, there's usually a lot of feedback after you review and find that some things don't work and need to be improved before the final delivery. I can't really say.
91. P1 It can also depend on the tool that you are working with, like depending on how you've created the system and what you can do in the tool, it can be difficult to change

stuff in the prototype. And maybe you don't have sketches to base it on, then it can take some time to create the basics. In that sense the first prototype can take a lot of time.

92. P2 For me it's usually not the tools that's the issue, that always works itself out eventually. Sometimes it's easier or more difficult but you solve it somehow. For me the difficulty as at the creative level. If you've developed the first version of a prototype then some requests show and they want you to add things, and that's challenging to fit in sometimes with the concept you have to mind. You have to change the pretty world that you have built up, you need to go back and reset to go back. That's when the compromising starts and that part is always difficult.
93. A very interesting
94. F how do you look at adding this extra step of adding invision, what is the background to adding invision (making it interactive)
95. P4 Usually if a problem is a bit more complex that makes it more difficult to just keep in static flows. Before we started using Invision you were sitting down doing wireframe documents with page references, you say that if you wanna do that then it's these pages that show how it's done, that's at least how I did it some years ago. But now with invision you can remove that complexity and build an interactive flow. Hmm, I forgot where I was going with this.. But as soon as it's more complex, you get value out of being able to click around in an interactive prototype?
96. P3 I think it's a great way to connect to the primal brain, look to feel how it works… P4 you showed me something this morning about expanding or minimizing, showing how the arrows should be.. With invision it's easier to trick yourself and feel like it's real
97. P1 but that wasn't invision that was used
98. P3 Yeah
99. P4 yeah you can…
100. P3 yeah what I mean…
101. P2 but yeah about invision, we need something to show interactive prototype. It can be Invision or it could be flash, or even old HTML. You can prototype in several different ways. Personally, I don't care if it's invision or some other tool. I'm tool agnostic
102. *laughter*
103. P2 you can use that too. But I think we wouldn't do a good job if we weren't able to make interactive prototypes. When you study, like if it's something as easy as clicking in google slides… as long as i've been a designer i've had a way of building a prototype.
104. P5 yes. But often the purpose is not specifying, but rather on communicating. Like when you can't explain with excel or with 10.000 screenshots.
105. P4 but it can be a timesaver, to disagree with you. Like I might make an invision for myself just to get an overview. But I agree with p2, any tool can work.
106. A to connect this a bit to requirements engineering, in this step requirement specification are often used, typically long spreadsheets, I'm not sure you've seen one
107. P2 Absolutely
108. P5 At Company C (changed for anonymity)
109. P2 absolutely at Company C!
110. *laughter*

111. A what is good or bad with specification as alternative to prototypes?
112. P2 I don't really see them as comparable.. Do you mean as a specification? Like usually you are provided with a specification when you are going into the design project, and the prototype is an output of the design process, so I don't see it as comparable. But I'm one of those people that like requirement specification lists
113. *laughter*
114. P2 then you can use that specification and check off boxes. It's good for your own sanity and feeling like you've done what you need to
115. P3 but isn't a shortcoming of these specifications that you don't know why something is a problem?
116. P2 yes, but that's not the point of a specification. That work has been done before you receive the specification. The specification doesn't solve everything, it just lists what the design needs to do. But you need to assume that the proper groundwork has been done, a specification is useless without proper groundwork. Like if they didn't understand the problem from the beginning.
117. A okay, very interesting! Franz, do you have anything you'd like to ask about?
118. F nothing to add!
119. A Okay, in that case, let's move on to step 4.

STEP 4

120. A Okey, so surrounding this step. I feel like we already discussed it a bit earlier, but I feel like it would be interesting to focus on the aspect of iterative improvement with user testing in this step. On the speaker list I have Maria for this one
121. P3 How fortunate that I get more room to talk!
122. *laughter*
123. P3 I think that, surrounding the question of how early you should test your prototype, I think it's interesting to think about what testing your prototype means. Because the first line you draw, you are testing the system for yourself. So even the first sketch you make you have rejected a 100 other options. Because for each decision you have moved things, removed stuff and made a different product. It's like when I send emails, for about a third of the mails I try to send, I realise when typing that I already have a way to solve the issue I'm writing about
124. A Okey. And if you find yourself stuck with a problem, do you ever create a prototype to just kind of throw the problem out there? Do you ever use testing in that capacity?
125. P2 Yeah sometimes
126. P3 Naaah
127. *Laughter*
128. P2 But yeah sometimes, it depends on if we are viewing a prototype as a simple sketch. If I get an idea, most recently this happened Friday, I tested that idea with a developer and asked what they thought about what I had in mind. It can be alot easier to do it that way rather than put it down in words. Sometimes you don't know if you have a good or bad idea and it's useful to send it off to find out.
129. A Oh okey. And related to that, does it also happen that they solve the issue for you
130. P2 Yeah, that can happen

131. P3 Yeah sometimes
132. A How many times do you test a given design?
133. P3 What is testing in this case?
134. A Like, just having a user try it
135. P5 For us, with each other in the design team all the time
136. P2 Something I think we are bad at is reaching out to a potential end-user and having them test the system. For me that is something more than just asking a customer proxy to click through something and ask for feedback. But as stated before, we test the system with eachothers and some
137. P1 The CEO sometime
138. P2 Yes even the CEO sometime
139. P5 Something else we do is roll out stuff gradually and beta test it with users. We don't roll out everything at once and the test is done with actual users, and that is a powerful way of doing it. And another thing we do sometimes is call in students and newly hired people and have them sit in a conference room, testing a system and using scenarios. We maybe do that two times a year. Beta-rollout is something we do all the time
140. P2 Yeah, but I am not sure if we're always that great at following up during beta-testing. We talk a lot about getting better. We talk about getting better at metrics.
141. P5 But we often get the feedback from lead tech and the (development) teams themself follow up crashes and so. So it's not only us that's building it. There are other parties involved that take a lot of responsibility
142. A And if the users do have feedback about design, does it ever come back to the development team or do the developers sometimes solve it straight away?
143. P5 well that depends. Developers resolve a lot of crashes straight away when they get the report. But regarding design. If the design needs to be changed, the responsibility for that sits with the design team, so it comes back to us then. But it's not always something we react to. Sometimes we don't agree. And sometimes we do agree and see a better way of doing it, but there are more important things in the pipeline and then we need to focus on those things. So feedback does not always lead to changes.
144. A okey, very interesting. And what is important to reach a good result in this step? Is it getting good feedback?
145. P1 Is P4 next on the speaker list?
146. P4 Hmm are we still on step 4? I think it's important what you ask for and what kind of questions you are putting out there
147. P3 Usually the first question during a feedback session is what kind of input would like. Because you can get feedback like, I think the whole system is wrong. It can be difficult to relate to because like. We never make the best design, because that means you spent too much time designing, you just make it good enough for testing
148. A Okey great. I have something else I'd like to ask. What people should be involved? I know you talked a bit about testing with each other and with users, but should you be talking to someone else too?

149. P1 Well for instance, if the design is related to the mobile application, it’s good to talk to a mobile application developer
150. A What is it that makes that important? Is it to see if it’s feasible to make the design?
151. P1 Yes for someone like me who doesn’t write code, I sometimes wanna find out what’s the easiest way for them to implement the solution. Like if I have 3 different options and if they have a given component already maybe that can make it easy for them, and otherwise they might need to spend 2 weeks developing. That kind of thing can be decisive in the design decision later, so I atleast wanna check up with them. That’s my thoughts on it
152. P2 It depends. If I’m building a new invoice handling system, it can be useful to get a stakeholder that knows economics. There can be other stakeholders as well. But the order of priority if I'm required to give one, first off there is the end-user or somebody that represents them. Then next somebody within the design team. And then finally a developer. At least that is what I think.
153. A perfect. I’m thinking we can move on to the final step.
154. P5 I unfortunately need to leave but it sounds like you have good flow going.
155. A okay that’s no problem. Take care P5 and thanks for participation. But so regarding the final step, but this is something we see as a more formal step. You mentioned sometimes even the CEO participates as a tester. I’m wondering, related to, well first let’s have a look at the speaker list, P4 is next. So I’m wondering for this particular step. Do you need to get approval from the business side of things, how do you go about getting that kind of approval?
156. P1 Do you mean before it gets picked up by the development team?
157. A Yes exactly
158. P4 We’re a bit spoiled since we have our lead designer in our team. He’s senior in the company and there’s not a lot of people that would disagree if he says something, often what he says goes..
159. P2 He’s not here so you can say whatever you want to
160. *laughter*
161. P4 It’s not common that we prototype all the way through the process. And you could see the beta as a prototype. But it’s rare that we would go back on a design.
162. P2 i’m not sure..
163. P4 We don’t take a lot of chances, I feel like sometimes. It’s very rare that we put a design on hold and redo it.
164. P2 I’m not sure I can view beta-testing as prototyping. I don’t agree with it. It’s like…
165. P1 It’s not everything that goes into beta
166. P2 Yeah and even if it goes into beta, it’s rare that we go back and look at it again. Maybe it’s different now, but I feel like it’s still rarely we revisit things after they go into beta. The video-conference for instance, it’s not really a prototype
167. P4 No you’re right. But we do have the potential and the beginnings of doing it in a more test-focused way. For instance by releasing a beta to friendly customers. So I think we have an ambition to work that way. But I’d like to do some work with actual running

prototypes with code, but maybe that would be resource heavy and require a developer resource

168. A What does it take in order for you to reach agreement with the lead designer about a design? What do you need to show him in order to get clearance to go ahead?

169. P4 We have a discussion throughout the entire design process. In this phase it's very rare that we still have disagreement. He's usually at least partly the person requesting the feature as well.

170. A How do you imagine it would look if he wasn't part of the process, like in another company?

171. P4 I imagine then maybe you would get a software requirement specification, saying the design needs to fulfill certain criteria. And then you need to pitch the design, maybe not in this step, but during some earlier step. That's something that we are protected from

172. A How do the others feel about this?

173. P1 As early as last fall, we introduced our own process within the team. This process was intended for how to go about approving each other's work. We're sticking to that and everybody feels like it's working well. We decide that during a feedback meeting we show designs that are between 10 to 70 percent complete. And then we can bounce ideas off of each other and discuss the prototype, and from an early stage we are all involved in each other's work. And then you do some updates and show your design and display again next week. And maybe you agree that if you change a given button and add a certain thing, then the next iteration will be the good enough final version. In addition we have a team meeting on every Tuesday, and before we as a team approve a design, we need to show a final design on a Tuesday and get agreement from everyone. And that makes the design approved, and at that point everybody has been given the chance to talk a couple of times before and they can't suggest changes that come totally out of the blue. When I first started here, sometimes it was more individual, designers would pass on designs that the other designers never even saw. But we moved away from that, now we can all stand behind a design and for me that's a good thing, the designs we send down the chain are confirmed designs.

174. A Okey very interesting. And as for agreeing with developers, do they also need to approve the design in some capacity?

175. P2 I think we try to do that in a good way. Sometimes there is a bit of friction. If you don't communicate with developers during the design process and you just show up dropping work on them that they haven't been consulted about, then they might feel irritated. So to not show up and surprise them and have them go, oh what you have done here, we involve them throughout the process. They test prototypes and discuss solutions. Sometimes we do it in a good way and sometimes in not as good a way, and that's probably how it's always gonna be, but we're trying and a lot of time we can avoid the problems that can arise here. I try to involve them even in smaller features, just to check in that it feels reasonable to them as well. And maybe they will say, no sorry I think that's f'ed up, and then you work with that

176. P1 And then sometimes there's not a clear receiver, and that can make it complicated. I might have a confirmed design that's complete, but I might go back and redo some sketches so it looks more incomplete, then the developer can feel like they were involved at an earlier stage. It can take some time but it's worth the effort, it makes the work much easier. I've been a consultant before and I took away that you should consider what the specific person you are working with wants. Like some people you need to involve much more and discuss the design several times with, while others are happy to just receive the final design
177. P3 Yes some people wanna be really involved
178. P1 Exactly
179. P3 And others just wanna find out what are the components I need to build
180. P1 Yes, so what you need to do is find out a bit about the person picking up the design for development
181. A Very interesting, I'm very happy with the answers you have provided. I was thinking we could take two quick minutes to discuss the process in entirety. We can use the speaker list for order. So P1 what are your thoughts on the five step and what we have talked about today?

Summary discussion

182. P1 Hmm..
183. A Have you gotten any special thoughts, and how has it made you look at the way you are currently working?
184. P1 It has given me the feeling that our current mode of working is good and that the steps proposed here match pretty well with those steps. But then to be honest, I don't feel like these are entirely new steps, if you're a trained interaction designer this is probably the way that you'll be working.
185. A Okay, very good. And if you want you're very welcome to point out any flaws in the steps we present here, that is just as welcome as telling us that the steps are good. P2 what are your thoughts about what we talked about today
186. P2 I'm just thinking, how are these steps intended to be used, are they like a suggestion for method?
187. A Oh yes. We can talk a bit about the steps and their background, perhaps we should have done that earlier
188. P2 it's possible that you did and that I just missed it.
189. A So what we've done is we've taken 3 activities from requirements engineering. One activity is elicitation, getting an understanding of customer needs, the next one is specification which is putting things down in writing and lead to the creation of a requirement specification. And then there's also validation where you get an approval for the design. And we've combined that with the research we did on prototyping to generate these steps. We've focused on how can you perform these activities by using prototypes
190. P2 Ah okey. Well the thing is I don't really consider a simple paper sketch a prototype, but if you do then this matches pretty well. But if you don't see it that way, I don't think you can work with prototypes all the time. For me a prototype is something

interactive that you can test out and really get a feeling for. And I don’t think that the effort required to produce that kind of prototype is justified for all of these steps. And if you look at for instance step 2 it could even be a bad idea to start prototyping. You can get locked into an idea, you’ve put time and effort into it and that makes it hard to reject it, you kind of develop a bond to it and it’s your baby. But maybe it missed the target completely. The first design you spit out is usually horrible, like you get anxiety looking back at it. But you do need to put something out there and start thinking about the problem. There’s a danger in putting to much love into a given design at an early stage. I think there was also a question about how the way we are working today is working. I’m a cynic, I always complain, so I’m gonna put out that I miss getting the approval on when the design is good enough, sometimes we’re a bit messy company but we’re also very fast moving. We typically don’t have definitive roles like product owner, project manager. Sometimes I miss that sort of structure.

191. A P1 I noticed that we are now approaching a time where you need to leave.
192. P1 Yes, I need to go pick up my kids
193. A that’s totally Ok, thank you for participating
194. P1 Bye everybody
195. A P3, you are next up one
196. P3 I think this model is interesting because it gives some perspective on what you need to find out with prototypes. I don’t think I’m always that aware about what I’m looking for. Initially maybe a prototype should serve to explore. And this model can help give an idea of how to focus my goals. But then I also feel like it is a bit hard because of some of the terminology. Some words like prototype, test, customer needs and so on that I have a hard time interpreting. Like what they actually are in this context. Like what is a test and what is a prototype. That can make it hard. And also the parameters are not always clear. What do you have initially and what do you output?
197. P2 If I can ask an interesting question, do you have a definition for prototype as defined in your thesis?
198. A Hmm, maybe we can define it a bit more clearly. But our view is that even the sketch is a prototype. In our own (case study) work at one point your lead designer requested we make some clearer drawings to build on, and to be honest at first we didn’t totally see the value, but then we noticed how much clearer it got what we were discussing, even if it was a primitively drawn google drawing that we used to communicate. I also noticed that during the process we avoided prototyping because it felt hard to present something incomplete. I hope that kind of answers your question.
199. P2 Okay, yes kind of. So to flip it. I feel like what we are talking about here is more like sketches.
200. A Oh okey, great point. And P4, your thoughts on everything?
201. P4 I don’t have a lot to add apart from what’s been said already. But concerning our own process and specifically prototyping. I feel like there’s a lack of high-fidelity code based prototypes. But maybe I said that already.

202. P2 A certain challenge is that if you construct a high-fidelity prototype, sometimes it’s very close to the final implementation, like particularly in HTML. Then you have almost built the whole entire thing.
203. P3 Maybe that would be different if we worked at Netflix, then maybe we would afford a lot of tweaking and doing A/B testing
204. A Yes. And maybe more tools will surface that change the landscape and how you work. I like Invision but it does have limitations
205. P4 Yeah I think easier tools will show up soon
206. F One of the aspects we have noticed is that there’s a difference in building a product for a company end-user and just a regular consumer, that can drastically change the feedback. Hmm there was another thing as well, but I forgot it now
207. P3 Tools and things maybe?
208. F Nah.. Anyways, it doesn’t matter.
209. A Okey, in that case lets finish up for today. Thank you so much for participating. I for one was happily surprised that this matched so well with your design, and maybe we need to move things around a bit, but we haven’t based this in any way on how you do things. And then obviously stakeholders look differently depending on the given company. Have a great afternoon

INSTITUTE OF COMPUTER SCIENCE | LUNDS UNIVERSITY OF TECHNOLOGY | PRESENTED 2020-06-11

MASTER THESIS Prototyping as a Requirements Engineering Technique
STUDENTS Franz Lang, Alexander Mjöberg
SUPERVISORS Maria Blomberg (Telavox), Elizabeth Bjarnason (LTH)
EXAMINER Björn Regnell (LTH)

Prototyping as a Requirements Engineering Technique

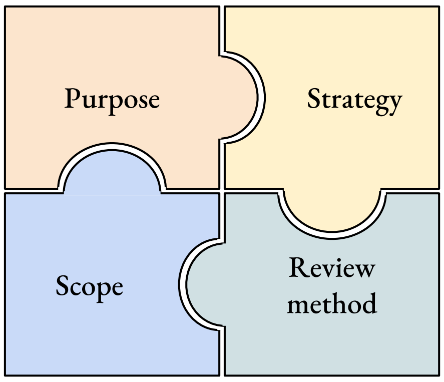
POPULAR SCIENTIFIC SUMMARY **Franz Lang, Alexander Mjöberg**

Prototyping is a widespread technique used for anything from illustrating ideas of a product to providing the foundation for a sales negotiation. This thesis presents a model that depicts the domain of prototyping and explores its use in the context of requirements engineering.

Software prototyping is a consolidated term for the creation and evaluation of a prototype that depicts a simplified version of a software product. Prototypes are used to explore certain aspects of a product throughout the product development process, such as testing the usability of a mobile application before release. There is a large number of different types of prototypes as each type is suited for exploring certain aspects. For example, evaluating physical properties of a flood barrier may necessitate the use of physical materials in a physical environment while user interaction in a mobile application may only require digital means. Subconsciously these are plausible connections but what factors do we take into consideration to determine what necessitates the use of a physical prototype and when it is sufficient to use a digital one?

We have set out to create a model that can be used to understand the domain of prototyping. We identified four aspects in a literature study of the domain; *purpose*, *strategy*, *scope*, and *review method*. The knowledge captured in our model can be used to identify the best prototyping approach for a given context.

The model was evaluated in an exploratory case study where we manipulated the aspect of scope, yielding three prototype variants. Subjects were shown one of the prototype variants after which they responded to a survey.



Our results indicate that manipulating the aspect of scope in prototyping can encourage users to focus on certain attributes such as how something looks or how it works. The feedback received from the user will vary depending on the scope of the prototype. This means that if you are looking to perfect the choice of colours in an interface you should focus your efforts on creating a prototype that looks good. If your goal is to evaluate a new functionality in a software you should create an illusion of functionality, such as with the use of *Sketch*, thereby increasing the amount of conceptual feedback.