

MASTER'S THESIS | LUND UNIVERSITY 2018

Chatbot for Configuration

Niklas Lindvall, Robin Ljungström

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2018-07



Chatbot for Configuration

Niklas Lindvall
ada08nli@student.lu.se

Robin Ljungström
ada09rlj@student.lu.se

May 17, 2018

Master's thesis work carried out at Axis Communications AB.

Supervisor: Daniel Andersson, Daniel.B.Andersson@axis.com
Johan Rönnåker, Johan.Ronnaker@axis.com
Ulf Asklund, ulf.asklund@cs.lth.se

Examiner: Martin Höst, martin.host@cs.lth.se

Abstract

With the rise of artificial intelligence and the way of communicating is shifting over to instant messaging, the natural conjunction is intelligent chatbots. These chatbots can be used to either talk about personal issues, technical issues with a system or customer service.

The problem of understanding a system diminishes with a chatbot. Instead of needing to understand a system, the system can understand the user. This is also the obstacle to developing a chatbot. To develop a chatbot that can understand natural language, and take sufficient decisions to understand the important information from natural language.

Development of our chatbot was inspired both by other existing chatbots and from conducted user tests. The user tests were observed and difficulties were identified, difficulties that the chatbot needed to handle.

The result of the work is a prototype chatbot that can help guide a user to configure a system and help with any question that might arise during the configuration. The burden of understanding the system is shifted to the chatbot rather than the user.

The prototype chatbot is evaluated with a concluding user test to see if it is a suitable way of configuring a system, evaluate if the chatbot could help with the problem of configuring a system and how it compares to the existing graphical user interface.

From the evaluation we can draw the conclusion that a chatbot is suitable for a configuration tool. It is best used as a complement to an existing tool, as to help understand the underlying system.

Keywords: Conversation User Interface, chatbot, Configuration

Acknowledgements

We would like to thank everyone who made our master thesis possible. We would like to express our gratitude to our supervisor Ulf Asklund for his patient guidance and all of the great advice during many discussion. We wish to acknowledge Axis Communications for their hospitality and engagement, which surpassed all expectations. We are thankful towards everyone who participated in the user tests, discussions and the feedback they provided. Finally we would like to extend a special thanks to Daniel Andersson and Johan Rönnåker at Axis Communications for their encouragement, invaluable feedback and expertise.

Contents

1	Introduction	9
1.1	Background	10
1.2	Hypothesis	10
1.3	Research Questions	10
1.4	Disposition	11
2	Theory	13
2.1	Conversational User Interface	13
2.2	Chatbot	14
2.3	Access Controller	15
2.4	Application programming interfaces	15
2.5	JSON	16
2.6	JSON Schema	17
2.7	Framework	17
2.8	Bot frameworks	18
2.9	Natural Language Processing	19
2.10	Previous work	20
3	Methodology	23
3.1	Scope	23
3.1.1	Limitations	24
3.2	Framework selection	25
3.2.1	Framework identification	25
3.2.2	Framework Requirements	25
3.2.3	Evaluation	26
3.3	Development	26
3.4	Evaluation	27
3.4.1	Scenarios	27
3.4.2	User Test	28
3.5	Analysis	29

4	Implementation	31
4.1	System overview	31
4.2	Integration	32
4.3	Dialogs	32
4.4	Logical path	32
4.5	Input Validation	34
4.6	Choice	34
4.7	Memory	34
4.8	Optional values	34
4.9	Confirmation	35
4.10	Natural language processing	35
5	Result	39
5.1	Framework Selection	39
5.2	Framework evaluation	40
5.3	User test	40
5.3.1	Test overview	40
5.3.2	User test 3	41
5.4	Focus areas	43
6	Threats to validity	45
6.1	Internal validity	45
6.2	External validity	46
7	Discussion	47
7.1	Evaluation of framework	47
7.2	Implementation	47
7.2.1	Access Controller integration	48
7.2.2	Error handling	48
7.2.3	Natural Language Processing	48
7.3	User tests	50
7.3.1	User test 1	50
7.3.2	User test 2	50
7.3.3	User test 3	50
7.4	Chat context	51
7.4.1	Starting point	51
7.4.2	Getting started	51
7.4.3	Learning period	51
7.4.4	Timed measurements	52
7.5	Functionality overview	52
8	Conclusion	53
8.1	Future work	54
8.1.1	Expert system	54
8.1.2	Linguistics	54
	Bibliography	55

Appendices	59
A Framework - Overview	61
B Dialog relations	63
C Form	65
D User Test Scores	71
E Test observations	73

Chapter 1

Introduction

We interact with many computerized systems in our daily life, in everything from the payment method at the grocery store to traffic lights at an intersection. Sitting at a desk with a screen and keyboard to interact with computerized systems is no longer the norm[1]. Laptops and smartphones brought us away from our desks and made us mobile. The last decade has been revolutionized by smartphones, which made a compact, touch-based interaction available everywhere. Recently a surge of Virtual Personal Assistants, such as Apple Siri, Amazon Alexa, and Google Assistant and is shifting the way to interact again[2][3].

The vision of conversational interfaces is that we would be able to communicate with technology as freely as with another person. A user can simply, in a natural language, state his intent and the system will parse what is needed to perform the intent.

The concept of technology imitating human interaction was popularized by the Turing Test by Alan Turing in 1950 [4]. The foundation of the test is to see if a computer can communicate in such a way that it is indistinguishable if communicating with a human or a computer.

Joseph Weizenbaum is often contributed as the creator of the first chatbot ELIZA, which tried to emulate Rogerian psychotherapist and was created in 1966. Up to 1980 language processing was based on written rules. But in 1980 the introduction of machine learning gave way to more complex versions of language processing.

In early 2000, chatbots got popular again. Numerous chatbots were developed as customer service helpers, such as Anna by IKEA[5] and Erik by the Swedish National Tax Board[6], but they were limited in both use and capabilities. They were mainly controlled by specified rules and could only answer questions.

During 2016 Microsoft introduced their bot framework and later Facebook opened up their Messenger Platform[7][8]. This made it so that many companies started integrating their services into chatbots. It is now possible from Facebook Messenger to order a car from Uber, purchase a pizza from Dominos and receive breaking news from CNN[9][10][11]. Axis Communications is now interested to see how a chatbot could be

used to interact with their devices.

1.1 Background

Axis Communications is a security company based in Lund, with a large portfolio of security products such as security cameras, door access stations and access control systems.

The Access Control System is a device containing the logic for user permissions in a physical security environment. The device handles the information received from a card reader, decides if the card is allowed access and controls the opening of the associated entrance. For small to medium business, Axis Communications provides a web application available on the device to configure it, such as add new cards or change permissions. Axis Communications are always looking to further improve the usability of their products and simplify the ease of use of their products. Can a chatbot make the device easier to configure?

1.2 Hypothesis

Using a chatbot to interact with a system, can move the burden of understanding from the user to the system. A user can often formulate their intent for an interaction and a chatbot can from the given sentence construct a perception of the intent of the interaction. The system also knows if any information is missing and can proceed to ask the user for any additional information. If at any point the user doesn't know what to do, a chatbot can guide the user to what needs to be performed for the interaction.

Chatbots can be viewed as a human-like machines, and will therefor give a system a friendlier face[12]. Describing a problem is often easier than trying to find the solution yourself, even if the listener is a computer with predefined answers[13]. A common collection name for chatbots is Conversation User Interface.

1.3 Research Questions

From the hypothesis, we arrived at the following questions. Since we did not want to develop the entirety of a chatbot a bot framework should be used.

As there already exists a tool for configuring the Access Controller, that gives us the option to compare the two different approaches to configuration tools.

As describing a problem is easier than trying to find a solution by yourself, we have to make the chatbot understanding and be able to describe solutions in a friendly manner.

RQ1 Are there any bot framework that can be used for this kind of work?

RQ2 Is it possible for a chatbot to be used as a substitute or complement to an existing tool for configuration?

RQ3 How does a conversational user interface compare to a graphical user interface?

RQ4 Is it possible to have the chatbot be clear enough as to not require any external help or information in order to make use of the system?

1.4 Disposition

This is a brief overview of the thesis rapport and a short introduction to each chapter.

Chapter 2 - Theory gives a short theoretical background to the thesis work.

Chapter 3 - Methodology describes the methods and tools we used to develop the chatbot

Chapter 4 - Implementation describes the approach we used to solve the problems we encountered.

Chapter 5 - Result shows the result from the different parts of the thesis work.

Chapter 6 - Threats to validity explains the ways we might have been wrong or how our methods might have skewed the result

Chapter 7 - Discussion Discussion of what went wrong and how we interpreted the data we gathered.

Chapter 8 - Conclusion is our ending statements, including future work, and answers the research questions.

Chapter 2

Theory

This chapter will explain the theory behind the many parts that make up the chatbot. The features used to connect the chatbot to the Access Controller is explained in sections 2.3 - 2.6. The features used to make the chatbot understanding is explained in sections 2.9 and 2.9. The bot framework itself is explained in section 2.8.

2.1 Conversational User Interface

Conversational User Interfaces is the vision of trying to make the human-machine interaction more similar to a human-human interaction. By having a dialog where the information or instructions are exchanged during a dialog, where the burden of understanding is moved to the machine understanding humans, rather than humans understanding the machine. In the ideal scenario, a person would be able to communicate in a natural language and the machine would understand the intent of the dialog, similarly to how humans would perform the interaction. Unfortunately, language is complex and constantly evolving, which makes this a hard, if not impossible, task to achieve fully. Therefore two broad categories are used to describe a systems approach; conversational and command-based[14].

Conversational systems encourage communication in free form and are meant to be more of a conversation companion, rather than an interaction with a system. As these types of systems do not look to understand the intent, they instead look to understand the context to have a meaningful answer to any given sentence. They seldom strive to achieve more than the actual conversation.

Command-based systems share the burden of understanding between the human and the machine. The user needs to communicate with clear actions and intent, with a structure suited for the system. This way it is clear that you are communicating with an interface for a system.

Most often a system cannot be purely defined as one of these approaches, but it is in most cases clear which approach it has aimed for as a Conversational User Interfaces.

Command-based and Conversational system share many methodologies, in which the most common one is domain limitation. When limiting the operational domain of a system, it becomes easier to understand the intent of a user. This does require the user to gain or have a certain understanding of the operational domain, so a user can communicate within the set domain. For example, the scenario of ordering a pizza; The domain is the action of purchasing and the subject is a pizza, the result is pizza. Assuming that you previously enjoyed pizza, the conversation for this domain won't confuse you and the system can be constructed with the assumption of some expertise.

Command-based

HUMAN: I would like to order a pizza.
AGENT: Eat here, delivery or take-away?
HUMAN: Take-away.
AGENT: Would you like a Vesuvio, Capricciosa or Calzone?
HUMAN: Vesuvio
AGENT: Would you like to order another pizza?
HUMAN: Yes.
AGENT: Would you like a Vesuvio, Capricciosa or Calzone?
HUMAN: Calzone.
AGENT: Would you like to order another pizza?
HUMAN: No.
AGENT: Your order will be ready in 10 minutes.

Conversational

HUMAN: I would like to order a Vesuvio and Calzone.
AGENT: Would you like it delivered?
HUMAN: No, I would like it take-away.
AGENT: Anything more?
HUMAN: That is all.
AGENT: Your order will be ready in 10 minutes.

Figure 2.1: Example of Command-based vs Conversational

2.2 Chatbot

Chatbots are the subcategory of Conversational User Interfaces which are restricted to written communication. This removes the subject of speech recognition, the process of interpreting sound waves to sentences, and the uncertainty that comes from this process. In a chatbot, it can be assumed that written sentences provided do not contain any uncertainties, which is not necessarily the case after speech recognition. This makes chatbots simpler to integrate than speech conversational user interfaces.

The popularity of chatbots are increasing, but they have been around for a long time. One of the first being ELIZA which was released in 1966. However, improvements to AI have made bots smarter and more coherent in their conversational skills to the point where they can hold meaningful conversations about a given topic. The growth in messaging platforms has also increased the demand for automated response systems.[15]

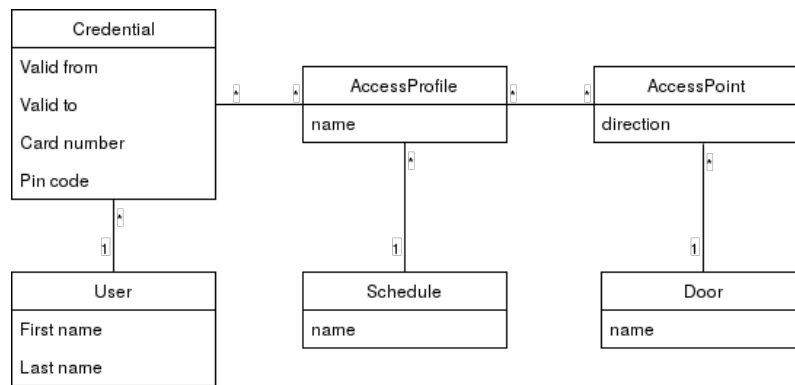


Figure 2.2: Relations in user management

Chatbots are best used when helping users with menial tasks. Chatbots are efficient where the solution to a problem is easy and predictable. In many cases, you don't need to try and understand what the user wants, instead present every available command. If a user for instance chats with a customer support bot, the chatbot can show everything it can help with and if the chatbot is unable to help with the specific problem a human agent is available to take over.

In cases where there are chatbots available to help users with these easier tasks, a chatbot can solve the problem without any human interaction in about 80% of cases [15].

2.3 Access Controller

Access Controller serves the purpose of controlling access to an entrance, most commonly a door or an elevator. The Access Controller contains which user owns which card, what card have access to which door, during what hours of the day, if the door requires a PIN-code etc. Based on this the Access Controller decides if you are authorized to enter.

For a user to gain access he must have a credential belonging to a group, a schedule and a set of doors, as shown in Figure 2.2. This is most commonly configured through the web clients user management screen, shown in Figure 2.3.

2.4 Application programming interfaces

Application Programming Interface (API) is a general term for a way of interacting application-to-application. The API exposes functionality within and can be used to build external implementations, which can expand the functionality of the system or integrate the functionality over multiple different systems.

The most commonly used API on the web is called a Representational State Transfer (REST) API. This is a way to mainly exposing the representation of the data within a system. By sending a query to the API a certain set of data is returned, based on the query. The query of data is in this type of API described in the PATH, GET and/or POST field of the HTTP request.

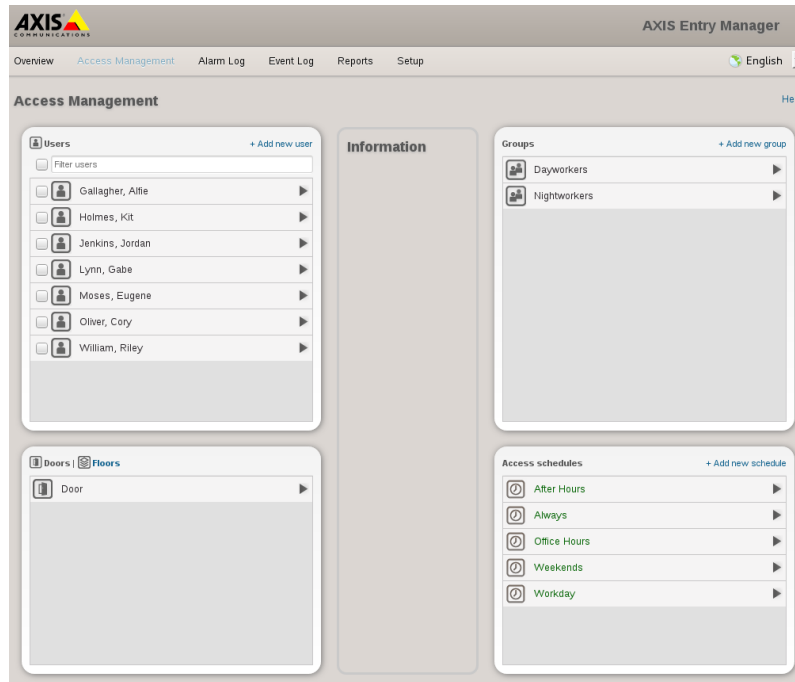


Figure 2.3: Interface for user management

2.5 JSON

JavaScript Object Notation is an open-standard data representation derived from JavaScript and is used to transfer data between different systems running different programming languages. Values can be an object, list, string, number, `true`, `false` and `null`. A simple example can be seen in Figure 2.4.[16]

```
{
  "name": "John Doe",
  "age": 23,
  "nicknames": [
    "Doey",
    "Johnny"
  ]
}
```

Figure 2.4: Example of JSON Object

```
{
  "title": "Person",
  "description": "Attributes of a person.",
  "required": ["name", "age"],
  "name": {
    "type": "string",
    "description": "Proper first and last name."
  },
  "age": {
    "type": "number",
    "minimum": 0,
    "description": "Number of years the person has been alive."
  }
  "nicknames": {
    "type": "array",
    "description": "Familiar form of the proper name",
    "items": {
      "type": "string"
    }
  }
}
```

Figure 2.5: Example of JSON Schema for Figure 2.4.

2.6 JSON Schema

JSON Schema is a complement to the JSON standard. It uses a JSON-based format to specify the valid formation of a JSON object. The standard consists of a JSON object describing each field present, what is required to contain, what is optional and other limitations, such as for example length of the value. This opens up for a lot of possibilities, such as automated validation or testing, since the specification is written in an easily traversed format. It can act as a contract for local validation in a server-client solution, which has led to its inclusion in the OpenAPI specification[17][18].

2.7 Framework

A framework is a software with generic abstract methods that can be expanded with user-written code and enables the possibility to be changed for a specific application. Most commonly frameworks consist of functionality shared by all applications within the domain, providing a reusable structure and base functionality. Often the framework provides an abstraction of platform integration so that the application can be executed on different platforms without separate implementations.

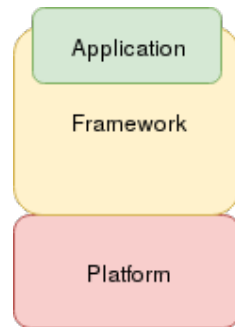


Figure 2.6: Illustration of a framework

2.8 Bot frameworks

A bot framework is an implementation that assists you when building a chatbot and provides functionality most chatbots require. Specific frameworks differ in their approach, responsibilities, and level of abstraction, but have many similarities. The most fundamental functionality provided is a structured representation of a conversation, such as whenever a message is sent to the bot, a response is generated and sent back. This can be represented in simple trigger-action scenarios or more complex dialog flows with multiple interactions. Many provide simplified ways of asking questions, receiving valid data and storing the data in association with a dialog, a conversation or a user. Since language is inherently a big part of chatbots, many also provide a format to keep track of and implementing functionality towards multiple languages. A lot of them provide a way to interact with messaging platforms, providing a unified way to integrate and deploy to multiple messaging platforms, such as Messenger, Whatsapp, and WeChat. Many also provide some kind of implementation to parse the intent of a user, through different kind of approaches. Since communication takes place in a human written form, many frameworks also provide some kind of natural language processing engine.

Microsoft Bot Framework

This section will briefly explain how Microsoft Bot Framework handles conversations.

Whenever a message is sent to the framework a so-called *conversation* is started. Within that conversation, different *dialogs* can be started. A dialog consists of one or more steps, which either gives information or expects additional input. Within a conversation, several dialogs can be held and freely switched between. Information within a dialog is held within that dialog, and not accessible from other dialogs. However, information held in the conversation is accessible for all dialogs.

To start a conversation an *intent* have to recognized from the input of a user. The intent comes either from an internal class called `recognizer` or an external module supplied by Microsoft called *LUIS*.

2.9 Natural Language Processing

Natural Language Processing (NLP) is the theory of a computer being able to understand human written form and to translate computer output to natural language. This is necessary for a chatbot to understand more than just predetermined commands.

The following subsections are theories that are commonly used in understanding natural language. These theories are in turn used either fully or partly in our chatbot.

Classifier

Classifiers are used to define which set of categories a new observation should belong to, based on previous training data. In natural language processing, this is used to classify a sentence into a group of commands that the chatbot can handle. A common technique for this is the Naive Bayes Classifier [19]. Classifiers can be used to categorize larger sets of text blocks to extract keywords.

Stemming

Stemming is a technique in which you can scale a word down to its base version. This is not to be confused with the base version as singular, undefined. This is a base version done computationally. For example, the base version of "universal" is "univers". This makes it easier to detect different inflections of words, such as plurals or defined versions. The issue with stemming is also that some words gain the same base version. Such as "universal" and "university" both have "univers" as the base stem.[20]

Inflection is a subcategory of stemming. As most inflections in the English language is changes to the end of the word, it will still manage to get the correct stem. However, stemming can destroy vital words and vital information in a certain sentence which is why lemmatization or morphological segmentation is done before stemming. These methods first detect if the word is referring to something or if it is referring to a plural version of a word.

Bag of words

Bag of words is a technique used when classifying sentences. In natural language, ordering of words is important. In English, the word ordering is Subject - Verb - Object (I gave him the book, subject being I, the verb is gave, and book is the object)[21]. The bag of words technique does not take this ordering into consideration while classifying a sentence and strictly considers which words have been mentioned.[22]

This technique makes it easier to classify sentences but some information may be lost due to the fact that ordering is ignored.

Part of Speech

Part of speech tagging is the technique used to tag each part of the sentence as its type of word, such as nouns, verbs, adjectives etc. This is done by using a big learning set and a

set of rules to train the tagger on what it should do. The learning set contains words with their respective tags. The rule set contains rules for words with more than one possible tag. For example, the word "tackle" in "fish tackle" is a noun, but the same word in "the player did a dirty tackle" is a verb. Using preceding tags will in most cases solve this.[23]

Word segmentation

A big block of text is difficult to handle and because of that chunks of text is usually divided up into smaller blocks. These smaller blocks are then broken up into n-grams or by word lists.

N-grams divides the sentence up into chunks which are n words long. Google, for instance, uses n-grams a lot to guess which word is most likely to come next in its predictive search[24]. N-grams are mostly used in speech recognition to see if the word our speech recognizer found is even likely to be there.

The smaller lists of texts are usually broken up by sentences. A good divider is a period or comma. Within these smaller, texts you can either keep the order of words or treat it as a bag of words and ignore ordering.

Feedforward Neural Network

Instead of using a Naive Bayes method to classify sentences, a neural network can be used.

Neural Networks is the practice of making a computer create a solution for a problem based on a set of data, commonly referred to as a learning set. The system randomizes values in a model, evaluates the result towards the learning set and adjust the model to get a smaller error. This is performed a certain number of times or until a specified error is achieved, then the model is considered trained. The models used differ based on what kind of problem that needs to be solved. For a finite classification, a Feedforward model is often used[25]. The Feedforward model consists of multiple layers, whereas the result is fed into the following layer in a unidirectional manner. The layers are consisting of weights, which are the values adjusted when training the model. The input data is an array of data points, and the output consists of an array with the likelihood that the input can be classified as any of the output parameters.

2.10 Previous work

We studied other chatbots and how they interacted. Some have a narrow domain of usage, so talking about a certain subject will get a response while not talking about a certain subject will yield an "I do not understand" response. Other have a larger domain of usage, so it will respond to almost anything you write. Since our work is domain restricted, we studied the chatbots which had a narrower domain restriction in more detail.

In order to aid with the configuration of a system, tools are often developed to help users with difficult tasks. These tools generate a complete configuration based a few inputs provided by the user[26].

ELIZA

ELIZA is considered one of the first chatbots ever created and was trying to emulate a psychiatrist[27]. ELIZA could identify some words and put them into a sentence in the type of a question. If you told ELIZA "My head hurts", it would answer with "What makes you say your head hurts?". It gives the impression of a human talking, but it was purely rule-based, with rules concerning certain trigger words. These include family-oriented words (such as "father", "mother" etc.) and the response was different in those cases. Most responses were hard-coded and only switched out the trigger word used in the previous sentence.

Ellie

Defense Advanced Research Projects Agency (DARPA) has performed a lot of research into ways of identifying and treating people with depression and PTSD. A lot of time diagnosing the problem is taking up a lot of resources. The study was meant to see if you could have a virtual persona to perform the first interaction and do a basic diagnosis[28].

Ellie was the name of the virtual persona. They studied three approaches: Human, wizard of oz and completely automated. They concluded that it is plausible to use a virtual persona to perform the initial assessment of depression and PTSD. They found it interesting that participants who thought they were interacting with the automated persona acted differently than the participant who thought they communicated with another human. Participants provided more information, reported a lower fear of evaluation and permitted more emotional response when they thought they were interacting with an automated system[29].

Tay & Zo

Microsoft Tay was a purely conversational chatbot encouraging free dialog and trying to imitate a fourteen-year-old girl. It used unsupervised learning on twitter to learn how to converse. While this was an interesting project, the unsupervised nature of Tay was taken advantage of and was shut down after a mere 24 hours[30]. While Tay itself was controversial, it spawned Zo which is arguably one of the better conversational chatbots¹.

Iris

Iris, on the other hand, uses a complex engine to formulate its answers while interacting with a mathematical system[31]. Iris was introduced as a helper with mathematical problems. It distinguishes itself because of its ability to use memory and understanding references. It uses what was the subject earlier to formulate answers, which is very common in conversations. This referencing can be incredibly difficult for a computer to understand. Iris solves that by saving all information and word data for the following incoming sentences.

¹<https://www.zo.ai/>

HUMAN: Can you calculate the calculate a person correlation between psitpe22 and takr?

IRIS: Sure, the correlation is 0.817 with p-value of 0.0032.

HUMAN: Is that statically significant?

IRIS: Yes, p-value is less than 0.05.

Figure 2.7: Example from "Iris: A Conversational Agent for Complex Tasks"[31]

Chapter 3

Methodology

To evaluate our research questions a prototype chatbot was integrated for an Access Control System. We identified the bot frameworks available, evaluated their capabilities and selected one to be used in this implementation. The bot framework evaluation is described in more detail in section 3.2. To gain knowledge in chatbot interaction, we studied previously created chatbots, which are summarized in section 2.10.

When implementing the chatbot, we divided the work into three phases, which had different focus areas and concluded with a user test to evaluate the implementations. The user test was designed from scope (see section 3.1) and the same test scenarios were used during every evaluation. The user test observations and user feedback were analyzed as a focus for the next phase.

An overview of the work and how each phase was designed can be seen in figure 3.1.

Phase 1 was focused mainly on the integration towards the Access Controller, the underlying structure of our expert system and functionality observed from the study of existing chatbots. During phase 2 and 3, the focus was more user-centric, which is the reason primarily user test 2 and 3 will be acting as a basis for analysis. Phase 2 focused on making the chatbot understand the user and suitable communication for users with prior knowledge of the Access Controller. Phase 3 was focused on making the chatbot suitable for interactions with users with no prior knowledge of the system. After phase 3 the proof of concept chatbot was considered implemented and the user test of the chatbot was performed in comparison to the existing interface for the Access Controller.

3.1 Scope

The Access Controller has a lot of functionality for installation and configuration of the environment which it will be working within. We choose to limit our scope to be the operational use of the Access Controller because these were deemed to be a likely use case for a chatbot. Operational use is big enough to describe advanced relationships and

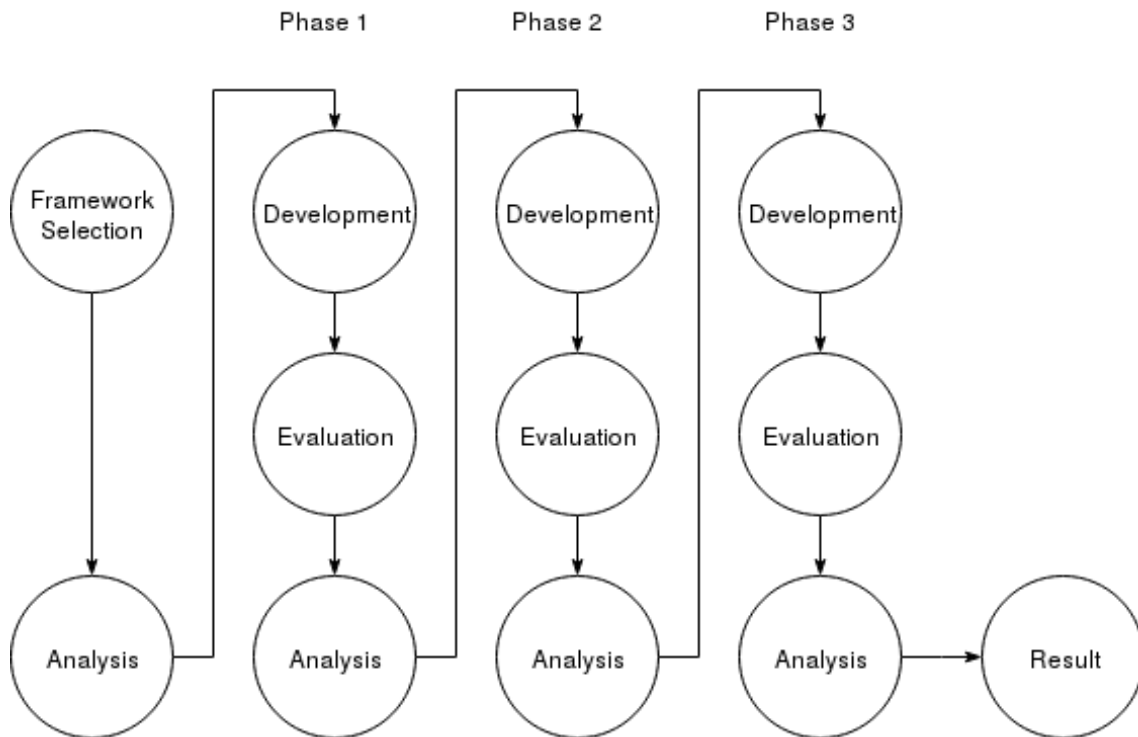


Figure 3.1: Overview of methodology

interactions, but small enough for a fully functioning chatbot to be constructed. The system functionality we are limiting our scope to includes the actions;

- Add/edit/remove a user.
- Add/edit/remove a credential.
- Add/edit/remove an access profile.
- Associate a user with a credential.
- Associate a credential to an access profile.
- Associate a door to an access profile.
- Associate a schedule to an access profile.

Doors are set during installation and are assumed to be present and without change. A set of schedules are present by default, such as workday, weekend and always.

3.1.1 Limitations

Some parts of this work is limited to make it plausible to implement, evaluate and analyze. There are many factors and perspectives when studying the interaction of a chatbot, but for the sake of this project we have chosen to limit our scope as following:

Framework A bot framework will be used as a foundation for this work. The choice of bot framework will be highly subjective and should not be taken as a defining fact of superiority.

Environment The Axis Physical Access System is an existing product and will not be subject to change during this work. All implementation will be performed towards the openly accessible API called Vapix®.

Security Restrictions such as password and access security will not be part of this work and therefore not taken into consideration.

Sample group User tests will not strive to be a truly unbiased test group, but rather a convenience sample. It will be geographically restricted to the southern part of Sweden and people of a higher than average computer skill.

3.2 Framework selection

There are many frameworks available that can be used as a base for a bot. They provide different levels of abstractions and approaches to integrating a bot. Our approach was to find as many framework candidates as possible, if they fulfilled our requirements they were evaluated and the best one was chosen to be used in the development.

3.2.1 Framework identification

We identified bot frameworks which had a certain level of popularity, which excluded personal (lesser number of contributors) and abandoned projects (inactive for more than a year). Identification of candidates was mainly performed from popular repositories, on the service GitHub (which contained the tag "Bot"), and from a comparative list of bot framework from published by Chatbots Journal[32]. The list A.1 contains the possible candidates and their basic attributes.

3.2.2 Framework Requirements

Requirements were described as easily identifiable attributes. We considered attributes that would aid or be required to perform our work. We choose to use a MoSCoW (acronym derived from "Must have, Should have, Could have and Won't have") formation to our requirements. The "Must have" requirements are critical to the project. "Should have" requirements are important but not necessary for the project. "Could have" requirements are nice to have but won't have any significant impact on the decision. "Won't have" requirements are not important to the project and won't have any weight at all on the decision.

We arrived at these requirements partly based on inquiries from Axis Communication and partly based on our own constraints and abilities. The "Must have" requirements are based on Axis Communications constraints, while the rest of the requirements are based on our own constraints.

Local environment The framework *must* be able to be hosted locally. Using an external service could obscure functionality and make it harder to understand underlying structure and functionality of the framework. It also creates a dependency on a service outside our control. This dependency can affect our work in unforeseen ways. Furthermore, the Access Controller can only be accessed locally.

Open source The framework *must* have an open source license. Open source provides transparency of functionality and possibility to expand upon functionality if needed. Open source frameworks often form a community around it, which in turn makes the source code vetted and issues are resolved faster. The community also gives stability in the case that the owner abandons the project, the community can fork the source code and continue maintaining the framework.[33]

Support The framework *should* have one or more medium to large organizations or communities supporting the framework. Knowing that larger companies are working with the framework makes it less likely to be abandoned. It also gives an indication of a certain level of stability and quality.

Language The framework *could* have a familiar programming language to the participants. Adding the complexity of learning a language on top of using a new framework is not desirable.

Integration The framework *could* have integration towards familiar services. Having integrations to services would enable us to deploy the chatbot in a familiar environment, such as Facebook Messenger.

3.2.3 Evaluation

An evaluation was performed on a smaller number of frameworks, that was deemed relevant. To evaluate the final candidates' suitability for the project a prototype activity was performed for each framework. The prototype was a proof of concept of a chatbot parsing basic data structure, constructing a dialog and performing simple web request. Each framework was assigned a limited amount of time. When the activity started nothing related to the bot frameworks was installed on the computers used to evaluate.

The activity aim was to give a deeper knowledge of the setup process, tools used to develop and the quality of the documentation. The selected framework was used during the development of the bot. The selection was based on the experience during the prototype development.

3.3 Development

Development was performed in a Debian environment with the Access Controller accessible within the local network. Microsoft Bot Framework acted as the foundation and the programming language Node.js was used to develop the chatbot. To understand how the API and the relationships it contained, the official documentation was studied to gain an overview of what was needed to configure a valid data object[34]. The valid configuration can be seen in Figure 2.2.

The development of the prototype chatbot was done in a total of three phases. Each phase ended with a user test to verify the goals of the phase had been met and to provide goals for the next phase. The last phase's user test was used as a ground to the conclusion of the thesis work.

3.4 Evaluation

To test our chatbot we used a version of GUI testing[35]. We had a start state and a given goal state. This both tests our bots behavior and the user interaction. The scenarios in the test were designed based on our project scope in section 3.1. During the user test, the chatbot was interacted with through Facebook Messenger, because of its popularity in Sweden[36][37].

When testing chatbots, usually the human-like behavior (such as trying to pass the Turing test) of the chatbot is tested[38]. As our chatbot was more of an assistant and not trying to be human-like this was not be considered. The chatbot should communicate in a human language form but only be able to converse about the topics that are present in the Access Controller.

The participants were a sample of convenience from workplace and university. They therefore shared some attributes; locality of the southern part of Sweden, native language Swedish, but fluent in English, moderate to high computer knowledge. In user test 1 and 2, the participants were employees of Axis Communications. In user test 3 the participants were only consisting of people who have not been employed at Axis Communications.

Before each test the participants were given the following information:

- All interactions are recorded and will not be directly shared.
- Data that is used, will preserve anonymity.
- Tests will be performed individually.
- The purpose is to test the system and not the user.

The test participants received a short introduction to the chatbot and its context; A chatbot to be used to manage access cards for a buildings physical access control system. The participants were asked to follow the instructions and fill in their experience in the form between the scenarios. After completing all scenarios, the participants were asked to provide personal information in form of age group, occupation, computer skill and amount of previous experience with chatbots.

3.4.1 Scenarios

The user test scenarios were created to from our scope and common user scenarios for the Access Controller.

Scenario 1

Create a new user with a new card and in a new group.

Exploratory scenario without any specific values, but presenting a general scenario to configure a valid configuration. This scenario served the purpose of introducing the user to the general concept of the chatbot and let the participant have a chance to learn before we collected our data, so not to get skewed results based on the learning curve to be introduced to a new system[39].

Scenario 2

Create a user with a specified name.
Create an associated, specified card with limited validity.
Associate it with a specified group.

Creating scenario. A specified task and a specified set of values, to emulate a use-case of introducing a new user to the system.

Scenario 3

Edit a specified user.
Remove a specified user.

Correction scenario. Updating or removing information already persisted in the system.

Scenario 4

Retrieve group membership for a specified user.
Retrieve pin code for a specified user.

Retrieval scenario. Retrieval of specific information whereas the user searches for information that isn't readily available.

3.4.2 User Test

At the end of each development phase, a user test was conducted to validate the goals of the development phase. These tests were done by people in our immediate surroundings, except for user test 3. User test 3 was done by people in our immediate surroundings with the exception of people employed at Axis Communications.

Each tester was given approximately 30 minutes to complete the test (in all different user tests).

The user tests were recorded with a screen recording software in order to be able to determine why a test participant answered in a specific fashion.

The test consisted of the scenarios mentioned before, a sample of how the test looked like can be found in appendix C. The tests were done without any input from the observers and included all information the test participant needed in order to complete the test.

User test 1 Scenario 1, 2 and 3

User test 2 Scenario 1, 2, 3 and 4

User test 3 Tested on both the chatbot and the web application supplied by Axis Communication. Which system a participant started with was randomized. Scenario 1, 2, 3 and 4 as well as doing scenario 2, 3 and 4 on the opposite system.

The user tests were done during the development, both to identify focus areas and to test if the focus area of the development phase had been met.

Phase 1

As of the first phase the chatbot had a very basic functionality of understanding a user, and the chatbots answer was inherently difficult to understand. The functionality that was considered to be done is the implementation towards the Access Controller. The chatbot had good interaction with the Access Controller and this had to be tested.

The user test 1 was performed in an informal setting where the participants were encouraged to share their thought process during the tasks. The tests were recorded by notes from the test observers based on observation and feedback from the participants.

Phase 2

This phase was focused on being able to properly understand a users intent and the bots ability to communicate actions taken. The focus of the second phase was the development of a coherent chatbot. Being able to communicate well enough to be understood by experts. The tests were recorded for study afterward.

Phase 3

As the second phase was finished we had to get data on how the chatbot (Conversational User Interface) compared to a web application (Graphical User Interface).

Participants in the user test had to test both systems and were randomized which system they would start with. Since this will test the current web application a requirement from our part was to have an independent tester, so no Axis employees were allowed to participate in this user test.

3.5 Analysis

After the user test was completed, the results were analyzed to provide goals for the next development phase. Each user test yielded a list of bugs that also had to be solved.

The screen recordings were analyzed where test results surprised us. From these, we could extrapolate behaviors from the prototype chatbot that either had to be improved (where the test showed bad results) or encouraged (where the test showed good results).

The focus of each phase was determined in advance to be Integration - Expert users - Novice users. However, the user tests helped us shape and fine-tune how the prototype chatbots interactions.

Chapter 4

Implementation

The implementation of the chatbot was divided up into three phases, each phase had different targets for the chatbot in mind. The first phase had the Access Controller in focus, to make sure the chatbot could "talk" to the Access Controller. The second phase had an expert user in mind, to make sure the chatbot could communicate in a way such that a user which have knowledge of the system could use it. The third phase had a novice user in mind such that a user who had never used the system still could use the chatbot.

4.1 System overview

The prototype chatbot acts as a tool to interact with an external system. Different sections of the bot handle different areas. As seen in figure 4.1 there are three sections which each have an individual problem to handle.

Language Processing will handle the conversion of natural language into an intent that can be used to start a specific dialog and also extract any additional information that can be of interest.

Bot Logic is the section that handles the intent and constructs a dialog that should handle that intent. It also handles the conversion of the answers from the dialog into a valid data object that is suitable for the external system.

External System is in our case the Access Controller.

Answer Synthesis is the section that handles the data coming from the external system or the logical section. This data is usually not in a readable form so this section converts data objects into a human-readable form.

The next sections will go into further detail on our own specific solutions.

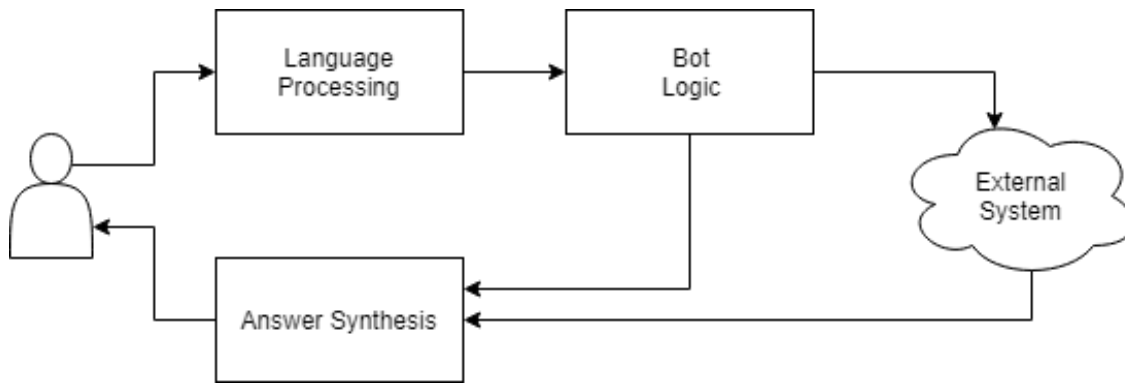


Figure 4.1: Overview of how the chatbot communicates with an external system

4.2 Integration

The chatbot was integrated towards the Access Controller openly accessible API called Vapix. Since the data was persisted within the Access Controller, no data was stored in the chatbot. To be able to communicate a representation the API was needed. The representation consisted of two parts: Definition of the interaction points in the API and specification of data objects. The interaction points described the paths to the interaction point, commands available and expected input/output objects. The objects were specified using JSON Schema consisting of the structure of the object and information to construct a valid data object.

4.3 Dialogs

The dialogs are directly representing the API commands available in the interaction points, such as "SetUser", "GetCredential" etc. These are separated into "action" (set, get, remove) and "subject" (user, credential, group). The action identified the type of dialog to be started and the subject identified what JSON Schema was to be used to create the expected input. The dialog uses the JSON Schema to generate which questions need to be answered to be able to generate a valid data object. If the dialog was performing an action which would result in a persistent change a confirm prompt will be presented.

4.4 Logical path

The logical path to create a valid data object is User then Credential then, if necessary, Group. These three data sets had to be connected in order to give a theoretical user access to a door.

As a foundation, we set rules on where the chatbot will lead a user. This has the characteristics of an expert system, which is what we were aiming for. The chatbot will continue to ask for information to fill all the fields needed to achieve what we consider to be a valid data object, and if the chatbot is somehow interrupted nothing will be created

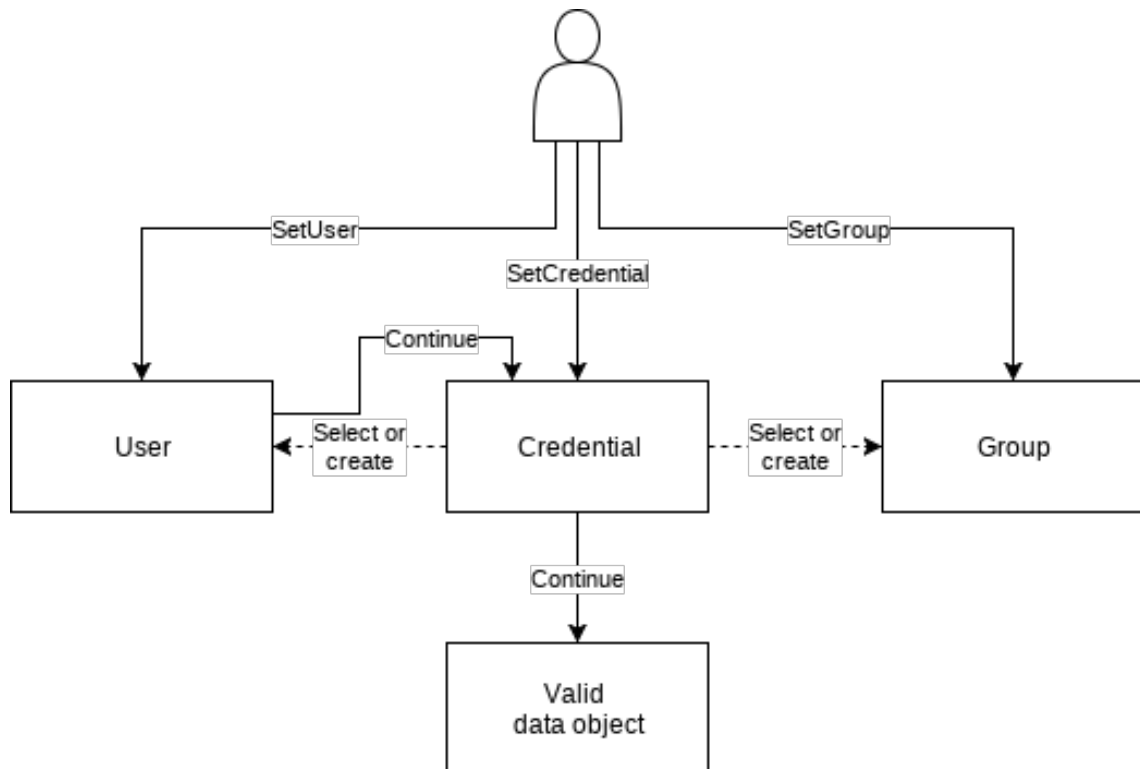


Figure 4.2: The different paths available to create a valid data object.

and the information entered will be discarded.

The different paths to create a valid data object can be seen in figure 4.2. "SetUser" will start the dialog to create a User object. When that object is created it will then continue on to create a Credential object with the User object automatically selected, then a Group object will either be created or selected from a list of previously created Group object. Once all of these things are fulfilled, a complete valid data object is created and the task is done.

A different path is starting in Credential, then you need to either select or create a User object and then either select or create a Group object.

Since the Group object can handle multiple Credential objects we decided to not try to continue after successfully creating a Group object, because Group object is not contained in any other object.

Using this, we can assure that every created entity is a valid data object, with any and all requirements fulfilled. The way the chatbot is constructed it is theoretically impossible for a data object to be created without the proper requirements fulfilled.

In appendix B the rest of the available dialogs are shown. They function in practice the same as mentioned above.

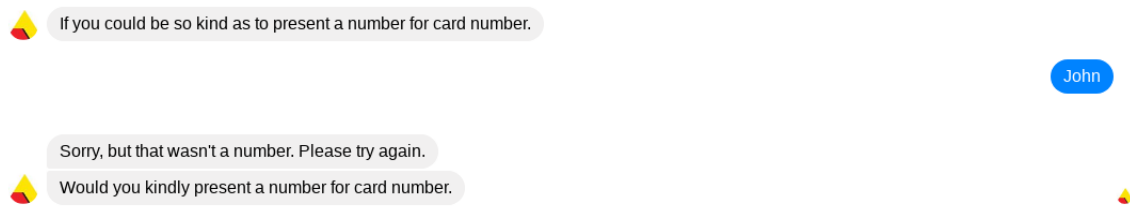


Figure 4.3: The chatbot asks for a number, but a string was entered.

4.5 Input Validation

During dialogs, some question the chatbot asks expect an exact type of data to use in the construction of a data object. These include alphanumerical names, a number for PIN-codes and card numbers and dates etc. The framework supplied some functionality to enable this, for instance, the validation and parsing of dates. An example of how the chatbot answers when an incorrect type of data is supplied can be seen in figure 4.3.

4.6 Choice

The fields which required to be populated by a relation to an already existing data object was a special case of input. This relationship is described as an alphanumerical string identifier and could not be expected to be put in manually. Therefore we constructed a choice dialog, where the chatbot let the user choose a relationship with an entity through the human-readable content for said entity. The choice dialog also presents the user with the option to create a new entity. Creating a new entity would trigger the creation dialog as a sub-dialog, after which the dialog would use the newly created entity as the selected choice in the choice dialog.

4.7 Memory

Memory is where the chatbot remembers and understands the use of previous data, and doesn't ask the user to repeat data previously mentioned. One very clear example of memory is whenever a user is created the chatbot continues on to create a credential for that user. Creating a credential is a different dialog and needs the information about which user the credential is supposed to be added to, the chatbot knows that a user was just created and will therefore use that user object while creating the associated credential object. An example can be seen in figure 4.4.

4.8 Optional values

As seen in figure 4.5 there is a lot of information needed for a valid credential object. Some of these values can be seen as optional values, as they are not specifically required to make



Figure 4.4: A user has been created and then the associated credential is about to be created. The previously created user is automatically entered.

a fully functional credential. As the chatbot needs an input for every field, some seemed redundant to ask as they would not be used most of the time. To combat an overly talkative bot, we decided to make some fields optional. These fields have a predetermined value, such that the chatbot will not ask for it but can still be changed should that need arise. For example, the fields "Description", "ValidTo" and "ValidFrom" are determined to be optional values. As such the chatbot will not ask for input on these fields unless the user specifically asks to change them. These include the values defined as memory values.

4.9 Confirmation

Before sending information to the system the chatbot has to ask if all the information the user entered was correct. There are two different methods of using confirm dialogs, the first being to confirm every step of the way (so confirming every input after the user has typed it), this method is secure but very inconvenient. The other is to collect all information and then present the information with options to change it. This method adds convenience but makes it confusing if dialogs are overly long[40, p. 71]. We decided on going with the latter of the two.

The confirmation prompt is shown every time a data object is about to be changed and includes information on what is about to happen to the data object (if it will be removed, created or changed etc.). At this prompt, there is also support to change data that was entered during the dialog and change the default values that were predetermined by us. See figure 4.6 of an example on how the confirmation prompt looks.

The response is whether the operation was a success (the object was edited, created or removed etc.). This can be seen partly in figure 4.4.

4.10 Natural language processing

The language processing engine is a classifier that classifies the sentence as one of our dialogs. The classifier is a neural network which was implemented with the help of an

```
{ "pacsaxis: SetCredential": {  
  "Credential": [{  
    "token": "Axis-00408c184bdb:1351593020.016190000",  
    "UserToken": "user-token1",  
    "Description": "Credential description",  
    "ValidFrom": "",  
    "ValidTo": "",  
    "Enabled": true,  
    "Status": "Enabled",  
    "IdData": [ { "Name": "Card",  
                  "Value": "12345678" },  
                { "Name": "PIN",  
                  "Value": "1234" } ],  
    "Attribute": [],  
    "AuthenticationProfile": [],  
    "CredentialAccessProfile": [ {  
      "AccessProfile": "Axis-00408c184bdb:1351591416.539133000",  
      "ValidFrom": "",  
      "ValidTo": "",  
    } ]  
  } ]  
}]}
```

Figure 4.5: An example of a JSON request sent to the Access Controller in order to create a credential object.

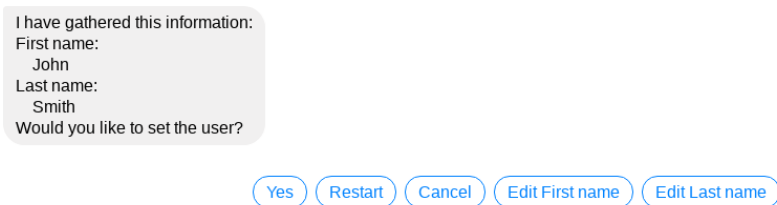


Figure 4.6: The confirmation prompt when creating a user.

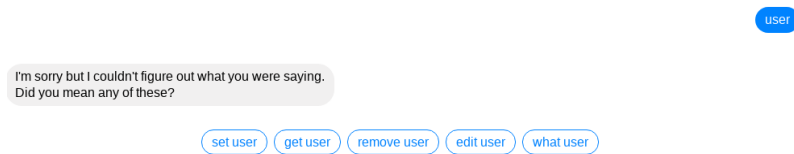


Figure 4.7: Unrecognized response, user was the input.

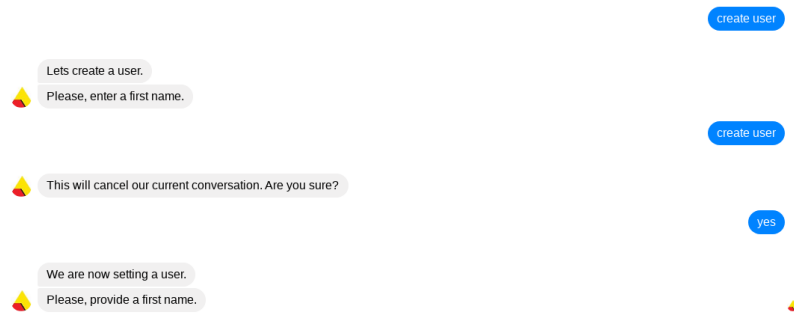


Figure 4.8: Classifier works during a dialog.

external library called Brain.js[41]. It classifies each sentence and supplies a score on how certain the neural network was that the classification is correct, ranging from 0 to 1. If the classifier fails to classify a sentence, the score is below a certain threshold, a dialog will not be initiated and instead number of high scoring intents will be suggested as guesses. For example, if "user" is entered the classifier will not be able to with certainty guess which dialog is intended and instead will show its' guesses, shown in figure 4.7. This is a way of asking for more information about a subject rather than saying it is wrong.

The result from the NLP engine is handled differently whether the chatbot currently is in or outside a dialog. While the chatbot is currently inside a dialog, the unrecognized dialog is not used, instead, the message is handled by the dialog. While outside a dialog the message is either used to start a dialog or the unrecognized dialog is used. For instance, if a name is entered as the chatbot asks for a name the NLP engine will try to classify the message sent, but will not send the unrecognized prompt as it fails to classify the message, instead the dialog will handle the message. However, if during a dialog a message is sent that is classified as a command, the chatbot will attempt to exit the current dialog and start whichever dialog the message was classified as. An example of this can be seen in figure 4.8.

In order to handle different inflections of words sent to the NLP engine, we used a stemmer to scale the words down to their base stem.

To detect names outside of an ongoing dialog we use a huge dataset of the most popular names in America. We check each word if it present in the dataset if it is then we save it as a name. Numbers are simply extracted from a sentence using a regular expression method. If a part of the sentence is only numbers, then we store it as a number.

Chapter 5

Result

This chapter contains the results from all parts of the work. Starting with the framework selection and continuing on to the results of the user tests.

During the work, we found some areas that were important in order to have a chatbot be coherent in a natural setting. These areas are explained in section 5.4.

5.1 Framework Selection

We searched for larger projects that had not been abandoned by the author and had a medium to large company backing project. The resulting framework that we found is shown in appendix A.1.

In table 5.1 the results of the evaluation are presented. Our requirements of the framework, being able to be run in a local environment and the project being open source, left us with three choices. Hubot, Microsoft Bot Framework, and Botpress.

Framework	Organization	Local environment	Open source
Watson	IBM		
Bot Framework	Microsoft	X	X
BotPress	BotPress	X	X
API.ai	Google		
Wit.ai	Facebook		
Messenger Platform	Facebook		
Hubot	GitHub	X	X
Botkit	BotKit	(X)	X

Table 5.1: Evaluation results

5.2 Framework evaluation

After looking further into HuBot, we discarded it based on its functionality. Much of its functionality revolved around doing silly things. We wanted a framework that had a more professional feel to it.

Botpress had an advanced setup process. When testing the chatbot it needed to be connected to a service, such as Facebook Messenger, which presented additional setup of the service and configuration for connections. The framework had no official tools which presented additional choices of the development environment. It was heavily geared towards developers with experience in developing web. It used a proprietary format for conversations called UMM (Universal Message Markdown) which was based on the more commonly used YAML standard. At the end of the activity, we had the chatbot running and answering with the static conversation format.

Microsoft Bot Framework had a simple setup, whereas we could run a "Hello World" within minutes of installing the corresponding development environment Visual Studio. The chat emulator made it simple to get started with the chatbot in a local environment, without setting up additional services. When deploying the chatbot through Azure a variety of chat channels are available, such as Facebook Messenger, Skype or Cortana. Dialogs were represented in an intuitive way, whereas it uses a waterfall method. At the end of the activity, we had constructed a basic proof of concept which could construct dialogs from simple data and could trigger web requests and present it to the user.

After the evaluation of the two different frameworks, we decided upon using Microsoft Bot Framework.

5.3 User test

There were three user tests done during the thesis work, the first and the second test was mainly used for implementation goals while the third is used as a base for our conclusion. The tests were done at different times during the thesis work. The first user test was conducted early on, the second user test was conducted around the halfway mark, and the third user test was done at the end of the thesis work. The third user test is detailed in this chapter.

5.3.1 Test overview

In figure 5.1 the reported difficulties are shown. These are self-reported, where one is "the chatbot was easy to use" and five is "the chatbot is difficult to use". Scenario 1, 2, and 3 all have a satisfactory curve. Scenario 1 during user test 2 was considered "very easy" so the small rise in difficulty for user test 3 was expected, due to the participants in user test 3 had never seen the Access Controller before. However, the rise in difficulty in scenario 4 was not expected, the cause of this will be further discussed in chapter 7.

Scenario 1 improved between user test 1 and user test 2 but had higher reported difficulty for user test 3. Scenario 1 was reported to be the easiest scenario.

Scenario 2 showed a decrease in difficulty over the user tests.

Scenario 3 showed a decrease in difficulty over the user tests.

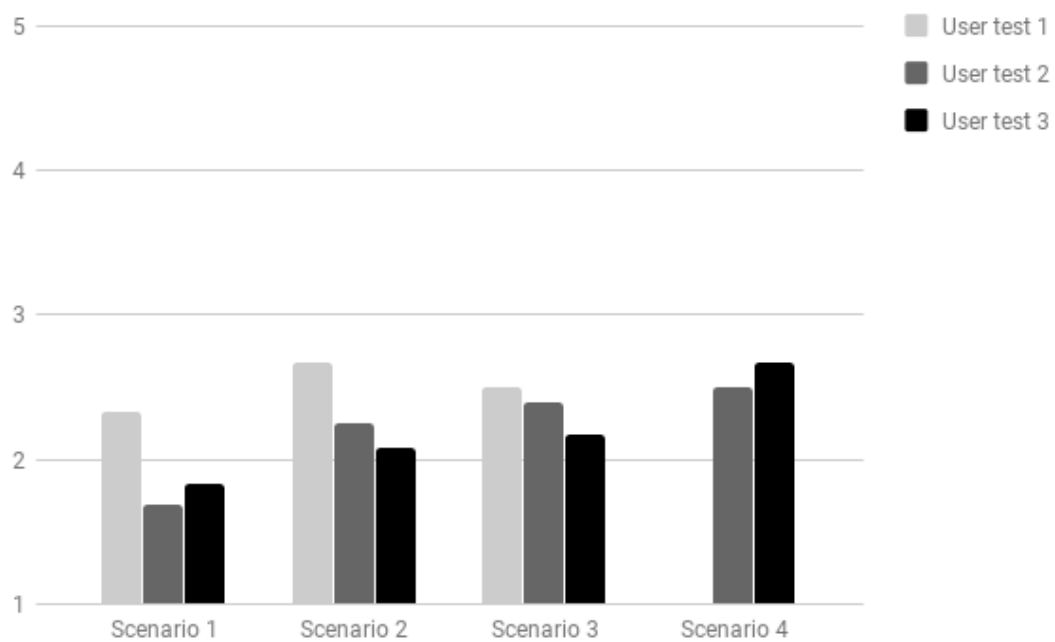


Figure 5.1: Average reported difficulty per user test and scenario. 1 was considered easy and 5 was considered difficult.

Scenario 4 showed an increase in difficulty and was noted as the most difficult scenario.

Since reported difficulty was used as a guideline improvements to the chatbot a decrease in difficulty was desired.

In table 5.2 the participants' occupation and age group are shown. Our goal was to use mostly "expert users" for user test 2 and "novice users" for user test 3. The occupation row in the table shows that we met that goal.

5.3.2 User test 3

User test 3 featured both the current web-application and our chatbot. Users were randomly assigned to a starting version of interacting with the Access Controller and half of the participants started using the chatbot and the other half started using the web-application (in total 12 people participated in user test 3, 6 started with the chatbot and 6 started with the web application).

If the chatbot was used first (see table 5.3) the fourth scenario was considered medium difficulty, however, the answers were mostly represented on the extremes of easy or difficult. In all other cases, the chatbot did not have little effect on being first or last in the user test. The perceived difficulty of the bot was increasing as the test participants were using it.

The web-application, on the other hand had decreasing perceived difficulty as the test participant continued using it. The perceived difficulties can be seen in table 5.4, and the average difficulty of use, independent of whether the web-application was used first or last, were decreasing over the scenarios.

The overall reported ease of use (see table 5.5) of the web was significantly improved

		User test 2	User test 3
Participants		25	12
Occupation	Student	0.0%	83.3%
	Developer	44.0%	0.0%
	Manager	36.0%	0.0%
	Thesis student	20.0%	0.0%
	Other	0.0%	16.7%
Age group	21-30	56.0%	100.0%
	31-40	20.0%	0.0%
	41-50	12.0%	0.0%
	51-60	4.0%	0.0%
Previously used a bot	Yes	60.0%	83.3%
	No	40.0%	16.7%

Table 5.2: User test 2 and 3 - Participants

Bot	first		last	
	average	median	average	median
Scenario 1	1.83	1	N/A	N/A
Scenario 2	2	2	2.167	2
Scenario 3	2.167	2	2.167	2
Scenario 4	2.83	3	2.5	2.5

Table 5.3: User test 3 - Bot rating

1 was considered easy and 5 was considered difficult.

Web	first		last	
	average	median	average	median
Scenario 1	2.5	2	N/A	N/A
Scenario 2	1.333	1	2	1.5
Scenario 3	1.333	1	1.5	1
Scenario 4	1.167	1	1	1

Table 5.4: User test 3 - Web rating

1 was considered easy and 5 was considered difficult.

	first		last	
	average	median	average	median
Bot	3.16	3	2.833	3
Web	2	1.5	3.833	4

Table 5.5: User test 3 - Overall ease of use rating

5 was considered intuitive and 1 was considered unintuitive.

Bot	first		last	
	average	median	average	median
Scenario 1	4 m 38 s	2 min 43 s	N/A	N/A
Scenario 2	4 min 51 s	4 min 5 s	3 min 12 s	3 min 5 s
Scenario 3	3 min 54s	4 min 2 s	3 min 6 s	3 min 2 s
Scenario 4	3 min 38	2 min 50 s	3 min 8 s	3 min 12 s

Table 5.6: User test 3 - Average time measurements for bot

Web	first		last	
	average	median	average	median
Scenario 1	2 min 32 s	2 min 33 s	N/A	N/A
Scenario 2	1 min 44 s	1 min 39 s	2 min 39 s	2 min 32 s
Scenario 3	57 s	57 s	56 s	45 s
Scenario 4	31 s	27 s	25 s	20 s

Table 5.7: User test 3 - Average time measurements for web

whether the chatbot was used before or not. However, the chatbot did not see any significant improvements whether the web was used before or not.

convenience

Tables 5.6 and 5.7 shows the average time to complete each scenario in chatbot and web.

When the chatbot was used first some outliers were present for Scenario 1 and 4, which is indicated by the average deviating from the median. The chatbot saw an overall decrease in time from being the last interaction.

Whenever the web-application was used for the first time (whether being used before or after the chatbot), the time to complete that scenario was considerably longer than in other scenarios. In the other cases, the time showed no significant difference between being the first or the last. An overall comparison between the chatbot and the web, the web interface was significantly faster.

5.4 Focus areas

We found four focus areas regarding Conversational User Interface during our work that the chatbot was required to handle. These areas were derived from the user tests (See appendix E) and study of other chatbot integrations (See section 2.10).

Integration The chatbot should always be functioning alongside the Access Controller.

The chatbot should be able to construct valid data object that can be sent to the Access Controller without containing faulty information such that the Access Controller won't accept the data object.

Error-handling Handling faulty sentences is a form of error-handling in a conversational user interface. Everyone expresses intents in a different way and as many of these variations as possible have to be handled. Incorrect information should not

be considered wrong, instead the chatbot should ask for clarification or for more information.[14]

Input requirements Creating a valid data object requires a lot of input and some of it should be able to be skipped. Values that can be guessed, should be guessed and later given the option to change it rather than asking too many questions. Input has to be as easy as possible to generate, users should not be required to know exactly what is needed to create a valid data object. Instead, the chatbot should ask what is needed and give ample suggestions.

Feedback As the chatbot answers, the bots sentences cannot be overly long, neither can it hide information. The feedback from the chatbot needs to be limited, to combat information overload[42]. The chatbot has to talk in a human-like way, as to displaying computer variables hastily thrown together will not be acceptable.

The focus areas were used as guidelines during the development phases. As predicting how a certain individual will interact with a chatbot is impossible, these areas gave us an idea on how to handle the interaction in an average case.

For example, during the development of new functionality, we had to make sure the bot could skip as much as possible during the conversation while creating the data object. During the conversation, the chatbot should not send messages that are unimportant and the messages that the chatbot does send should not be longer than a sentence.

These focus areas are the result of the evaluation of the conducted user tests and should be taken into consideration when developing a chatbot as a tool for configuration.

Chapter 6

Threats to validity

Threats to validity are the observed factors that could indicate that the conclusions and results of this work are unreliable, inconsistent or incorrect. Within this chapter, we will discuss the two categories of threats to validity affecting our work: Internal validity and External validity.

6.1 Internal validity

Internal validity is whenever the results are caused by other factors than the ones observed[43]. The threat to our internal validity comes from two aspects: The subjectivity of self-reported difficulty and sample size.

The sample size, consisting of a total of unique 38 participants, could be considered to be small. Some of our user tests and measurements contained extreme data points, which within our smaller sample had a greater influence on the resulting averages. To combat this threat we chose to display all averages next to their medians so that larger inconsistencies would be easily identifiable.

In the user tests performed the difficulty was reported by the participants, with only the extremes as references ("difficult" vs "easy"), which makes the results subjective. The perceived difficulty of a task is highly individual and dependent on previous knowledge and experience.

Since these threats to validity were present for all user test we conclude that they do not greatly affect the interpretation of the results since the results is interpretation are relative relationships between tests.

6.2 External validity

External validity discusses to what extent conclusions from this work can be generalized and applied in similar situations, not strictly within the scope and limitations of this work[43]. This work has two major threats to external validity: Product dependency and selection of test participants.

The product used for this work is an Axis Communications Access Control System. The scope used, user management, is a common type of configuration present in many products and not unique for the Access Control System. The underlying data structure does not greatly differ from other systems implementing some kind of user permissions and it would therefore be plausible to use the results and conclusions of this work to for a similar system.

Our biggest threat to external validity is sample bias since test participants consisted of a sample of convenience. This included employees at Axis Communications, thesis workers at Axis Communications, students at LTH and acquaintances. Most of which are professionals, students or enthusiasts within the field of computer science, which is very strong selection bias. This has probably affected their interaction and reported experience, and could account for their overall low rating of difficulty. Therefore it could be concluded that the conclusions are applicable for people with medium to high computer knowledge.

Chapter 7

Discussion

In this chapter, we will discuss the process' and results from the different parts of the thesis work.

7.1 Evaluation of framework

When we started to evaluate we had little to no knowledge about chatbot frameworks. We constructed requirements from previous experiences of other types of frameworks. We have been very satisfied with the choices we made.

Especially the attribute to be supported by a larger company, since many of other frameworks we evaluated has by the end of this work been integrated into larger companies. Wit.AI was acquired by Facebook[44] has been focusing more and more on working with Messenger. API.AI was acquired by Google[45] and rebranded as DialogFlow.

We performed the evaluation on Windows computers which ended up being less the ideal, because the computers we used for this work was Debian based. One of the reasons we liked Microsoft Bot Framework was because of its use of C#, which is not very suitable to be used in a Linux environment. This led us to switch over to the NodeJS implementation of the framework, which was a blessing in disguise since the untyped nature of JavaScript has been of great benefit for this work.

7.2 Implementation

During the development, many features were refined, improved and specified. These are not decisions we actively took, but rather a natural evolution of the chat experience. As such the areas where natural refinement was the case are not further explained more than the result mentioned in the previous chapters. However, on some features, we had to decide

how to approach and which approach suited our work and these decisions will be explained and detailed in this chapter.

7.2.1 Access Controller integration

The initial thought was that we would be able to automate the whole process of understanding the capabilities when the chatbot was started, by traversing the APIs. This was simply not possible to do without some manual work to specify the API. This made it so that we could add content to the Access Controller, then generate the JSON Schema from the populated objects. This worked until we realized that it was not possible to distinguish identifiers by their format, so that object references was just interpreted as a string identifier, without any information about what kind of identifier this was. This had to be manually edited to make the chatbot correctly understand object references. The format we did this on was according to the JSON Schema specification.

It can be of note that during our work, Microsoft released support for using JSON Schema to design chatbot dialogs[46], unfortunately only for the C# implementation. We wouldn't have used this functionality even if available since we wanted to maintain full control over the dialogs, but we see it as an indication that the choice of JSON Schema as an underlying structure was a good one.

The process of automation could have been feasible if a more descriptive API was provided, such as the OpenAPI specification. Since it was not within the scope of this work to make changes to the existing implementation of the Access Controller, it was not attempted to implement this standard.

7.2.2 Error handling

A lot of time was spent on the focus area Error Handling. During user test, this was the most common cause of increased perceived difficulty, so this was the highest priority to get right. We noticed that no matter how much time we spent trying to understand a sentence that was missing information, someone managed to construct a sentence which the chatbot was unable to handle.

Seeing how difficult it is to handle every incorrect (from the bots point of view) sentences, we are unsure if we should have dedicated less time on Error Handling and more time on any of the other focus areas.

As mentioned, our observations told us that this should have been the most important area, but maybe instead of making the chatbot understand supply the user test participants with a user guide to make sure they know the available commands, instead of having the chatbot trying to understand an intent from a participants who don't know exactly what is available.

7.2.3 Natural Language Processing

Our solution was mainly a classifier based on a neural network. During the work, we also tried using a Naive Bayes as an approach to the classification issue. While Naive Bayes is great at classifying a sentence as a single intent, the difference in our intents where so

	Hirate
Natural Naive Bayes	52.2%
Custom Naive Bayes	56.3%
Neural network	68.2%

Table 7.1: Hirate of the different versions of the NLP engine

small that it struggled to classify longer sentences. As an experiment, we used all the messages sent to the chatbot during user test 2, around 2000 sentences, to see how much of the set of messages the different classifiers managed to successfully classify. As the set of sentences included answers to questions etc. which the classifiers should not manage to classify, the perfect score is not 100%. The amount of sentences that were actually classifiable is 75%. The different hitrates of the different NLP engines can be seen in table ???. The Natural Naive Bayes managed to classify 52.2% of the sentences in the test set, while the Custom built Naive Bayes managed to classify 56.3% of the sentences. The Neural Network classifier managed to classify 68.2% of the sentences.

We did not validate whether the classification was correct or not, we only tested if the NLP engine managed to classify it or not.

During the development, we used an external library called Natural[47] to help with natural language processing. While we thought that library would solve the problem for us, it was quickly discovered that it lacked a lot of desired functionality. It used a Naive Bayes classifier to classify sentences, which prompted us to develop our own Naive Bayes classifier. Our own solution could be custom made to cover more of our specific cases. However, it struggled with edge cases and when multiple subjects were mentioned in a single sentence. As a sprint we looked into neural networks and how they can be used as a classifier. We developed a simple one layered neural network just to see how it could handle simple words and noticed how simple it was to set up. The Brain.js module helped us implement a multilayer neural network to be used as a classifier. The improvement can be followed in table 7.1.

Every version of the classifier had issues when multiple subject or actions were mentioned in the same sentence. This was commonly used during our testing but how classifiers work, they will never be able to classify a sentence as more than one intent. This is just one of our limitations, based both on our knowledge of NLP and linguistics. Stop words are words that can divide a sentence into smaller individual sentences, stop words are words such as "to", "from", "when" etc. We never got around to using this information in our NLP engine.

As the system handles users and their corresponding cards, names are a natural part of the written language. Test participants often wrote sentences containing both a dialog trigger and a name, such as "Remove user John Smith". Detecting the intention in the sentence was easy due to the classifier, however, the name "John Smith" was more difficult. The best solution we found was to use a large dataset containing the most common names. This will still generate errors as with names such as "January" which can both be a name and a month, or "Robin" which can be a bird or a name. To solve this a proper part of speech tagger have to be implemented with sufficient rules concerning what often preludes a name. We did not have enough knowledge of linguistics to confidently do this. We manually removed most of the names with dual contextual meanings.

We put a lot of effort into making unrecognized work as well as possible. Having the chatbot tell you that something is wrong, and nothing else (which is basically did during user test 1) was very unhelpful. Seeing how we theorized on the strength of a chatbot is to have the system understand the user rather than the other way around we devoted a lot of time to build on this strength. Guessing what the user talks about is almost always better than just announcing something is wrong. The art of guessing is difficult but made easier since we have few commands to choose from. Better guesses could have been made, such as if a name is mentioned the chatbot could assume a user is the subject. This is also a more natural sentence, "delete John Smith" sounds better than "delete user John Smith". However, we never implemented this only talked about it in theory.

Our limited knowledge of linguistics made us take the decision on only implementing a classifier, and not try to further the NLP engine more than that. We had some extras developed, such as identifying names and stem support. To be able to further the functionality of the NLP engine, we needed to dive much deeper into the field of linguistics which wasn't the focus of the thesis work and thus disregarded.

7.3 User tests

We decided on conducting a series of user tests to validate the prototype we developed during the thesis work. The user tests are also the foundation of the conclusion. Some comments on the user tests are necessary to strengthen the validity of the conclusion.

7.3.1 User test 1

The results from user test 1 are not considered in our conclusion. The version as of user test 1 was a prototype in where most functionality was very primitive. Which can be shown in the difficulty reported during user test 1. The main goal of user test 1 was to begin identifying the focus areas in section 5.4. Therefore this data was not included in results, but can be viewed in Table D.1. The majority of the participants, 5 out of 6, from user test 1 also participated in user test 2.

7.3.2 User test 2

The goal of the second user test was to test it on "Expert users". These should be people who have higher than average knowledge of the underlying system, we considered employees at Axis Communication to be "Expert users". During the second user test, the chatbots functionality had improved greatly compared to the first user test. The main goal of this user test was to validate the behavior of the chatbot and to provide the foundation of the conclusion.

7.3.3 User test 3

The third user test was supposed to test the chatbot on "Novice users", people who have little to no previous experience with the underlying system. We considered students at LTH

university to be part of such a group. The functionality of the chatbot had been improved since the second user tests to more easily convey what the chatbot was talking about.

The main goal of the third user test was to provide data to whether a chatbot should be used as a replacement or a complement to an existing tool for configuration.

7.4 Chat context

We choose to deploy our chatbot to Facebook Messenger. Our decision to use Messenger was based upon Messenger being the most commonly used in Sweden. Microsoft bot framework allows deploying towards many of other channels: Cortana, Skype, Kik, Slack or Telegram etc. What we didn't consider is if popularity necessarily is the best choice in this situation. It gives a familiar context, which is good, but it also comes with a natural bias. Messenger is mostly used to communicate with friends and family, and bots are a comparatively new phenomenon. Deploying it into a context where people are more commonly interacting with chatbots, such as in Slack and Discord, would maybe have given different observations.

7.4.1 Starting point

Participants were randomly assigned to the interface which they should do first. Our theory was that whichever interface they did last would have a better experience. This theory was seemingly correct as the chatbot and the web application were perceived to be a better experience if they had used the other interface first. The web application experience was lower if they started with the web system, but if they started with the bot, the experience of the web were higher.

The chatbot did not seem to have this trend, it was perceived to be approximately the same difficulty, as seen in Table 5.3. This is a good indication that the chatbot is easy to use and learn from, and having previous knowledge of the system in question does not greatly influence the experience.

7.4.2 Getting started

As of phase 2, we received feedback that participants had difficulties getting started which we wrote off since the perceived difficulty of the scenario 1 was comparatively low. This might have been an incorrect assessment. When a participant was presented with an empty chat they had little insight in what the chatbot actually could do, and our "help" dialogs were almost more harmful than helpful (as it only explained the simplest set of functionalities). This seems to have caused some participants to limit their input to simpler commands, which was not intended.

7.4.3 Learning period

An interesting observation is that the first interaction with the web client has a significantly higher time measurement than the following. This can be seen in Table 5.7 when comparing first scenario 1 and last scenario 2. This is concluded to be from the visual elements

has to be inspected before the user feels comfortable to start performing the scenario. If the scenario 1 was present for the last interaction, we theorize that this increase in time would have been present for scenario 1 rather than scenario 2 if this had been done.

7.4.4 Timed measurements

The measurement of time for the bot, declared in Table 5.6 was very distributed, as can be indicated by the comparison between average and median. The recordings of the extreme outliers were studied and identified to have been caused by external circumstances, such as slow or unstable Internet connection. This did not as heavily affect the web client since it used an internal cache. Therefore we did not use this measurement to study minor correlations, but rather the larger ones we can conclude. We could observe that being first or last did not affect the time of the interaction and that in general, the interaction was faster to perform in the web client.

7.5 Functionality overview

One of the observations during all our user test was that it is inherently hard to communicate what functionality the chatbot has. We choose to communicate functionality in relation to error handling or when it is specifically asked for. We discovered no suitable way of teaching the user more functionalities, which in some cases gave the user an insufficient overview of the chatbots functionality.

This could be observed especially in the Scenario 4 interactions during user test 2 and 3, which was concluded to be the most difficult scenario during our user testing. Before user test 3 specific functionality to handle this was developed so that the more advanced interactions could be described in a natural language.

Short path "Get pin-code for John Smith"

- The chatbot will respond with the pin-code. Test participants found the scenario to be easy.

Long path "Get user", "John Smith", *Lookup the card number*, "Get card" "*card number*".

- This is, according to the test participants, an excessively annoying amount of steps to fetch specific data. Test participants found the scenario to be difficult.

Even though this functionality was present for user test 3 the perceived difficulty increased. When investigating this phenomenon we found that few users realized there existed a short path and their perceived difficulty directly reflected if they did. We theorize that if we could have communicated the functionality of the chatbot better, the result would have been a lower perceived difficulty.

Chapter 8

Conclusion

The goal of the thesis work was to implement a prototype of a chatbot which could be used to configure a system. Axis Communications supplied such a system for us to integrate the chatbot to, the Access Controller. We found some problems which we think are common while developing a conversational user interface, these areas are mentioned in section 5.4. The conclusion of the thesis work is whether or not a chat could be used, and identify the strengths of using a chatbot.

With our current solution and the results from user test 2-3, we can with confidence say that it is plausible for a chatbot to substitute or complement a web application for user management.

RQ1 We choose the Microsoft Bot Framework as our ground stone for this thesis work, and in hindsight, we are very happy with our choice. The continued support from Microsoft and the Open Source community have been helpful and encouraging.

RQ2 As the perceived difficulty for the web application improved if they were using the chatbot before, and seeing how much faster the web application is, the best way of using a chatbot should be as a learning experience or as a complement to a graphical user interface.

RQ3 The overall ease of use rating (seen in table 5.5) tells us that knowledge of the underlying system had little impact on the usage of the chatbot. The initial reaction to each interface was in favor of the chatbot but the later usage shows the web was more favorable. As a conclusion, a chatbot has an easier time introducing a new system while a web application might be easier to use eventually.

RQ4 We did not find the suitable way of conveying the full functionality of the chatbot in a non-intrusive way. Many of the difficulties test participants had with the chatbot stems from the participant not gaining the knowledge of the full functionality of the chatbot.

8.1 Future work

As we did not find a satisfying answer to **RQ4**, that would be a good starting point. How to convey functionality in a conversational user interface.

8.1.1 Expert system

As the expert system is the written references in the JSON Schema, these can be automatically detected with a better tool for generating the Schema. We used a very simple one where it took an already finished data object and substituted the data with the schematics from a Schema. If the data object field contained numbers, then it is number, if there are letters, then it should ask for a string. Since dates and references are either numbers or alphanumerical strings, they will by our tool be marked as number or string, not date and references. Developing a tool that can better understand these, should help the underlying system to better generate the JSON Schema. Optional values cannot be set by our tool and have to be manually added.

Instead of using an expert system which makes a decision on where to lead the user when generating the JSON Schema, you could use a neural network to take a decision based on previous conversations. This solves the problem of optional values, as the neural network can decide on its own which field should be optional.

8.1.2 Linguistics

Understanding of natural language is essential when developing a chatbot. In order to cover the problem of error handling the need for understanding natural language is great. Be able to guess with information which is not included is also a necessity.

While our solution is functional, it leaves much to be desired. It cannot handle multiple intents, it struggles to identify names and correctly map them to the corresponding field in the data object.

An obvious improvement for the chatbot is better natural language processing. Have it understand intents better than just a classifier, and have it respond to natural language better.

The answers from the chatbot are also automatically generated from a pool of words with intents that the chatbot uses to construct some answers, although many answers are still hard coded. This can also be improved to get a more human-like behavior for the chatbot answers.

Bibliography

- [1] Aodhan Cullen. Mobile and tablet internet usage exceeds desktop for first time worldwide. <http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide>.
- [2] Bret Kinsella. Gartner predicts 75% of us households will have smart speakers by 2020. <https://www.voicebot.ai/2017/04/14/gartner-predicts-75-us-households-will-smart-speakers-2020/>, Apr 2017.
- [3] Bret Kinsella. Forrester smart speaker forecast: 22 million amazon echo sales in 2017, 66 million us households 2022, Oct 2017.
- [4] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.
- [5] Chatbots Org. Chatbot anna ikea. https://www.chatbots.org/virtual_assistant/anna_sweden, 2005. Accessed 6-March-2018.
- [6] Chatbots Org. Chatbot erik, the swedish national tax board. https://www.chatbots.org/virtual_assistant/erik/, 2005. Accessed 6-March-2018.
- [7] Microsoft. Introducing microsoft bot framework. <https://blog.botframework.com/2016/03/30/botframework/>, 2016. Accessed 23-March-2018.
- [8] Messenger platform - documentation. <https://developers.facebook.com/docs/messenger-platform/changelog>. Accessed 28-March-2018.
- [9] Rahul. Say hello to uber on messenger. <https://www.uber.com/newsroom/messengerlaunch/>, Dec 2015. Accessed 24-March-2018.
- [10] Dominos facebook messenger bot. <https://www.dominos.com.au/inside-dominos/technology/messenger-bot>.

- [11] CNN. News gets personal with cnn for facebook messenger. <http://cnnpressroom.blogs.cnn.com/2016/04/12/news-gets-personal-with-cnn-for-facebook-messenger/>, Apr 2016.
- [12] Ingrid Sørensen. Expectations on chatbots among novice users during the onboarding process. 2017.
- [13] Barry J. Wadsworth. *Piagets theory of cognitive and affective development: foundations of constructivism*. Longman, 1996.
- [14] Gabriel Skantze. *Error Handling in Spoken Dialogue Systems-Managing Uncertainty, Grounding and Miscommunication*. Gabriel Skantze, 2007.
- [15] Chatbots in customer service. https://www.accenture.com/t00010101T000000__w__br-pt/_acnmedia/PDF-45/Accenture-Chatbots-Customer-Service.pdf.
- [16] ECMA International. The json data interchange format. <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>.
- [17] A Wright. Json schema: A media type for describing json documents. Technical report, Internet Engineering Task Force, 2016.
- [18] Oai/openapi-specification. <https://github.com/OAI/OpenAPI-Specification>. Accessed 5-March-2018.
- [19] Haiyi Zhang and Di Li. Naïve bayes text classifier. In *Granular Computing, 2007. GRC 2007. IEEE International Conference on*, pages 708–708. IEEE, 2007.
- [20] Julie Beth Lovins. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2):22–31, 1968.
- [21] Harry F Gollob. The subject-verb-object approach to social cognition. *Psychological Review*, 81(4):286, 1974.
- [22] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.
- [23] Atro Voutilainen. Part-of-speech tagging. *The Oxford handbook of computational linguistics*, pages 219–232, 2003.
- [24] Jeffrey Barlow. Google’s ngram viewer. 2011.
- [25] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.
- [26] Tamar Eilam, Michael H Kalantar, Alexander V Konstantinou, Giovanni Pacifici, John Pershing, and Aditya Agrawal. Managing the configuration complexity of distributed applications in internet data centers. *IEEE Communications Magazine*, 44(3):166–177, 2006.

-
- [27] Joseph Weizenbaum. Eliza — a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 26(1):23–28, Jan 1983.
- [28] David DeVault, Kallirroi Georgila, Ron Artstein, Fabrizio Morbini, David Traum, Stefan Scherer, Louis-Philippe Morency, et al. Verbal indicators of psychological distress in interactive dialogue with a virtual human. In *Proceedings of the SIGDIAL 2013 Conference*, pages 193–202, 2013.
- [29] Jonathan Gratch, Ron Artstein, Gale M Lucas, Giota Stratou, Stefan Scherer, Angela Nazarian, Rachel Wood, Jill Boberg, David DeVault, Stacy Marsella, et al. The distress analysis interview corpus of human and computer interviews. In *LREC*, pages 3123–3128. Citeseer, 2014.
- [30] James Vincent. Twitter taught microsoft’s friendly ai chatbot to be a racist ***** in less than a day, Mar 2016.
- [31] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael Bernstein. Iris: A conversational agent for complex tasks. *arXiv preprint arXiv:1707.05015*, 2017.
- [32] Olga Davydova. 25 chatbot platforms: A comparative table – chatbots journal. <https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aefc932eaff>, May 2017. Accessed 29-September-2017.
- [33] Karim R Lakhani and Robert G Wolf. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. 2003.
- [34] Vapix - axis communications. <https://www.axis.com/global/en/support/developer-support/vapix>.
- [35] Atif M Memon, Martha E Pollack, and Mary Lou Soffa. Hierarchical gui test case generation using automated planning. *IEEE transactions on software engineering*, 27(2):144–155, 2001.
- [36] Joseph Schwartz. The most popular messaging app in every country. <https://www.similarweb.com/blog/worldwide-messaging-apps>, May 2016.
- [37] Liron Hakim Bobrov. Mobile messaging app map - february 2018. <https://www.similarweb.com/blog/mobile-messaging-app-map-2018>, Feb 2018.
- [38] Nicole M Radziwill and Morgan C Benton. Evaluating quality of chatbots and intelligent conversational agents. *arXiv preprint arXiv:1704.04579*, 2017.
- [39] Charles R Gallistel, Stephen Fairhurst, and Peter Balsam. The learning curve: implications of a quantitative analysis. *Proceedings of the national academy of Sciences of the united States of america*, 101(36):13124–13131, 2004.
- [40] Johannes Pittermann, Angela Pittermann, and Wolfgang Minker. *Handling emotions in human-computer dialogues*. Springer, 2010.
-

- [41] Brain.js. <https://github.com/BrainJS/brain.js>.
- [42] Pattie Maes. Agents that reduce work and information overload. In *Readings in Human–Computer Interaction*, pages 811–821. Elsevier, 1995.
- [43] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.
- [44] The Wit.ai Team. Wit.ai is joining facebook. <https://wit.ai/blog/2015/01/05/wit-ai-facebook>, Jan 2015.
- [45] Making conversational interfaces easier to build. <https://developers.googleblog.com/2016/09/making-conversational-interfaces-easier-to-build.html>, Sep 2016.
- [46] Define a form using json schema and formflow - bot service. <https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-formflow-json-schema>.
- [47] Natural. <https://github.com/NaturalNode/natural>.

Appendices

Appendix A

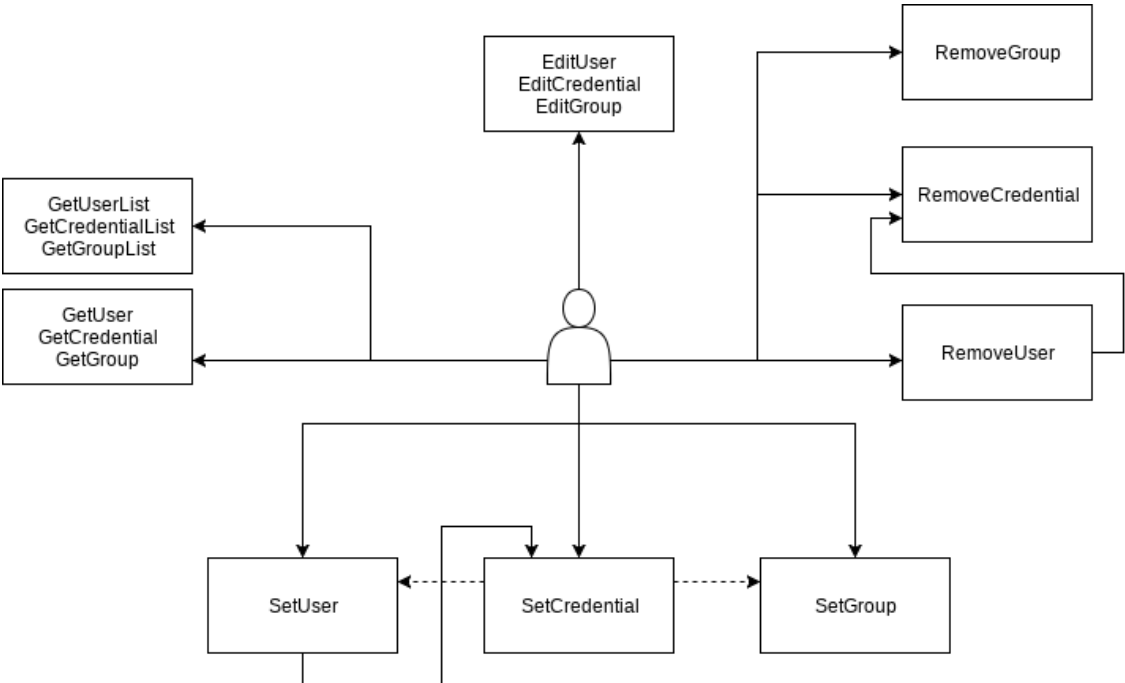
Framework - Overview

Bot Framework	Organization	License	Language	Integrations
Watson	IBM	Monthly	Java, Python	
Bot Framework	Microsoft	Free	.NET C#	Bing Cortana Facebook Messenger Kik Telegram Slack Skype
Botpress	Botpress	Free	Node JS	Facebook Messenger Telegram Kik Slack
API.ai	Google	Free	Java Swift JavaScript .NET Ruby C++ Python PHP	Google Assistant Facebook Messenger Slack Kik Line Skype Telegram Twitter Viber
Wit.ai	Facebook	Free	Java Swift C# JavaScript Ruby Python PHP	
Messenger Platform	Facebook	Free	Web API	Facebook Messenger
Hubot	GitHub	Free	CoffeeScript node.js	Campfire Gitter HipChatt IRC Slack XMPP
Botkit	Howdy	Free	Javascript	Slack Facebook

Table A.1: Framework - Overview

Appendix B

Dialog relations



Appendix C

Form

Scenario 1

Godtycklig användare

* Required

Mål

En användare med kort och tillhörighet i en ny grupp.

1. Hur svårt var det att klara uppgiften? *

Mark only one oval.

	1	2	3	4	5	
Lätt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Svårt

2. Vilken del var svårast att förstå?

3. Vilken del var lättast att förstå?

4. Ytterligare kommentarer

Scenario 2

Specifik användare

Mål

En användare vid namn John Smith.

Ett kort: nummer 45, pin kod 1234 och giltig mellan förste januari 2018 till december 31 2020.

Sammankoppling mot tidigare skapad grupp.

5. Hur svårt var det att klara uppgiften?

Mark only one oval.

	1	2	3	4	5	
Lätt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Svårt

6. Vilken del var svårast att förstå?

7. Vilken del var lättast att förstå?

8. Ytterligare kommentar

Scenario 3

Uppdatera användare

Instruktioner

Användaren John Smith namn ska vara Jonathan Smith

Ta bort Jonathan Smith samt hans kort 45.

9. Hur svårt var det att klara uppgiften?

Mark only one oval.

	1	2	3	4	5	
Lätt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Svårt

10. Vilken del var svårast att förstå?

11. Vilken del var lättast att förstå?

12. Ytterligare kommentar

Scenario 4

Stort system

Mål

Ta reda på vilken grupp Gabe Lynn tillhör.
Vilken PIN-kod Kit Holmes använder.

13. Hur svårt var det att klara uppgiften? **Mark only one oval.*

	1	2	3	4	5	
Lätt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Svårt

14. Vilken del var svårast att förstå?

15. Vilken del var lättast att förstå?

16. Ytterligare kommentarer

Helhetsupplevelse**17. Helhetsupplevelse?**

Mark only one oval.

	1	2	3	4	5	
Usel	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Fantastisk

Allmänna frågor**18. Ålder?**

Mark only one oval.

- 10-20
 21-30
 31-40
 41-50
 51-60
 61+

19. Datorvana?

Mark only one oval.


	1	2	3	4	5	
Låg	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Hög

20. Har du använt en chatbot förut?

Mark only one oval.

- Ja, många!
 Ja, någon enstaka
 Nej

21. Sysselsättning

Powered by
 Google Forms

Appendix D

User Test Scores

Subject	Scenario 1	Scenario 2	Scenario 3
1	3	4	4
2	4	2	3
3	4	4	4
4	3	3	3
5	4	4	4
6	4	3	3
Average	3.6	3.3	3.5

Table D.1: User test 1 - Scores

Appendix E

Test observations

- Typing a lot of information on a single line must be supported. Sentences such as "Create a user, the name is John Smith" is a common way of interacting with the bot.
- Optional values being skipped does not have to be shown, as long as they are easy to notice at the end of the dialog (at the confirm prompt).
- During creating, removing etc. the chatbot has to make clear at the end what is about to happen as well as being clear on what just happened, when participants agreed to the confirmation prompt.
- At more difficult decisions the chatbot have to supply ample suggestions to the question such that a participant knows from context what is needed to answer the question.
- When the chatbot cannot understand which dialog it is supposed to start, a well formatted message has to be sent which includes guesses on which dialog the participant meant.
- Incorrect information has to be able to be changed or further explained, never let the chatbot say something is incorrect without ample reason.
- As the chatbot jumps between different dialogs a clear message has to be sent based on which dialog the chatbot is currently in.
- Information should only be displayed when it is necessary. If it is not important to the current question, it can be said at a later time or not at all.
- Never ask for information twice. If information has been mentioned earlier (such as when starting the dialog, or objects being created during the conversation), the

chatbot should not ask for it again. Instead keep a memory of all things mentioned and created.

EXAMENSARBETE Bot for Configuration**STUDENT** Niklas Lindvall, Robin Ljungström**HANDLEDARE** Ulf Asklund (LTH), Daniel Andersson & Johan Rönnåker (Axis Communications AB)**EXAMINATOR** Martin Höst (LTH)

Chattbot för konfiguration av produkter

POPULÄRVETENSKAPLIG SAMMANFATTNING **Niklas Lindvall, Robin Ljungström**

Chattappar är de mest använda apparna enligt rapporter från Business Insider. Detta arbete undersöker huruvida en chattbot kan användas till att konfigurera en produkt och på så sätt kringgå behovet för en applikation.

Allt fler produkter i våra hem är digitala och måste konfigureras. Detta sker vanligtvis via en app eller en hemsida specifik för produkten, vilket kräver att du sätter dig in i hur produkten fungerar. Hade det inte varit bättre om du kan säga vad du vill åstadkomma och produkten förstår dig?

Med en chattbot kommunicerar du i ett naturligt språk och kan säga vad du vill utföra. Chattbotten försöker förstå vad du vill och kan ställa motfrågor om ytterligare information behövs. Om något är oklart så kan du be om förtydliganden eller be om hjälp. På så sätt behöver du endast en yttlig förståelse av de underliggande relationerna och vilken information som behövs.

På senare år har intresset för chattbottar ökat, framförallt från många större företag. Facebook har öppnat upp sin plattform, Messenger, för chattbottar och Microsoft har tagit fram ett ramverk för utveckling av chattbottar. Detta gör det möjligt att utveckla en chattbot och introducera den i en chatt där användaren redan pratar med vänner och familj.

Ur en enkel mening kan mycket utvinnas, såsom ny information, referenser till innehåll i tidigare dialoger och/eller till existerande konfigurationer. För att chattboten ska kunna förstå en mening krävs det att chattbotten förstår naturligt språk, vilket är svårt, men har gjorts lättare av framsteg inom neurala nätverk och maskininlärning.

I vårt arbete har en prototyp av en chattbot tagits fram till Axis Communications AB Access Controller, vilket är en produkt som hanterar kortaccess för byggnader. Vår prototyp genererar dialoger från en beskrivning av vad produkten kan utföra, vilket gör det enkelt att expandera funktionaliteten utan ytterligare utveckling. Tanken med dialogerna är att du kan säga en enkel mening, såsom "Ge Ohlsson tillgång till garaget på torsdagar" och chattbotten ska hjälpa dig att utföra det. Chattbotten vill bara ha en avsikt om vad användaren vill göra och därefter leds användaren igenom konfiguration, med enklare frågor när ytterligare information behövs.

För att utvärdera prototypen fick en testgrupp konfigurera produkten med hjälp av chattboten. Resultaten visar att testarna tyckte chattbotten var lättare att använda än den produktspecifika applikation och den produktspecifika applikationen upplevdes som lättare ifall chattbotten hade använts först. Från detta drar vi slutsatsen att en chattbot lämpar sig framförallt för nya användare eller vid kortare interaktioner.