CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Flexible Integration of Voice Recognition Components for an Automotive Android Platform:

A Design Science Research

Master's thesis in Software Engineering

David Johansson
Marcus Eliasson

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Göteborg, Sweden 2020

Master's thesis 2020

Flexible Integration of Voice Recognition Components for an Automotive Android Platform:

A Design Science Research

David Johansson
Marcus Eliasson

UNIVERSITY OF
GOTHENBURG

Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
University of Gothenburg
Göteborg, Sweden 2020

Flexible Integration of Voice Recognition Components for an Automotive Android Platform:
A Design Science Research

David Johansson
Marcus Eliasson

Supervisor: Eric Knauss, Software Engineering Division
Examiner: Jennifer Horkoff, Software Engineering Division

Master's Thesis 2020
Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Göteborg
Phone: + 46 31 772 1000

Typeset in LaTeX
Göteborg, Sweden 2020

Acknowledgements

We would like to thank our academic supervisor, Eric Knauss, and our company supervisor, William Leeson, for their constant support and advice throughout this thesis. We would also like to say thank you to Aptiv, and to all of the employees who helped us by participating in our interviews, or answered any questions we might have had. Thank you also to Supriya Kotrappa Sirbi who has supported us with her knowledge of the Alexa infrastructure. Finally we would like to say cheers to our friends and family who has supported us throughout the last six months while we have been working on this thesis.

David Johansson | Marcus Eliasson

Abstract

There are several voice recognition interfaces available on the market capable of being integrated into a variety of platforms. Mobile phones, speakers, and even refrigerators can be equipped with a voice assistant. Having a fully integrated voice assistant in a car has been shown to increase the performance of the driver and reduce distractions. The subcontractor Aptiv aims to provide their Android platform with a voice assistant meant to be sold to car manufacturers, but they aspire to avoid locking their platform to one voice component. This thesis investigates the possibilities of mitigating the difficulties of integrating multiple voice recognition components into one unified framework using a design science research approach over the course of two cycles.

Data has been collected through both formal and informal interviews that address the problem at hand. An Ishikawa diagram was constructed to illustrate an overview of the problem space at Aptiv and through evaluation, a new narrowed scope could be determined. The new scope resulted in an investigation of the four voice recognition components Aptiv showed interest in, which were Amazon Alexa, Google Assistant, Mycroft, and Houndify. This investigation lead to the creation of a tool containing 15 parameters developers can use to compare the functionality of different voice recognition components. The tool was then evaluated by developers at Aptiv through a workshop.

The tool together with its illustrations is a helpful start for developers tasked with integrating, either one or several, voice recognition components for an Android system. The presented details of each component are prone to change in the future due to uncertainties regarding their future development. Further investigation of each voice recognition system is necessary in order to fully create the general solution Aptiv aims for. Other companies may benefit from the process that has been used in this thesis when trying to integrate components. The artifact presented in this thesis could be applied to other cases, where multiple components are present and the causes and effects of choosing one or several have implications that need to be considered.

Keywords: voice recognition components, automotive software engineering, design science research, developing strategies, interchangeable components, software reuse.

Contents

1
Introduction

Voice recognition interfaces are constantly improving and are integrated more and more into technologies used everyday. Most modern vehicles have infotainment systems that support a voice controlled interface to perform various tasks such as media control, navigation and managing calls. Research [1, 2] has shown that voice recognition systems reduce the amount of time the driver spends focusing on managing the infotainment system in their car. This results in less distractions, lowered risks and an overall better driving performance. The advantages of voice recognition interfaces are beginning to outweigh its disadvantages. An advantage proven by research, is that using touch screens as interfaces tends to distract the driver for longer periods than voice controlled systems or physical buttons do [1].

Many companies within the software engineering field struggle when creating software that is flexible enough to meet the demands of all customers. Integrating components into a platform requires significant upfront work and planning. Multiple strategies try to make this process easier, both in terms of elicitation and creating thorough specifications [3, 4], as well as integration strategies. Creating a detailed plan makes it possible to reduce costs, time, and highlight possible problems. Aptiv has presented a problematic case, in which there is a need to provide a flexible solution due to the varying demands of their customers. Their product is an Android platform suitable for customers in the automotive industry. The platform mainly considered in this thesis consists of a complete software and hardware solution with its primary focus on the infotainment system of a car. Aptiv is investigating how they would proceed when implementing voice recognition systems into their current platform, while still catering to different customers with distinct demands that each require a unique solution. Whenever a voice component is chosen by a customer, there are implications to that choice and Aptiv wants to be knowledgeable regarding how to tackle these implications by providing a solution capable to catering to these variations. Aptiv can not choose components by themselves and has therefore decided that they want to be flexible within their platform in regards to what VRCs that can be used. When referring to choosing components, the goal is to choose a set number of VRCs that can be incorporated into the platform, providing a baseline that can be expanded upon over time. Companies will not be able to choose any VRC that are outside the bounds of this assortment, but over time, most needs will be covered.

Depending on the speech recognition components, there are major differences between them, each of which has its characteristics that influence the end-users and the

implementation strategies. The systems can for example vary in terms of their capabilities of technologies, user interaction, privacy and comfort. This implies different solutions, since manufacturers have distinct requirements. Due to each component requiring to be integrated in a way that is component dependent, a unifying framework that allows for integration of components within a family of products without major implementation differences would be beneficial. The voice recognition components (VRC) that Aptiv has decided to investigate are *Amazon Alexa, Google Assistant, Houndify and Mycroft.*

The purpose of the study is to systematically investigate and mitigate the implications of integrating different VRCs for automotive use cases. The mentioned implications needs to be assessed according to associated risks and trade-offs as well as available mitigation strategies. By investigating and gaining insights as to how these implications affect the outcome of choosing a specific speech component for a platform, the knowledge gained can be used to help other software engineers within the field. The nature of the problem is not only present within the automotive industry nor voice interfaces in general. It applies to companies that have the need to integrate new complex software components, such as A.I. (Artificial Intelligence) based, into their platforms that also deals with interaction between users and the software.

The research goal of the thesis is to investigate speech recognition interfaces to find a set or subset of similarities and differences on a structural and functional level.

RQ1: *Which challenges exist with flexible integration of voice recognition components for an existing platform meant to be sold to customers with varying demands within the automotive industry?*
This step aims to investigate the complexity of the problem domain. By evaluating the problem domain, useful insights and a clearer overview regarding the challenges and problems will be gained, which will benefit the work towards what later will be used to create the artifact.

RQ2: *Are there any mitigation strategies capable of decreasing time, complexity and economic costs when incorporating a variety of speech recognition components?*
Applying the knowledge gained from RQ1 by creating an artifact that supports possible mitigation strategies.

RQ3: *To what extent will these mitigation strategies support software engineers and companies exposed to this process?*
After RQ1 and RQ2 have been answered their results will be evaluated to determine their impact

The study aims to generate knowledge by creating an artifact that will aid in developing an unifying framework, capable of supporting multiple varying speech recognition interfaces. Flexibility within a platform is vital to companies that deliver solutions based on customer requests that has to adopt and support varying

components depending on demands. Therefore, systematically exploring different solutions to this problem would generate knowledge and insights that could be useful to others within the software engineering field.

Overview
The structure of the thesis is as follows:

1. **Introduction**
   This section introduces the thesis and the topic. The research questions are described along with the purpose of the study.
2. **Related Works**
   This section presents related works taken into consideration when executing the thesis.
3. **Method**
   This section describes all methods used throughout the thesis.
4. **Results**
   This section presents the results of the study.
5. **Artifact**
   This section describes the developed artifact and its functionality.
6. **Discussion**
   This section argues for the results presented and how they correspond to the conclusions drawn. It will also elaborate on the threats to validity.
7. **Conclusion**
   This section presents the conclusions drawn from the results, the significance of the study and provides directions for future work.

2
Related work

The software engineering field is very broad with many different areas to explore. By focusing on the relevant literature about the surrounding topics of the thesis, background knowledge necessary to execute the thesis can be attained. A literature review has therefore been conducted that investigates voice recognition interfaces, component encapsulation and requirements engineering for components.

2.1 Voice Recognition Interfaces

The usage of voice interfaces in cars has increased. The most common applications supported are navigation, music selection and telephone related actions. These were shown to be less distracting than using regular visual-manual interfaces and improved drivers driving performance. It is however unknown according to the authors how frequent these features are used in vehicles [1]. Winter et al. [2] investigates the typical use cases of voice control if drivers were to be unconstrained by any limiting factors. The results of this research is then used to guide the development of natural speech applications.

There are mainly two types of speech recognition interfaces. The first one is the command language type, which utilizes short commands, without regular dialog flow, that each have their specific purpose. Much like pressing a button that triggers a given action. The second type is the natural language type, which is the one recognized in voice assistants such as Siri, Google assistant and Alexa [5]. The main purpose of this is to allow for users to freely speak their language naturally. Instead of using simple commands, a natural language speech processor can detect commands through context and grammatical rules. One major human factor in speech recognition is that all people speak differently, with alternating pronunciation and dialects. Under ideal circumstances, speech recognition software is capable of interpreting the correct words 97-98% of the time. Beyond that, there are also conversational techniques that us humans depend on as we speak to one another. Cues, such as nodding and humming which indicates that you understand and attentively listen to the conversation, are tough to implement because they are complex and nuanced on another level than recognizing individual words or coherent sentences. Another human factor that complicates the use of speech recognition is short-term memory. A regular human can keep 5-9 things in the short-term memory. This does not cause any problems when using ordinary visual interfaces, since the interface itself contains all necessary information that can be re-acquired by the user when

needed. When no visual help is available, the options that are available to the user tends to be forgotten due to the short-term memory and lack of visual cues [6].

There has been limited research conducted on who and how drivers interact with the voice interfaces in their cars [1]. There are three primary types of tasks related to operating a vehicle. Voice recognition interfaces together with touch and physical buttons falls into the third category which is related to entertainment and comfort, also known as infotainment systems. The Society of Automotive Engineers recommends that a task that takes more than 15 seconds while stationary should be disabled when in motion. With current voice interfaces there is discussion that they should be disabled when in motion since the task requires a high cognitive load and it harms driving performance. The voice recognition systems have been implemented as a secondary type of input. It is an independent type of system that is not fully integrated into the vehicle. This is because the speech recognition engines has never been able to be as reliable as other physical types of inputs are [7]. Researchers are unsure and have not been able to agree upon any standards when it comes to designing voice recognition interfaces [8]. C.J. Lim states in his paper that there are eight major ergonomic problems associated with designing interfaces based on speech [9]. Another paper written in 2017 states that there is an error rate of 20.6% when using speech input [10]. Still, the same paper concludes that the experienced feeling of safety was greater when using the voice interface compared to manual input.

2.2 Encapsulating Components

The integration process of new components within an already existing platform can be complex, expensive and time consuming. Especially when the components require large parts of the actual system to be reworked in some way. By encapsulating components in a framework, a way to reduce and mitigate the upfront costs can be achieved. Research on this has been done in different ways and areas of the software engineering field. Sneed [11] discusses the advantages of wrapping legacy software components when integrating them into newer systems. This reduces costs and risks of re-engineering or redeveloping the components. Huang et al. [12] discuss how a generic framework can be used to integrate different ECA components. ECA stands for Embodied Conversational Agents and are used to communicate with humans face-to-face through multi-modal conversations. Because of the complexity of ECA components, a common framework to use when integrating the components would reduce redundant effort. The conclusions that can be drawn from the paper is that it is extremely difficult to develop a generic framework for this specific case, but that it still is the ultimate goal that will be the most profitable.

2.3 Requirements for Components

The requirements engineering (RE) research conducted prior to implementing software is a time consuming but important task. The better the research, the easier and better the final product is likely to become. There are numerous ways of going about RE and an interesting article for our case is to look into the CARE/SA approach to find new improved processes about matching, ranking and selecting commercial of-the-shelf (COTS) components [3, 4]. Several issues have been identified when working with eliciting non-functional requirements for COTS products [13]. Conducting elicitation, specification, and evaluation of quality requirements is crucial at an early stage to improve our chances of producing quality software [14].

Through further investigation of the problem space, it has been concluded that the problem of the thesis is not requirements engineering related. Choosing components is something Aptiv does not have full control over and their ultimate goal is to be able to adapt to their customers needs with as little coding effort as possible. That being said they need to offer a number of components their customers should be able to choose from. The components they offer has to fit both Aptiv's platform and the demands of their customers.

2.4 Product Line Engineering

The process of Product Line Engineering (PLE) means to streamline and reduce the cumulative cost as the amount of similar products are produced. The goal is to take advantage of the common aspects and predicted variabilities to engineer a more efficient line of production [15]. E.g having a structural platform all cars build upon. See the Volvo SPA (Scalable Product Architecture) platform [16].

The difference between the case Aptiv has presented and the common practices of PLE are mainly related to what type of product they are trying to produce. PLE aims to take a family of products and deduce the commonalities to streamline their production of their products. Pohl et al. [17] mentions in the book "*Software Product Line Engineering: Foundations, Principles and Techniques*" an example of a product line engineering scenario in which the production of cars evolve from only having one or a couple of models available for production to being able to produce multiple varying models for specific user groups. This is achieved through finding a general subset of car parts that all models can use, while producing custom tailored parts only for the components that differentiate them. Whilst Aptiv wants to integrate four different COTS components into their platform. The components they want to integrate have not been built by Aptiv, and they can in several cases not make changes to the software. Each component is also prone to evolve through continuous development and future changes may force Aptiv to change their implementations. PLE can not be applied to the case at Aptiv because Aptiv is not producing a product line. They are creating one product which is deducing the functionality of multiple products, which is more likely to be an engineering problem according to

industry experts.

3
Methodology

This investigation follows a Design Science Research (DSR) methodology over the course of two iterations, known as cycles. All guidelines for this method have been provided through the papers by Knauss and Wieringa [18, 19]. The investigation itself and its results are the artifacts produced in this thesis. The comparison sheet has been constructed through elaborate research through interviews, reading documentation, and a workshop. Various methods have been utilized in order to understand the problem space, analyze data, and to create the final artifact.

3.1 Design Science Research

DSR is a constructive method that aims to complete a tangible design artifact, while at the same time provide valuable knowledge about the problem and the solution space of the domain [19]. The method uses an iterative approach in order to create its artifact and each cycle aims to provide its researchers with more knowledge meant to further develop the artifact. The regulative cycle consists of five steps and can be found with a detailed explanation in [20]. Each cycle has its focus on one or more research questions and revolves around the artifact. Once a cycle transition is done, the focus shifts towards the next research question and re-evaluates its current problem statement before it starts. This study has been conducted in two iterations where the first cycle focuses on understanding the problem to solve, and is an essential preparation step for the coming cycle(s). The second cycle mainly focuses on the implementation of the artifact together with an evaluation.

1. **Problem Investigation**
   Seeks to understand the problem in order to be able to describe and explain it. [20] The first iteration starts with what is known as a problem-driven investigation. The emphasis of this step will be to investigate and describe the problematic phenomena. The second iteration evaluates the data collected during the first iteration before it further investigates the problem domain to best answer RQ1.
2. **Solution Design**
   Aims to create a specification for a potential solution. Using the knowledge gained from the previous step, solution design is the creative step in which possible solutions will be developed.
3. **Design Validation**
   The task within which we validate whether or not the proposed solution if

implemented correctly, will bring the involved stakeholders closer to their goals.

4. **Solution Validation**
   Attempts to implement the design which has been validated to be the most valuable for stakeholders during the current iteration.
5. **Implementation Evaluation**
   After the implementation is done, it is valuable to evaluate both the process and the artifact in order to gain knowledge. The insights from the current iteration are meant to improve the following iteration and serve as building blocks.

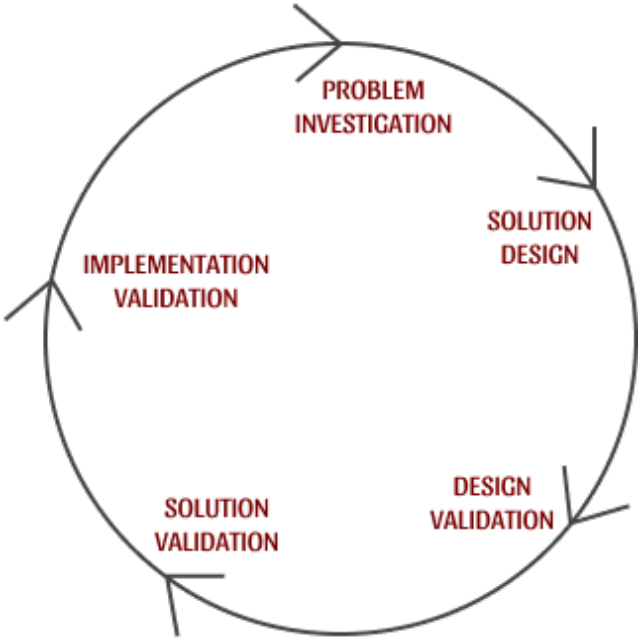

Figure 3.1: The regulative cycle of Design Science methodology.

By using a DSR study, each choice of a component can be analyzed in relation to risks and trade-offs. The analysis will result in a deeper understanding of the problem space along with tools to help developers create possible mitigation strategies. The knowledge gained from the study can help other software engineers within the field facing similar problems.

3.1.1 Problem Type

It is important to define the problem type when using the DSR-method. Depending on the problem type, appropriate evaluation criteria can be established early on. Wieringa [20] makes a distinction between *practical* and *knowledge* problems through the definition of whether or not the problem calls for a change of the world. For a practical problem, it is of importance to satisfy customer requirements through the creation of a tangible artifact. The evaluation involves the stakeholder's opinions of how well the results suit their needs. The solutions to knowledge problems do however provide stakeholders with new knowledge but without necessarily changing the world. The evaluation means to determine the value and accuracy of the gathered knowledge.

There is a clear practical problem regarding creating an interchangeable framework for voice recognition components, but the related sub-problem of knowing whether or not it is practical to attempt such a task is a knowledge problem. This investigation started out as a practical problem but evolved into a knowledge problem when the difficulty of creating such a system became overwhelming and scope creep began.

3.2 Data Collection

This section encapsulates all methods used for collecting any sort of data. Through the two cycles, a number of interviews have been conducted, both formal and informal. A workshop has been organized and investigations have been done through both semi-structurally doing literature reviews and reading all available documentation for each voice recognition component.

3.2.1 Interviews

The interviews that were conducted during this thesis were first face-to-face, but later all handled through online communication tools due to the COVID-19 pandemic. Each interview focused on one of the three main aspects of DSR (*Problem Investigation*, *Solution Design*, *Solution Evaluation*), depending on what stage the thesis was in. All participants were interviewed separately in order to reduce potential bias caused by participants influencing each other. Each interview was also following the same predetermined structure of an interview guide. The questions were semi-structured whilst the emphasis was to have more open questions giving the participants more possibility to answer freely. All closed questions were a part of the introductory questions that were used to verify the background of the interviewee. The rules listed by Kuniavsky et al. [21] were considered during the creation of the questions fitted into the interview guide.

A pilot test [22] of the questions was done after the interview guide was completed. The goal of the test was to determine its length and find questions in need of modification or questions that were missing or outright wrong to include. Transcribing the interviews were done through both taking notes during the interview and having audio recordings. The first step of each interview was always to ask for consent to record the session.

3.2.2 Informal Conversational Interviews

Because of the complex nature of the problem, several informal conversational interviews [22] were conducted with both experts at Aptiv, as well as university representatives. Informal conversational interviews are used to offer a flexible approach to data collection, where most questions come up naturally as the interview proceeds. They can be conducted without much preparation, and are to a great extent guided by the participant. Unstructured interviews are very similar to informal

conversational interviews, but often require more preparation or detailed knowledge about the topic. Since informal conversational interviews do not follow any particular structure, a set of goals were defined prior to the meeting to help lead the discussion towards important topics. Depending on which goals were defined, each interview varied in length. All participants of the meetings were informed to speak freely and the questions were generated as the interviews proceeded. The interviews were documented through transcription.

3.2.3 Workshops

Workshops offer a flexible way to teach or introduce its participants to new practical skills, techniques, or ideas. They can also be used to generate solutions to a problem or feedback. A good workshop offers stimulating activities that are interesting and facilitate creativity and discussion [23].

One workshop was held through an online communication tool. The workshop focused both on the *Solution Design* and *Solution Evaluation* of the artifact. All participants were given the same information through a presentation that had been created beforehand. After the presentation, a number of questions were asked and these were all open-ended questions aimed to be discussed by all participants. During the workshop, notes were taken which were later sent to the participants so they could change mistakes and elaborate their answers.

3.2.4 Literature Reviews

A full systematic literature review has not been accomplished during the work of this thesis. Instead, a lightweight version inspired by the procedure described by Kitchenham and Charters [24] has been completed. A systematic literature review aims to evaluate and interpret available research within a particular topic in a systematic manner. It mainly covers three phases: planning, conducting, and reporting the review.

The plan for the review was created by deciding a number of keywords to search for that all were related to one or more research questions/topics associated with the research. The process of finding material was then realized through searching and finding relevant papers using Google Scholar, IEEE Xplore, Research Gate, and Springer. The goal was to find papers with many citations that could be considered works of impact within the field. All the important papers that were found were first saved, then gone through once more to find the most relevant ones. They were then summarized during section 2.

3.2.5 Software Investigation

Each voice component has its own way of handling and creating its functionality through the use of "skills/actions/custom commands", and they are therefore structured differently. For the remainder of this thesis, this functionality will be described using the term "skills", since their purpose is similar even though they have different

names. To better understand these differences, the documentation for each voice interface was investigated. Any additional information that could be found online was also used, such as online forums, video tutorials, etc. A description of each voice interface with corresponding skill structure was then written to compile the information gathered so that a comparison could be made. Informal conversational interviews with experienced developers were conducted in order to fill in any gaps and to confirm that the gathered information was correct.

3.3 Data Analysis

Qualitative data can be analyzed in many ways using multiple different tools. Patton [25] explains that there is no perfect way to analyze qualitative data. It is rather a matter of choosing an appropriate method for the context that is presented, coupled with putting enough time and effort into the actual analysis. Once the researcher has gained enough knowledge from the data itself to be able to accomplish what he/she set out to do, the analysis is complete.

During the work of this thesis, the method of choice for analyzing qualitative data was thematic analysis. It is described as one of the most common methods for analyzing qualitative data in the research field. The method emphasizes identifying, analyzing, and interpreting themes or patterns in qualitative data. The steps that are used offer a flexible approach to analyzing data. Many aspects of the method are not unique, as it shares many commonalities with other qualitative analysis methods [26].

The thematic analysis in this thesis was done manually, mainly using a whiteboard and post-it notes to code different themes that emerged during the analysis. The first iteration of finding themes was done separately, and then were merged through analysing cooperatively. By both analysing separately and cooperatively, a wider range of themes could be established and discussions could be held to prevent that important aspects of the data was overlooked. An example of a constructed theme was "Technical" which described aspects gathered from the interviews that were centered around technicalities such as implementation and integration of software components. The theme contained comments such as "how a component not being Android centric can cause problems", "no clear testing strategy for voice interfaces" as well as "the documentation is poor and inadequate".

4
Results

4.1 Cycle I

The main objective of the first cycle was to investigate the problem space, in order to create the first iteration of a solution. The cycle consisted of five steps that follow the regulative cycle, established by the design science research method [18, 19].

4.1.1 Problem Investigation

Three interviewees were selected from the Human Machine Interface team at Aptiv, known as the HMI team. The reason for choosing interviewees from this specific team is because they are responsible for developing all applications related to the Android Platform, which is delivered by Aptiv, and also all possible future solutions regarding voice recognition components within the same platform. Each interviewee had different backgrounds and worked in different areas of the projects such as a developer, a product owner, and a project manager. By choosing interviewees with different specializations within the company, a wider range of knowledge could be gained. Every interviewee had experience with software development previously, and two still mainly work as developers. All interviews followed the same structure given in the interview guideline found in the appendix A section. Audio recordings and notes were taken in order to properly analyze the data post interviews. Consent was given in all interviews concerning being recorded. A number of problems could be identified through the interviews, and the following is a numeration of each of these together with a short description.

1. **Finding and creating generic solutions**: Aptiv needs to be more generic towards their customers in order to satisfy the demands and to mitigate possible problems that arise with case-specific solutions. There are still many cases where only parts of the platform are suitable for generic builds and much still has to be developed solely for one customer, which is known as custom engineering. This is expensive and lacks purpose for reuse. Component integration varies based on the differences between the components and the demands of the customers. By having a more generic platform it increases their capability of swiftly switching between software components that the customers desire. However, when the components are complex (such as voice interfaces) it becomes problematic to have a generic implementation. This results in loosely coupled software where parts do not have a strong connection to each other. Some functionality will also be difficult to integrate. Having a framework

that allows for seamless integration of new components would help in regards to development times, costs, and convenience. There are no current specific strategies for developing generic software at Aptiv. If a case arises in which a generic solution can be utilized, they usually try to implement it. Most of the time, however, they simply create what the customer demands with the cost being dependent on the implementation needed. By creating generic solutions, Aptiv hopes to negate most of these problems while still catering to many customers.

2. **Dealing with varying demands**: Different customers have different demands varying from letting Aptiv have full control of the development/design, while others want to do everything themselves. It can also be a problem when the customers have too much say in the matter, resulting in sub-optimal design choices.

3. **Setbacks amongst developers**: There are multiple aspects that are ever-changing as a developer. Their way of working has to be able to quickly adapt to change whilst still being fast with innovations. The agile method of working enables this in several ways today, but there are still problems to be addressed. If new technology conflicts with the current way of working, it will cause problems among the team and developers. This will lead to them having to rethink their way of working, which, hopefully temporarily, slows down their development. An example of this kind of problem would be if a team has to integrate a new tool for continuous integration, and their current git flow is not compatible with the new tool. This leads to the team having to deal with the problem caused by the integration of the new tool.

   When working with new technologies that have poor documentation, due to being relatively new, it would be beneficial/nice to work in close collaboration with the original developers in order to quickly get help and feedback on how to work. This is wishful thinking of developers since they rely on the documentation available which at times is faulty and incomplete.

   From time to time, developers at Aptiv face problems that require them to scrap their code and start over. This is something we want to avoid in favor of making improvements instead of re-writings. Starting over leads to developers having to re-learn stuff and it will lead to new bugs that previously were solved and it will slow down the development. It takes time to cover all issues that comes up and it presents many problems that previously already have been solved but just with a new look.

4. **Having expert knowledge and providing guidance**: Aptiv would like to change the way they interact with their customers. To be more of an informed company that can help its customers with suggestions rather than having them say what they want and try to provide it. Aptiv currently does not take initiative themselves, but wait for the customer, and then try to replicate what

they want to the best of their abilities. Customers choose what they want, and Aptiv listens and tries to cater to the demands.

5. **Being bound to Android**: If a library is not developed with Android in mind then it will lead to more integration problems while integrating it into the platform. Most libraries that Aptiv works with today stem from and are meant to be used with Android and are therefore easy to integrate and provides a smooth integration process.

6. **The costs of rewriting code**: Another important topic that came up during the interviews was that creating new software (rewriting) does not always imply better software. It can be the cause of multiple new bugs since the software has not been tested and iterated upon as much as an older software would have been. Newly written software, however, does not confine to old design and structure and can be created in an entirely new way, sometimes even being an improvement in relation to the older implementation. It was also mentioned that testing software can be very difficult, especially when it comes to specific components such as voice interfaces. Resource constraints also play a role in testing and can be a hindrance. When scrapping old software, the tests that were created go to waste, meaning that many tests have to be re-written, possibly multiple times.

7. **Fulfilling feature standards for each customer**: The most important insights that were gathered from the discussions regarding voice interface components were that the components need to be shipped with all of their features readily available. Most components on the market have a basic set of functionality that has the capability to be expanded through custom skills created by developers. This implies that the developers have to be able to implement new custom skills on each component. Another problem is that most voice components lack high-quality documentation. There are many times were problems occur that can not be solved through reading the documentation.

The knowledge derived from the interviews can be summarized as a number of challenges. The enumerated challenges are very dispersed, and it is not clear in what ways they are related, as well as how they affect each other. The scope of the thesis can not tackle all of these problems and need a clear goal to pursue. The *lack of overview* of the problem space prevents this, and needs to be addressed.

4.1.2 Solution Design

During the problem investigation, a variety of problems were discovered. Each problem was broad, and together they covered many different areas within the software engineering field. This made it difficult to understand which the most important problems of voice recognition components to mitigate were. Through discussion with supervisors, the idea to model the problem space was brought up. A number of models were discussed, and ultimately it was decided that the 4+1 model would

be used to try and get a better overview of the components and to create a first possible solution architecture. Creating models representing how a solution could look like help to create an overview of how the components could co-exist.

Prototypes of three different views of the 4+1 model were created, but the work came to a stall using this approach. Considering the early stage of the thesis, lack of experience of VRCs, and the absence of experts, it was not feasible to produce models capable of representing a solution that was detailed enough. The models ended up being too general and vague, can be seen in appendix Section B, resulting in moving away from the 4+1 model for the time being. By discussing with supervisors, it became clear that there still was a lack of overview and understanding of the problem space. It could be argued that the lack of overview of the cluttered problem space is one of the main reasons why it is hard to tackle the problem of integrating multiple different speech recognition components. If a more clear view of the most vital problems were available, it could potentially help remove confusion and misguidance and therefore allow for a solution to be created.

An efficient way of dividing and organizing problems within a problem space can be achieved by using a cause-and-effect diagram. An example of such a cause-and-effect diagram is the Ishikawa diagram [27] which helps its users by using a single root cause to facilitate the organization of sub-causes to the root cause. Each sub-cause helps define what makes up the root cause or the effect that it brings. This will supply an overview of the problems that can be tackled in order to help solve the main cause which is the effect of the sum of sub-problems.

4.1.3 Design Validation

In order to validate the design suggested in the solution design step, meetings with supervisors were held to discuss how appropriate it would be to create an Ishikawa diagram to help solve the problem. It was decided that the Ishikawa diagram would help understand the problem and aid in finding potential solutions. Since the prototype models created using the 4+1 model were not sufficiently developed they were not considered for validation (appendix B).

4.1.4 Solution Validation

The Ishikawa diagram in 4.1 has been constructed based on the data gained from interviews. When summarizing the data from the interviews, the seven problems accumulated one main root cause and three sub-causes. It has been identified that the main root cause implies a need for a framework that allows for changing the VRCs, because of the difficulty in reusing code between customers. In other words, there is a desire to become better at reusing code between customers by providing generic solutions. This will aid in being a better competitor in the industry.
There are three sub-causes that point towards the main cause in the diagram. They tackle three different areas of Aptiv's development of having a voice recognition interface. The sub-cause "Hard to exchange one VRC for another" touches upon

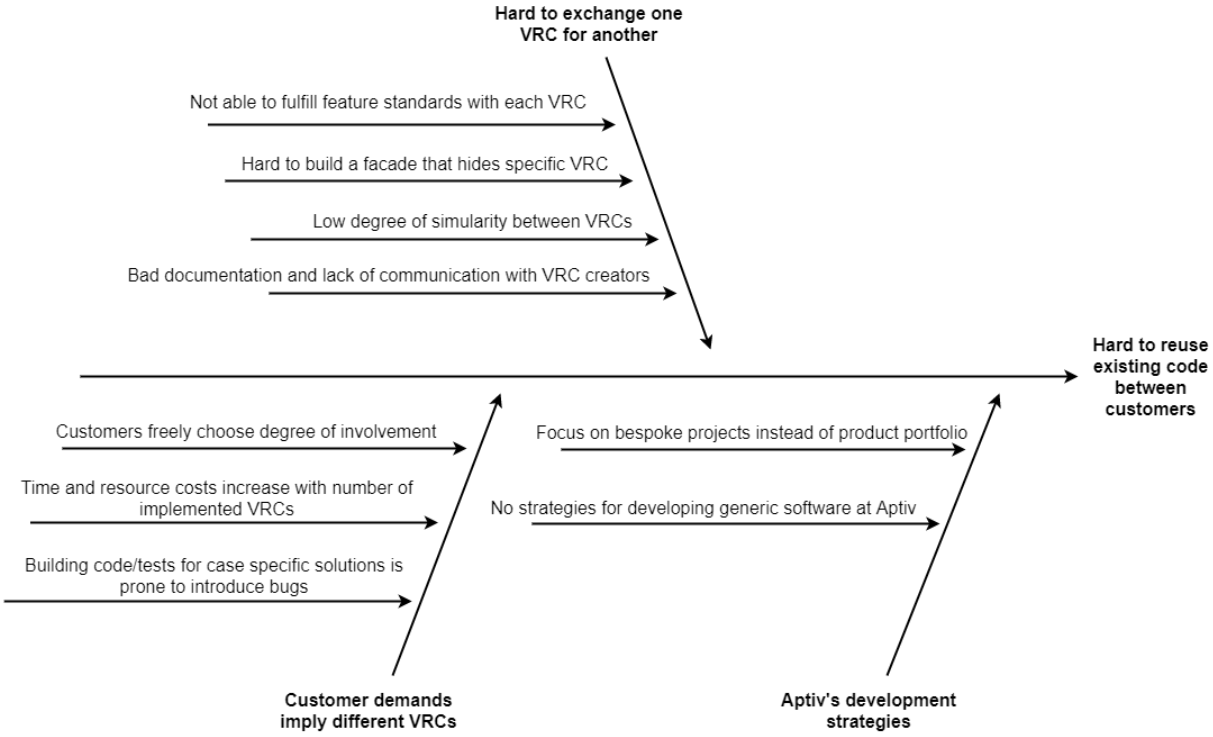

Figure 4.1: A cause-and-effect diagram visualizing the problem space.

the main difficulties of creating an environment where multiple VRCs can be implemented and exchanged for one another. The sub-problems of this branch all express different problems that make up the complex task of creating the mentioned environment. The second sub-cause "Aptiv's development strategies" takes into account what internal organizational concerns there are within Aptiv that makes the integration difficult. The third and last sub-cause is "Customer demands imply different VRCs". It tries to convey the difficulties that appear at Aptiv when customers have different demands, different levels of involvement, and expertise within the field. The focus of these issues are not outwards towards the customer, but rather inwards towards Aptiv.

4.1.5 Implementation Evaluation

This diagram has been shown to three employees at Aptiv. These were developers and managers at that were used in order to evaluate the correctness, completeness, usefulness, and feasibility of the diagram. A small interview was conducted with each employee and the following questions were asked:

1. Are the statements provided in the diagram valid and correct in your opinion?
2. Is there anything you would like to add to this?
3. Is this form of representation of causes and effects in any way useful to you as a company?
4. Which of these causes would be the most feasible and valuable to address dur-

ing the coming cycle?

The following are the summarized answers to each of the questions asked during the evaluation of the Ishikawa diagram:

1. **Are the statements provided in the diagram valid and correct in your opinion?**
   All three participants agreed that the main problems brought up in the fishbone diagram were correct. However, two stated that the branch "Aptiv's development strategies" is more of a past problem that they are moving away from.
2. **Is there anything you would like to add to this?**
   It was pointed out that "generating skill" was not included in the diagram. Also instead of saying "low degree of similarity" it would be more correct to say that some components have similarities while some do not.
3. **Is this form of representation of causes and effect in any way useful to you as a company?**
   It was considered useful according to all participants in regards to understanding the problem space and it was mentioned that it has been used previously at Aptiv to break down problems. However, this diagram in its current state will not be able to help developers to integrate multiple voice recognition components to the Aptiv platform.
4. **Which of these causes would be the most feasible and valuable to address during the coming cycle?**
   All three participants agreed that continuing to work on the problem "Hard to exchange one VRC for another". It was mentioned that customer demands are difficult to control, hence the problem "Customer demands imply different VRCs" is harder to create an effective solution for that has applicable use-cases.

Based on the evaluation of the Ishikawa diagram, it has been concluded that there is a need for a more detailed solution that enables developers to better understand the problems of integrating voice recognition components. This implies that the overview, although not lacking purpose, mostly showcased aspects already known to the company. To further investigate the problem, a new artifact was considered during the following cycle which builds upon the knowledge gained.

4.2 Cycle II

The second cycles follow the same procedure as the first, only differentiating in its shift of focus towards the solution and evaluation rather than the problem. It utilizes the knowledge gained from the previous cycle in order to improve the understanding of the problem and thus creating a better solution.

4.2.1 Problem Investigation

The first cycle was evaluated through a meeting with the representative supervisors at Aptiv and Chalmers. It was concluded that the artifact from the first cycle did not provide enough knowledge for Aptiv to be able to create a complete solution. While the Ishikawa diagram did provide an overview of challenges as well as a means to validate, prioritize, and focus for the work, it did not solve the underlying challenges by itself.

To continue the thesis, further problem investigation was needed. This led to having more meetings with supervisors, as well as industry experts. The meetings with supervisors were concerning the scope and focus of the thesis and sought for a different angle to tackle the problem that both gives value to Aptiv and contributes to the scientific field. A brief investigation of each of the voice recognition components was conducted which lead to a furthered understanding of the complexity of the involved systems. This new information was brought up during meetings with supervisors. During the discussions, a new perspective of the complexity and intricacy of each component was gained. This led to the involvement of industry experts to verify the difficulties of solving this problem. The meetings with industry experts were mostly with developers, and they were of a more casual nature than the interviews performed in Cycle I in order to promote more open discussions. It was confirmed that the systems differed drastically from each other and that there was a lack of understanding for the entire set of components. To be able to formulate a solution for a problem of this magnitude, an understanding of in which areas the structure of the components overlap and differ would be beneficial. By focusing the artifact on this, knowledge for future implementations of a unifying framework would be gained.

Instead, the focus will be on identifying similarities and differences among the VRCs. This approach will result in a more tangible artifact that has the potential of creating value for both Aptiv and the SW engineering field in the future. By reducing the complexity and vagueness of the problem statement, it becomes easier to find appropriate methods to apply in order to come closer to a complete solution.

4.2.2 Solution Design

The problem investigation concluded that the focus should shift from Aptiv's capabilities as a company coupled with the problems associated with creating a general solution for voice interface integration, to the components themselves instead. There is a need to understand each of the voice recognition components (*Amazon Alexa, Google Assistant, Houndify, and Mycroft*) to have the necessary knowledge for integrating them in a general solution. All the mentioned components have their respective documentations, which all are complex in their own way. It is hard to grasp in what way they differ or could possibly be generalized when looking at them one at a time, whilst investigating them at the same time makes the task even more confusing and complex as can be seen in the sub-problems "Low degree of similarity between VRCs" and "Bad documentation and lack of communication with VRC creators" of the Ishikawa diagram (Figure 4.1). It is, however, necessary for Aptiv

to gain this knowledge in order to reach their goals. Gaining this knowledge will lay a better foundation for future work possibly preventing many mistakes that would be made without this knowledge.

A summary and overview of each component could be realized through the creation of a data sheet containing the most important parameters that would help future work in creating a unified framework. To discover the most important parameters for Aptiv's case, a thorough investigation and evaluation has to be executed. Developers need to be involved throughout the process to make sure that the outcome will be as useful and correct as possible. Through careful structuring of the parameters, it will be possible to compare each component against the others in order to identify differences and similarities, making it more tangible to grasp the functionality and requirements of each component. Producing an artifact which removes some of the complexity of the components can reduce the amount of time developers would need for investigation, and instead provide them with more time to plan their solution strategy.

4.2.3 Design Validation

To validate the suggested design of creating a data sheet containing the most important parameters for all the components, meetings were held with supervisors to discuss this. A first draft of the sheet was created and showed to the supervisors to discuss if the suggested design was appropriate and would contribute to both industry and the academic field.

It was concluded that creating a comparison sheet for all the available information of each component was too large of a task. However, there are certain parameters leading further toward the goal of creating a unifying framework. Therefore, the scope was reduced to focus on the integration of the components into the Android platform and the handling of skills. Due to other parameters having less impact concerning the creation of a unifying framework, such as climate control and navigation, they were disregarded. Another example of this is the setup phase of each component, which includes the rigid process of authorizing each component to its respective server and other miscellaneous tasks. These are only done once and thus not worthwhile to generalize.

4.2.4 Solution Validation

The comparison sheet examines 15 parameters in total, which are deemed to be the most important for Aptiv's case. Each parameter has been identified through a comprehensive investigation that is based on reading documentation and Software Development Kits (SDK), talking to industry experts, and experienced developers as well as discussions with supervisors. The entire investigation had its focus on helping to solve the sub-cause cause "Hard to exchange one VRC for another" identified in the Ishikawa diagram (Section 4.1).

To begin with, the integration support for each VRC was investigated. This was examined first due to the nature of Aptiv's case. All identified aspects regarding integration were documented in order to convey each component's capabilities of being integrated into the Android platform. This is vital information for future work since it can potentially reveal differences leading to problems regarding integration of specific components.

During the investigation, it was identified that the skill composition and handling (see Section 5.2.1) aspects of the interfaces were the most customizable and had the least amount of restrictions in terms of "black boxing". Most functionality outside of integration and the skill systems is hard to control and is primarily managed by the creators of the VRCs. This provides great potential for generalization. Input from Aptiv gathered at interviews in cycle I implied that Aptiv would like to create generic solutions especially when it comes to creating and handling skills.

Due to the investigation of VRCs, it was decided to revisit the idea of creating models, although not the ones used in the 4+1 model. The new models were supposed to convey the structure of each VRC, based on knowledge gained from the investigation and were therefore created from the comparison tool. By visualizing the structure of the VRCs, a clearer picture of the differences can be displayed.

4.2.5 Implementation Validation

A workshop was constructed to which two experienced developers were invited to participate. The workshop included a presentation followed by a discussion regarding the completeness, correctness, and usefulness of the comparison sheet. During the presentation, it was established which the problems at stake were and what the goals regarding the content of the comparison sheet were. Also, an explanation of how these goals are meant to mitigate the mentioned problems was given. The problems revolved around the root cause established in the Ishikawa diagram seen earlier, 4.1, which is finding a way to reduce the amount of code and time to develop a unified solution for several VRCs. Using the comparison sheet and models the goal is to help Aptiv understand and prepare for the difficulties of integrating VRCs. The following list tries to convey the goals the comparison sheet tries to help with reaching:

1. Focus on the integration of skills into the platform.
2. Find differences and similarities between components in order to help developers understand and find potential solutions.
3. Identifying problems due to differences among components.
4. Reduce the lack of knowledge regarding the integration of VRCs through the visualization of these parameters.
5. This problem is complex and has a large scope. It has to be reduced to smaller pieces.

Through identification of the most vital parameters regarding integration and how each of the components differs the goal is to identify potential problems at an early

stage. This understanding could help developers create general solutions. The models were validated indirectly through the validation of the comparison tool.

Together, both participants and facilitators actively participated in the discussion concerning the data present in the comparison sheet. The first subject of discussion was to identify missing parameters (completeness). A number of interesting parameters were discovered by the participants as missing and have been investigated and considered for the final artifact. Next was the matter of understanding whether or not the comparison sheet is clearly structured and correct (correctness) and helps developers understand (usefulness) the characteristics of VRCs. It was argued that the sheet provides an overview and summarizes the components well. By providing an overview that quickly gives a developer an understanding of how the VRCs differ, it makes it possible to quickly discard some types of interfaces that lack the functionality needed or to find problems regarding how they are implemented as well as how they are supported in relation to the platform they are to be implemented in. However, there was some ambiguity regarding the explanation of each parameter, therefore it was concluded that a revision of the parameters would be beneficial. The last subject concerned finding potential solutions and problems using the sheet. Looking at the differences and similarities the participants were tasked with discussing whether or not there was any information that would help or halt the success of this project. The conclusion was that the more information they are given, the better they can tackle the problem. Also, knowing that the VRCs currently evolve and change makes knowing what their current development status and problems are very useful. Another example of a problem recognized by the participants was the lack of support for Python within Android. This means that Mycroft without an officially supported Android version would need a Python interpreter in order to be integrated and fully functional within the platform.

5
Artifact

This chapter will describe both the process behind creating the artifact of this thesis and its final form. Each iteration is meant to improve the existing artifact, which then will be evaluated at the end of each completed cycle. These evaluations will guide the coming iterations in order to improve the problem statement and find better solutions.

5.1 Problem Space Overview

In order to get a better overview of the problem space, an Ishikawa diagram (cause-and-effect/fish-bone-diagram) has been created [27]. They are considered as one of the seven basic quality tools and are widely used for visualizing and categorizing potential causes of a problem. Because of the complex problem space of this thesis, the Ishikawa diagram was meant to help identify the root cause and its associated sub-problems and to structure them in a comprehensible way.

The Ishikawa diagram has been constructed based on the data gained from interviews. When summarizing the data, one main root cause and three sub-causes could be accumulated. It was identified that the main root cause implies a need for a framework that allows for as much code reuse as possible when integrating different VRC. In other words, there is a desire to become better at reusing code when constructing products, by providing generic solutions that fit many divergent customers. This will aid both time and costs of development and in being a better competitor in the industry.

There are three sub-causes that point towards the main cause in the Ishikawa diagram. They tackle three different areas of Aptiv's development of having voice recognition interfaces available on their platform. The sub-cause "Hard to exchange one VRC for another" touches upon the main difficulties in creating a framework in which multiple VRCs can be implemented and exchanged for one another. The sub-problems of this branch all express the difficulties that make up this complex task. The second sub-cause "Aptiv's development strategies" takes into account which internal organizational concerns that exist at Aptiv that make the integration process difficult. The third and last sub-cause is "Customer demands imply different VRCs". It tries to convey the difficulties that appear at Aptiv when customers have different demands, different levels of involvement, and expertise within the field. The focus of these issues are not outwards towards the customer, but rather inwards towards

Aptiv.

The Ishikawa diagram has been evaluated by showcasing it to developers and managers at Aptiv. The evaluation verified its correctness, completeness, usefulness, and feasibility. Improvements based on the evaluation were made creating the final diagram displayed in Section 4.1.4.

5.2 Comparison Tool for VRCs

A comparison tool (see Table 5.1, 5.2 and 5.3) was created using excel to provide an overview that will help developers understand the differences between separate VRCs regarding a selected number of parameters fundamental to each VRCs design. A total number of 15 parameters were found that describe the functionality of each VRC. All of the information has been found through systematically reading the documentation of each VRC, and by investigating respective SDKs. Parameters have been discussed with industry experts and experienced developers in order to verify completeness, correctness, and value. The tool has been continuously improved when new important parameters have been discovered. The value that this tool provides to developers is mainly an overview of which parameters that are the most vital when gathering knowledge about a specific VRC. Having defined parameters to look into enables the comparison of different VRC when considering which to choose, or how to determine what sets them apart and what makes them similar when creating a generic solution such as in Aptiv's case.

Table 5.1: Comparison tool part 1.

	Amazon Alexa	**Mycroft**	**Google Assistant**	**Houndify**
Android Support The component has an Android client and support for being integrated into Android.	Yes	Yes	Yes	Yes
Gradle integration The source code has their own libraries in Gradle.	Yes	No	Yes	Yes
Has an SDK The component has an official Github repository containing their SDK.	Yes	Yes	Yes	Yes
Open source If the source code is publicly available.	Yes	Component dependent	No	No
Sample application Has an available sample application for quick testing.	Yes	Yes	Yes	Yes
Client registration Requires each client to register for the service.	Yes	Yes	Yes	Yes

Table 5.2: Comparison tool part 2.

	Amazon Alexa	**Mycroft**	**Google Assistant**	**Houndify**
Authorization The method of authorizing each client.	Amazon developer account and device registration. Security profile needed on the Alexa Voice Service (AVS) developer portal.	Mycroft account and device pairing required.	Device registration required. OAuth 2.0 access tokens to authorize your device to the Assistant service.	A Client ID & Key on your device required for authorization to the service.
Create new skills The program associated and used for creating skills.	Alexa Skills Kit	Mycroft Skills Kit (CMD Terminal)	Dialogflow for dialogues, Actions by Google for actions	Houndify Custom Commands (website)
Create custom skills Allowing developers to create custom skills and how.	Yes, by defining phrase, endpoint and response.	Yes, using the Skills kit to generate basic a skill and tailor functionality using Python.	Yes. Using custom device actions. Write code in JSON and register the action making the assistant aware of the command. Device takes care of response.	Yes, can define phrases and responses. Take care of response on device
Skills language Languages available for developing skills in.	Node.js, Java, Python, C#, Go	Python	Python, C++, Node.js, Android Things (Java) + JSON	Web based for developing interpretation of utterance. Managing the received commands is done in the client.

Table 5.3: Comparison tool part 3.

	Amazon Alexa	**Mycroft**	**Google Assistant**	**Houndify**
Skill storage Where will the created skills be stored and accessed.	Alexa cloud service	Locally	Google cloud	Houndify domains (cloud)
Custom wake word Capability to invoke a custom wake word.	No	Yes	No	Yes, behind paywall
Message communication The communication protocol type.	HTTP/2 with AVS Protocol (JSON)	Mycroft's MessageBus (JSON)	gRPC -> HTTP/2 (Protocol Buffers)	HTTP (JSON)
Response package The data package type that is sent between clients and the service. Contains voice output and possibly commands.	JSON	Mycroft "Message" (Can include JSON)	AssistResponse [EventType, DialogStateOut, AudioOut]	HoundServer JSON
Media player support possibilities Available media players and integration options.	Android Media Player & Gstreamer with integration support for others if a wrapper is built.	Supports different audio players through their AudioService. Unclear if one can choose/add players.	Built in speaker + HDMI/USB + I2S	User defined/implemented solution. None available directly through Houndify.

5.2.1 Skill Structure Overview

The following headlines describe the basic aspects of handling and creating skills (see Table 5.1: row 5-6, 5.2: row 1-4, 5.3: row 1-4) for each VRC. The images associated with each component illustrate the basic structure of each VRC and their purpose is to complement the comparison tool with a simple and comprehensible visualization.

Mycroft Skills
All skills in Mycroft are created through the Mycroft Skills Kit (MSK), which is run through the command line in a terminal. All skills are generated by answering a number of queries about the properties of the skill. Each skill consists of a number of files, with one of them being a Python file that provides most of the functionality of the skill. A part of the Mycroft core called “Skills” handles all skill activities such as loading, managing, and updating skills. This includes a “Skill repository” that stores all skills on the device and loads them into memory when needed. A "MessageBus" (a kind of web-socket) is used for communication between processes (e.g. the Skills-process) within the core and other devices. There is most likely a need for some sort of “Python parser” for Android integration or to run commands on Android, due to the skills being written entirely in Python without any current support for other languages. The Mycroft core is also written in Python and does not have an officially supported Android version. There is however a companion app for Android, which servers as a client that communicates with the core. The core has an unofficial version released on GitHub which has not been proven to work when tested.

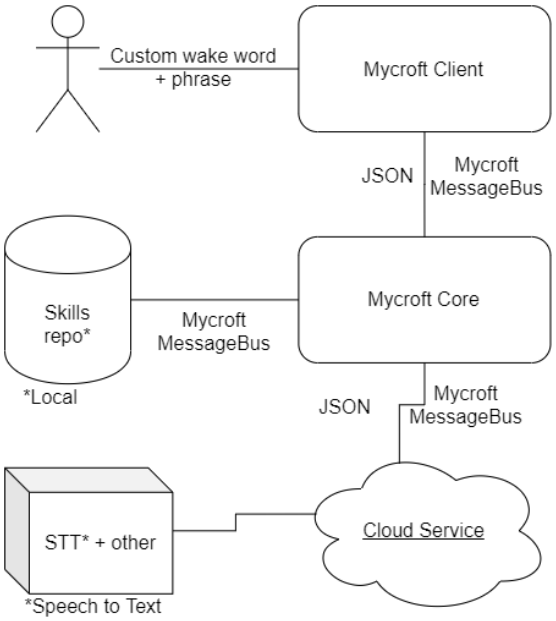

Figure 5.1: Mycroft structure model.

Google Assistant Actions
Google provides an SDK for Android Things which makes it possible to call the Google Assistant Service using gRPC (Google Remote Procedure Calls). It records a spoken request from the connected microphones, sends it to the Google Assistant API, and plays back the Assistant's spoken response on the connected speaker. The response given by the assistant has a protocol buffer containing commands that the developer can implement however they like. With Device Actions, it is possible to control hardware connected to the device. You will have to create these Actions separately using either Dialogflow or Actions SDK. Note that Dialogflow provides a Natural Language Understanding-component (NLU) while the Actions SDK does not and requires the developer to have their own solution for interpreting the user.

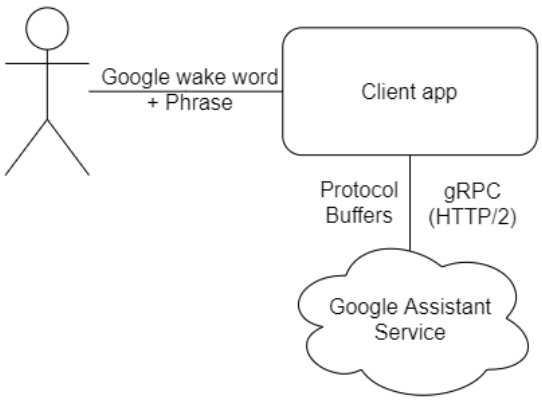

Figure 5.2: Google Assistant structure model.

Alexa Skills
Alexa utilizes a cloud-based service that accepts intents as structured requests to perform different tasks. Since the service is cloud-based, an internet connection is required for access. Alexa uses "endpoints", which is the destination Alexa sends requests to when a user invokes a specific skill. An endpoint must be provided when configuring skills. An endpoint can be either a hosted web server or a component that has some sort of functionality. What the endpoint is, depends on what kind of skill it is. In order for Alexa to communicate with a service, it uses the request-response mechanism HTTP over SSL/TLS. When a user interacts with an Alexa skill, the service receives a POST request containing a JSON body. The request body contains the parameters necessary for the service to perform its logic and gen-

erate a JSON-formatted response.

Amazon offers the "Alexa Skills Kit" for creating skills. "Request handlers" handle the requests sent to the cloud service and they define the actions that are taken based on the request type. Request routing is done automatically by the SDK, but can also be done manually by the developer. Once a request is processed, the handler sends an appropriate response (e.g. a JSON object) back to Alexa. The response can contain whatever information the developer may need to take the correct action.

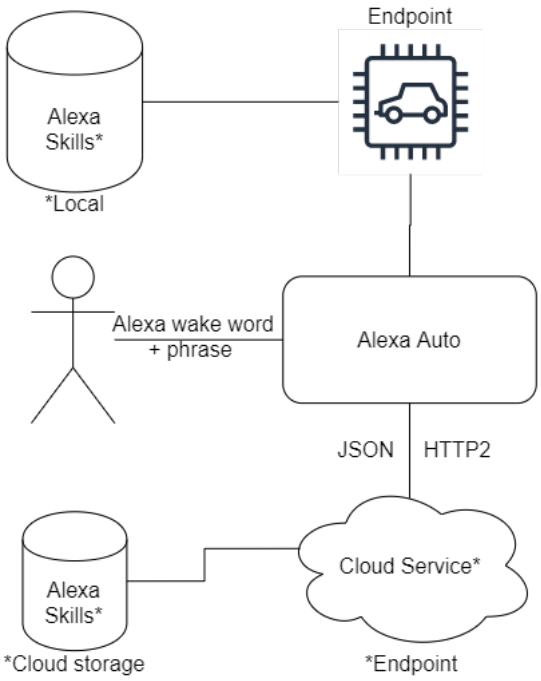

Figure 5.3: Alexa structure model.

Houndify Custom Commands
It is possible to build custom commands on the Houndify website. The commands need to specify an expression to match, and the result to be returned if the expression is matched. Responses are in JSON format and have different properties depending on the domain. The developer then decides what to do with the information stored in the JSON container. The feature of creating custom commands and domains are currently only available by invitation only.

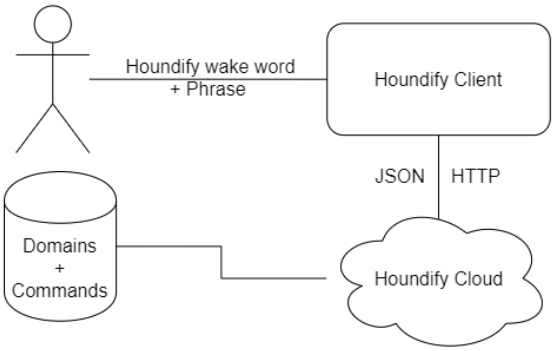

Figure 5.4: Houndify structure model.

5.3 Re-using the Process

The process that has been established (see Figure 5.5) during this thesis has been used to help solve the problem of incorporating multiple VRCs into one platform. When dealing with a problem that is complex and vague, it is necessary to split it up into something comprehensible that can be achieved within a given time frame. This section will describe the steps needed to be taken in order to re-create a similar artifact such as the one presented in this thesis, as well as a description of how to use the artifact.

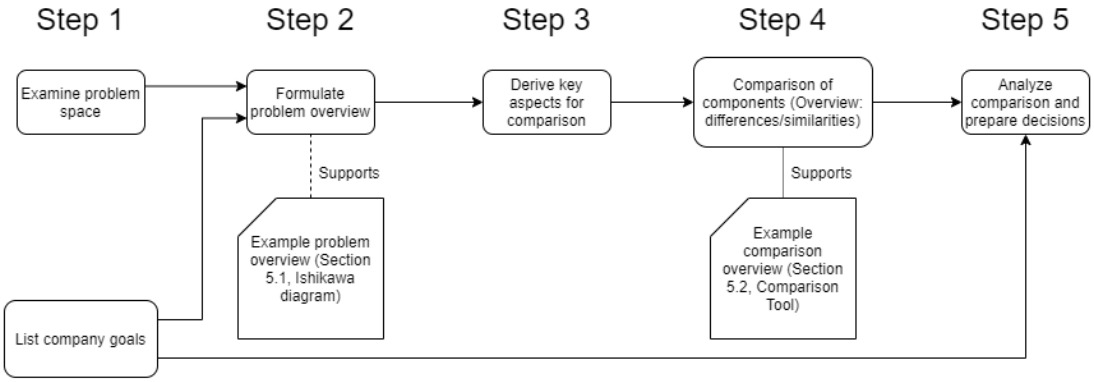

Figure 5.5: The established process.

Step 1 of the process is to investigate the problem space on an organisational level, and to compile a clear list of the goals within the company. In order to proceed to the next step, it is vital to gather information from stakeholders to better understand the problem space, and to make sure that it aligns with company goals. Gathering

information regarding the problem space and company goals should mainly be done through interviews with employees that have a relation to the problem. This can be on a technical, management, or product level. Choosing the right employees to interview is vital in order to gain the most knowledge. After the interviews has been conducted, the data should be analyzed to identify themes within the problem space to use for Step 2. The company goals are meant to be considered throughout the entire process, and to be used for making decisions during Step 5 in order to determine whether or not the solution reaches the intended goals.

Step 2 of the process is to formulate the problem overview based on the themes identified in Step 1. This can be done using a Ishikawa-diagram which intends to visualize the causes and effects that have been identified through the problem investigation. When creating an Ishikawa-diagram, begin by determining the root-cause, and then continue to add sub-root-causes. The sub-root-causes is thereafter expanded with sub-causes related to its respective branch. The Ishikawa-diagram created in this thesis could be used as a source for inspiration and knowledge when conducting this step (see Section 4.1).

Step 3 of the process focuses on using the produced problem overview to determine which key aspects to focus on. These ones should be identified as the most important and feasible for the problem space. The focus can be either be on one or multiple aspects, as long as it is feasible within the context of the project. In order to determine which aspect(s) to focus on, either utilize interviews, informal conversational interviews, or focus groups with developers and supervisors who are responsible for determining which direction to take. Always make sure that the choice aligns with the established company goals.

Step 4 of the process is where the investigation and solution design takes place. By researching each relevant component through its available resources, a comparison overview can be constructed. Focus on interviewing developers, industry experts and other people with knowledge of the components or other technical aspects. When a high-level of understanding of the components has been achieved, start investigating which parameters that are the most relevant to consider when comparing or choosing a component for the given problem space. Also conduct interviews that facilitates discussion regarding these parameters in order to find the most suitable ones. When creating the comparison overview, the presented comparison tool (see Section 5.2) of this thesis can be used to gain knowledge of which aspects that has been considered important and how the overview has been crafted.

Step 5 of the process is to analyze the comparison. This can be done either through a workshop or focus group with the developers that will be using the comparison in practice. By asking questions and discussing how to improve the comparison, it can be determined whether or not the developers find it useful for their future work.

How to Use the Comparison Tool

The comparison tool has two main purposes; to provide an overview for comparing different components in relation to each other when considering certain parameters, and to provide knowledge that allows developers to make an informed decision of what components to use, based on their platform and context. This requires knowledge about the problem space, whilst the tool highlights the most important aspects, and serves as a stepping stone for future work.

6

Discussion

6.1 Implications to Research

The study set out to answer three research questions, depicted in Section 1, as well as help Aptiv in the task of solving their problem of integrating multiple VRC into one unified framework. During the thesis work, the focus has been shifted several times due to ambiguities regarding the problem space and scope. It is also a part of DSR to iterate and improve upon the artifact during each cycle, which can in turn force the artifact to take on different shapes and evolve into what becomes the final artifact. Based on the problem investigation and the complexity of the problem, we believe that the approach we took generated an artifact that can be of much use as well as teachings that contribute to the research field.

Three research questions were formulated at the beginning of the study. The following are the research questions with motivation as to how they have been answered by the study.

RQ1: *Which challenges exist with flexible integration of voice recognition components for an existing platform meant to be sold to customers with varying demands within the automotive industry?*
The Ishikawa diagram presents the causes and effects of the problems that were identified through the interviews in cycle I. By solving the problem "Hard to exchange one VRC for another", the effect "Hard to reuse existing code between customers" can be mitigated. The problem of "Customer demands imply different VRCs" will then diminish due to Aptiv becoming more flexible. This, in turn, aids them and their "Development strategies". When designing the comparison tool (see Section 5), these aspects from the Ishikawa diagram was kept in mind together with the first research question. By allowing each component to be systematically documented using the structure of the comparison sheet, the challenges can be analyzed more clearly by the developers. Each VRC is very complex and therefore requires structure to be comprehensible. The information is not easily accessible due to over-saturation, meaning that the quantity of available information is very high but not structured and therefore confusing, as well as the fact that they are continuously developed which makes information outdated quickly. By providing clear instructions as to what kind of support each VRC has, a risk-analysis can be performed to ensure that it is feasible to integrate said VRC. Integration problems that may arise can be identified and tackled early, and the decision to not choose a specific tool because

of the effect it will have on the development times and costs can be made before resources go to waste. Due to the "black box" nature of the components, certain aspects were difficult to investigate. However, information regarding skills was not limited in the same way and could be integrated into the tool. Skills are also one of the most important aspects when considering a VRC since skill integration is what allows developers to tailor functionality to their platform and user needs. Therefore, the causes and effects related to this were very important to consider, which lead to much of the comparison tool to be about this. Even though the tool helps developers understand many of the challenges related to choosing a specific VRC, it primarily creates a format to systematically document these and to interpret how the differences between VRCs could be used to their advantage.

RQ2: *Are there any mitigation strategies capable of decreasing time, complexity and economic costs when incorporating a variety of speech recognition components?* We believe that the results of this thesis will help to mitigate the mentioned problems through the usage of the presented tool (Section 5). To find a specific mitigation strategy has been extremely difficult due to the complex nature of the problem, so therefore the problem has to be divided into smaller parts. The artifact provides a mitigation strategy that helps in the first step that needs to be taken to create a full solution, which is the research of VRCs, their requirements, and planning. It is, therefore, possible to decrease the time spent on researching different VRCs through the overview provided through the tool. By reason of enabling a way to quickly find relevant information about each VRC and its complexity, costs can be reduced through appropriate planning. Developers tackling the problem can use the research conducted in this study to get a foundation of how to start their research, which later will evolve into developing an appropriate mitigation strategy for the implementation such as a unified framework or architecture, which is the next step in providing a full solution.

RQ3: *To what extent will these mitigation strategies support software engineers and companies exposed to this process?*
Due to the increased complexity of components today, it is becoming more difficult to abstract from multiple components into one API (Application Programming Interface). The characteristics that differentiate the VRCs differ in such a way that providing a simple solution is not possible, therefore requiring more than just an API to solve the problem. This is due to many factors, such as the way each component is built from the ground up, its way of handling and managing skills and messages, the procedure of controlling different parts of a car (climate control, etc.), as well as security and authorization protocols. When keeping all of these factors in mind, it becomes extremely difficult to abstract and create a unified framework that allows for seamless integration of VRCs. This does however not necessarily mean that it is impossible to solve the problem. When looking at today's software engineering industry, companies are trying to cater to a wider range of customers. In the same way that PLE helped industry to streamline the mass production of customer-tailored products, we believe that a need to abstract more complex components is becoming more relevant for companies to satisfy a larger consumer base.

When taking into account the lack of control of the components, it is safe to say that new ways of solving this are necessary, since older methods are becoming less applicable. We also believe that the reason for the gap in research regarding the integration of COTS-components is that this problem is relatively new, and will continue to grow as more complex systems such as VRCs emerge. After discussions with industry experts, it has become clear that the problem that Aptiv faces is not unique, and that other companies face similar problems, especially within the automotive industry. Due to the lack of methods and solutions for flexible integration of VRCs, many have considered building their own custom VRC solutions instead.

The created tool tries to provide support through creating a format in which components can be investigated and compared when trying to choose which to integrate. The process used in this thesis can be used at other companies to try and replicate a comparison tool tailored to other types of components (see Section 5.3). The knowledge gained from the research can be used to mitigate possible stalls in the investigation process. By describing challenges and problems that arose regarding developing a unified framework in the Ishikawa diagram (see Section 4.1), other companies can learn from these and investigate if similar problems are present in their context. Much of the emphasis in this thesis was on the problem investigation, which displays how important it is to distinctly define the problem and its scope. The reuse of code was a fundamental part of the problem at Aptiv, with the goal being to reuse as much code as possible. This first became clear after conducting the interviews during cycle I and the creation of the Ishikawa diagram.

6.2 Implications to Practice

The content and results presented have been entirely fixated on the problem presented by Aptiv. The drawn conclusions and the help provided by the artifact are evaluated to provide a helpful contribution to their developers who are attempting to integrate VRCs into their platform. The findings of the study show that VRCs differ fundamentally when compared. Many parameters need to be accounted for, such as integration support, skill execution, and structure as well as what kind of baseline functionality they offer. All the knowledge generated by investigating these parameters can be applied to other VRCs than the ones discussed in this thesis. This in practice gives Aptiv the possibility to widen their understanding as much as they want to find the most suitable components to use by comparing their most vital parameters. After they have gathered a sufficient amount of knowledge on all the VRCs that they possibly want to use, they can successfully plan the next step in creating a full solution. As mentioned in Section 6.1, the problem that they are facing has to be divided into sub-problems, in which the tool provided solves the first sub-problem which is the investigation of VRCs, their requirements, and planning. Once an appropriate plan has been created, taken into account the complexities described by the artifact, an appropriate implementation strategy can be developed. As mentioned in the motivation to the third research question in Section 6.1, other companies can use the process described in this thesis (see Section 5.3) to create an appropriate comparison tool for their own problem space. They can also learn from

the challenges described in this thesis to possibly avoid being in the same situations.

6.3 Threats to Validity

This section will discuss the possible threats to validity regarding this thesis. The discussion will be based on the types of validity threats described in research by Wohlin et al. [28], which in turn is based on the results from the research by Cook et al. [29]. Four types of validity threats are described: conclusion, internal, construct, and external threats. The following segments will consider any threats to validity of the work process and the conclusions drawn from this thesis, as well as how any of the achieved results could be wrong [30].

Conclusion Validity

Conclusion validity is the reasoning of whether or not the conclusions drawn are correct or "reasonable". The problem space of this thesis was complex from the start, resulting in the reduction of scope multiple times. This caused the work to focus on a solution to a sub-problem rather than the original problem. By assuming that there is no way to solve the original problem of unifying all the VRCs into one framework during the time-frame of a master thesis, certain aspects might have been overlooked. There might have been answers to the complex problems encountered that would have enabled an architectural solution, but not found by the researchers.

Internal Validity

Internal validity concerns the trustworthiness of the relationship between a treatment and the given outcome that it provides, i.e. cause-and-effect. The interviews conducted during the thesis posed some threats to internal validity. The interviewees had experience within the automotive industry and with software development, but not directly in relation to VRCs. Due to this, there is a risk that some of the questions that were asked were outside of the interviewees' knowledge and therefore gave less optimal answers. Also, some of the interviewees did not have the same technical expertise within software development preventing them to clearly understand all questions and elaborate on the answers on a sufficient technical level resulting in more ambiguous answers.

Construct Validity

Construct validity is the measure of whether or not the construction of the study claims to measure the intended construct. To safeguard against threats to construct validity, some safety measures were put in place. First off, every interview/workshop always started with a detailed presentation followed with questions that were answered to prevent any confusion regarding the thesis subject and its goals. The participants were also encouraged to ask any questions that might come to mind during the interviews to make sure nothing is misinterpreted or confusing. Secondly, the pilot runs of all interviews/workshops were conducted to make sure that the

questions were clear and that the structure was proper. By simulating an interview, errors previously overlooked could be found.

Some threats to construct validity can be identified due to using informal conversational interviews. Since these interviews lack preparation and proper documentation, the results may vary. This can be problematic, but because most informal interviews were held to discuss very specific topics that only one or a few people attended, there was no need to conduct the same interview with the same questions more than once. Another aspect is that the questions are formulated as the interviews proceed, which makes it possible to miss things. It is also very dependant on the interviewers and the participants' ability to come up with questions and discuss topics, which usually is reflected in the results. When considering the iterative nature of DSR, some of these threats can be mitigated to a certain extent since the method allows for missed issues to be brought up again.

External Validity

External validity concerns how the study conducted in this thesis can be generalized across other situations. Due to conducting all data collection within one company exclusively with their employees, the generalizability of the process is reduced. The thesis is quite case-specific and mainly focuses on the selected company's problem space and platform. To validate the generalizability, the same process would need to be executed on another similar company. However, we believe that the created Ishikawa-diagram (Figure 4.1) and its analysis may aid in this process.

During the thesis, it was concluded that the artifact could be used to investigate other VRCs, using the parameters that had been identified to be important. However, other VRCs might differ in such a large way that the parameters are not applicable, therefore making this conclusion incorrect. This was not the case during the investigation, but the possibility still exists. The conclusion that having an overview of the structure and differences of various VRCs would be helpful was made and later validated by developers at Aptiv. This appears to be correct, but there might be a bias present since all the research was conducted at one company. What this means is that there is no validation confirming that the helpfulness of the artifact extends outside of Aptiv.

7
Conclusion

The aim of this study has been to investigate and find potential strategies for mitigating the implications of integrating different VRCs to an Android-based automotive platform. There are currently multiple VRCs on the market available for integration into Android units. The case introduced at Aptiv however, presented the problem of finding a way to reduce the development costs of creating a framework capable of catering to multiple kinds of VRCs for a unified framework.

The approach of conducting Design Science Research allowed the work of this thesis to start with a large scope, and through iterative work narrow down the scope in order to produce something that is manageable in the given time frame. This approach allowed for the creation of artifacts that through the iterations became better and more tailored to the needs at Aptiv. Altogether, two iterations were completed in which the first cycle mainly concerned the investigation of the problem. The second cycle utilized the knowledge gained and focused on creating a valuable artifact and evaluating its contribution. Two artifacts have been produced during the course of this study. The Ishikawa diagram was the result of the problem investigation during the first cycle. It means to visualize the problematic framing of the problem which also serves as valuable knowledge to anyone aspiring to tackle the problem of integrating multiple VRCs. The second artifact is the comparison tool containing information for developers looking to compare the differences/similarities of the four VRCs considered by Aptiv. Models were crafted in order to complement the information of the comparison tool, by visualizing the differences/similarities identified in the sheet, providing an overview.

The study aspires to answer the three research questions seen below:

RQ1: *Which challenges exist with flexible integration of voice recognition components for an existing platform meant to be sold to customers with varying demands within the automotive industry?*

RQ2: *Are there any mitigation strategies capable of decreasing time, complexity and economic costs when incorporating a variety of speech recognition components?*

RQ3: *To what extent will these mitigation strategies support software engineers and companies exposed to this process?*

Each VRC is similar on a basic structural level in how it interprets input from

the user, sends it to its service, processes it, and generates a response that is sent back to the user while simultaneously executing related functionality. It is however in the details of each of these steps where the main differences lie. Multiple challenges have been discovered (see discussion in Section 6.1) that affect how a framework would be implemented as well as how a developer would approach the integration of new VRCs. The tool that has been created can serve as a guiding post towards understanding the challenges of choosing to utilize one or more of the investigated VRCs. The parameters have been found through investigating multiple VRCs, looking for properties that must be taken into consideration. The ones found are deemed to be the most important and can be used to investigate VRCs other than the ones brought up in this thesis. By enabling developers to have a tool providing an accessible way to compare and understand the basic functionalities of VRC, a decrease in time and economic costs can be achieved preventing them from spending additional time making their own investigation. The complexity gets reduced thanks to the provided overview and knowledge that it contributes.

7.1 Significance of the Study

The findings presented in this thesis contributes both to the academic field and industry. For the academic field, it is showcased which the difficulties and challenges are regarding dealing with component integration and the process of systematically identifying similarities and differences between the mentioned components. The produced artifact presents numerous challenges faced by industry when trying to generalize a solution for COTS components.

Contribution towards Aptiv and their case has been met through the knowledge gained regarding the problem space as well as a tool meant to mitigate possible challenges for further development in this area. The parameters presented in the comparison tool can be used for future development of the framework. They can also be applied when investigating other VRCs available on the market in order to both pinpoint vital information about them and to compare them to VRCs already investigated by Aptiv. Similar parameters can be found for other types of components which in turn provides companies that have a similar problem space with a strategy.

7.2 Future Work

At the beginning of this study, the ultimate goal was to create or find possible mitigation strategies for implementing multiple VRCs into one unified framework. The work done in this thesis will serve as a stepping stone for future researchers and developers looking to continue the work. There are improvements to be done regarding both the tool and the complementary models attached to it. The identified information displayed in the comparison sheet should not be considered complete, and if continued, further investigation of the documentation and the components is advised. Changes to either documentation or the components will most likely

happen due to the continuous development. Therefore, the information in the sheet can become outdated in the future meaning both the challenges and solutions may be prone to change. Using the tool for its actual purpose would reveal more of its strengths and weaknesses, in turn expanding the knowledge of what helps Aptiv reach their goals and ultimately construct a better product.

The evaluation during cycle II provided feedback on aspects that could be included in the comparison sheet to further improve it. From the developer's point of view, an understanding of the baseline functionality of each component would provide them with an overview of what is available out of the box and what they need to implement themselves. Another topic of interest is the possibility of converting skills in order to be compatible with other components. It is unknown if this is possible, but future work in this area could prove beneficial for Aptiv and potentially other companies.

During the course of this study, attempts have been made to test the components in order to better understand their functionality. These tasks have mostly been unsuccessful, due to various reasons, and future researchers should, therefore, continue these attempts, using guidance from experienced developers. Finding and contacting technical experts is a recommendation because of the complex nature of the components, and firsthand experience will be valuable for gaining insights and knowledge.

Bibliography

[1] Ei-Wen (Victor) Lo and Paul Green. Development and evaluation of automotive speech interfaces: Useful information from the human factors and the related literature. *International Journal of Vehicular Technology*, 2013, 03 2013.

[2] Ute Winter, Tim Grost, and Omer Tsimhoni. Language pattern analysis for automotive natural language speech applications. pages 34–41, 01 2010.

[3] L. Chung and Kendra Cooper. Matching, ranking, and selecting components: A cots-aware requirements engineering and software architecting approach. 06 2004.

[4] L. Chung, Kendra Cooper, and Sam Courtney. Cots-aware requirements engineering: The care process. 01 2004.

[5] David B. Yoffie, Liang Wu, Jodie Sweitzer, Denzil Eden, and Karan Ahuja. Voice War: Hey Google vs. Alexa vs. Siri. *Harvard Business School*, page 25, 2018.

[6] The Human Factors of Speech-Based Interfaces A Research Agenda. *ACM SIGCHI Bulletin*, 27(2):61–67, 1995.

[7] Christian Müller and Garrett Weinberg. Multimodal input in the car, today and tomorrow. *IEEE MultiMedia*, 18:98–103, 01 2011.

[8] Zhang Hua and Wei Ng. Speech recognition interface design for in-vehicle system. pages 29–33, 01 2010.

[9] Human Factors and Design Issues in Multimodal (Speech/Gesture) Interface. Technical report.

[10] Ulrich Gärtner, Winfried König, and Thomas Wittig. Evaluation of manual vs. speech input when using a driver information system in real traffic. pages 7–14, 09 2017.

[11] Harry Sneed. Encapsulation of legacy software: A technique for reusing legacy software components. *Ann. Software Eng.*, 9:293–313, 05 2000.

[12] Hung-Hsuan Huang, Toyoaki Nishida, Aleksandra Cerekovic, Sedlyarov Igor, and Yukiko Nakano. The design of a generic framework for integrating eca components. pages 128–135, 01 2008.

[13] Ljerka Beus-Dukic. Non-functional requirements for cots software components. 09 2001.

[14] Rafael Capilla, Muhammad Ali Babar, and Oscar Pastor. Quality requirements engineering for systems and software architecting: Methods, approaches, and tools. *Requirements Engineering*, 17, 11 2011.

[15] D.M. Weiss and C.T.R. Lai. *Software Product-line Engineering: A Family-based Software Development Process*. Addison-Wesley, 1999.

[16] Noah Joseph. Volvo details upcoming xc90's new scalable platform. *Autoblog*, 2014.

[17] K. Pohl, G. Böckle, and F.J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Berlin Heidelberg, 2005.

[18] Eric Knauss. Constructive master thesis work in industry : Guidelines for applying design science research.

[19] Roelf J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014. 10.1007/978-3-662-43839-8.

[20] Roelf J. Wieringa. Design science as nested problem solving. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, pages 1–12, United States, 2009. Association for Computing Machinery (ACM). 10.1145/1555619.1555630.

[21] E. Goodman, M. Kuniavsky, and A. Moed. *Observing the User Experience: A Practitioner's Guide to User Research*. Interactive Technologies. Elsevier Science, 2012.

[22] Daniel W Turner III. Qualitative interview design: A practical guide for novice investigators. *The qualitative report*, 15(3):754, 2010.

[23] Jan Horsfall and Cleary Michelle. Planning and facilitating workshops. *Journal of continuing education in nursing*, 39:511–6, 12 2008.

[24] Kitchenham BA and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2, 01 2007.

[25] Michael Patton. Qualitative research and evaluation methods. *http://lst-iiep.iiep-unesco.org/cgi-bin/wwwi32.exe/[in=epidoc1.in]/?t2000=018602/(100)*, 3, 01 2002.

[26] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[27] Ishikawa K and Loftus JH. *Introduction to quality control*. Tokyo, Japan: 3ACorporation, 1990.

[28] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Computer Science. Springer Berlin Heidelberg, 2012.

[29] Ray Hyman. Quasi-experimentation: Design and analysis issues for field settings (book). *Journal of Personality Assessment*, 46(1):96–97, 1982.

[30] J.A. Maxwell. *Qualitative Research Design: An Interactive Approach*. Applied Social Research Methods. SAGE Publications, 2012.

A

Interview guideline

Introduction

1. **Personal Introduction**
   Name, education and a brief explanation of how the interview will be conducted.

2. **Thesis Explanation**
   We are working on a thesis investigating the problems Aptiv are facing when it comes to integrating the different voice recognition components available on the market into your current platform. The systems available today are not adapted to be used in vehicles and each of your customers have different demands on what exact component to use. Therefore, we are trying to increase Aptiv's flexibility towards its customers but also investigate and assess the risks and trade offs concerning having to choose between components. You have been invited to this interview to help us understand and discuss what the main challenges are and how to potentially mitigate them.

3. **Consent**
   Ask for consent regarding using audio recordings, explain what it will be used for.

Interview Questions

Questions are located in the table on the next page.

Background of Interviewee 1. Are you okay with being recorded? 2. What is your role at Aptiv and what do you do? 3. How long have you worked at Aptiv? 4. Have you had any previous work experience in the automotive industry? → If yes: How much experience? 5. What projects have you worked on previously at Aptiv? 6. What projects have you worked on previously at Aptiv?
Organisational Questions 7. Can you summarize what your department at Aptiv does?
Understanding How Component Integration Works 8. How would you describe the integration process of new software into the existing platform? Individual components such as libraries and frameworks, not entire system components. 9. Are there any notable problems that come to mind when thinking of the integration process? 10. What aspects of component integration would you like to improve, if there are any and how?
Questions on Knowledge of Voice Interfaces 11. Have you've ever used any type of voice assistant? 12. Do you agree that for you as a company, it is difficult to integrate speech recognition systems into your current platform? → If yes: What are the main concerns associated with integrating speech recognition systems?
Discussing the Problem 13. What are you doing in order to satisfy different customers with varying demands? 14. What are the biggest problems with having customers with varying demands? → If problems are mentioned: What aspects of the company does these problems affect the most? → How has problems like these been solved previously?
Discussing Solutions 15. If you want to satisfy each customer with varying demands, integrating multiple different components will be necessary. Are there any current solutions/strategies to this problem? 16. How would a framework that allows for a more streamlined process of component integration help the company? → If the interviewee needs clarification: The framework would be intended to allow for seamless integration of multiple varying components as a layer between the current platform and the components.
Wrap up 17. Is there anything else you would like to discuss or mention regarding the topic we've talked about today?

Figure A.1: The interview questions

B

Abandoned 4+1 Models

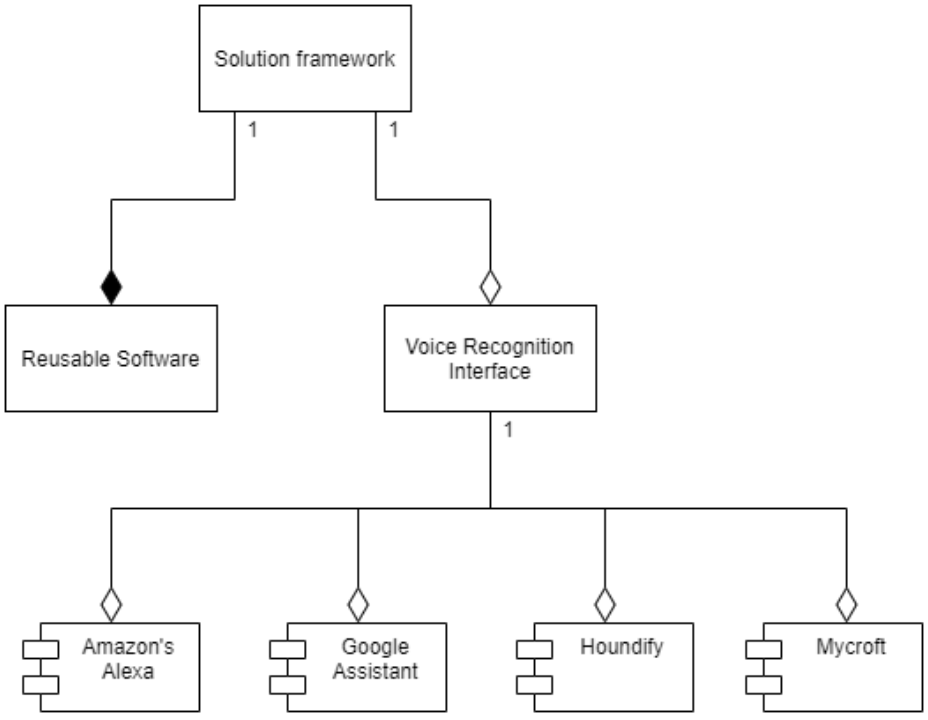

Figure B.1: Rough sketch of an interchangeable framework from the "Logical View".

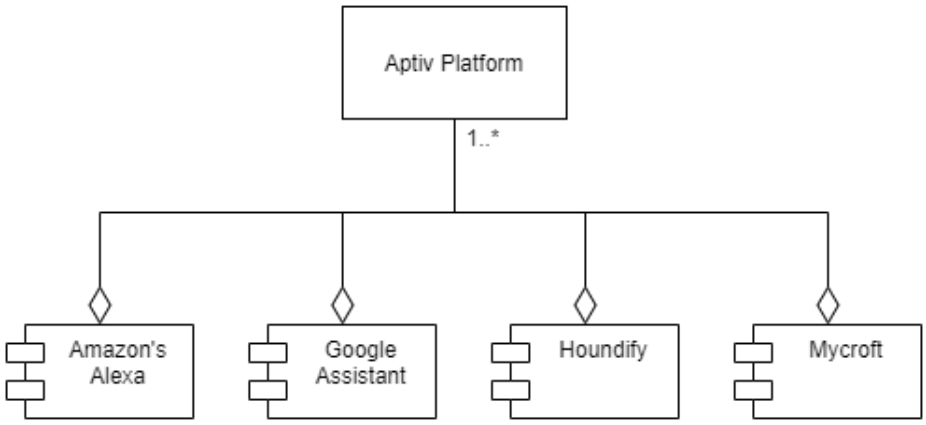

Figure B.2: Rough sketch of how a static framework would look from the "Logical View".

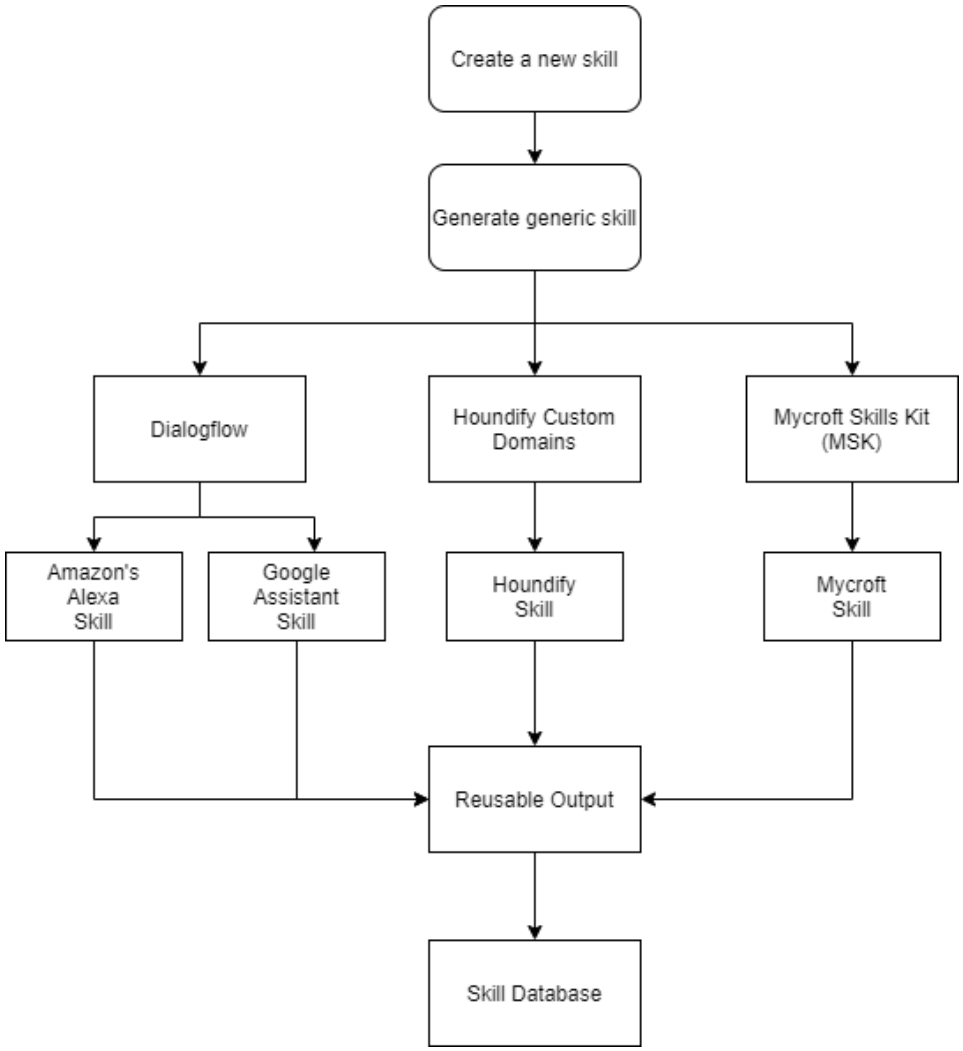

Figure B.3: Rough sketch of the process flow for creating generic skills from the "Process View".

Use-cases	Actors	Relevance (1-10	Complexity	Challenges	Goals	Stakeholders
Studying the most important parts of the documentation	Developers	8	* Lack of documentation results in relying on multiple sources	* Lack of documentation * Finding solutions to specific unique problems * All component are under development and may change	* To be able to successfully implement voice interfaces and make the process more effective	* Developers
Implementing a voice interface	Developers	7	* Lack of documentation * Proper integration	* Understanding the voice interface structure * Understanding how integrate components into the current architecture * Lack of documentation * Finding the appropriate voice interface	* To be able to provide a voice interface to customers * To be able to offer forefront technology	* Aptiv * Aptiv's customers
Choosing a voice interface to integrate	Customers, Reqs. Engineers	5	* Licenses and agreements * Understanding differences	* Knowing what each voice interface provides * Knowing what they want from the voice interface * Knowing the risks and tradeoffs between different voice interfaces	* Offer an attractive voice interface provider to their customers * Having a custom tailered voice interface	* Aptiv's customers * End-users
Creating a voice controlled UI	Developers, Designers	5	* Usability * Understanding how to work with voice components from a design aspect	* Separating the UI from the voice engine component * Making a UI that does not distract the driver * Validating the design to avoid missing functionality and visual representations	* To have a visual representation tailerd to our vision * To provide visuals corresponding to the voice interface	* Aptiv * Aptiv's customers * End-users
Building a unified framework for VRCs	Developers	10	* Multiple different components in one unifying architecture	* Understanding the implementation of each component * Creating an understandable description of how to use this framework * Lack of documentation for components * Encapsulating differences in components while still having reusable code * Changes to each component may cause changes to the framework	* Not having to re-implement each component individually * Save time and resources * Generic UI for customer flexibility and minimized workload	* Developers * Designers * Aptiv's customers
Creating a framework for generating skills/actions usable with multiple VRCs	Designers	7	* Very different approaches to each action/skill * Setup varies a lot, some require a server etc.	* Understanding how each skill/action is created * Lack of documentation * Making the output usable * Understanding the differences	* Not having to create skills/actions for each interface * Save time and resources * Do not have to learn each way of creating skills	* Aptiv

Figure B.4: Use-cases from the "Scenario View".