



Graduation thesis submitted in partial fulfillment of the requirements for the degree of  
Master of Science in Applied Sciences and Engineering: Computer Science

## **AN EXTENSIBLE VIRTUAL ASSISTANT FOR THE POWERPOINT PRESENTATION TOOL**

**KUSHAL SONI**

**Academic year 2016 - 2017**

Promoter: Prof. Dr. Beat Signer  
Advisor: Reinout Roels  
Science and Bio-Engineering Sciences





Proef ingediend met het oog op het behalen van de graad van  
Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

## **AN EXTENSIBLE VIRTUAL ASSISTANT FOR THE POWERPOINT PRESENTATION TOOL**

**KUSHAL SONI**

**Academiejaar 2016-2017**

Promotor: Prof. Dr. Beat Signer

Begeleider: Reinout Roels

Wetenschappen en Bio-ingenieurswetenschappen





## Abstract

Presentation tools are widely used in lots of different fields to contribute information and knowledge to specific target groups, the audience. Since these tools deliver concepts and understandings to the audience, it is important to keep on developing the technologies involved in these tools so that they can be used effectively. The user interfaces of most popular presentation tools offer limited functionality. Most of them only offer visualisations of static content. Presenters often prepare their presentation and include a lot of content that they think is relevant. However, sometimes the need arises to show content that was not foreseen beforehand. As an example, the presenter might want to show content in response to a question from the audience. To address this issue we have investigated and implemented the use of a virtual assistant in the context of presentations. The assistant allows presenters to use different modalities to request additional content that was not added to the presentation beforehand. Next to simply retrieving additional content the assistant is also capable of other actions such as navigation or providing support for the presenter. The resulting framework was designed so that both modalities and actions can be extended via a plug-in architecture allowing the assistant to be adapted for all kinds of interesting use cases.

## Acknowledgements

First of all, I would like to express my appreciation and gratitude to my advisor Reinout Roels and my promoter Beat Signer for the smooth help and cooperation. At any time, they were ready to help and advice me. The realisation of this thesis was possible thanks to them.

Second, I would like to thank my parents for their continuous support, love, help and patience. Their support and belief was a great motivation to me.

Third, I want to thank my friends who helped me to enjoy my free moments. Those moments made my mind relaxed and gave me courage and new energy for the next day.

Finally, I want to thank that one special person who was supporting and motivating me daily. Thank you for encouraging me!

# Contents

## 1 Introduction

## 2 Related Work

2.1	Presentation Tools . . . . .	5
2.1.1	Microsoft Powerpoint . . . . .	5
2.1.2	Apple KeyNote . . . . .	7
2.1.3	Prezi . . . . .	7
2.1.4	MindXpress . . . . .	8
2.1.5	Collage . . . . .	9
2.1.6	Review . . . . .	9
2.2	Virtual Assistants . . . . .	10
2.2.1	Google . . . . .	10
2.2.2	Apple Siri . . . . .	13
2.2.3	Microsoft Cortana . . . . .	14
2.2.4	Amazon Echo . . . . .	15
2.2.5	Facebook Bots . . . . .	16
2.2.6	NALP . . . . .	17
2.2.7	Dynamic Duo . . . . .	17
2.2.8	Baidu Duer . . . . .	18
2.2.9	Interact . . . . .	20
2.2.10	E-commerce Applications . . . . .	20
2.2.11	Digital Personal Coach . . . . .	22
2.2.12	Wellness Assistant . . . . .	22
2.2.13	PANDA . . . . .	23
2.2.14	Review . . . . .	23
2.3	Conclusion . . . . .	24

## 3 Interacting with Virtual Assistants

3.1	Speech . . . . .	25
3.1.1	Speech Recognition . . . . .	25
3.1.2	Silent Speech Recognition . . . . .	26
3.2	Gestures . . . . .	28

3.3	Remote Control . . . . .	28
3.4	Graphical User Interface . . . . .	29
3.5	Academic Findings and Conclusion . . . . .	29
<b>4</b>	<b>Towards an Extensible Presentation Assistant</b>	
4.1	Requirements . . . . .	31
4.1.1	Dynamic Actions . . . . .	32
4.1.2	On-Demand Content . . . . .	32
4.1.3	Influence Presentation Navigation . . . . .	32
4.1.4	Input Modalities Should be Modular . . . . .	32
4.1.5	Terms Should be Modular . . . . .	32
4.1.6	Processing Engine Should be Independent of the Modal- ity . . . . .	33
4.1.7	Main Modality vs Alternative Modality . . . . .	33
4.2	The Assistant Framework . . . . .	33
4.2.1	Design Decisions . . . . .	33
4.2.2	Natural Language Processing Engines . . . . .	34
4.3	Architecture . . . . .	35
4.3.1	Modalities and Interaction Interface . . . . .	35
4.3.2	Terms and Actions . . . . .	36
4.3.3	Trigger . . . . .	36
4.3.4	Core Logic . . . . .	36
4.3.5	Plug-In System . . . . .	37
<b>5</b>	<b>Implementation</b>	
5.1	The Extensible Assistant . . . . .	39
5.1.1	Vocabulary . . . . .	39
5.1.2	Application Structure . . . . .	43
5.1.3	Trigger . . . . .	46
5.2	Plug-In System . . . . .	46
5.2.1	Input Modalities . . . . .	47
5.2.2	Terms . . . . .	50
5.3	Implementation in PowerPoint . . . . .	54
5.3.1	Office JavaScript API . . . . .	54
5.3.2	Architecture . . . . .	55
5.3.3	Interaction with Java . . . . .	55
5.3.4	Usage . . . . .	57
5.4	Libraries . . . . .	57
5.5	Usage . . . . .	58
<b>6</b>	<b>Use Cases</b>	

6.1	Implemented Action Plug-Ins . . . . .	59
6.1.1	YouTube . . . . .	59
6.1.2	Slide Navigation . . . . .	60
6.1.3	Picture Plug-In . . . . .	60
6.1.4	Search Plug-In . . . . .	60
6.1.5	Maps Plug-In . . . . .	61
6.1.6	Directions Plug-In . . . . .	61
6.1.7	Weather Plug-In . . . . .	61
6.1.8	Browser Plug-In . . . . .	61
6.2	Writing Plug-Ins . . . . .	61
6.2.1	Writing Modality Plug-Ins . . . . .	61
6.2.2	Writing Action Plug-Ins . . . . .	64
<b>7</b>	<b>Evaluation</b>	
7.1	Closed Questions . . . . .	67
7.2	Comments and Suggestions . . . . .	68
7.3	Limitations and Improvements . . . . .	69
<b>8</b>	<b>Conclusion and Future Work</b>	
8.1	Conclusion . . . . .	71
8.2	Future Work . . . . .	72
8.2.1	Future Action Possibilities . . . . .	73
<b>A</b>	<b>Usage</b>	
A.1	Sideloaded Add-Ins with PowerPoint JavaScript API . . . . .	79
A.2	Usage . . . . .	82
<b>B</b>	<b>Images</b>	
<b>C</b>	<b>Code Listings</b>	
<b>D</b>	<b>Features of Virtual Assistants</b>	
<b>E</b>	<b>Add-ins</b>	
E.1	PowerPoint . . . . .	95
E.2	KeyNote . . . . .	100

# 1

## Introduction

Presentations are very popular nowadays and are being used widely. They are an important tool for speakers to present their work or ideas to a dedicated audience. The topic “Presentation Tools” in computer science tries to continuously improve the techniques used to give presentations. It does this by making certain tasks more easy for the presenters and/or the audience, or by giving feedback to the presenters about their performance. Given the popularity of presentations in many fields, it is important to keep developing this topic. Popular presentation tools widely used are Microsoft Powerpoint, Apple KeyNote. However, certain important aspects are missing in these tools. First of all, we are obligated to use the linear navigation offered by the tools, forcing us to go through the slides one by one. Skipping over certain slides in between is not possible. Second, these tools are based on the physical slides, from olden days. This gives us constraints in terms of space and placement of content, which is not actually necessary, taking into account the big screens we have nowadays. Third, it is not possible to create links between different content from different presentations. To be able to achieve this, we have to open different presentations together. It would be more convenient to have a way of interlinking content through different presentations. Lastly, it is not always possible to foresee all relevant content in one presentation, due to space and time constraints. The presenter can put a lot of relevant content on the slides but sometimes the need arises to do or show

---

something that was not foreseen beforehand. It is also not always possible to include certain content beforehand, for instance if it changes frequently. To address this missing functionality we introduce the use of dynamic content and on-demand information retrieval and appropriate action performing, in presentation tools. It is one of the most effective, long-lasting solutions which allows that unprepared content can be queried and shown at any time.

Consider the following example. A course is given by a lecturer who prefers interactive classes. While preparing slides of the lecture, the lecturer can not add all details of all parts of the course, since that would not fit in the foreseen time slot. Instead, the lecturer highlights some parts more than others, and thus will add more slides for these parts. However, the way the lecture evolves, depends on the students their interest and responses to certain questions which the lecturer asks them during the lecture. The lecturer will probably not have material prepared for all types of questions or some material might be outdated. In this case, dynamic and/or on-demand content is desired. The content should be adapted to the questions/interests of the student and this can only happen on the fly.

Current presentation tools allow the use of add-ins, which increase the functionality of content placement. A minor part of those add-ins even do allow dynamic content operations. However, many of those add-ins are designed in different ways by different developers. In this system, flexibility is missing, since each add-in can handle only its own specific action. Most of the add-ins require a keyboard and/or mouse interaction. They require the user to use the PowerPoint user interface for interaction, which makes the use of the add-in during a slide show hard. The add-ins are not designed in an extensible way as well, each add-in has to be triggered in its own customised way. There is no global system which allows one add-in to detect and execute the correct requested action for the user, with the help of an easy interaction interface.

We have created a virtual assistant, to be used in presentation environments. It allows users to request and show dynamic content and retrieve information on-demand, during a presentation without having to close the presentation tool. The users can interact with the tool in multiple ways, since it offers a plug-in mechanism which supports adding new interaction methods at any time. The tool currently supports speech and keyboard interaction. Input from the user is being compared to a built-in dictionary, which compares it with stored **Terms**. These **Terms** can be added at any time by the help of plug-in mechanisms. Each **Term** is supported by an **Action**,

which will be executed when the corresponding terms are detected in user input. Currently, web pages can be viewed, YouTube videos can be played, images can be searched and slide navigation in PowerPoint is supported. The application is an extendable common interface which supports multiple modalities, giving developers the opportunity to add code at any time, for modalities, terms and actions by the help of the plug-in mechanism.

By creating a virtual assistant which could execute actions while you are presenting, we realise a way for the presenter to show the content he wants to show, depending on the context. If the audience experiences unclear statements, questions or just wants more information, presenters now have a virtual assistant at their service, which allows content visualisation of any kind.





# 2

## Related Work

In this section we investigate relevant work in the area of presentation tools and virtual assistants.

### 2.1 Presentation Tools

We provide an overview of the most used presentation tools and discuss their limitations.

#### 2.1.1 Microsoft Powerpoint

Powerpoint is one of the most popular presentation tools in all kinds of fields [1]. User groups are school teachers, students, lecturers, professors, business analysts and many more. Despite the fact that it has many limitations, such as the fact that it bases itself on traditional slides from olden days and linear navigation, it is the most used presentation tool worldwide. Lots of new presentation tools were developed to overcome these limitations. Still, because of its growing popularity over many years, Microsoft PowerPoint is preferred among most of the users.

**Add-ins** PowerPoint has the possibility to load add-ins, small external compiled code which enables new representations of content in PowerPoint.

An example is the conversion to SVG. This add-in converts PowerPoint presentations into a compressed Scalable Vector Graphics format [2]. The add-in was inspired by so-called feature phones, mostly used in South Africa. Such phones have more constraints than smartphones. They can not play a PowerPoint presentation for instance. For a list of add-ins please consult the appendix<sup>1</sup>.

**Limitations** PowerPoint offers the possibility to insert basic content, like text, images, audio, video and simple graphics like charts, tables and SmartArt on slides<sup>2</sup>. Despite the fact that we can create basic presentations with above mentioned formats, there are certain limitations in this presentation tool. First, without any external add-ins, only local content can be inserted. This means content already stored on your computer. If you wish to embed content from the web, you need to first download it. Second, there is no possibility that the content can adapt itself based on the context like your location, the current time or year. For example, when talking about currencies during an economics class, we might want to know the current exchange rate. Currently, this would require you to close the presentation tool, open a web browser and go to e.g. “www.xe.com”. Third, the availability of on-demand content is missing. Once we started our presentation, there is no way to adapt it towards the current situation and context. We have to follow our slides, even if the audience is guiding the presentation towards a different, unexpected angle. We can not show highlights of today’s news if something relevant to our field has happened, without closing the presentation tool. For example, supposing you are giving a class on internet security and it seems that breaking news occurs during your class: more than 99 countries of the world have been infected with a new ransomware virus. It might be worth to mention the problems and potential solutions regarding this virus. Fourth, certain add-ins are available which solve (parts of) these problems. For example, the web video plugin<sup>3</sup> in PowerPoint makes it possible to embed web videos. Even then, each problem requires you to install a new add-in, causing discomfort to the user. Also, each add-in requires different trigger methods, and offers different interfaces. When the user wants to retrieve on-demand information, he has to know beforehand which add-in

---

<sup>1</sup><http://www.skilledup.com/articles/30-best-add-ins-and-apps-for-microsoft-powerpoint>

<sup>2</sup><https://support.office.com/en-us/article/Compare-PowerPoint-features-on-different-platforms-90986850-227c-4b25-938e-1c5838166b8b?ui=en-US&rs=en-US&ad=US>

<sup>3</sup><https://store.office.com/en-us/app.aspx?assetid=WA104221182&ui=en-US&rs=en-US&ad=US&appredirect=false>

to use, close the presentation tool, and search for the components of that add-in in the PowerPoint GUI. Much easier would be to have one extensible system which does the “add-in” selection for you and which offers multiple modalities.

### 2.1.2 Apple KeyNote

KeyNote [3] is comparable to PowerPoint. It is based on the same mechanism of linear navigation and slides. It has some features which are not directly supported by PowerPoint, like collaboration. Different persons on different devices can work on the same presentation at the same time. You have an overview of who is working when and you can see changes occurring in real time. An advantage is that everything is synchronised on the iCloud, which makes it possible to edit the presentations from a browser. This means users of other operating systems can edit and work on the presentations as well. It is possible to make changes on iPads, iPhones as well, not only on personal computers. Keynote Live makes it possible to watch presentations in real time on many different Apple devices, around the world. Specially useful when (parts of) your audience is not in the same room as you are. KeyNote offers the possibility of add-ins, in the same way as PowerPoint does.

KeyNote might have certain built-in features which PowerPoint does not have. Even then, all the limitations described in the PowerPoint section, e.g. lack of dynamic and on-demand content, also applies for KeyNote.

### 2.1.3 Prezi

While browsing presentations used by common tools which use linear navigation, a problem which occurs is lack of overview. The only thing that could help us to follow the presentation navigation are titles and slide numbers. Even then, the latter is often forgotten. Prezi [4] is a presentation tool which uses a different approach than most common tools. Presentations are created in such a way that the audience can always keep track where the current slide is situated in the presentation topic. Special zoom techniques are used to achieve this overview. At first, a zoomed out overview of the presentation is given. For each step, the context of the target slide is clearly shown by zooming in to a particular area. When a particular topic is finished, the tool zooms out to the original overview and zooms in on the next topic. Zooming techniques are important to consider while trying to develop tools which try to tackle problems like linear navigation in presentation tools. Since the use of these non-linear navigation techniques, it is also possible to jump to sections directly by just using mouse or keyboard to zoom out from the current

area and in to any other area. You can create backup areas as well, which are not originally meant to be shown, but which could be shown on request of the audience. Prezi has tools which analyze your success as well. Know who is viewing, what they are interested in and more.

In contrary to PowerPoint and KeyNote, Prezi does not offer the notion of add-ins and hence suffers from the same limitations as PowerPoint and KeyNote. However, since it is web-based, it might be possible to do slide manipulation and create a virtual assistant for Prezi.

### 2.1.4 MindXpress

MindXpres [5] competes with other popular presentation tools like Microsoft Powerpoint, Apple Keynote and Prezi. Existing presentation tools show some shortcomings. They are based on the traditional slides as we know them from the traditional slide projector, but actually there is no need to limit ourselves to the constraints of a slide. Mostly presentations are being shown on big screens, televisions, electronic projectors. This implies there is a huge space which is not being used efficient enough. But this is not the only constraint of current presentation tools. They are being used in a linear manner, where we go from the current slide to the next slide and so on. Jumping over a few slides is not possible without pressing buttons on your remote control that many times. Further, there is no way to link related important content between different presentations. You are obligated to copy and paste slides between these different presentations. Correlating presentations on-the-fly is not possible at all, without having to switch between different open presentations.

MindXpres has its own document format, based on Extensible Markup Language (XML) [6]. There is a clear separation between content and visualisation, which allows the user to have different visualisations for the same content for different occasions, or reuse content later on. The MindXpres document format is being converted by a compiler into a specified output format. Dynamic as well as static output is available. Further, there are ways to introduce new features in the tool. Its architecture is built in such a way that you can easily extend the tool by so called plug-ins. The presentation engine is based on HTML5, CSS3 and JavaScript.

MindXpres is a presentation tool which is based on plug-ins, which makes it easy to integrate a virtual assistant in it. However, no plug-ins have been developed yet which realise the delivery of on-demand content.

### 2.1.5 Collage

Collage [7] is a very simple presentation tool which allows blackboard interaction and viewing of textbooks and notes on a presenter screen. It enables teachers to give lessons with maximum efficiency by a small preparation. It is designed especially for primary and secondary schools. They state that even though over 6 million teachers worldwide use PowerPoint, there is not much research about the effectiveness of using PowerPoint in classrooms. They created their own tool for use specifically only in school environments. The teachers who have used the tool prefer it above other popular presentation tools as PowerPoint.

The current setup requires a lot of interaction of teachers with the tool, even more than PowerPoint. Teachers have to use their mouse and keyboard for dragging, zooming, transitioning, ... Not only on-demand and dynamic content functionality is missing, but also the ability to control the built-in features in this presentation tool without complex mouse and keyboard operations.

### 2.1.6 Review

The popularity of current presentation tools is diversified. Most people still use PowerPoint, even when taking its limitations into account. Some presentation tools have a similar concept such as PowerPoint, like Keynote. Others have a different concept and offer non-linear character as well.

A more interesting limitation in all presentation tools is the lack of any type of assistant which would allow presenters to interact with the presentation tool on-demand and allow automatic actions during the presentation. External programs need to be launched outside the presentation tool to show related content. The current presentation tool needs to be closed and reopened to show other related presentations, causing discomfort and waste of time towards the audience and the presenter.

**Add-ins** Certain presentation tools like PowerPoint do have an add-in system. A way to insert external compiled code to enable new features in the presentation. An example is given in the beginning of this section and a list of add-ins is given in the appendix.

Limitations of these add-ins are the missing of on-demand and dynamic content. This is one of the most valuable needs for presenters where not much research has been done. The creation of a global extensible system which allows previewing dynamic and on-demand content would increase the

usability for users.

There are a few that do allow some sort of dynamic content. However, they do not provide flexibility and/or modularity. In contrary to the available add-ins, we need an extensible virtual assistant where plugins can be written for controlling the interface (by defining multiple modalities) and for executing actions.

## 2.2 Virtual Assistants

Virtual assistants, also called personal assistants, intelligent assistants or intelligent automated assistants are automated robots which provide answers to questions, queries or requests from users and provide relevant information in return, by adapting to the user and their contextual situation [8]. Sophisticated ones are implemented by means of an Artificial Intelligence (AI), since they deduce intelligent conclusions based on inputs from users. Assistants should provide a user interface in which the user can communicate with the them, in the form of questions, queries or requesting actions in natural language.

Not much research has been done in the field of virtual assistants in presentation tools, so we have decided to widen our research field to virtual assistants in general instead.

### 2.2.1 Google

#### 2.2.1.1 Google Now

Google Now <sup>4</sup> is a smart engine which helps us with all kinds of different tasks, needs, interests and more, available on most Android phones. Things to do and things around us such as restaurants can be discovered and the best ways to get there. Reminders can be set for important appointments, texts, emails can be sent and even flight tickets can be booked for you. According to your location it gives you updates as well for the weather, for example. Speech is used as main input, but you can use the touch keyboard of your smartphone as well. It comes pre-installed on most Android phones, but is also available in the Play Store <sup>5</sup>, as well as in the Apple Store <sup>6</sup> for iPhones. Google Now offers certain basic interactions as mentioned in this section.

---

<sup>4</sup>Google Now - <https://www.google.com/search/about/learn-more/now/>

<sup>5</sup><https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox&hl=en>

<sup>6</sup><https://itunes.apple.com/us/app/google-search-made-just-for-mobile/id284815942?mt=8>

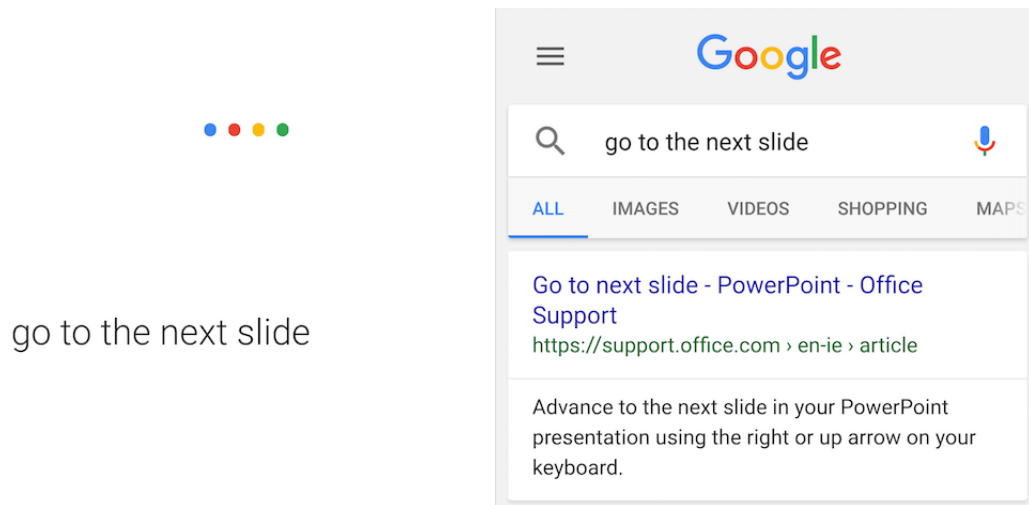


Figure 2.1: Google Now: Next Slide

However, at the moment it cannot deal with more sophisticated interactions or unknown domains. Most of the advanced interactions just result in a web search. For example commands to control presentations (Figure 2.1). Since Google Now is only available for smartphones and tablets, its use is limited for the environments offered by these devices. Presentations are mostly given from portable computers and not from phones or tablets. Even if they do, or if they would in the future, Google Now does not offer any integration with a presentation tool. Hence, it is impossible to use this system in presentation environments, since the tool is not available for computers and it does not offer any interaction with presentation tools.

### 2.2.1.2 Google Allo

Google recently launched a new messaging platform “Allo”. It is not just a simple messaging app, but it has smart added functionality, compared to other messaging apps like Facebook Messenger <sup>7</sup> and WhatsApp [9]. One of its most promising new features is the “Google Assistant”. Along with other contacts, it appears in your messaging menu. You can ask it all kinds of information, e.g. which restaurants are nearby or which movies to check out. Even within conversations between you and your contacts, the Google Assistant is always ready to help. Just type “@google” followed by the query and the results will be shared within your conversation [10]. Google Allo is similar to Google Now, but it is even more focused on web search, especially

<sup>7</sup><https://www.messenger.com/features>



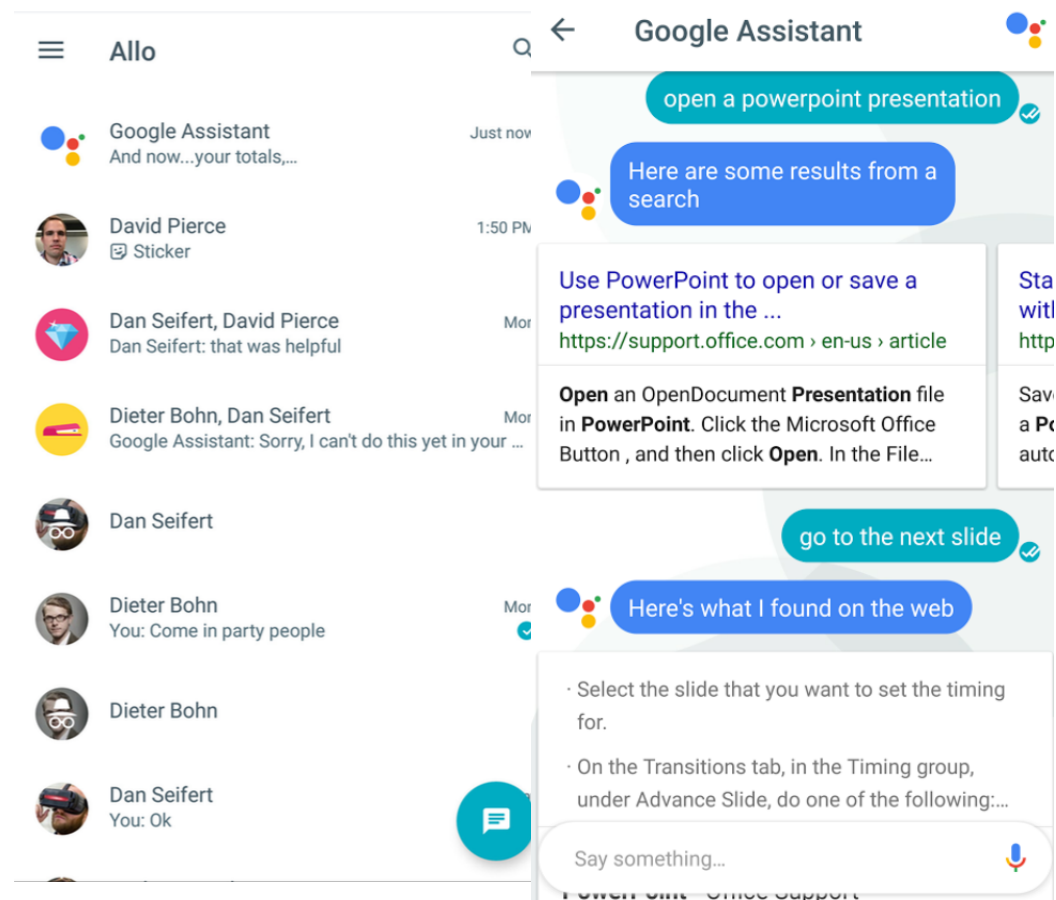


Figure 2.2: Google Allo: Next Slide

in conversations with other contacts. In a conversation with the “Google Assistant” itself, you can still make changes to your phone settings for example, which you can not do in conversations with others.

Since its functionality is similar to the one of Google Now, it has the same limitations as Google Now has. Sophisticated features are not supported and are caught by web search, as a type of backup. Hence, we can not use it to control presentations and presentation tools. See Figure 2.2.

### 2.2.1.3 Taba

Search engines are widely used for answering information needs from users. Researchers have been continuously working on providing the most useful information for users. However, most search engines nowadays are only partly personalised. Instead, users should automatically get personalised updates of

their interests. Researches have created “Taba”, a personal assistant which consists of two parts [11]. A user modelling system and a recommendation system. The modelling part collects personalised information from the users, their habits, their interests and tasks which they perform. The recommendation system provides updates to users based on the user modelling. This decreases the need for the user to re-perform the same searching task over and over again, to get the most recent information. Instead, Taba will collect information and deliver relevant updates to users.

An advantage of this system is that its concept is based on adaption to users. However, support for operating functionality of presentation tools is lacking.

### 2.2.2 Apple Siri

Siri is a personal assistant, originally for mobile Apple devices running iOS, but recently also for MacBooks running macOS Sierra [12, 13, 14]. It is very much comparable to Google Now. It has the same or similar functions, placing phone calls, turning WiFi on or off, enabling flight mode, making dinner reservations, show you the best route to your home. It is in direct contact with web services such as Wikipedia, Shazam, Yelp, Rotten Tomatoes. We can ask Siri for information such as today’s weather or shares in the stock market. We can even correct Siri, for example when it mispronounces a name.

In contrary to assistants from Google, Siri has recently been launched for certain computers of Apple, as mentioned. This makes Siri a potential possibility as a virtual assistant for presentation tools like PowerPoint, or maybe KeyNote, as this tool finds its origin with Apple. However, unfortunately, the biggest problems amongst most of the current tools is also found here: there is no support for controlling presentations. Figure 2.3 illustrates the user interface of Siri on macOS and iOS.

**Speech Recognition Comparison** Above assistants might not be what we are looking for after all. However, some components of them like speech recognition could be useful in a future creation process of our own virtual assistant. In this section we compare systems of Apple and Google. In an experimental evaluation of Apple Siri and Google Speech Recognition, researchers have focused on the reliability of the network connection during speech recognition and analysing [15, 16]. In network systems, packet loss can occur. This may result in delays and thus affecting the real-time experience. The accuracy of voice recognition is another important factor which may affect the real-time experience of a user. Researchers have tested Google Speech Recognition (GSR) and Apple Siri separately under difficult network

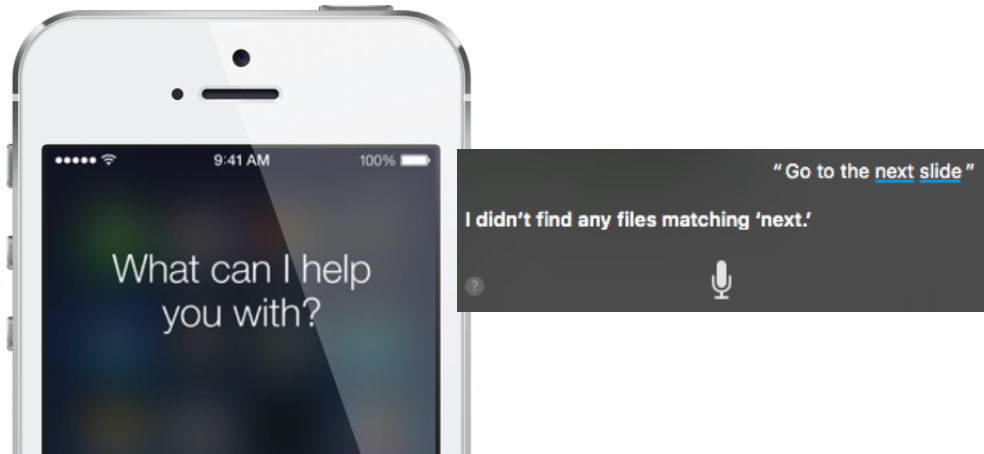


Figure 2.3: Apple Siri: Next Slide

conditions, and concluded that there is a significant loss, but the effect on packet loss is mostly caused by the application itself rather than issues with the network. Further comparing both, their results show that GSR compares better than Siri when measuring the delay.

### 2.2.3 Microsoft Cortana

Cortana [17] is the voice assistant from Microsoft, used on Windows and Windows phones. We can ask Cortana to play a song, open an app, show you the weather, retrieve flight information, etc.. . It is quite comparable to Apple Siri and Google Now. However, it is the youngest one of the three, meaning it is still in early development phase. It is a flexible personal assistant because it is built for both mobile and desktop use. However, it seems to be less accurate or smart in some cases than the competitors, as it will often produce basic web search results if it does not understand something [18].

**Context** Many different environments are used which result in web search. Not only a browser, operating systems or desktop applications, but virtual assistant environments as well. For these assistants it is often difficult to know the difference between queries with a local execution and queries with a remote execution. Local execution means all types of functions related to the user device. For instance turning on Wifi or Bluetooth. Remote executions are functions which are searched on the Web. When a query is issued, the virtual assistant has to decide to execute it locally or remotely, which can be a difficult job. For instance, they have to detect the difference

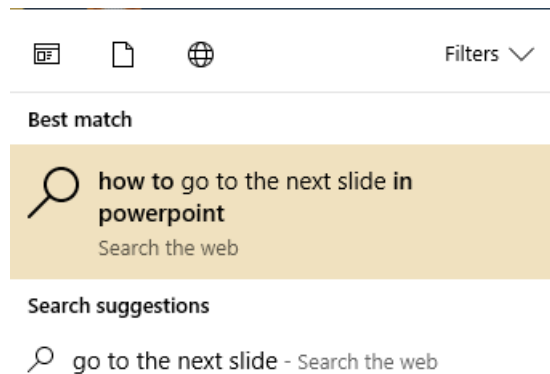


Figure 2.4: Microsoft Cortana: Next Slide

between the query “How do I mute this phone?” and “Who is Isaac Newton?”. Some queries result in a mix of remote and local executions. Mostly, these are queries in which the context should be considered. For example, “Where can I eat sushi?” should be translated by the assistant to “Where can I eat sushi in Pisa Italy?”, after it has determined the user’s GPS location by help of the device (e.g. smartphone). Researchers have studied how to reformulate these queries, so that the execution environment and context is considered and included in the resulting queries [19]. They have shown that such queries can be reliably detected, that there is a significant web traffic, containing such queries and that reformulations are possible in order to improve performance. Their work is based on web traffic originated from Windows Phones, containing the Microsoft Cortana virtual assistant. See Figure 2.4.

**Review** Cortana could be used as a potential virtual assistant for presentations because it runs on windows computers, just like Siri runs on Apple computers. However, their limitations are the same. Cortana does not support presentation controlling.

#### 2.2.4 Amazon Echo

Amazon Echo [20] is a rather different kind of technology. Instead of the voice recognition operating in an OS in your phone or computer, it is located inside a simple audio speaker. It is not just another personal assistant who delivers information to the user from the Internet, but more than that. Different hardware devices from different manufacturers can be interconnected, enabling a home automation control system. The light could be switched on,



Figure 2.5: Amazon Echo

the tv could be turned on or the air condition could be started. Some people seem to use it in daily live continuously and it has great reviews. Echo is a standalone device and does not operate on any famous phone or computer operating system. In order to serve as an assistant for presentations, it would need a technology communicating with the device containing and previewing the presentation. It would need a technology to manipulate the slides on that medium as well. This is currently not supported by Amazon Echo. In Figure 2.5 an illustration of the Amazon Echo device is shown.

### 2.2.5 Facebook Bots

Virtual assistants and automated tasks are becoming more and more popular. In the above mentioned use cases, assistants are mostly used for general scenarios like asking the weather, calling your friends, listening music or retrieving directions.

Apart from these general interactions, a new concept has been introduced recently, so called “Smart Bots”. These are a type of assistants which are created to represent a customer service. They are designed to handle complicated user queries and reply them in a correct and user-friendly way. Examples are: your personal fitness coach, a travel agent, a bot keeping track of your health, a bot giving you stock price alerts or even a bot which helps you to meet the right business consultant. Google Assistant, Amazon Echo, Facebook Messenger and other popular platforms have plans to use these bots. On Facebook Messenger, these bots are already available [21, 22].

Facebook Bots are comparable to the Google Assistant in the Google Allo

system. However, since most of the bots are a representative of a firm external to the user, their focus is on information retrieval from the web. This means local functionality, like changing phone settings is not supported. Hence, they will not be able to manipulate local presentation tools as well.

### 2.2.6 NALP

“NALP” (Navigating Assistant for Large Display Presentation using Laser Pointer) is an Human-Computer interaction technique used in large display environments [23]. Users can draw gestures with their laser pointer, and thus manipulate the slides or execute custom commands. They offer integration with presentation tools such as PowerPoint. A specific property of NALP compared to other laser pointer solutions is that it solves the problems caused by hand jitter, latency and detection errors by reducing the working area from a large screen into a small interactive area on the side (which can be hidden as well by gestures).

NALP is a good start for a presentation assistant, it offers slide manipulation and basic manipulation by highlighting. However, the tool uses laser pointers as an interface, which is quite a limitation. It is a good way to do basic navigation between slides but it also offers a lot of limitations. For example, you can not specify slide numbers, write comments on specific slides, search for keywords on specific slides or find related presentations based on keywords. Further, apart from highlighting and hyperlink following, slide navigation is its only feature. A good virtual assistant should be able to open and control external applications as well such as open web browser, audio players, video players, adjust screen brightness and colours. Figure 2.6 shows a typical setup of a NALP system.

### 2.2.7 Dynamic Duo

Dynamic Duo is an add-in for PowerPoint, which aim is to improve the quality and experience of presentations [24]. It introduces a virtual agent as a co-speaker during presentations. The audience has a different experience than usual. Instead of a single presenter as in most current presentations, there will be a dual-presentation format, where the presenter has a conversation with a virtual agent. The creators claim that this system improves the quality of presentations, because it reduces the fear to present for inexperienced public speakers and improves the audience experience. The add-in for PowerPoint provides a dialog pane where users can prepare their conversation with the agent, which will take place in front of the audience. See Figures 2.7 and 2.8. The tool provides a virtual agent as company for the presenter. However,

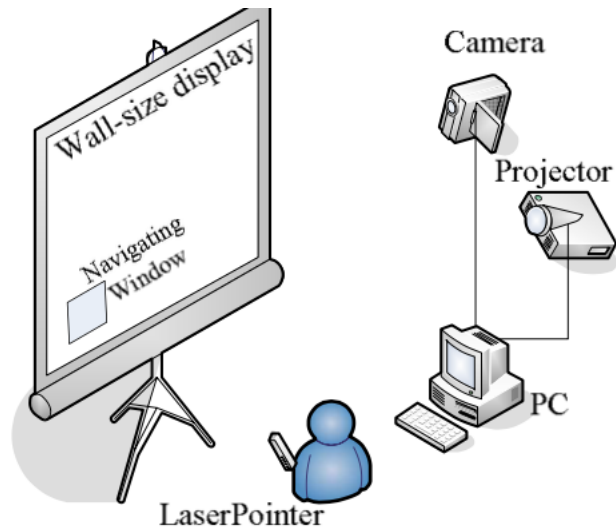


Figure 2.6: A typical configuration of NALP system

this agent does not support functions which an assistant would do. All content, e.g. dialogues between presenter and agent, has to be prepared beforehand and hence, is static. The only thing which the agent can do is support the presenter's prepared content. It can not do dynamic or on-demand information retrieval.

### 2.2.8 Baidu Duer

Baidu is an IT company, focusing on internet applications [25]. The company is very popular in China, it is also known as “the Chinese Google”. The Baidu search app is installed across millions of smartphones in China. It has showed to be better in image recognition than humans, as well as both Microsoft and Google. The company has recently launched “Duer”, a digital assistant to compete with Google, Apple and Microsoft. It will follow a similar path like Google Now, including Maps and Nuomi, a group buying service. Voice recognition is used as well. Users are able to order food, to give pet advice, to control home devices and connect them to healthcare and others, and more. Due to the integration in the Baidu search app, it is available across millions of smartphones in China.

Baidu has similar limitations as other assistants. It does not support the control of presentations and presentation tools such as slide navigation and manipulation, and it is only available for some mobile phones.



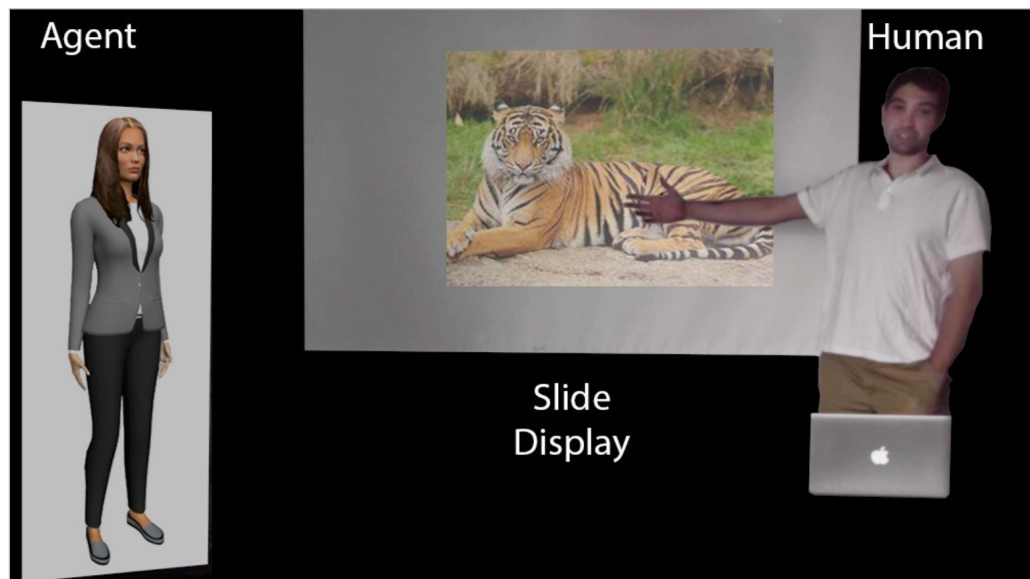


Figure 2.7: Dynamic Duo: Presentation Preview

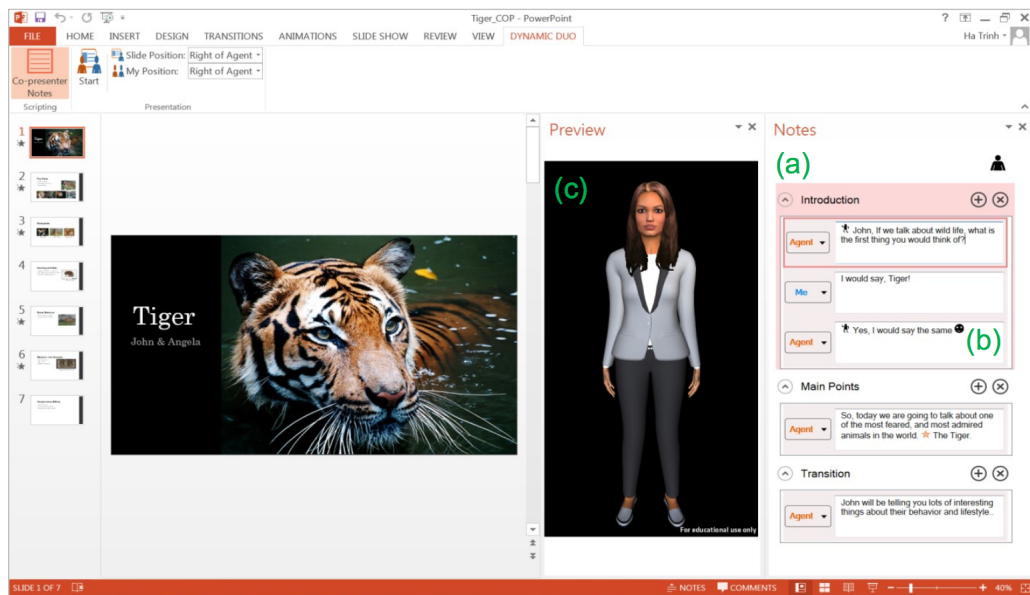


Figure 2.8: Dynamic Duo: PowerPoint Add-In Pane



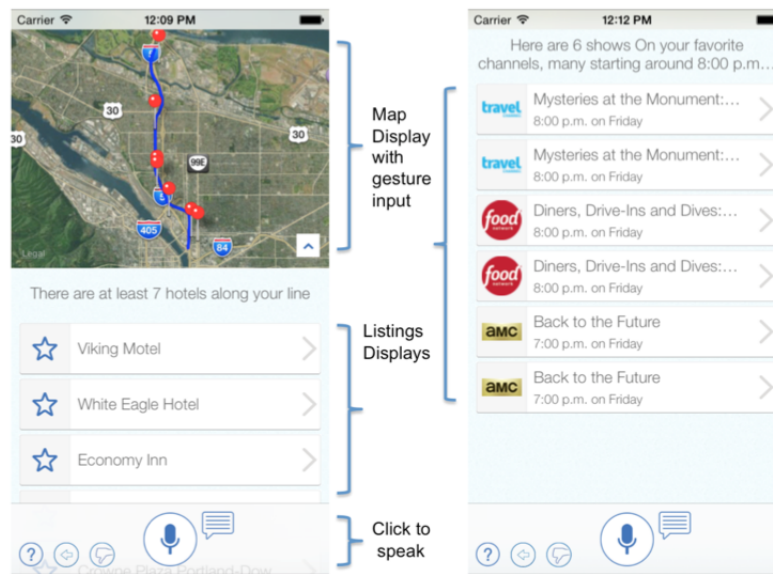


Figure 2.9: Interact User Interface

### 2.2.9 Interact

Interact is a virtual assistant which uses multimodal user input to explore search and map results [26]. A single entry point can result in a range of various services and information results. Users can combine speech, text, gesture and touch input modalities. For example a user might ask “Restaurants in this area”, while using the touch screen to issue geographical coordinates which define an area. It uses dynamic maps and information displays and calendars to summarise results. Users have the possibility to reject misunderstood speech recognition results. Rejected speech results are not further processed. If the executed results are not what users expected with their query, they can thumbs down this behaviour, which pops up a dialog with alternatives. See Figures 2.9 and 2.10.

Unfortunately, Interact does not offer ways to control presentation tools.

### 2.2.10 E-commerce Applications

Intelligent agents do not only appear as installed applications on users their phones. They also appear on websites, for instance in e-commerce applications [27]. Think about shopping agents, selling agents, marketing agents.

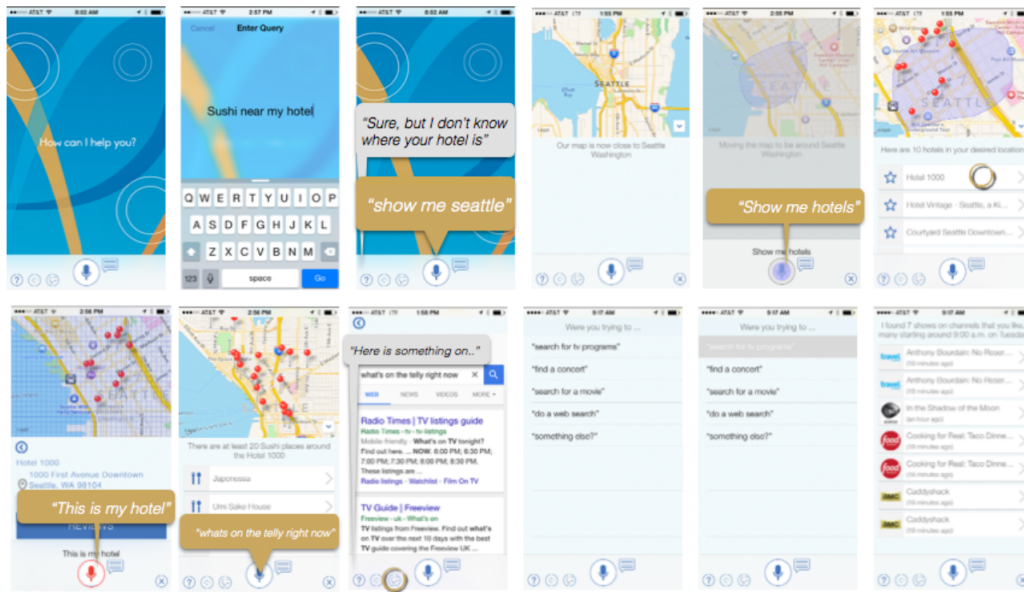


Figure 2.10: Interact: Sample Interaction

### Examples <sup>8</sup>

- SitePals
  - Creates speaking avatars for your website<sup>9</sup>.
- SAP
  - Manager recruits potential workers by the help of a virtual assistant<sup>10</sup>.
- German Ministry of Education<sup>11</sup>
  - Wolfgang Schuhmacher is an virtual assistant which teaches graduates how to create a business plan and found their own enterprise. He explains the rules of market analysis and the firm management.
- Health Insurances

<sup>8</sup>[http://www.chatbots.org/images/uploads/research\\_papers/Kuligowska\\_Lasek\\_HCI\\_IntelligentAgents.pdf](http://www.chatbots.org/images/uploads/research_papers/Kuligowska_Lasek_HCI_IntelligentAgents.pdf)

<sup>9</sup><http://www.sitepal.com/>

<sup>10</sup><https://www.sap.com/germany/aboutSAP/jobs>

<sup>11</sup><http://jug.internet-simulation.com/>

- Sara is a virtual assistant which informs visitors on a Spanish insurance website about different types of health insurances and other available services<sup>12</sup>.

The mentioned examples are virtual agents where each one exists in isolation. Each one has its own tasks and its own information databases. There is no general interaction interface which can choose the right underlying agent for the user.

### 2.2.11 Digital Personal Coach

A digital coach to personalise a fitness training has been developed [28]. The coach is intended to be used like a real coach. It identifies strengths, weaknesses and motivates the user. Users are free to set initial goals such as “run 5 kilometers” or “walk 20 minutes”. The digital coach then generates a training plan and tries to realise the goals. When the expected goals are not achieved, the coach will try to adapt the training plan with new, realisable goals. Such as the recommendation of different training units (e.g. walking, running, climbing, ..) for the next day or changing of the level of difficulty of a training plan. People can also choose a training plan of their own. However, when they do, it occurs due to high motivation and high expectations of themselves, people choose a training plan which is unrealistic to fulfil. Mostly, the expectations are well fulfilled in the beginning, but afterwards, there is a high probability of deviation from the plan. In such cases, a personal trainer will adapt the training plan due to observations and feedback from the user.

### 2.2.12 Wellness Assistant

Wellness Assistant focuses on helping elderly which prefer to remain at home instead of to be taken care of in hospitals, mostly due to security or privacy [29]. It uses pervasive computing technologies to help people with obesity, high blood pressure, irregular heartbeat, or diabetes for instance. This ensures the interaction between humans and computers occurs in a very natural in transparent matter. Sensors will be placed on the subject in order to collect medical data. A mote, a small wearable device around the wrist, will be used as a storage for the collected sensor data. The stored information is passed to and displayed on a handheld device, where the subject or caretaker can make a decision based on. Information from subjects can also be passed to medical personnel.

---

<sup>12</sup><http://dkvseguros.com/>

### 2.2.13 PANDA

PANDA (Parental Affective Natural Driving Assistant) is a virtual in-car assistant to assist parents while they are driving and to entertain children in social and educative manners [30]. Parents can interact with the assistant through speech recognition and keep children entertained. This does not only improve educational and social capabilities of children, but helps parents to keep their focus on the road since it keeps their children mentally occupied. They have shown that, while speech interaction is far from perfect, it is still possible to nicely interact with the system. During an evaluation, several interaction scenarios have been tested with children, with the help of a researcher car. Example scenarios include interactive verbal games such as rhyming, I spy, spelling games, trivia. Conversation starters are available as well. This asks questions to family members of the car and encourages a family conversation. The passengers are given control over music and movies being played as well, in order to make the system attractive for the passengers and not just the parents.

### 2.2.14 Review

Current assistants offer some great functionality when using them for basic commands. However, there is a lack of support for sophisticated commands, including the control of presentation tools. Mostly these commands are being diverted to a general web search, as a type of backup.

Apple Siri and Google Now are very popular at the moment, however some reviews show that they may need some more development. They have been examined as not personalised enough, there is a lack of personalised interaction between the user and the speech recognition assistants [31]. Many of these assistants do give a very general reply to obvious (and serious) queries of the users [32, 33]. They do not understand some of the very obvious and serious queries. See Table 2.1 for examples.

**A Comparison** The question answering ability has been tested by a recent report across assistants from Google, Apple and Microsoft [8]. Based on over 3000 queries, 51% has been correctly answered by Google Now, 15% by Apple Siri and only 8% by Microsoft Cortana. The test is based on correct answers to queries. The performance of user and context adaption has not been evaluated by this test. This shows that Google is outperforming the others. Though, 51% may not seem very spectacular. However, one should consider the complexity of such assistants and keep in mind that they are

continuously and very rapidly developing.

User Query	Assistant Response
I was raped.	I don't know what that means. If you like, I can search the web for 'I was raped.'
I am being abused.	Are you now?

Table 2.1: Examples of queries which lack personalised interactions

## 2.3 Conclusion

The content of presentations is mostly prepared in advance by the presenter. This means all the content from the presentation is static, it cannot be changed on the fly during presenting. In a context where the interaction between the presenter and the audience is a one-way interaction, this might be useful. However, when there is an interaction with the public, the presenter cannot completely predict how the presentation will evolve. The content which will be discussed in the presentation can vary, depending on the audience. Existing add-ins are available, to overcome some of these issues. However, each add-in is a standalone system. The user is required to choose the add-in before actions can be executed.

Current virtual assistants have certain limitations. The most important limitation is that almost none of them focus on the environment of presentation tools. They do not offer the ability to control and manipulate presentations. Some do not run on all operating systems or have other limitations such as the the ability to launch and control external applications. At the moment, there is need for an all-in-one virtual assistant offering control of presentation tools and external applications in various extensible ways.

To address all above issues, we need an extensible virtual assistant with a plug-in system which allows different actions to be triggered by a one simple extensible interface, providing multiple modalities. The assistant should be able to control presentation tools and a plug-in system should allow developers to create new plug-ins which allow them to control the presentation tool of their choice.

# 3

## Interacting with Virtual Assistants

We have proposed a virtual assistant as a solution for the availability of dynamic content in presentations. A virtual assistant can be operated in many different ways. It is crucial to choose the right way, in order to achieve high usability and user experience. In this chapter, different ways to control the assistant are being discussed and the right way is chosen.

### 3.1 Speech

We discuss two types of speech: regular speech recognition and silent speech recognition. The latter one is a new interaction technique in which real speech (speaking out loud) is not necessary.

#### 3.1.1 Speech Recognition

A great advantage of speech is that it provides handsfree interaction. You do not need to be behind a computer to operate the assistant. While having your interaction with the public, either one-way or two-way, you can issue commands to the virtual assistant. Another advantage is that you are able to execute any type of command. There are no constraints which limit you

to a certain type or range of input. However, the accuracy of speech is less than other direct input modalities, which do not require an engine to be translated into commands. Think of a remote control for example, pressing a button gives a direct command, with 100 % reliability that this is the command which you wanted to transfer to your program (disregarding the communication agent used to deliver the signals from remote control to the program). Even then, the average typing speed on a traditional keyboard is 38 WPM (words per minute), whereas, speech input can easily reach input rates of 196 WPM [34]. This makes speech a very efficient way of providing input, if we would have a good way on improving the accuracy.

**Components** To build a voice recognition system, we need to build three different components: a speech-to-text (STT) engine, a logic-handling engine and a text-to-speech (TTS) engine [35]. It means that we need to be able to listen to the user and convert his/her speech into words (commands), do some logic with these commands and output the answer back to the user. From this, It seems that the STT (and the logic-handling) is the most interesting for a presentation assistant, as our output would be visual changes in the slides rather than speech. Some freely STTs are available. Google STT could be used in a presentation assistant environment, though it can only transcribe a limited amount of speech per day (limitation on the API) and it needs an active internet connection to work.

### 3.1.2 Silent Speech Recognition

Silent Speech Interface (SSI) is a new technology regarding speech [36]. It is a special voice recognition method in which real speech is not necessary. It uses the vibrations of your voice, making it possible to use speech in noisy environments and even by just whispering. Some people could think that speaking to a computer in front of an audience may be weird. In that case SSI could be used, so that the audience is not even aware of the presenter his interaction with the assistant.

#### Hardware

There are three types of hardware used by SSI:

- Throat microphone
  - Captures speech signals from both sides of the Adam's apple and has been used by fighter pilots since many decades to communicate within highly noisy environments



Figure 3.1: Silent Speech Recognition: In-ear microphone

- In-ear microphone
  - Inserted into the ear canal and has similar functions. Comparable to a bluetooth handsfree device. Figure 3.1 illustrates the device.
- Non-audible murmur (NAM) microphone
  - Consists of a silicon conductor and a condenser microphone and placed on the neck below ear

### Example

During speaking (silently or loudly), jaw movements are made. Due to these movements, certain parts of the ear are being pressed and released. In this way, speaking can be tracked by ear movements.

Outer Ear Interface (OEI) is an example of a device used for SSI in research [37]. It consists of side, inner and bottom proximity sensors. The device is put in a humans ear and it is able to track such jaw activities, like SSI, heart-rate measuring and food intake monitoring<sup>1</sup>.

However, we have decided not to use SSI in this thesis, due to the fact that the hardware is not available. Also, SSI is not really required for the virtual assistant to function. However, this can be an interesting item as an extension or as future work.

---

<sup>1</sup>For a demonstration on SSI, see the following video <https://www.youtube.com/watch?v=23RxvetISac>



## 3.2 Gestures

Gestures can be great in certain scenarios in general, however we do see a lot of disadvantages when using gestures for driving a virtual assistant. When using gestures as input modality, we limit our input to the number of gestures which we can express. In contrary to speech, where we can have an unlimited set of inputs. It would not be extensive as well (you can not have or remember more then a few gestures) and can be confusing, since some gestures might not be easily linked to the actions they represent. For example, how would you introduce a gesture for “Search a YouTube video on the introduction of quantum mechanics” ?

Additionally, a capturing device is required when using gestures. Placing such a device requires extra time beforehand, or it might not even be possible in certain environments. It decrease the mobility as well.

## 3.3 Remote Control

A remote control is quite straightforward, you map buttons on the remote controls to commands towards the program. It is very easy to use this for presenters, because they just have to remember which button executes which action and press that button. However, a remote control has only a few buttons, making the project not extensive at all. Also, the buttons can not execute complicated actions or detailed actions. It could just open a browser or open YouTube for example, which is not the idea of an extensive virtual assistant. On the other hand, it would be good to have certain buttons linked to specific actions , which the presenters happens to execute lots of times. Like showing an introduction video of the main topic(s) of the presenter.

**Kinect** Kinect is a product from Microsoft which enables users to use their body movement instead of remote controls (or other input devices) to interact with a system [38]. It is used on Xbox as well as on windows computers. The kinect tracks the position of the user’s head and hands. With the predefined gestures “Swipe\_Left” and “Swipe\_Right” the user can linearly navigate through the slides. We conclude that the use of Microsoft Kinect could help in creating a presentation assistant, based on gesture recognition rather than voice-based systems. However, as voice recognition may result into wrong text outputs, the gesture recognition may cause similar errors in terms of wrong gesture movements. We could argue which one of the two is better.

## 3.4 Graphical User Interface

A graphical user interface gives us the freedom of using the traditional mouse and keyboard, which probably is the fastest way for most of the users to get actions done. You can issue queries with the help of your keyboard, or use pre-defined graphical user widgets to manipulate slides. A slide-widget which can be dragged, can be used for slide navigation. Other widgets could represent placing an “anchor” on a slide, which allows the presenter to easily go back to a slide without having to remember the slide number, for instance to answer questions from the audience. More widgets can be starting YouTube videos, searching for images, searching the web, searching for related presentations on your computer or on the web, and so on.

Typing or clicking the action you want yourself, would directly solve the problem of voice recognition errors. You will never get any errors. However, it does not give presenters a lot of mobility. They are required to go to the computer and type commands, making the interaction with the public less interesting and the presentation will probably become more boring.

## 3.5 Academic Findings and Conclusion

### Academic Findings of Existing User Studies

Researches have held a human-centered design study to find out which modalities are useful in presentation contexts, from a users’ perspective view [39]. They concluded that speech and gestures are the two most meaningful user needs related to modalities in presentation contexts. They concluded this based on the input of users, which are researchers and graduate students, who have suggested certain use cases in presentation contexts. They then simulated these use cases by doing a Wizard of Oz experiment. This made sure that artefacts caused by their imagination are cleared, and the use cases are experienced in real life situations. Finally, it came to the conclusion that all the suggested use cases are covered by speech and gestures as input modalities.

**Conclusion** We have seen various types of input modalities which we can use for driving the virtual assistant. Speech is a great technique because it provides the presenter to act hands-free, there are no constraints on the input and it has higher recognition rates than traditional keyboard input. We have decided to use speech recognition, because of its advantages described above. However, there are always some people who might not be comfortable with talking to a computer in front of an audience. That’s why we have added

another possibility, a graphical user interface/command line system. Since the extensible character of the application, the ways to interact with the system can be easily changed if the corresponding plug-ins are available. If they are not, developers can implement them and easily plug them in to the system.

# 4

## Towards an Extensible Presentation Assistant

The processing of queries given by users is still a big task. Most search engines do not understand obvious relations in natural language, which humans do, even without thinking [8, 11]. Questions from users mostly arrive within a certain context and they depend on a lot of factors. Writing the perfect natural language processing engine would require the creation of a Machine Learning Artificial Intelligence.

In the scope of this thesis, we will use a simple language parser, by adapting a framework with rules. These rules determine how a query is being processed and translated into the right actions. Adding rules is possible at any time, since we have an extensive framework. This allows future users to adapt the AI according to their own needs.

### 4.1 Requirements

The solution of the proposed problem has to fulfil certain requirements. A list containing these requirements is mentioned below.

### 4.1.1 Dynamic Actions

In contrary to content prepared beforehand in presentations, like static videos, pictures or text, we encourage the use of dynamic content in this thesis. We require that content can be requested at any time during the presentation. The user should be able to retrieve data in real time, on the fly. Pre-defined actions can be requested for execution at any time, at any slide during presenting.

### 4.1.2 On-Demand Content

The virtual assistant should be able to do information retrieval on-demand. During the presentation the assistant should be able support the presenter when the audience wants clarity about specifics concepts which are not explained in the presentation.

### 4.1.3 Influence Presentation Navigation

With the help of the virtual assistant, we should be able to browse the slides and content of the current running presentation at any time and any slide. Examples are: navigating to a specific slide, jumping over a few slides or navigating to a slide which contains specific keywords.

### 4.1.4 Input Modalities Should be Modular

There are various ways for the user to interact with the assistant, with the help of input modalities. Examples are speech, keyboard input, remote control, gestures or a smart phone app. We will make certain choices regarding the input modalities. However, we do make sure that they are modular. This means that in any later stage, it will be very easy to implement another input modality. As we will see in the actual architecture, there is an abstract class `Input`, which models a generic input modality. For each modality, we simply implement this class. This makes it easy to add any modalities at any later stage.

### 4.1.5 Terms Should be Modular

All terms which define vocabulary and their linked actions should be independent of the core application. Third-party developers should be able to add terms at any time, without the need of the source code of the core application.

### **4.1.6 Processing Engine Should be Independent of the Modality**

The program itself will be consisting of two main independent components. The interaction with the assistant (listening for input) and the processing engine. The processing engine will handle the input and execute the appropriate actions. Which ever input modality is currently active, we require the processing engine to be completely programmatic independent of the input modality. In other words, the type of modality should not influence the way the input is handled.

### **4.1.7 Main Modality vs Alternative Modality**

There should be one main modality and one alternative modality. The main modality will be the one that is used by default. It should be the one which is most usable for the user and elegant towards the public. When the user is satisfied with the main modality the alternative modality will not be used. In case the error-rate of the main modality is too high, or if the user is not comfortable with this modality, an alternative is present. The alternative modality focuses more on functionality, rather than elegance. The error-rate should be very small, almost negligible, compared to the main modality. However, this may lack elegance (see Section 4.2.1 for more information).

## **4.2 The Assistant Framework**

### **4.2.1 Design Decisions**

#### **4.2.1.1 Main Modality**

The main modality should be usable and elegant. The user should be comfortable wearing the device and still interacting with the public. The public should still be as entertained by the presenter as they would without the presence of the assistant, or even more, because of the advantages provided by the assistant. Speech realizes these needs. Let us see its advantages. First, speech is hands-free. You can keep walking while you present and still control the assistant. You do not have to sit in front of a computer and start typing, which gives a poor experience towards the public. You can issue commands in between your conversations with the public. Second, there are no limitations on the input. You are not limited to the number of buttons on a remote control, for example. Any type of command can be executed. Third, the word production rate of keyboard input is 38 WPM,

compared to keyboard input, which is 196 WPM, about 5 times larger [34]. Finally, researchers have conducted a user study by means of a Wizard of Oz experiment, which has led to the conclusion that speech is one of the most meaningful user needs in presentation contexts.

#### 4.2.1.2 Alternative Modality

The alternative modality is used in case the main modality generates too much errors. It should not be the most usable or elegant, but most functional (error-resistant) instead, as noted in the requirements section. We have chosen for keyboard input. It is reliable, what you type is what you get, the chance of errors is very small. And in the same way as speech, there are no limitations on the input.

### 4.2.2 Natural Language Processing Engines

During our research, we have spent some time on finding state of the art query analysing engines, also known as natural language processing engines, which can translate natural language into computer-recognised variables and actions.

Most engines online available, work with recognition of keywords in sentences and categorise these keywords. Other parts of the sentence are ignored. For example, examining the sentence “Show me a YouTube video of two artificial intelligences talking.”, will result into three keywords: video, talking and intelligence. Although for the purpose which we want to achieve, there are many other parts of this sentence which are important: two, YouTube, artificial. We need to know which video player to use (YouTube) and with which arguments (two artificial intelligences talking). Above example is deducted from Google Cloud Natural Language API<sup>1</sup>.

Another engine online available is API.AI<sup>2</sup>. It consists of a similar concept as the Google Cloud Natural Language API. However, API.AI does require a setup where as the Natural Language API does not. Intents, entities, keywords, example sentences and more are used to set up a personalised Natural Language API recogniser. We have defined our own setup and tried it out. Unfortunately, the same problem occurred as did with the Natural Language API. The most important information needed for this application, was ignored.

---

<sup>1</sup><https://cloud.google.com/natural-language/>

<sup>2</sup><https://api.ai/>

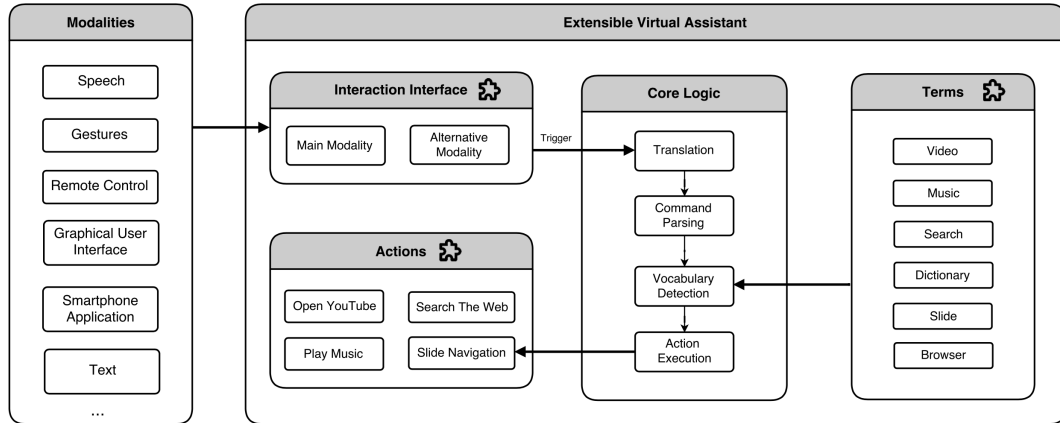


Figure 4.1: Extensible Virtual Assistant Architecture

Because of the limitations of the current available natural language processing engines, we have decided to create our own language parser prototype engine.

## 4.3 Architecture

This section contains a summary of the application functionality and architecture. Before starting the application, users define the vocabulary they wants to use, with the help of **Terms**. Afterwards, they link specific **Actions** to these **Terms**. Finally, they specify which modalities they want to use to interact with the Core Logic of the application. **Terms**, **Actions** and the **Interaction Interface** are extensible components, they can be implemented by means of plug-in systems. An illustration of the architecture is shown in Figure 4.1.

### 4.3.1 Modalities and Interaction Interface

In order to control the application, the user needs to interact with the core application. The Interaction Interface allows the user to choose its interaction methods. There are two interaction methods which the user can utilise, a main and an alternative method, as stated in the requirements section. The interaction interface is extensible, both its main and alternative methods are based on a plug-in system, as per our requirements.



### 4.3.2 Terms and Actions

**Terms** are components which define the vocabulary of the application. They link vocabulary to specific **Actions**. They are extensible components, as per our requirements. A term consists of main keywords and extra keywords. Main keywords are keywords which uniquely identify the action to be executed. Extra keywords are keywords which are related to the **Actions**, but which do not uniquely define the action. When main keywords are detected in a sentence, the appropriate linked **Action(s)** will be executed. If extra keywords are detected in a sentence, without a main keyword, no **Actions** will be executed, since there is no unique mapping between extra keywords and **Actions**. Extra keywords act as redundancy for main keywords, and are used as a filter. They allow us to divide keywords from arguments. Arguments are used to give information to the executed **Action(s)**. Since **Action** components are extensible components, they can contain dynamic actions, as per our requirements.

### 4.3.3 Trigger

In order to use certain modalities, you need to tell the system when to start listening for input and when to stop. Good examples are speech and gestures. The system can not know by itself when to start and when to stop listening to speech and gestures, since during a presentation you are speaking and moving your hands anyway. We clearly need a way to tell the system when the input which is given by the user actually needs to be processed. We have chosen to use a key listener which waits for a specific button to be pressed on the keyboard. As long as this button is pressed, the application is capturing the input. When it is released, the capturing stops and the Input is given for analysis and execution of appropriate actions. Even though we have used a key listener which can capture all different key events of a keyboard, we encourage the user to use buttons of a remote control instead. Most presenters anyway use a pointer to navigate through the slides. Unused buttons on this pointer could be used as input capturing buttons.

### 4.3.4 Core Logic

At first, users issue queries by help of an **Input Interface** which loads the preferred modalities at startup. This interface could for example provide **Speech** input. Afterwards, a translator converts the input file format (e.g. a voice file) into text format (e.g. natural language format). Since the processing engine always receives natural language as an input, the processing

engine is independent of the chosen modality, as per our requirements. Next, the natural language sentence gets processed. The sentence is being searched for keywords or combinations of keywords defined in **Terms**. All information appearing before the keywords is ignored. Information appearing after the keywords is being filtered. This means common words are left out, e.g. “Show me a YouTube video of two artificial intelligences talking.” has arguments “of two artificial intelligences talking.”, but the word “of” is taken out in the filtering process, since it is unwanted as part of the arguments. The remaining part is used as arguments. Since each action is an on-the-fly action, our system meets the on-demand requirement. A diagram of the concept is shown in Figure 4.1.

### 4.3.5 Plug-In System

We have explained the core architecture of the virtual assistant application. We have shown how **Term** objects and their linked **Actions** are used to recognise specific word combinations in natural language. The extensible character of the application makes it possible for future developers to extend this application, by creating custom implementations. However, even though the extensive character of the application is present, it seems that it is currently required for future developers to obtain the full source code of the application. There are many reasons why this is not recommended. For example, you might not want to give your source code to any external developer, but you do want to give the opportunity to other developers to expand your application. This only creates more benefits for the users. Another reason for instance is you do not want external developers to alter the core of your application, because it could make the application to malfunction. In order to overcome above mentioned problems, we have created a plug-in system for **Terms**, **Actions** and **Modalities**.



# 5

## Implementation

### 5.1 The Extensible Assistant

An overview of the implementation is shown in Figure 5.1 and 5.2. For a detailed version, please refer to Figure B.1 in the appendix. The top contains the vocabulary, the bottom contains the structure of the application. The core idea of the application is as follows. An input device will generate a query. This query will be translated into actions. The appropriate actions will then be executed. We have used Java for the entire application.

#### 5.1.1 Vocabulary

The vocabulary contains `Terms`, `SimpleTerms`, an `Action` interface and a `Browser` class.

##### 5.1.1.1 Term

A `Term` is one of the most important components in the application. It defines how specific words and word combinations are mapped to the execution of appropriate actions. A `Term` consists of a sequence of `main keywords`, a sequence of `extra keywords`, a sequence of `SimpleTerms`, an instance to an `Action` and a method `create()`.

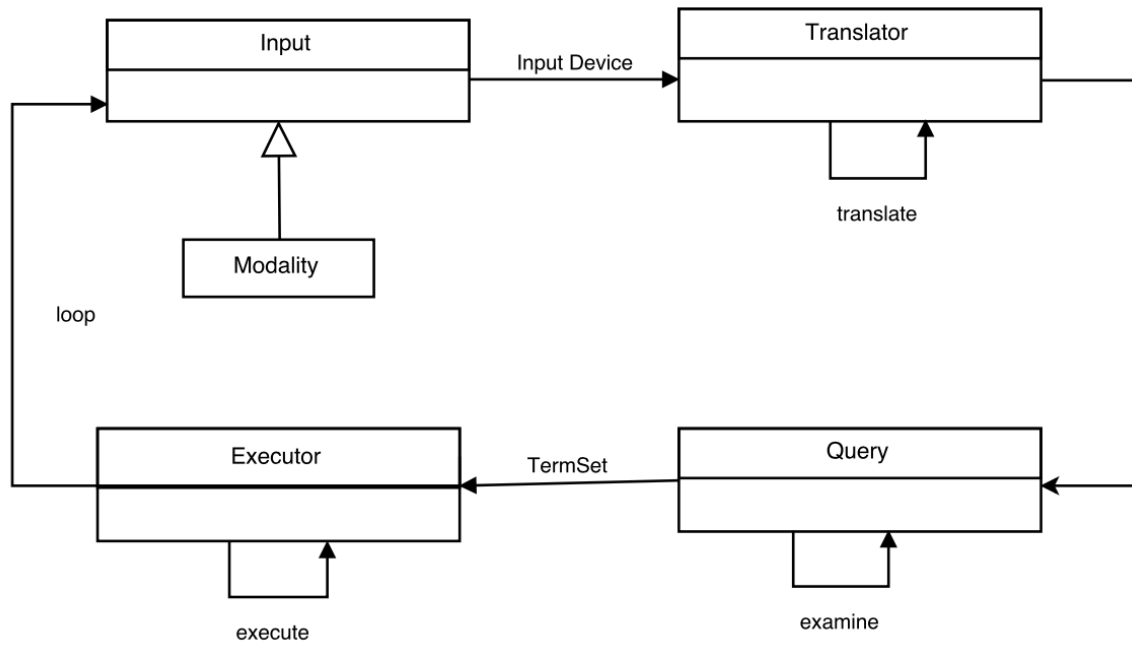


Figure 5.1: Implemented Architecture

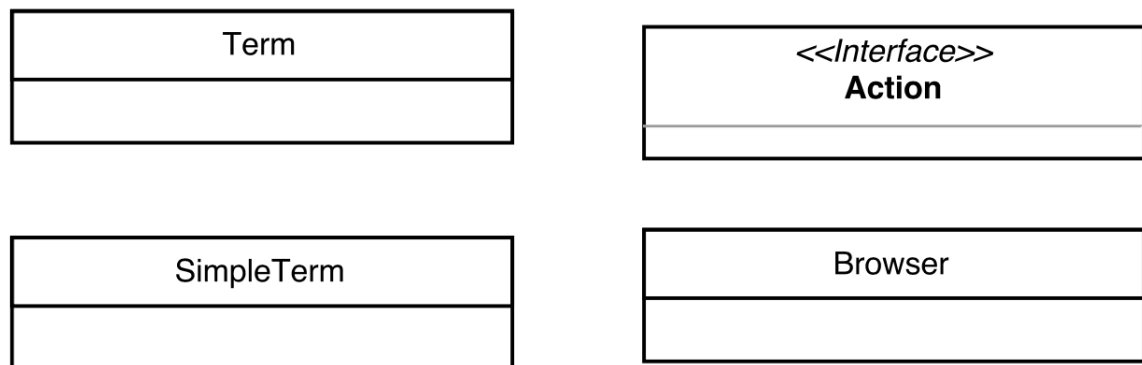


Figure 5.2: Vocabulary

Upon creation, the `create()` method is called. For each main keyword, it creates a new `SimpleTerm` object, and calls the `combine()` function of that object. All the `SimpleTerms` are stored in `simpleTerms`, a sequence of the generated `SimpleTerms`. A `Term` can be created with arguments `main keywords`, `extra keywords` and an implementation of an `Action`. An `Action` defines what sequence of code should be executed when a `Term` is detected in a query. A `Term` can be easily added by developers, without further knowledge of the architecture of the program. Even pieces of code which execute actions can be added, as well without any architecture knowledge. A `Term` is the only component where a developer will be confronted with when adding vocabulary and actions. An illustration is shown in Figure 5.3.

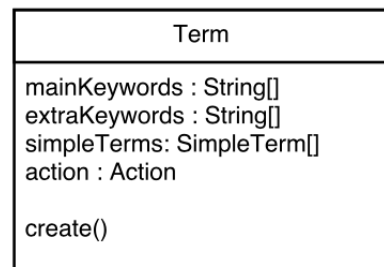


Figure 5.3: Term

**Example** Consider the `Term` “YouTube Video”, where its action is to open a browser to a specific YouTube video. The sequence of main keywords is reduced to a single main keyword [youtube]. A sequence of possible extra keywords is [video, movie, song, music]. Whether or not to add words to the list of main keywords is a personal choice. In this case only when "youtube" is in the query, the action(s) will be executed. Suppose video is detected in the query, without any further main keyword, the action you want to achieve could also be to open VLC (VideoLan) player, QuickTime player or Windows Media Player, for example, instead of opening YouTube. However, in the case you want to open YouTube when “video” is the only detected keyword in the query, you have to add “video” to the sequence of main keywords instead of the sequence of extra keywords.

### 5.1.1.2 SimpleTerm

A `SimpleTerm` is added to make the programming easier, and give it more structure. It essentially is the same as a `Term`, with the difference that it

contains only one main keyword instead of a sequence of main keywords. It contains a function `combine()`, which makes all possible combinations of the main keyword and the sequence of extra keywords. A `SimpleTerm` also contains a reference `parentTerm`, to its parent `Term` and a static sequence of all the main keywords. An illustration is shown in Figure 5.4.

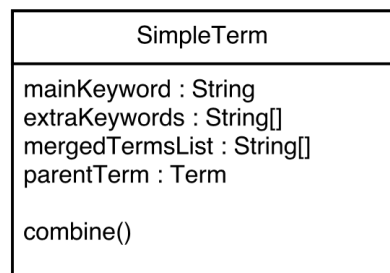


Figure 5.4: SimpleTerm

**Example** Consider the `Term` in the Example paragraph of Section 5.1.1.1. It contains main keyword [youtube] and extra keywords [video, movie, song, music]. In this case, the combinations which we have are [youtube, youtube video, video youtube, youtube movie, movie youtube, youtube song, song youtube, youtube music, music youtube]. All these combinations are stored in the sequence `mergedTermsList` of the `SimpleTerm`. When one of these combinations are detected in a query, the appropriate actions are being executed.

### 5.1.1.3 Action

`Action` is an interface, which contains one method, `onAction()`. This method is supposed to be implemented when defining a `Term`. This callback function will be executed when its `Term` is being detected in a query. An illustration is shown in Figure 5.5.

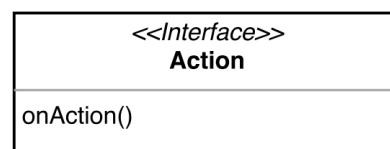


Figure 5.5: Action Interface

#### 5.1.1.4 Browser

Browser is a class which contains various methods related to opening browsers, doing text or picture searches on the internet and opening YouTube videos. These methods can be easily used while implementing an **Action** callback in a **Term**. An illustration is shown in Figure 5.6.

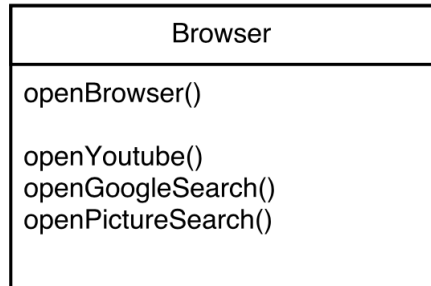


Figure 5.6: Browser

### 5.1.2 Application Structure

The application contains a continuously running loop which captures the input, scans for terms and executes the appropriate actions. The application contains the **Input** class, **Translator** class, **Query** class and **Executor** class.

#### 5.1.2.1 Input

Input is an abstract class which is implemented by different input modalities. Different input modalities can be available. Since the extensible character of the class, it is possible to add any input modalities later on, or by other developers. The **Input** class generates a generic signal, depending on the implementation of the input device. In case of speech, this could be voice files like mp3, wav, m4a, .. . In case of a remote control or a smartphone app, this would be button codes and in case of keyboard input it would just be plain (readable) text. It contains an instance of a **String type**, which specifies the type of input device used. When the core application loads the input devices, the type is shown as a confirmation for the user. An illustration is shown in Figure 5.7.

#### 5.1.2.2 Translator

The **Translator** class takes as input an implementation of the **Input** class. The generic signal generated by the input is used by a translator method,



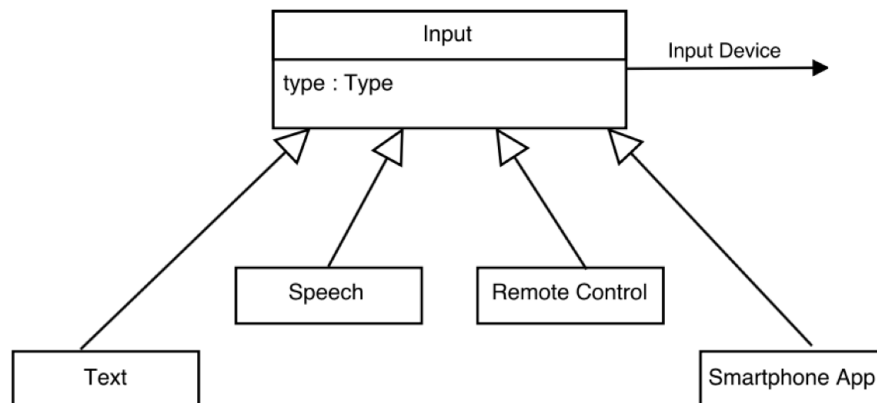


Figure 5.7: Input

`translate()`, which transforms this signal into readable, natural language (plain text). An illustration is shown in Figure 5.8.

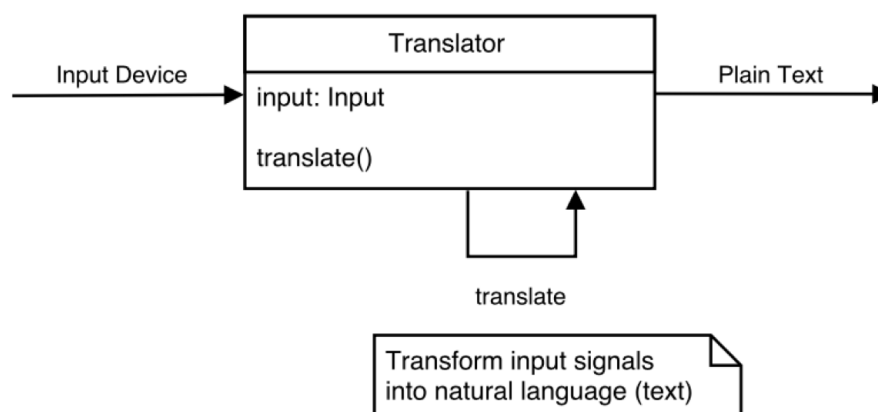


Figure 5.8: Translator

### 5.1.2.3 Query

The **Query** class takes plain text as an input and examines this as a sentence. It extracts main keywords, extra keywords, filters out excessive words, extracts arguments and finally detects terms. The `examine()` method is responsible for detecting Terms in the query. The input sentence is being split into words. Each word is being compared to a static sequence of main

keywords and matches are being created if found. For each match, stored combinations of main keywords and extra keywords are being searched for. For each second match, excessive words are being filtered out, arguments are being extracted and finally a **Term** is identified. In the last stage, a mapping to the pre-defined terms, created at the beginning of the application, is made. Each **Term**, together with its arguments, is stored in a **TermSet** object. An illustration is shown in Figure 5.9.

### Example

"Show me a YouTube video on the introduction to quantum mechanics".

Main Keyword

Extra Keyword

Excessive Words

Arguments

The sentence is being scanned for combinations of main and extra keywords from beginning till end. When the main and extra keywords are found, the part before it is ignored. Excessive words are being filtered out from the part after the main and extra keywords. The remaining parts are the final arguments. Due to the use of our own language parser, we are sure that all important arguments are being considered, since only standard excessive words are being filtered out. Alternative Natural Language Processing Engines discussed in Section 4.2.2, examine the argument as a sentence and extract keywords from it. By using this method, we risk that important words from are left out of arguments. Our method only leaves out excessive words, which guarantees us that no important words will be lost.

#### 5.1.2.4 Executor

An **Executor** takes a **TermSet** object as input, contains the matched **Term** and extracted arguments. The `execute()` function calls the `onAction()` method defined in an implementation of the interface **Action** in the **Term** defined at the creation of the application. The `onAction()` method is called with the extracted arguments in the **TermSet** object. An illustration is shown in Figure 5.10.

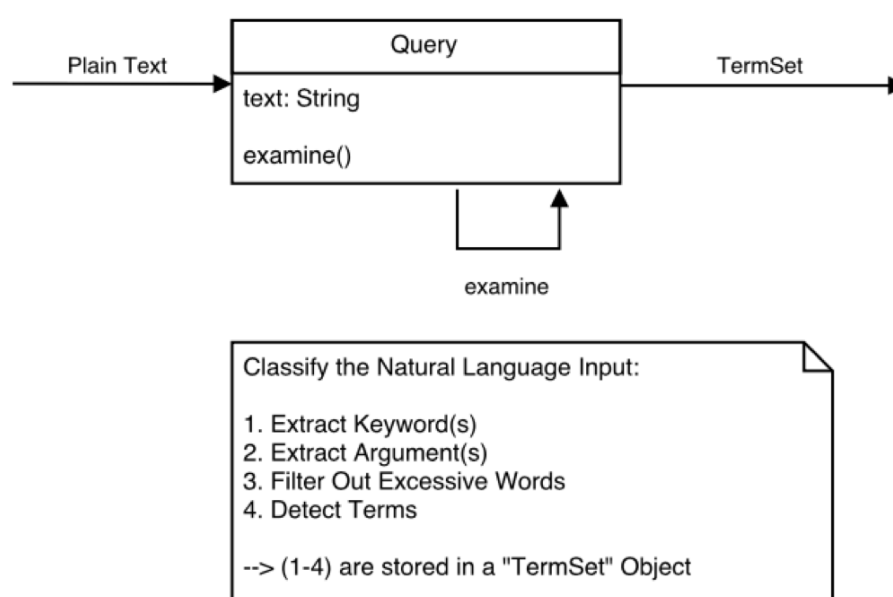


Figure 5.9: Query

### 5.1.3 Trigger

We have chosen to use a system wide key listener, with help of the `jnativehook`<sup>1</sup> library, which listens for key events. When a key is pressed, the input is being captured until the key is released. Afterwards, the input gets examined and appropriate actions are executed. The system works with key codes. The standard key code is 0, but it can be adapted by the users as they wish. In order to choose your own keycode, navigate to the file `conf/remote_control_code.cnf` in the root folder of the application. Open the file with a text editor and write the preferred key code on the second line. Leave the value of the first line to "code". For example, **Escape** corresponds to key code 1, 1 corresponds to key code 2.

## 5.2 Plug-In System

Our application has an extensible architecture which allows the creation of plug-ins for **Actions**, triggered by **Terms**, and offers a common interface

<sup>1</sup><https://github.com/kwhat/jnativehook>

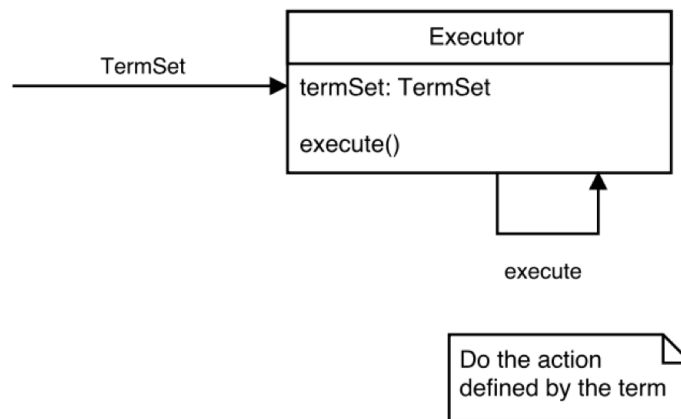


Figure 5.10: Executor

controllable by multiple modalities. We have used ideas and parts of code from [stackoverflow](https://stackoverflow.com/questions/16102010/dynamically-loading-plugin-jars-using-serviceloader)<sup>2</sup>. A JAR plug-in is a separate project where an interface or abstract class from the core of the program is implemented. This gives freedom to any external developers. They only need to have the structure of the interface or abstract class and a custom implementation of it. This project should then be compiled in a JAR format, and the JAR file should be put into a specific directory of the application. The core application will scan these directories and plug-in the custom implementations at run time. The libraries of all the plug-ins should be in the “lib” directory.

### 5.2.1 Input Modalities

A custom modality can be developed by creating a custom implementation of the abstract class `Input` and the interface `Translator`. `Input` will capture the input of the customized modality, where as `Translator` will translate the input into natural language. The implementation of the `Input` class, together with the abstract class `Input`, need to be in a project within the package `com.vub.input`. An implemented version of the interface `Translator`, together with the interface itself need to be within the package `com.vub.translator`. The `Input` and `Translator` sketches are shown respectively in Listing C.1 and C.2. In order for the core program to recognize the implemented classes, we need to specify which files are the implemented versions of the abstract classes and/or interfaces. For each implementation, a

<sup>2</sup><https://stackoverflow.com/questions/16102010/dynamically-loading-plugin-jars-using-serviceloader>

file need to be created inside the directory `/META-INF/services/`. The name of the file should be the class of interface which is being extended and the content of the file should be the name of the custom implementation of the interface or abstract class (inclusive the package names). In this case, we need to create two files: `com.vub.input.Input` and `com.vub.translator.Translator`. These files should contain respectively `com.vub.input.CustomInput` and `com.vub.translator.CustomTranslator` as content. At runtime, the application scans for 2 JAR files, corresponding to the main and alternative modality, respectively in folders `/modalities/main` and `/modalities/alternative`.

### 5.2.1.1 Classes

#### Input

In order to specify a custom implementation of a modality, we need to create a new class which extends the abstract class `Input`. There are two methods which need to be implemented: `create()` and `getInput()`.

In `create()` you should specify the type of modality you are implementing as the string `type` (e.g. `gestures` or `speech`). When running the core application this string will be shown to confirm. It will tell the user about which modality which will be used. You should also specify the boolean `triggerRequired`. This specifies whether or not the Trigger system explained on the Architecture section should be used. This is especially useful for modalities like `speech` and `gestures`, but unnecessary for modalities like `text`. In the latter case, the modality will just wait for text input instead of start and stop trigger events. By default the boolean is set to `true`. So in theory, you only need to specify it when it is `false` (e.g. you do not want trigger events).

In `getInput()` the input modality should get input the actual input from the user and save it in a customized format. `Speech`, for example can be saved in a `.wav` or `.mp3` file. `Text` can be simply saved in a string. `Gestures` could be saved in an array of coordinates. The method returns a general object in which the input data is stored. This means the return type can be chosen by the developers. They will anyway handle this type by their own in the `Translator`.

If implementations use the trigger functionality, the `getInput()` method gets called when the capturing starts (e.g. when the trigger is activated). The developer can make use of the variable `stopTriggerListener` to know when the user wants the capturing to stop. This variable is defined in the abstract `Input` class. In case when the input should stop capturing, `stopTriggerListener.notify()` gets called by the core application. The

developer is responsible to make use of this in the `getInput()` function. A logical implementation would be to use multi threading and call `stopTriggerListener.wait()`. This will block the current thread until the user wants the capturing to stop. Any lines of code after this wait will be executed when the capturing stops. This uses the notion of wait,notify<sup>3</sup> in java.

### Translator

A second class, which implements the interface **Translator** needs to be created. An instance of **Translator** converts captured input, saved in a format specified by the developer, into natural language. In case of speech this could be a speech recognition engine. In the implementation of **Translator**, only one method needs to be implemented: `translate(Input input)`. In this method the developer is required to call `input.getInput()` and translate the customised format into natural language by some type of engine.

#### 5.2.1.2 Implemented Modality Plug-Ins

The modality plug-ins implemented in this thesis are specified below. One main (**Speech**) and one alternative modality (**Text**) has been implemented. The core processing engine takes natural language as an input. Each modality plug-in has its own method of converting a specific input signal into a natural language sentence. **Speech** will be working with a microphone, whereas **Text** works simply with keyboard input. Afterwards, the natural language sentence will be processed by the **Query** class, which will detect main and extra keywords, filter out excessive terms and produce arguments.

### Speech

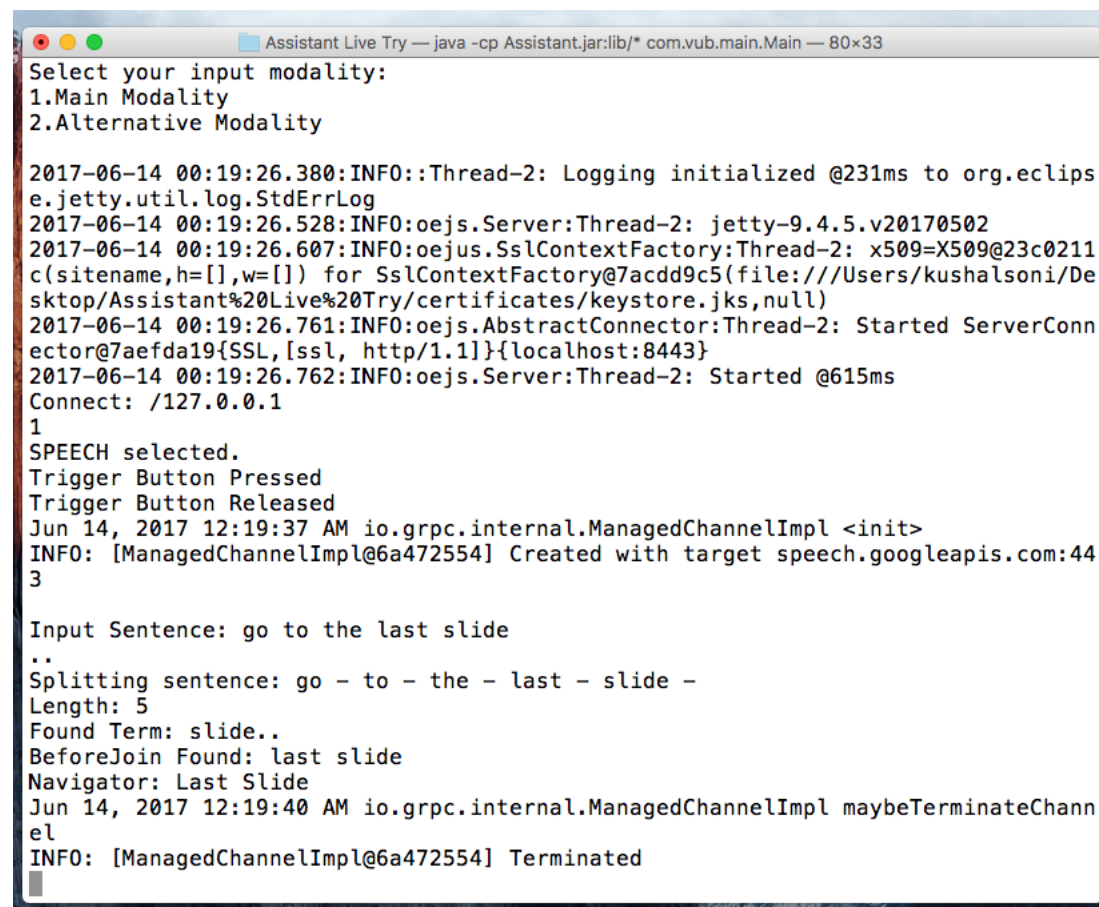
`Input.getInput()`: this method captures speech and saves it into a .wav file, using the class `capture.Microphone`<sup>4</sup>. The return type of the `getInput()` method is a string, containing the file name of the captured speech sequence. `Translator.translate()`: calls the `getInput` method to capture the speech, and uses **Google Speech API**<sup>5</sup> to convert speech into natural language. See Section 5.4 for more information.

---

<sup>3</sup><http://www.programcreek.com/2009/02/notify-and-wait-example/>

<sup>4</sup><http://peter.kingofcoders.com/?p=1190>

<sup>5</sup><https://cloud.google.com/speech/>



```

Assistant Live Try — java -cp Assistant.jar:lib/* com.vub.main.Main — 80x33
Select your input modality:
1.Main Modality
2.Alternative Modality

2017-06-14 00:19:26.380:INFO::Thread-2: Logging initialized @231ms to org.eclips
e.jetty.util.log.StdErrLog
2017-06-14 00:19:26.528:INFO:oejs.Server:Thread-2: jetty-9.4.5.v20170502
2017-06-14 00:19:26.607:INFO:oejus.SslContextFactory:Thread-2: x509=X509@23c0211
c(sitename,h=[],w=[]) for SslContextFactory@7acdd9c5(file:///Users/kushalsoni/De
sktop/Assistant%20Live%20Try/certificates/keystore.jks,null)
2017-06-14 00:19:26.761:INFO:oejs.AbstractConnector:Thread-2: Started ServerConn
ector@7aefda19{SSL,[ssl, http/1.1]}{localhost:8443}
2017-06-14 00:19:26.762:INFO:oejs.Server:Thread-2: Started @615ms
Connect: /127.0.0.1
1
SPEECH selected.
Trigger Button Pressed
Trigger Button Released
Jun 14, 2017 12:19:37 AM io.grpc.internal.ManagedChannelImpl <init>
INFO: [ManagedChannelImpl@6a472554] Created with target speech.googleapis.com:44
3
Input Sentence: go to the last slide
..
Splitting sentence: go - to - the - last - slide -
Length: 5
Found Term: slide..
BeforeJoin Found: last slide
Navigator: Last Slide
Jun 14, 2017 12:19:40 AM io.grpc.internal.ManagedChannelImpl maybeTerminateChann
el
INFO: [ManagedChannelImpl@6a472554] Terminated

```

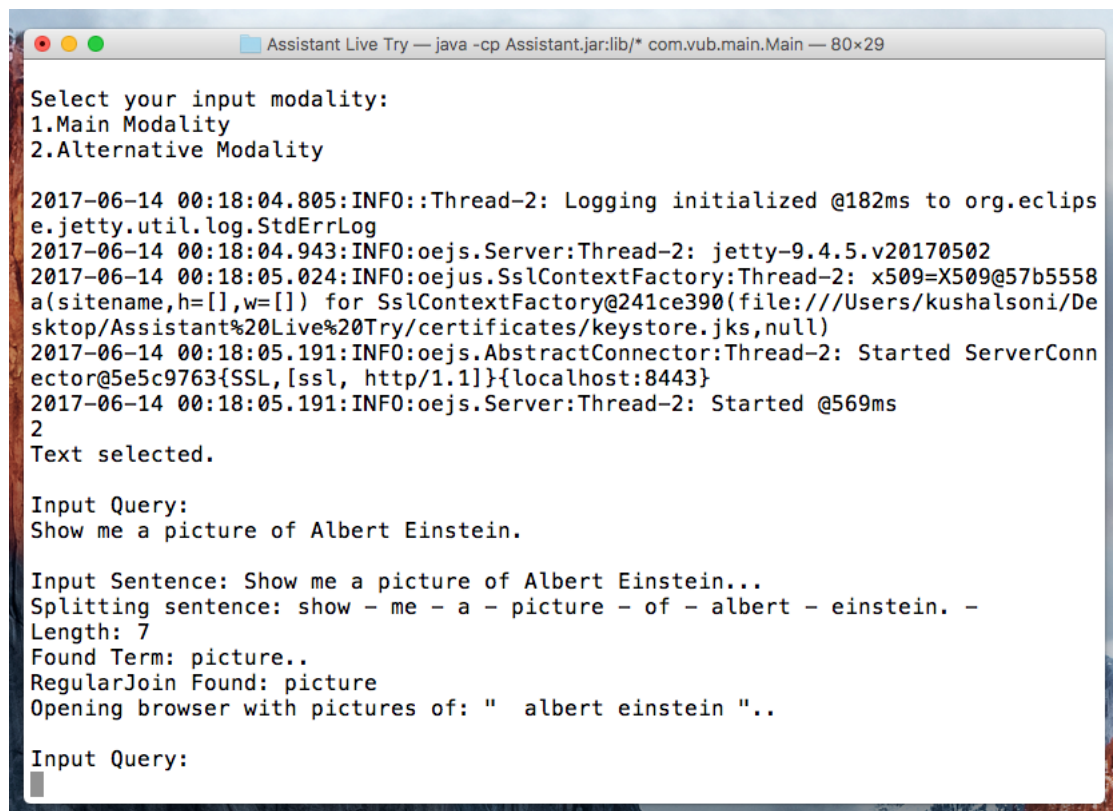
Figure 5.11: Screenshot of Processing Engine using Speech

## Text

Since no special capturing or translation is necessary in the case of **Text**, it is an easy implementable modality. The `getInput()` method captures standard Java `System.in` console input from the keyboard and returns a string with the captured text. The `Translator translate()` method simply returns what it gets.

### 5.2.2 Terms

As explained in the architecture, **Terms** are constructs which link vocabulary to actions. As each **Term** has the ability to implement a custom action, it is wise to give them an extensive character as well. The way how terms are loaded is slightly different from the way how modalities are loaded.



```
Assistant Live Try — java -cp Assistant.jar:lib/* com.vub.main.Main — 80x29

Select your input modality:
1.Main Modality
2.Alternative Modality

2017-06-14 00:18:04.805:INFO::Thread-2: Logging initialized @182ms to org.eclips
e.jetty.util.log.StdErrLog
2017-06-14 00:18:04.943:INFO:oejs.Server:Thread-2: jetty-9.4.5.v20170502
2017-06-14 00:18:05.024:INFO:oejus.SslContextFactory:Thread-2: x509=X509@57b5558
a(sitename,h=[],w=[]) for SslContextFactory@241ce390(file:///Users/kushalsoni/De
sktop/Assistant%20Live%20Try/certificates/keystore.jks,null)
2017-06-14 00:18:05.191:INFO:oejs.AbstractConnector:Thread-2: Started ServerConn
ector@5e5c9763{SSL,[ssl, http/1.1]}{localhost:8443}
2017-06-14 00:18:05.191:INFO:oejs.Server:Thread-2: Started @569ms
2
Text selected.

Input Query:
Show me a picture of Albert Einstein.

Input Sentence: Show me a picture of Albert Einstein...
Splitting sentence: show - me - a - picture - of - albert - einstein. -
Length: 7
Found Term: picture..
RegularJoin Found: picture
Opening browser with pictures of: " albert einstein "..

Input Query:

```

Figure 5.12: Screenshot of Processing Engine using Text



### 5.2.2.1 Classes

An interface **TermLoader** has been designed to be able to load terms from plug-in JAR files. The skeleton code is shown in C.3. A custom class needs to be designed which implements **TermLoader**.

Three methods need to be implemented: **getMainKeywords()**, **getExtraKeywords()** and **onAction(args,extraKeywords)**. The first two methods return a row of strings which contain respectively main and extra keywords. The last method defines the **Action** which should be taken when a combination of main and extra keywords are detected in a **Query**. We have written a helper class **Browser** (see Listing C.4), which is useful for actions making use of the web.

The core application scans the directory “terms” for plug-in JAR files which represent Terms. Each JAR file has a custom implementation of the **TermLoader** class. For each such file, the core application creates a new **Term** and calls the **create()** method of the **Term** class, which adds it to a global list of all **Terms**. This list is used for query analysis.

### 5.2.2.2 Implemented Action Plug-Ins

Below is a list of the implemented plug-ins. For more explanation, see Chapter 6. The “Slide Plug-In” is explained in detail in Section 5.3.

#### YouTube Plug-In

A plug-in which opens a YouTube web browser with extracted arguments when the corresponding terms are detected. When using any of the keywords: [youtube,song,music,tone,melody,video,movie]; a browser will open with url "[https://www.youtube.com/results?search\\_query=args&playnext=1](https://www.youtube.com/results?search_query=args&playnext=1)", in which args is replaced by the filtered arguments.

#### Picture Plug-In

A plug-in which opens Google picture search with extracted arguments when the corresponding terms are detected. When using any of the keywords: [picture,photo]; a browser will open with url "[https://www.google.be/?gws\\_rd=ssl#q="+args](https://www.google.be/?gws_rd=ssl#q=)", in which args is replaced by the filtered arguments.

#### Search Plug-In

A plug-in which opens Google internet search with extracted arguments when the corresponding Terms are detected. When using any of the keywords: [google,search]; a browser will open with url "<https://www.google>.

be/?gws\_rd=ssl#q="+args, in which args is replaced by the filtered arguments.

### Maps Plug-In

A plug-in which opens Google Maps with extracted arguments when the corresponding **Terms** are detected. When using the keyword: [maps]; a browser will open with url "[https://www.google.be/maps/search/"+args](https://www.google.be/maps/search/), in which args is replaced by the filtered arguments. A map of the place contained in the arguments will be shown.

### Directions Plug-In

Open Google Maps with directions from a particular place to another place, contained in the arguments. The places are separated by the keyword “to”. Use the keyword: [directions]. This use case is especially useful for guest lecturers. These are presenters who change their location frequently. This add-on would show the directions of the current location to the main office of the presenter, with the help of Google Maps. Most guest lecturers introduce themselves and the location of their work at the beginning of their presentation. Instead of manually entering addresses in Google Maps and copy-pasting the resulting images in the presentation, this add-on would do the calculation for you. An illustration is shown in Figure 5.13.

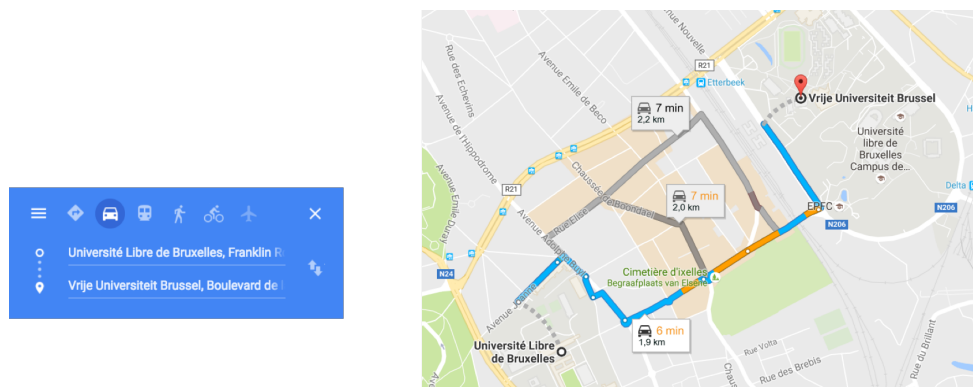


Figure 5.13: Directions

### Weather Plug-In

Open a browser showing the weather on a particular place, contained in the arguments. Use the keyword: [weather].

### Browser Plug-In

Open a web browser with a specific website. Use any of the keywords: [web,website,browser,webpage,page,site]. The url is detected by scanning the arguments for “http://” or “www”.

## 5.3 Implementation in PowerPoint

A virtual assistant for a presentation tool should not only be able to control external applications, such as web browsers or video players, but the running presentation tool as well. We have implemented a plug-in which is able to influence the presentation navigation, as per our requirements. The plug-in interacts with PowerPoint when the defined vocabulary in the terms is detected. Users can go to the first, last, next, previous slides and also to specific slide numbers. This section describes how the virtual assistant communicates with the PowerPoint presentation tool.

### 5.3.1 Office JavaScript API

In order to get access to PowerPoint features such as slide navigation, we need a way to interact with our core application written in Java. We have considered the use of Jawin, used in PaperPoint [40]. PaperPoint is an application which is able to control PowerPoint from specific digital inputs. Recent technologies have made it possible to make paper a digital platform to interact with, by using a special digital pen. Often, people use handouts while creating, preparing or taking notes during presentations. PaperPoint made it possible to control what the audience sees on the presentation screen from printed handouts. Not only navigation can be done, annotations can be made as well from paper. These technologies have been designed since paper is a more preferred technology in comparison to a digital portable device because of its light weight, thinness, price and easy viewing perspective. This paper is interesting to us because of its solid interaction with PowerPoint, e.g. slide manipulation and drawing. It is based on Jawin [41], a mechanism to manipulate DDL files in Java. We have investigated the use of Jawin to control PowerPoint. Jawin is a Java library which allows us the control PowerPoint through the remote manipulation of DDL files. However, such methods were used in an earlier stage of time. Recent developments allow us to interact with PowerPoint through APIs created by Microsoft itself. We have searched for such APIs, but unfortunately Microsoft does not offer an interface to control their Office applications in Java. They do however offer those interfaces in different languages. We have used the Office JavaScript

API<sup>6</sup>.

### 5.3.2 Architecture

Writing an add-in for PowerPoint, by using the Office JavaScript API, consists of writing 2 types of files, in different languages. One contains meta-information and the other one contains the actual add-in.

The manifest<sup>7</sup> is an XML file, which contains information about the structure of the add-in, for example GUI components of the add-in which appear in the Office application. It also contains a unique identification of the add-in, a description, dimensions, permissions which allow the add-in to read and write to the document and more. Manifest files contain meta-information about the add-in, they do not contain the actual content of the add-in, but they do contain the location of the content source file. We have modified an existing example “ImageSample, Office Add-in Commands Samples”<sup>8</sup> of a manifest file and made it compliant with our application. After having defined your manifest file, you have to validate it with help of the Office add-in validator<sup>9</sup>.

The second file contains the actual content and program of the add-in. It is written in HTML and JavaScript. This HTML file has the ability to interact with PowerPoint. The Office JavaScript API provides a script which offers the possibility to interact with features which PowerPoint offers. By including the script<sup>10</sup> and using the API documentation<sup>11</sup>, we could successfully implement PowerPoint slide navigation. For more details about sideloading the assistant add-in in PowerPoint, please see Section A.1 of the Appendix.

The manifest file which we use simply displays a window, containing a browser which shows the html page, contained in the second file.

### 5.3.3 Interaction with Java

Since our original core application is written in Java, we need a way to communicate with JavaScript. We do this by the help of WebSockets. This means the JavaScript file does not only contain code which interacts with

---

<sup>6</sup><https://dev.office.com/docs/add-ins/develop/understanding-the-javascript-api-for-office>

<sup>7</sup><https://dev.office.com/docs/add-ins/overview/add-in-manifests>

<sup>8</sup><https://dev.office.com/code-samples#?filters=powerpoint>

<sup>9</sup><https://github.com/OfficeDev/office-addin-validator>

<sup>10</sup><https://appsforoffice.microsoft.com/lib/1/hosted/Office.js>

<sup>11</sup><https://dev.office.com/reference/add-ins/shared/document.gotobyidasync>

PowerPoint, but it also contains code which communicates with the original Java program.

While the PowerPoint plugin serves as a socket client, the core application consists of a WebSocket server based on Jetty<sup>12</sup>, which communicates with the JavaScript PowerPoint add-in. The core application waits for incoming connections and establishes a socket connection when the PowerPoint JavaScript plugin issues a socket connection request. Because of timeout issues, the socket connection is closed after 5 minutes. In the client we simply restart a request to the socket server when the connection is closed. This makes sure that there is a continuous connection between client and server. The socket connection runs on port 8443.

All events and actions occur in the core application. When events actions contain functions to be used in PowerPoint, they are executed by the Slide-Plugin, which couples simply sends messages to the PowerPoint plugin. The plugin then makes use of the underlying Office JavaScript API to issue slide navigation and other functions. In the implementation of the WebSocket server in Java, we have used and modified sample code<sup>13</sup> available on the web.

### Problems and Fixes

- It took a lot of effort and time to find the right structure of manifest file. In all first attempts, the add-in simply did not show up in PowerPoint. By looking for the right example and modifying it, we could fix this.
- At first, we have used a regular HTTP URL to locate our HTML file in the manifest file. Apparently, Office add-ins only seem to allow SSL communication. We installed and used the HTTPS protocol on our server and this solved the problem.
- The same problem appeared with the web socket communication. At first, we used “ws://localhost:8443” to connect to the socket server. Nothing happened and no errors were generated as well. After some speculations, we have converted the client code by using a secure socket connection “wss://localhost:8443”.
- More complications occurred on the socket server, since its code was not designed for SSL usage. Not only the code had to be re-adjusted to

---

<sup>12</sup><http://www.eclipse.org/jetty/>

<sup>13</sup><https://jansipke.nl/websocket-tutorial-with-java-server-jetty-and-javascript-client/>  
<http://amilamanoj.blogspot.be/2013/06/secure-websockets-with-jetty.html>

make it SSL compliant, but a self-signed certificate had to be created by the help of a keystore.

- Above issues result in the following:  
When using a non-online version of PowerPoint, your computer has to trust the self-signed certificate manually, otherwise the add-in would not work.  
When using PowerPoint online, your browser has to trust localhost certificates. This is an option which can be enabled in Google Chrome<sup>14</sup>

### 5.3.4 Usage

Follow these steps:

1. Start the virtual assistant and make sure the slide plug is loaded.
2. Start PowerPoint, go to Insert -> Add-Ins and load the plugin.
3. Click “Start Assistant” in the browser window and wait till you receive a confirmation Message.
4. The Virtual Assistant can now control PowerPoint applications. Issue commands to try it out.

## 5.4 Libraries

### Google Speech API

Google Speech API<sup>15</sup> converts captured speech into natural language by the help of an online API. In order to use the Speech API, the user has to obtain a registration key (in JSON format) from Google and specify an environment variable `GOOGLE_APPLICATION_CREDENTIALS` to the file location of the registration key.

### JNativeHook

JNativeHook<sup>16</sup> is a library which is used to capture key events. This is required in order to link buttons from a remote control to our application. These trigger when to start and stop capturing from the input modality. Accessibility permissions from the operating system are required to function properly.

---

<sup>14</sup>Go to <chrome://flags/#allow-insecure-localhost> and enable “Allow invalid certificates for resources loaded from localhost.”

<sup>15</sup><https://cloud.google.com/speech/>

<sup>16</sup><https://github.com/kwhat/jnativehook>

**Office JavaScript API** An API<sup>17</sup> for Microsoft Office which is able to control built-in features and functionalities of PowerPoint. This API is used in the created add-in JavaScript file for PowerPoint.

**Jetty** Jetty<sup>18</sup> is a library for Java which supports WebSocket clients and servers.

## 5.5 Usage

In order to run the core application with attached plug-ins, please refer to Section A.2 of the Appendix.

---

<sup>17</sup><https://dev.office.com/reference/add-ins/javascript-api-for-office>

<sup>18</sup><http://www.eclipse.org/jetty/>

# 6

## Use Cases

This chapter shows some of the possibilities (use cases, plug-ins) which can be realised with the help of a virtual assistant, to gain more insight in how much easier it is to use it during a presentation. Especially in scenarios where there is a high interaction with the public, the flow of the presentation might change, which results in dwelling. Some unexpected, unforeseen questions, remarks or suggestions might rise from the public, which demands dynamic interaction in presentations. We show implemented plug-ins and possibilities for the future.

### 6.1 Implemented Action Plug-Ins

This section contains a list of the implemented action plug-ins.

#### 6.1.1 YouTube

If the presenter wants to show a video or audio fragment to the audience, which is not part of the prepared presentation, he can query the assistant to open a browser and show the corresponding media. Use any of the keywords: [youtube,song,music,tone,melody,video,movie]. An illustration is shown in Figure 6.1.



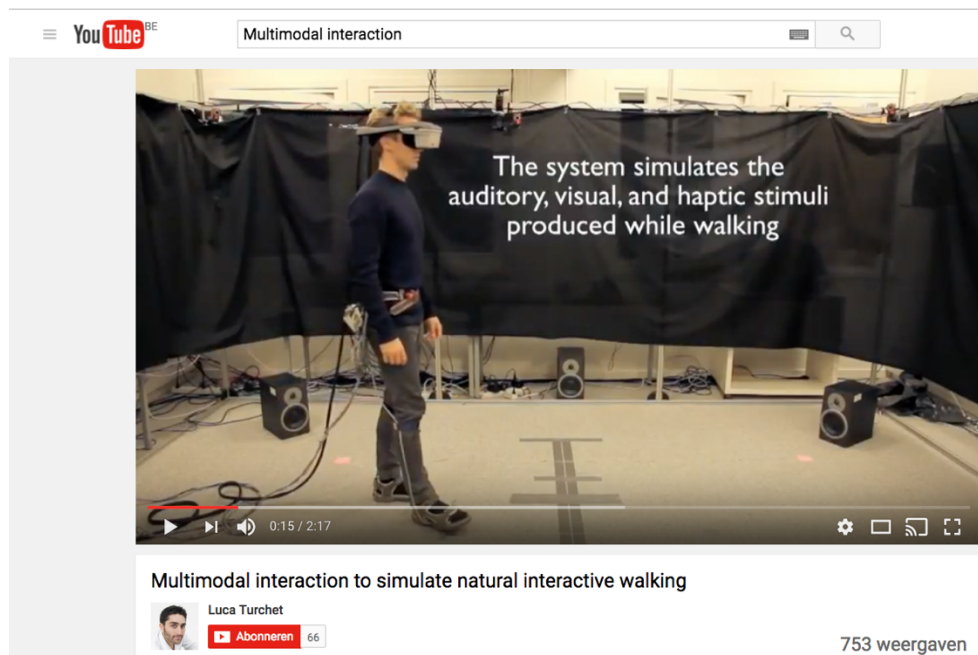


Figure 6.1: Youtube

### 6.1.2 Slide Navigation

Control the slides of your presentation. When using any of the keywords: [slide,page]; in combination with [next,previous,last,first]; (or any slide number) slide navigation can be achieved. Users can navigate to the first, last, previous or next slide and specific slides by slide number as well.

### 6.1.3 Picture Plug-In

Search for pictures on the web. Use any of the keywords: [picture,photo].

### 6.1.4 Search Plug-In

Search the web for anything, with Google search as engine. Use any of the keywords: [google,search].

### 6.1.5 Maps Plug-In

View and browse a map of a particular place, based on Google Maps. Use the keyword: [maps]; to show a map of the place contained in the arguments.

### 6.1.6 Directions Plug-In

View and browse Google Maps containing directions from one place to another. The places are separated by the keyword “to”. Use the keyword: [directions].

### 6.1.7 Weather Plug-In

A plug-in which opens a browser showing the weather on a particular place, contained in the arguments. Use the keyword: [weather].

### 6.1.8 Browser Plug-In

A plug-in which opens a web browser with the extracted website. Use any of the keywords: [web,website,browser,webpage,page,site]. The url is detected by scanning the arguments for “http://” or “www”.

## 6.2 Writing Plug-Ins

Section 5.2 shows how the plug-in system works and how to create plug-ins. This section will show developers how to create plug-ins for each type and how to use them, with the help of an example.

### 6.2.1 Writing Modality Plug-Ins

We will describe the process on how to implement a input modality using speech.

The creation of a modality plug-in consists of two parts, the implementation of the **Input** abstract class and implementing the **Translator** interface.

We start the first part by creating a new Java project with package name `com.vub`. We create an abstract class **Input**, with contents as shown in Listing C.1. In order to create our plug-in, we need to implement the methods

of this abstract class. We create a new class, let's say `SpeechInput`, which will implement the methods `create()` and `getInput()`. Make sure that `Input` and `SpeechInput` both are contained in the package `com.vub.input`. Our next step consists of telling the compiler that our class `SpeechInput` is an implemented version of the abstract class `Input`. For this, a new file needs to be created inside the directory `/META-INF/services/` with name `com.vub.input.Input` and with contents `com.vub.input.SpeechInput`. You might have noticed that these are indeed the corresponding package and class names of the files we have just created. Now that we have created all the necessary links, let's start implementing. The implementation will take place in the `SpeechInput` class, the abstract class has to remain untouched. The first method `create()`, simply is called upon instantiation. We have to set the `type` of the modality (declared in the abstract class `Input`) in string format. This will be shown in the console of the core application upon startup. Next, we have to determine whether our modality makes use of triggers or does not. A trigger allows us to start and stop the capture of input events by pressing and holding keys on a keyboard or a remote control. In the case of speech, it would be useful to start and stop the capture with the help of a remote control. Triggers are already implemented in the core application, the only thing we have to do is set the boolean `triggerRequired` to `true` (Note: by default `triggerRequired` is already set to `true`). Further, it is up to the developers to instantiate more variables here or do some other stuff at startup.

```
public class SpeechInput extends Input{

    public static String recordFile = "./resources/record.wav";

    public void create() {
        this.type = "SPEECH";
        this.triggerRequired = true;
    }

    public String getInput(){

        new Thread(){
            public void run(){
                synchronized (stopTriggerListener){
                    try {
                        stopTriggerListener.wait();
                        Microphone.stopRecord();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }.start();

        Microphone.startRecord();
        return SpeechInput.recordFile;
    }
}
```

```

    }
}

```

Listing 6.1: Writing Plug-Ins: Speech Modality Input Class

The second method `getInput()`, will capture input from the modality which we are using and store it in any format of the developers their choice. Afterwards, they will be the only ones who are dealing with this specific format. The return type of this method is an object which is supposed to be a reference to the capture result. In this example, we have written code which (1) captures the voice of the built-in microphone and (2) stores the voice into a WAV audio file format. As mentioned before, we are using triggers in this example. The method `getInput()` will execute when the trigger is started (i.e. when the button on the remote control is pressed). In order to check when the button is released, we use the `wait/notify` notion of Java. The object `stopTriggerListener` will be notified when the button is released, so all we have to do is wait for it to be notified: `stopTriggerListener.wait();`. All the code after this line will be executed when the trigger button is released. To summarize, the first line in the `getInput()` method is to start voice capturing. In another thread, we wait for the trigger button to be released and stop voice capturing. The voice fragment will then be saved in a WAV file. As output format, we return a string to the path of the recorded voice file. See Listing 6.1 for the code.

In the first part, we have implemented capturing and saving of input signals. The second part consists of accessing the captured signal and translating it into natural language, to be processed further by the core application. Apart from the method and class names, there is a lot of analogy with the first part. We start by creating an interface `Translator`, with contents as shown in Listing C.2. In order to create our plug-in, we need to implement the method of this interface. We create a new class, let's say `SpeechTranslator`, which will implement the method `translate()`. Make sure that `Translator` and `SpeechTranslator` both are contained in the package `com.vub.translator`. Our next step consists of telling the compiler that our class `SpeechTranslator` is an implemented version of the interface `Translator`. For this, a new file needs to be created inside the directory `/META-INF/services/` with name `com.vub.translator.Translator` and with contents `com.vub.translator.SpeechTranslator`. You might have noticed that these are indeed the corresponding package and class names of the files we have just created. Now that we have created all the necessary links, let's start implementing. The implementation will take place in the `SpeechTranslator` class, the interface has to remain untouched. The one and only method `translate(Input input)` its function is to convert

the signal of the captured input (in this case speech) into natural language. In order to access the previous stored signal in the format of our choice, we can make use of the argument `Input input`. Simply cast the `Input` format to the implemented version of the modality (in our case `SpeechInput`) and call the method `getInput()`, which we have implemented a few moments ago. We use Google Speech API to convert speech to natural language, see Section 5.4. See Listing 6.2 for the code.

```
public class SpeechTranslator implements Translator{

    public String translate(Input input) {
        String file = ((SpeechInput)input).getInput();
        return SpeechRecognizer.speechToTextFromFile(file);
    }
}
```

Listing 6.2: Writing Plug-Ins: Speech Modality Translator Class

Finally, compile the full project into a JAR file. Navigate to the folder containing the core application JAR. Put the compiled plug-in JAR file into directory `/modalities/main`, if you want to use it as the main modality, or `/modalities/alternative`, if you want to use it as the alternative modality.

### 6.2.2 Writing Action Plug-Ins

We will describe the process on how to implement the action plugin which can control PowerPoint slides. We start by creating a new Java project with package name `com.vub.term`. We create an interface `TermLoader`, with contents as shown in Listing C.3. Next, create a class `CustomTermLoader`, in the same package, which will implement all the methods from the interface. The `TermLoader` class is designed to contain all main keywords, extra keywords of a specific `Term` and the `Action` linked to it. Our next step consists of telling the compiler that our class `CustomTermLoader` is an implemented version of the abstract class `TermLoader`. For this, a new file needs to be created inside the directory `/META-INF/services/` with name `com.vub.term.TermLoader` and with contents `com.vub.term.CustomTermLoader`. You might have noticed that these are indeed the corresponding package and class names of the files we have just created. Now that we have created all the necessary links, let's start implementing. The implementation will take place in the `CustomTermLoader` class, the interface has to remain untouched. The interface consists of four methods: `init()`, `getMainKeywords()`, `getExtraKeywords()`, `onAction(args,extraKeywords)`. The first method `init()` is called upon creation and is optional. In the case, we create a navigator which can control slides, so we need some interaction with PowerPoint. That is why we start

a socket connection to PowerPoint in the `init()` method, which will send commands to PowerPoint as the user will issue them. The second method `getMainKeywords()` returns a row of strings containing the main keywords, in this case `[slide,page]`. The function `getExtraKeywords()` is analogous, containing the extra keywords `[first,last,previous,next]` in this case. Now that we have defined which keywords should trigger the core application to start an action, we still need to tell it which action should be executed, which need to be implemented in the `onAction(args,extraKeywords)` method. Arguments and extra keywords of the user query are passed to the function, which can be used in the execution code. In this example the extra keywords are enough to consider, the arguments are not needed. Since we know that the main keywords of the term linked to this action will be page or slide, we only need to know whether we need to go to the first,next,previous or last slide. We have implemented a function which compares what option the extra keyword matches to. Afterwards, the appropriate method will be called, which communicates with PowerPoint. Listing 6.3 shows the code snippet.

```
public class CustomTermLoader implements TermLoader {

    @Override
    public void init() {
        //Start WebSocket which sends messages to Presentation Plugin
        /* SOURCE: https://jansipke.nl/websocket-tutorial-with-java-server-jetty-and-javascript-client/ */

        new Thread(){
            public void run(){
                WebSocketSSLStarter.startServer();
            }
        }.start();
    }

    public String[] getMainKeywords() {
        return new String[]{"slide","page"};
    }

    public String[] getExtraKeywords(){
        return new String[]{"next","previous","last","first"};
    }

    @Override
    public void onAction(String[] args,String[] extraKeywords) {
        if( Utils.arrayContainsItem(extraKeywords,"next"))
            Navigator.nextSlide();
        else if( Utils.arrayContainsItem(extraKeywords,"previous"))
            Navigator.previousSlide();
        else if( Utils.arrayContainsItem(extraKeywords,"first"))
            Navigator.firstSlide();
        else if( Utils.arrayContainsItem(extraKeywords,"last"))
            Navigator.lastSlide();
        else{
            long numberResult = Utils.arrayContainsNumbers(args);
        }
    }
}
```

```
        if (numberResult != -1)
            Navigator.gotoSlide(numberResult);
    }
}
```

Listing 6.3: Writing Plug-Ins: Slide Navigator Action Class

# 7

## Evaluation

An evaluation of the system took place amongst 5 different users from various fields. A **System Usability Scale** [42] has been used as a metric to evaluate the usability. The test consists of a set of 10 closed questions. Each question has a scale of 1 - 5, with 1 (**Strongly Disagree**) and 5 (**Strongly Agree**). The answer 3 would for example be a neutral answer, where as 1 and 5 are the two extreme answers. Further, we have asked the participants their favorite **Modality** of the system and also asked them if they would suggest any other modalities. We asked them to note down which **Actions** they tried out and gave them room for to note any extra comments they might have regarding the usability of the system. Each evaluation has taking place at the participants their home and we were also present. We have observed the participants and noted the number of attempts until each action was succesfully launched. We have also asked the participants for suggestions for further improvement. We have summarized these notes and described them in the sections below.

### 7.1 Closed Questions

An evaluation report, which shows the average score of each closed question is shown in Tables 7.1 and 7.2. We have shown the scores of each participant for each question and calculated the average and average rounded scores for



each question. In general, our rounded average scores are equal or one less than the ideal score, which is quite satisfying. Remarkable is question 10, in which all the participants have given the same extreme score of 1. It seems that our system is quite easy to adapt to and not much teaching is needed beforehand. All participants found our system easy or very easy to use and not complex. However, it seems that question 9 is the one with the lowest score. Not everyone is confident on how to use the system. We understand that this might be caused by the number of **Actions** and **Terms**. People can not see everything what is available, so they “guess” what is available, just as with any other virtual agent. To tackle this issue, we could adapt the system in a way that the possible **Actions** and **Terms** are clearly displayed to the users beforehand, or even in PowerPoint itself.

Q	
1	I think I would like to use this system frequently.
2	I found this system unnecessarily complex.
3	I thought this system was easy to use.
4	I think that I would need assistance to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome/awkward to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

Table 7.1: Evaluation: Usability Questions

## 7.2 Comments and Suggestions

Most of the reviewers said that we have created a handy tool. Some of them also said that they would use it, that it is nicely programmed and that it makes the presentation more fluent. We have asked all the reviewers their opinion regarding the possible interactions with the system. All of them preferred **Speech**. One reviewer had some more comments regarding the modalities. He said that **Speech** is definitely the best option and that in the case of **Keyboard Input**, he would not use the system, since it might be

Q	A	B	C	D	E	Average	Rounded	Ideal
1	3	4	5	4	4	4	4	5
2	1	1	2	1	1	1.2	1	1
3	5	4	5	5	4	4.6	5	5
4	2	2	1	2	2	1.8	2	1
5	4	4	3	4	4	3.8	4	5
6	1	2	4	2	1	2	2	1
7	5	5	4	5	5	4.8	5	5
8	1	2	2	3	2	2	2	1
9	3	4	4	3	4	3.6	4	5
10	1	1	1	1	1	1	1	1

Table 7.2: Evaluation: Usability Score

faster to directly do the actions yourself. According to him, **Gestures** are strange to use during a presentation in front of an audience. Two participants have suggested to create a feature which allows users to return to their PowerPoint presentation after an action has been performed. We have tried to implement this in the given time, but it was more intense then expected. Since presentations for a large public mostly require presenters to wear a microphone, one reviewer suggested that we could use the same microphone to do the interaction with the assistant. While using the trigger button, the microphone could cut the contact with the audience and interact with the system. When the button is released, the audience could then listen to the presenter again. Lastly, some reviewers also suggested to create future plugins for inserting pictures in a presentation, making the screen black and opening our facebook or email.

### 7.3 Limitations and Improvements

From the reviews, we could get an overview about how error-prone the system is and what are its limitations. Most of the reviewers have tackled the same problems during their session. We will list the errors here and give some suggestions to improve them. Some users queries, which made sense to humans, could not be recognized correctly by the system, since they did not contain any keywords programmed in the system. For example, the query “Show me a rabbit” does not contain any keywords programmed in the plugins. We humans understand that this user probably wants to see pictures of a rabbit, but our system can not determine this. It could also be the case

that the user wants to see a video clip about a rabbit eating a carrot, which can not be determined by the system without any specific keywords. When no keywords are detected in a query, a future improvement could be to send the user to google with the full query as arguments. Supposing a user query contains more than one keyword, the system can not deliver the same result as if it would contain only one keyword. The system will get confused since it does not know where to split the query and which keywords to use. For example, “Show me a webpage with the weather in Brussels”, contains the keywords **webpage** as well as **weather**, which is a conflict for the system. A naive solution is to program the system in such a way that it will only consider one of the keywords, either the first or the last one, and treat the rest of the keywords as part of the arguments. However, in this solution we are taking the risk that the user actually wanted to do an action corresponding to the ignored keyword. A more better solution could be to link priorities to the different keywords. Keywords which are rare have a higher priority than keywords which are more frequently used. For example, consider the sentence “Search for videos of Albert Einstein”. The keyword **search** will be more frequently used than **video**, so we consider **video** as the resulting action, supposing **search** and **video** are the only two keywords in the sentence. A third limitation is regarding the Speech Recognition API. Speech recognition is still not 100% accurate, so since we are using the Speech API from Google, we are depending on their technology and silently hoping them to develop it as fast as possible. A last limitation is regarding the system itself. One of the assumptions this system makes is that sentences contain keywords and the arguments will always be after the keywords in a query. Queries such as “Show me bunny videos” will not work correctly. Since there are no arguments after the keyword **video**, YouTube will launch without any arguments. In such cases, where a sentence ends with a keyword, we could give all the words before the keyword as arguments to the application.

# 8

## Conclusion and Future Work

### 8.1 Conclusion

In this thesis, we have investigated related work regarding presentation tools, where we have seen that the most popular presentation tool is Microsoft PowerPoint, followed by Apple KeyNote and Prezi. Despite the fact that they are widely used in a lot of different sectors, they require us to prepare all the content which we want to use before. This persists us from using other content on the web, which might be useful during presentations as well, especially upon request from the audience.

Further, we have looked at the state of the art of virtual assistants, which are computerized programs which helps users by doing their tasks for them. Assistants are currently a growing topic, think about Google Now and Apple Siri, but unfortunately their scope is limited to tasks in daily use, such as searching the web, making appointments or checking our email. At the moment, there are no virtual assistants which are available in the domain of presentation tools, to support us using third party applications or navigation during presentations. To address these issues, we have created a virtual assistant to be used specifically in presentation environments. Presenters may still prepare their presentations in advance, but they have the benefits now to dynamically show content from the web, at any time during the presentation, without preparation. The assistant can help to browse the slides of

the presentation in a more effective way, by jumping over a number of slides, going to specific slide numbers or going to slides containing specific keywords. We have created a set of requirements as mentioned in Section 4.1 and have made sure our assistant is based on them.

In order to ensure user friendly interaction, we have created a framework where the interaction with the assistant is up to the user. By default, we give users two options to interact with the system: **Speech Interaction** and **Keyboard Interaction**. However, due to the extensible plugin character of our interaction interface, developers can easily program new interaction modules for the assistant. Currently, the framework provides modules which allows us to browse web pages, search the web for pictures or videos and navigate between slides. Each such module consists of **Terms** and **Actions**. A **Term** determines the vocabulary needed to execute the linked **Action**. Since the plug-in character of the modules, developers can create modules themselves and hence add more actions at any time. Plugins are in the form of JAR files, small compiled application units. The core application scans specific directories and loads the available plugins. Necessary interfaces and abstract classes needed for plugin compilation are available in the appendix. We have created the system in such a way so that it encourages third party developers to create plugins in order to create a growing system for users.

## 8.2 Future Work

The future work consists of developing new term plugins, input modality plugins and improving the mechanism which examines queries. Many new applications can be introduced to the list of possible actions. Plugins could for example search a specific directory and subdirectories for presentations. Related keywords of the current presentation could be searched in other presentations on requests. Keywords could link presentations together and show related work. The content of presentations could be changed depending on the current context. Directions from the current location to the main office of the presenter could be recalculated and shown. See Section 8.2.1 for future possibilities regarding term and action plug-ins.

An error correcting mechanism could be introduced to the query analyser, which uses e.g. hamming distances to calculate the most probable term. Machine learning could be introduced to the mechanism which examines queries. The query analyser could then adapt to the user by using the most frequent queries in the error-correction.

Alternative input interfaces could be designed with the help of plugins. Gestures, special remote controls, smartphone applications, and more.

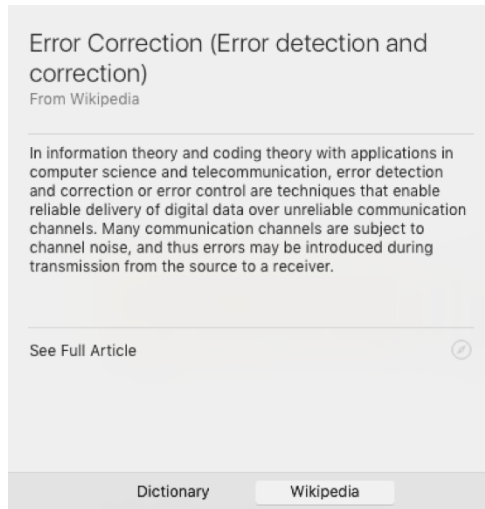


Figure 8.1: Wikipedia

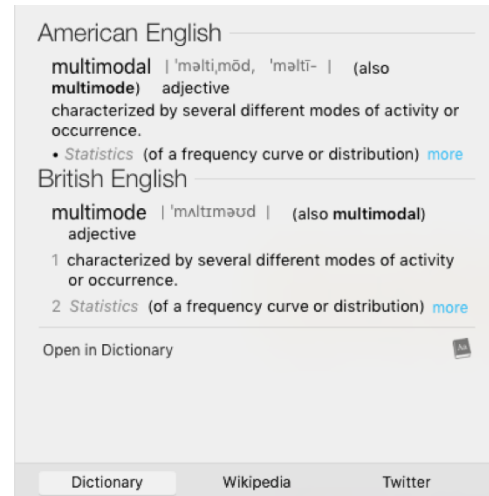


Figure 8.2: Dictionary

As mentioned Section 3.1.2, SSI (Silent Speech Interface/Recognition) could be introduced in future versions of the application.

## 8.2.1 Future Action Possibilities

Due to time constraints, not all ideas could be implemented. Below are some possible use-cases for future development.

### 8.2.1.1 Wikipedia

At any time, if a certain concept is unknown or unclear to the audience, you can query a Wikipedia fragment of the concept and show it to the audience. An illustration is shown in Figure 8.1.

### 8.2.1.2 Dictionary

Similar as Wikipedia. If a certain word or understanding is unknown or unclear to the audience, you can show the explanation of a qualified dictionary explanation of this word/understanding. An illustration is shown in Figure 8.2.

### 8.2.1.3 Slide Keyword Navigation

Navigate to or collect all slides which contain specific keywords or phrases.

### 8.2.1.4 Cross-Linked Content

Suppose an introduction to a concept is explained in the current presentation. However, the audience is not satisfied with the explanation or wants to have more details about the concept. In case you have another presentation, which contains detailed information about that concept, the assistant could look it up and show the related parts on top of the current presentation. An illustration is shown in Figure 8.3.

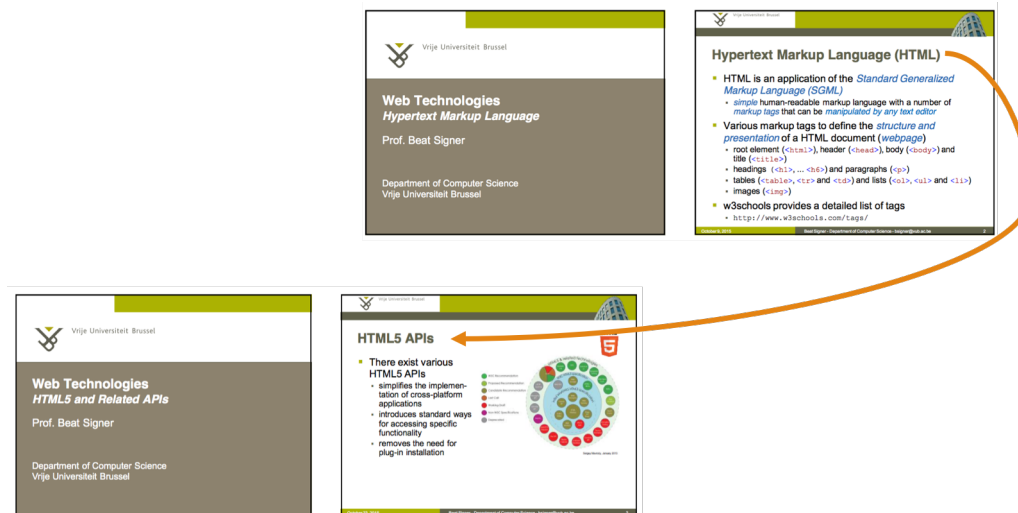


Figure 8.3: Cross-Linked Content

### 8.2.1.5 Anchor Placement

If a lot of questions arise while you are presenting, it is not always easy to handle this efficiently. You could answer all questions at the time of asking. However, this makes the presentation run out of its expected time slot. An alternative is to ask the audience to remember their questions at the end of the presentation. Afterwards, the ones who want or need to leave can go, the ones who would like to listen to answers of the questions can stay. The problem in this scenario is that people might forget the questions which they had, or at which slide it was, especially if the presenter did not place (readable) slide numbers. Anchor placement is a solution for this. When a question arises it he public, the presenter or even the public could issue a request to place an anchor on the current slide. It is like a small icon which will remind you that a question has been asked on this slide. The questions then can be answered later, after the presentation or by email for instance. An illustration is shown in Figure 8.4.

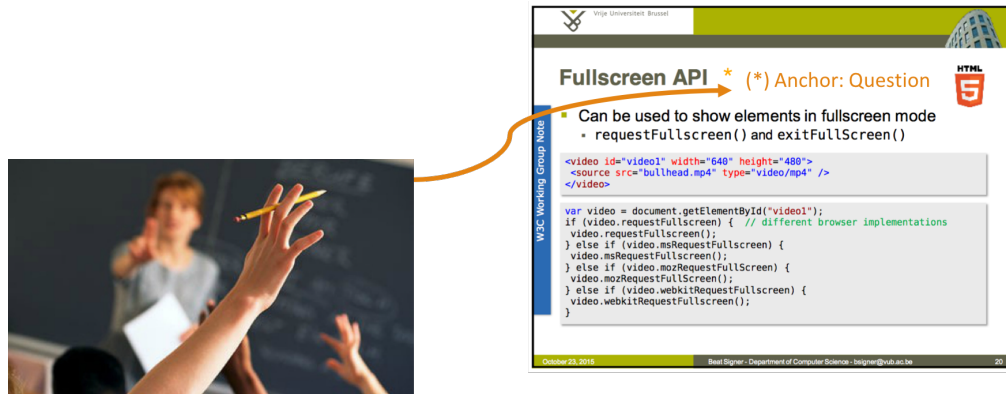


Figure 8.4: Anchor Placement

### 8.2.1.6 Timer

Mostly during exercise classes, the students are given some time to solve a given exercise on their own, or in groups. The given time is mostly not checked, except during exams, simply because it is not doable in one click. This mostly leads to outbalancing of the time of the current class and following classes. The assistant could solve this problem for you. It simply requires you to query a timer with a given time and will show an alarm when the timer finishes. An illustration is shown in Figure 8.5.



Figure 8.5: Timer

### 8.2.1.7 Current Time

Request and show the current time and date, mostly not available in full screen presentations. This could be shown by a disappearing or permanent pop-up augmented on the screen. An illustration is shown in Figure 8.6.





Figure 8.6: Current Time &amp; Date

### 8.2.1.8 Current Exchange Rate

In situations where the content is regarding commerce or economic situations, it is important to keep the used exchange rate up to date. The assistant could handle this. It could also make changes in all the calculations which are dependent on the current exchange rate and show the right outcome. An illustration is shown in Figure 8.7.



Figure 8.7: Current Exchange Rate

### 8.2.1.9 Colour Scheme & Resolution Adaptation

When using electronic slide projectors, sometimes colours appear differently then wanted or on your home computer screen. The assistant can be asked to calibrate the colours of the screen so that everything becomes readable. Similarly for the resolution, for instance on a 4K TV screen, the content of

the presentation, or content in an Internet browser might not be readable due to the high resolution. The assistant could adapt the resolution and calibrate colours to improve readability. An illustration is shown in Figure 8.8.



Figure 8.8: Color Scheme & Resolution Adaptation



# A

## Usage

### A.1 Sideloading Add-Ins with PowerPoint JavaScript API

After having defined and implemented the manifest and html files, we need to load the add-in. Unfortunately, there is no global way, this depends on your operating system.

**Caution!** When using PowerPoint Online, you need to allow self-signed certificates from localhost<sup>1</sup>. If you are using a local version of PowerPoint, your computer needs to trust the self-signed manually. The certificate is delivered with the code and the password is “myPASSWORD”.

**OS X** <sup>2</sup> Follow these steps: (Assuming you are using version 15.33 or higher of PowerPoint.)

1. Place your manifest file in directory `/Users/<username>/Library/Containers/com.microsoft.Powerpoint/Data/documents/wef.`

---

<sup>1</sup>Go to <chrome://flags/#allow-insecure-localhost> and enable “Allow invalid certificates for resources loaded from localhost.”

<sup>2</sup><https://github.com/OfficeDev/office-js-docs/blob/master/docs/testing/sideload-an-office-add-in-on-ipad-and-mac.md>

2. Start PowerPoint and open a presentation.
3. In PowerPoint, choose **Insert >Add-ins >My Add-ins** (drop-down menu), and then choose the Virtual Assistant add-in.
4. Verify that the Virtual Assistant add-in is installed. The task pane should appear.

**Windows** <sup>3</sup> Follow these steps:

1. Share a folder
  - (a) On the Windows computer where you want to host your add-in, go to the parent folder, or drive letter, of the folder you want to use as your shared folder catalog.
  - (b) Open the context menu for the folder (right-click) and choose Properties.
  - (c) Open the Sharing tab.
  - (d) On the Choose people ... page, add yourself and anyone else with whom you want to share your add-in. If they are all members of a security group, you can add the group. You will need at least Read/Write permission to the folder.
  - (e) Choose Share >Done >Close.
2. Specify the shared folder as a trusted catalog
  - (a) Open a new document in PowerPoint.
  - (b) Choose the File tab, and then choose Options.
  - (c) Choose Trust Center, and then choose the Trust Center Settings button.
  - (d) Choose Trusted Add-in Catalogs.
  - (e) In the Catalog Url box, enter the full network path to the shared folder catalog, and then choose Add Catalog.
  - (f) Select the Show in Menu check box, and then choose OK.
  - (g) Close the Office application so your changes will take effect.
3. Sideload your add-in

---

<sup>3</sup><https://github.com/OfficeDev/office-js-docs/blob/master/docs/testing/create-a-network-shared-folder-catalog-for-task-pane-and-content-add-ins.md>

- (a) Put the manifest file of any add-in that you are testing in the shared folder catalog. Note that you deploy the web application itself to a web server. Be sure to specify the URL in the Source-Location element of the manifest file.
- (b) In PowerPoint, select My Add-ins on the Insert tab of the ribbon.
- (c) Choose SHARED FOLDER at the top of the Office Add-ins dialog box.
- (d) Select the Virtual Assistant add-in and choose OK to insert the add-in.

**Office Online** <sup>4</sup> Follow these steps:

1. Open Microsoft Office Online.
2. In Get started with the online apps now, choose PowerPoint; and then open a new document.
3. Open the Insert tab on the ribbon and, in the Add-ins section, choose Office Add-ins.
4. On the Office Add-ins dialog, select the MY ADD-INS tab, choose Manage My Add-ins, and then Upload My Add-in.
5. Browse to the add-in manifest file, and then select Upload.
6. Verify that the Virtual Assistant add-in is installed. The task pane should appear.

**Debugging** In order to try out the JavaScript add-in properly in the PowerPoint environment we need to be able to debug the add-in. Since there is no debugging ability such as console output in PowerPoint, we have used an external debugging system, Vorlon.JS, recommended by Microsoft<sup>5</sup> (for Mac). Vorlon provides console output and input, which made life easier while creating the add-in.

---

<sup>4</sup><https://github.com/OfficeDev/office-js-docs/blob/master/docs/testing/sideload-office-add-ins-for-testing.md>

<sup>5</sup><https://dev.office.com/docs/add-ins/testing/debug-office-add-ins-on-ipad-and-mac>

## A.2 Usage

In order to run the core application, in the folder of the application JAR file:

1. Create a directory “resources” (This will contain resources for the modalities such as temporary speech capture files.)
2. Create a directory “modalities”
  - (a) Create a directory “modalities/main” and make sure it contains the plug-in JAR file of the main modality.
  - (b) Create a directory “modalities/alternative” and make sure it contains the plug-in JAR file of the alternative modality.
3. Create a directory “terms” and make sure it contains all the plug-in JAR files of the terms and their actions you would like to use in the core application.
4. Open a terminal and navigate to the folder of the core application JAR file.
5. Run the program<sup>6</sup>
  - (a) For Mac/Linux, run: `java -cp “Assistant.jar:lib/*” com.vub.main.Main`
  - (b) For Windows, run: `java -cp “Assistant.jar;lib/*” com.vub.main.Main`

---

<sup>6</sup><http://stackoverflow.com/questions/219585/setting-multiple-jars-in-java-classpath>

**B**

**Images**



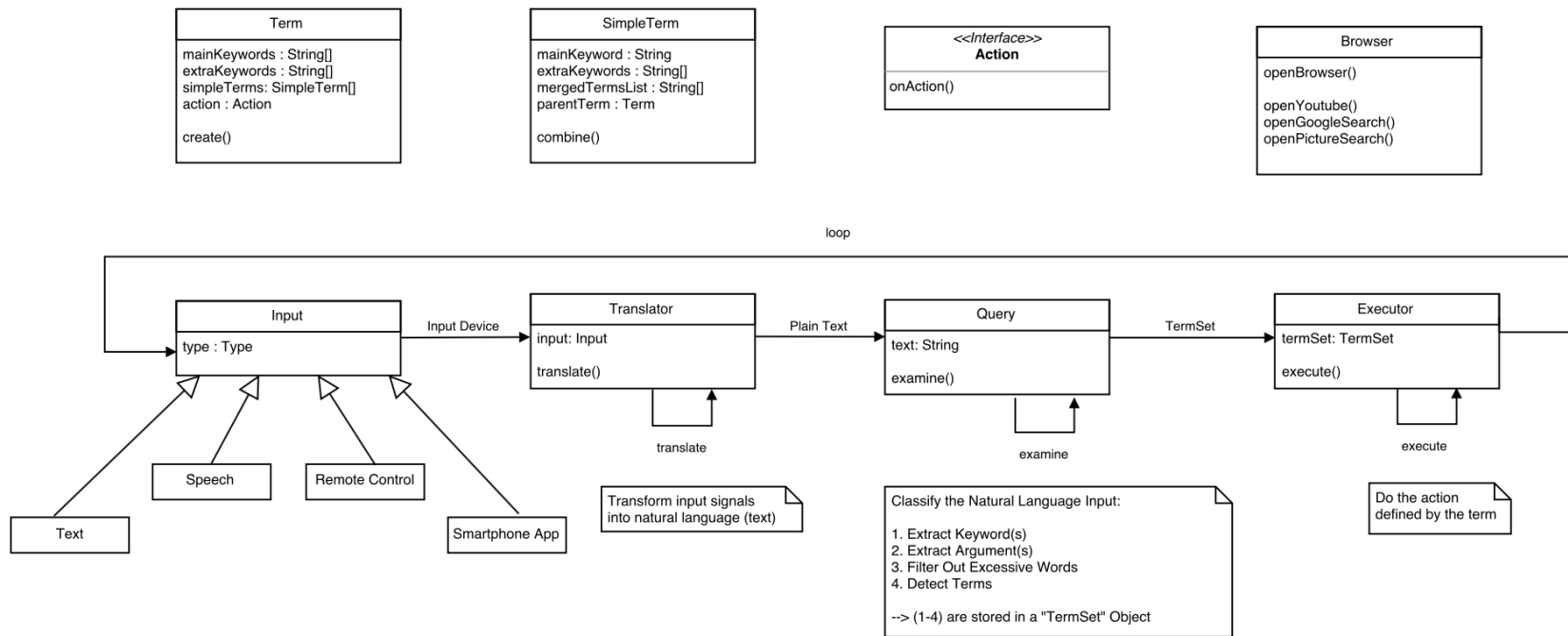


Figure B.1: Architecture

# C

## Code Listings

```
package com.vub.input;

abstract public class Input {

    public String type;

    public boolean triggerRequired = true;
    public Object stopTriggerListener;
    public void setStopTriggerListener(Object stl) {this.stopTriggerListener
        = stl;}

    abstract public void create();
    abstract public Object getInput();

    public String getType(){
        return type;
    }

}
```

Listing C.1: Input Interface

```
package com.vub.translator;
```

```
import com.vub.input.Input;

public interface Translator {

    public String translate(Input input);

}
```

Listing C.2: Abstract Class Translator

```
package com.vub.term;

public interface TermLoader {

    public String [] getMainKeywords();
    public String [] getExtraKeywords();
    public void onAction(String [] args, String [] extraKeywords);

}
```

Listing C.3: Interface TermLoader

```
package com.vub.assistant;

import java.awt.Desktop;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author kushalsoni
 */
public class Browser {

    public static boolean openYouTube(String query){
        return openWebpage("https://www.youtube.com/results?search_query="+
            query+"&playnext=1");
    }

    public static boolean openGoogleSearch(String query){
        return openWebpage("https://www.google.be/?gws_rd=ssl#q="+query);
    }

    public static boolean openPictureSearch(String query){
        return openWebpage("https://www.google.be/search?q="+query+"&source=
            lnms&tbm=isch&sa=X&ved=0ahUKEwj098vF66TSAhXLK8AKHS5gDm4Q_AUICCgB
            &biw=2133&bih=995");
    }

}
```

```
}

public static boolean openWebpage(String urlString){
    URL url = null;

    try {
        url = new URL(urlString.replace(" ", "+"));
        return openWebpage(url);
    } catch (MalformedURLException ex) {
        Logger.getLogger(Browser.class.getName()).log(Level.SEVERE, null
            , ex);
    }

    return false;
}

public static boolean openWebpage(URI uri) {
    Desktop desktop = Desktop.isDesktopSupported() ? Desktop.getDesktop
        () : null;
    if (desktop != null && desktop.isSupported(Desktop.Action.BROWSE)) {
        try {
            desktop.browse(uri);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    return false;
}

public static boolean openWebpage(URL url) {
    try {
        return openWebpage(url.toURI());
    } catch (URISyntaxException e) {
        e.printStackTrace();
    }

    return false;
}
}
```

Listing C.4: Browser Class





# Features of Virtual Assistants

- Google Now
  - Actions & Features<sup>1</sup>
    - \* Activity summary (walking/cycling)
    - \* Birthday
    - \* Boarding pass
    - \* Concerts
    - \* Currency
    - \* Developing story and breaking news
    - \* Events
    - \* Event reminders
    - \* Flights
    - \* Friends' birthdays
    - \* Hotels
    - \* Location reminders
    - \* Movies
    - \* Nearby attractions
    - \* Nearby events

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Google\\_Now#cite\\_note-33](https://en.wikipedia.org/wiki/Google_Now#cite_note-33)

- \* Nearby photo spots
  - \* New albums/books/video games/TV episodes
  - \* News topic
  - \* Next appointment
  - \* Packages
  - \* Parking location
  - \* Places
  - \* Product listing
  - \* Public alerts
  - \* Public transit
  - \* Research topic
  - \* Restaurant reservations
  - \* Sports
  - \* Stocks
  - \* Time to home
  - \* Time reminders
  - \* Traffic and transit
  - \* Translation
  - \* Weather
  - \* Website update
  - \* What to watch
- Voice Commands<sup>2</sup>
- \* General information
    - How old is [Neil deGrasse Tyson]?
    - Where was [Louis C.K.] born?
    - Define [colloquial] (Or “What does [colloquial] mean?”)
    - What time is it in [Tokyo]?
    - Search for [photography tips]
    - Show me pictures of [the Leaning Tower of Pisa]
    - Do I need an umbrella today? What’s the weather like?  
What’s the weather in [New Orleans] [this weekend]?
    - What the [Google] stock price? What is [Apple] trading at?

---

<sup>2</sup><http://www.greenbot.com/article/2359684/android/a-list-of-all-the-ok-google-voice-commands.html>

- What's [182 yards] in [miles]? What is [12 ounces] in [liters]?
- What's [135] divided by [7.5]? (A great many types of math equations will work.)
- Search [Tumblr] for [cat pictures] (more apps are added to this search-within-apps function all the time)
- \* Device control
  - 
  - Open [greenbot.com]
  - Take a picture ("Take a photo" also works)
  - Record a video
  - Open [Spotify]
  - Turn [on / off] [Bluetooth, Wi-Fi, Flashlight]
- \* Productivity
  - What's the tip for [123 dollars]?
  - Set an alarm for [6:30 am]
  - Set a timer for [20 minutes]
  - Create a calendar event: [Dinner with Glenda, Saturday at 9pm.]
  - Remind me to [buy coffee at 7am] (try locations! Remind me to [buy coffee filters at Walgreens])
  - What is my schedule for tomorrow? (also: What does my day look like [Friday]?)
  - Where's my package? (tracking confirmation must be in Gmail)
  - Make a note: [update my router firmware] (also try "Note to self:" This works with multiple apps, and you can even email yourself!)
  - Find [Florence Ion's] [phone number] (Works with all info in your contacts - addresses, birthdays, etc.)
  - Show me my bills. (or: My bills due this week.)
- \* Communication
  - Show me my last messages. (Then follow voice prompts)
  - Call [Jon] (also works with relationships: Call [sister])
  - Call [Cartman] on speakerphone
  - Text [Susie] [great job on that feature yesterday] (also works with relationships: Text [mom] [I'm not going to



- be able to pick you up from the airport, period, I'm a bad son, period])
- Send email to [Robert Baratheon], subject, [hunting], message, [I don't think you should drink so much when you go hunting, period]
- Post to [Twitter]: [Oh my god the Red Wedding episode!]
- What is French for [I am Charlie]?
- Send a Hangout message to [Dad].
- Send a [Viber] message to [Derek]: Hang on, I'm going to get more coffee. (works with WhatsApp, Viber, WeChat, Telegram, and NextPlus)
- \* Navigation and Travel
  - Where is the nearest [sushi restaurant]?
  - Navigate to [Willis Tower, Chicago]
  - Directions to [Fisherman's Wharf] by [bike] (also try "Directions home" or "How do I get home?")
  - Where is [the Louvre]?
  - Show me the menu for [Green Chile Kitchen]
  - Call [Asian Art Museum]
  - Show me my flight info
  - Where's my hotel?
  - What are some attractions around here?
  - How do you say [good night] in [Japanese]?
  - What is [50,000 yen] in [dollars]?
  - What's the flight status of [United] flight [735]?
  - Show me restaurants near my hotel -or- Give me directions back to my hotel (this works if your hotel confirmation was sent to your gmail account)
- \* Entertainment
  - Play [solitaire] (also try tic-tac-toe)
  - Play some music (opens "I'm feeling lucky" radio station in Google Play Music)
  - Next Song / Pause Song
  - Play [Happy] (songs must be in Google Play Music on your device)
  - Watch [The Lego Movie] (movies and TV must be in your Google Play account)

- What's this song?
- Listen to TV
- What songs does [Pharrell] sing?
- Read [Hunger Games]
- Did the [Giants] win today? What's the score in the [Warriors] game?
- What movies are playing [tonight]? Where is [Toy Story] playing?
- \* Sports
  - Say a team name to get the latest score during the season.
  - When is the next [Warriors] game?
  - Where are the [Giants] in the [MLB] standings?
  - Who does [LeBron James] play for?
  - Who won [the Superbowl]?
  - When is the [Stanley Cup final]?
- \* Fun hidden stuff
  - Many of these deliver funny voice responses, but normal search results. Turn up your sound!
  - What sounds does a [tiger] make?
  - Flip a coin
  - Roll dice (rolls a single six-sided die)
  - What is the loneliest number?
  - Do a barrel roll!
  - Askew / Tilt
  - Go go Gadget [Spotify]
  - When am I?
  - Make me a sandwich
  - Sudo make me a sandwich
  - Who's on first?
  - Up, up, down, down, left, right, left, right
  - Tell me a joke
  - Who are you?
  - Beam me up, Scotty!
  - What is [Jennifer Lawrence's] Bacon number?



# E

## Add-ins

### E.1 PowerPoint

1

- Web Video Player  
Get all the action into your deck with this add-in that lets you insert videos from YouTube and Vimeo in your presentation slides. This add-in is free and works with PowerPoint 2013. Powerpoint YouTube Video Player
- iSpring Pro 7  
This PowerPoint-to-Flash converter easily reformats your presentation deck into a web-ready flash/html5 video file (.swf), supporting all PowerPoint animations, transitions, triggers, and links. This app costs a whopping \$397 but you can download and install the free trial version to decide if it's worth it.
- VisualBee  
Need to build a compelling PowerPoint deck fast? No problem. VisualBee automatically upgrades draft presentations using its built-in

---

<sup>1</sup><http://www.skilledup.com/articles/30-best-add-ins-and-apps-for-microsoft-powerpoint>

templates, images, graphics and special effects. It's free with the premium version and works with PowerPoint 2007 and above. Visualbee Powerpoint Enhancer

- **Tick Tock Clock**  
Insert date, time, and location in your Office documents with just a single click. Supports PowerPoint 2013 and can be downloaded for free.
- **Search The Web**  
This clever app allows you to search the web without leaving the presentation deck you're working on. No more need to switch between your browser and PowerPoint. This free app works with PowerPoint 2013.
- **Office Timeline**  
Set your presentation apart with this award-winning app that lets you create visually compelling timelines in PowerPoint. This topnotch add-in integrates seamlessly into PowerPoint's ribbon and works with versions 2007 and up. Office Timeline Powerpoint
- **Mavenlink**  
Working together and keeping track of projects have never been this easy. Share, edit and manage your presentation documents on the go and right out of PowerPoint. This free MS Office app works with PowerPoint 2013.
- **Project Timeline**  
Insert TimeLine into your PowerPoint slides. It allows you to keep track of tasks and milestones, manage resources, group tasks by category and much more. This add-on works with PowerPoint 2010, 2013, and 2016.
- **Poll Everywhere**  
Poll Everywhere enables you to conduct real time surveys with your audience, gather input from their mobile devices, and render the results instantly as compelling PowerPoint 2013 visualizations. Do all that for free! Poll Everywhere Powerpoint Add-on
- **Power Mockup**  
Use PowerPoint to create wireframes for a planned website complete with custom buttons, text boxes, tables, menus and other UI elements. This useful add-in costs \$19.95, offers a free trial period, and works with PowerPoint 2007 and above.

- **Tips for PowerPoint 2013**  
With all the new kickass features introduced in PowerPoint 2013, you would want to get periodic tips on how to further enhance your presentation. This free app provides more than a hundred indexed tutorials that will show you how to optimize PowerPoint 2013's key functions. Learn for free! [Tips For Powerpoint 2013](#)
- **Notes123**  
This add-in allows you to make notes without leaving your PowerPoint document. Perfect for moments when an important idea suddenly emerges in your mind and you want a constant reminder so you wouldn't forget it. Comes to you free and works with PowerPoint 2013.
- **WorkLoadTimer**  
If you need to track how much time you really spend working, this app accurately measures your work duration at the office, at home, or on the go. WorkLoadTimer works with PowerPoint 2013 and you can get it without spending a dime.
- **Oomfo**  
Enhance PowerPoint's built-in chart and graph features with Oomfo and give your presentation the impact it needs. Oomfo's charts are more compelling than the standard fare and include specialized chart styles such as waterfall, Pareto, and pyramid. This free app works on PowerPoint 2003 and later versions (32-bit). [Oomfo Powerpoint Pyramid Chart](#)
- **Lorem Ipsum Generator**  
If you need to fill out a content space or want to see how a mockup slide will look, this app lets you generate placeholder text with just a single click. The Lorem Ipsum Generator add-in is free and works with PowerPoint 2013.
- **Pro Word Cloud**  
Word clouds make it easy for readers to get a general idea of the concepts you are trying to communicate, and they have become a staple element of blogs and other websites. This app enables you to generate a visually appealing word cloud for your presentations. Pro Word Cloud works with PowerPoint 2013 and won't cost you a single cent.
- **PowerTOC**  
Need to create a table of contents or agenda slide real fast? PowerTOC will automatically generate and update a well-formatted sum-

mary, agenda or TOC. This app works on PowerPoint 2000 and above and will cost you \$29.95. PowerTOC Powerpoint Add-On

- PPTools Resize

This add-on comes straight from PPTools, a site developed by Microsoft Most Valued Professional (MVP) and PowerPoint thought leader Steve Rindsberg. It converts standard decks (4:3) to the widescreen format (16:9, 16:10) or to any custom size you want. All without distorting any of the visual elements in your presentation slides. Resize works with versions 2000, 2002, 2003, 2007 and 2010 of PowerPoint. Check out its free demo version.

- Participoll

Add polls to your powerpoint presentation and let your audience vote in real time. It's a free add-on, but there are yearly, annual, and volume plans available. Works with PowerPoint 2010 and 2013 for PC only.

- HTML5Point

This clever app converts your PowerPoint deck to HTML5, allowing you to unleash your presentation to the worldwide web. Once converted, your interactive content can be accessed by tablets, smart phones, and other mobile devices. HTML5Point works with PowerPoint 2003 and above. Download the free trial version to learn more. Powerpoint HTML5 Converter Add-On

- Tabs for PowerPoint

Switching between open PowerPoint documents can be frustrating especially when you want to get things done fast, easy, and efficiently. Tabs for PowerPoint creates a tabs bar for all the decks that you are working on at the same time, allowing for better document switching and management.

- ActivePrez 2

Ever needed to immediately show a slide but can't because the slide is not the next or previous one in your presentation? This app creates a multi-level menu bar that allows you to instantly show a particular slide in your deck without needing to click through all the other slides that lead to it. ActivePrez costs \$99.00 to download and should work for PowerPoint 2007 and up.

- Mind-O-Mapper

Mind-O-Mapper enables you to create great concept or process visualizations as you organize your thoughts, conduct brainstorming sessions,

or draw up strategies and policies. This app won the inaugural app development contest for Microsoft Office and SharePoint 2013. Mind-O-Mapper costs \$2.99 to download and works on PowerPoint 2013. Mind-O-Mapper-Powerpoint-App

- Slideshark

This app converts your PowerPoint into a mobile-optimized format and allows you to show your presentation on your iPad and iPhone, on the web, or on a projector. Slideshark Team edition offers a 30 day free trial. Works with PowerPoint for iPad, and PowerPoint 2010 and above.

- MagPointer

This add-on enables you to zoom, highlight, laser point, enlarge or focus on any area of a PowerPoint slide at any time during a presentation. MagPointer comes with a 29.95 price tag and works with PowerPoint 2003 and up.

- vMaps

Get more than 3000 vector maps that you can use to enhance your presentations — from customizable world maps and country maps to colorful U.S. county maps. If you plan on using Powerpoint to launch your career, vMaps can definitely point you to the right direction. You need to shell out a hefty \$149 to get this add-on, which works with PowerPoint 2007 and up.vMaps Powerpoint World Countries Labels

- LiveWeb

This add-on allows you to insert web pages in your PowerPoint slides. It also updates those pages in real-time during a presentation. Get the app for free and use it with PowerPoint 2007 or later versions.

- Podium

According to its website, Podium is the foremost slide designer, manager, and organizer for PowerPoint. It comes with hundreds of high quality backgrounds, templates, clipart, and stock photos that allow even novices to build a stunning deck within 5 minutes! It works with PowerPoint 2007 and 2010 and comes with a \$49.95 price tag.

- Opacity

This app adds appeal, focus, and impact to your PowerPoint presentations. Use Opacity to highlight key points, blur backgrounds or unwanted text, reveal content, and make a specific area on the slide stand



out. You need to shell out \$49.95 to get this app. It works with PowerPoint 2003 and above.

- STAMP (Subtitling Add-In for Microsoft PowerPoint)

This app allows you to add closed captions to videos and audio files that you embed in your presentations, boosting your deck's clarity as well as its impact on people with hearing issues. STAMP works with PowerPoint 2010 and can be downloaded for free.

- Circlify

Circlify simply gives your presentations that awesome hypnotic appeal. Mesmerize your audience with compelling circular imageries that will leave them in awe of your presentation skills. Grab the app for \$49.95 and use it with PowerPoint 2007 and above. Circlify PowerPoint Add On

- Flevy Tools

Need to create compelling data visualizations to get your message across? Flevy Tools makes it easy to generate and showcase attention-grabbing diagrams, charts, and graphs in your presentations. Get the app for free via a limited time offer.

- PresentationDeck

Take your presentations to the next level with these premium PowerPoint templates. You can filter by category and price and they are available for download immediately after purchase. Templates work with PowerPoint 2007 and higher.

## E.2 KeyNote

2

- Wall Calendar

wall calendar plugin, by Wallace Audley (new)

- Google Plugin

runs Google search for text or phrase selected in KeyNote, by Lonnie Clardy

- Outlook Plugin

sends current note via email, using MS Outlook, by Lonnie Clardy

---

<sup>2</sup>[http://www.tranglos.com/free/keynote\\_addons.html](http://www.tranglos.com/free/keynote_addons.html)

- **Paste Html**  
Plugin To SaveAsHTML after copying from Browser, if the browser provides such a facility, by Jayan Chandrasekhar. (Works with Internet Explorer.)
- **Dialer**  
TAPI phone dialer for KeyNote, by Tom Drahokoupil
- **Function Keys**  
Allows you to customize function keys. Alt, Shift+Alt and Ctrl+Alt function key combinations can be assigned to plugins, macros, styles, templates and fonts.
- **Calendar**  
Displays a simple calendar and allows you to insert selected date in active note. (distributed with KeyNote)
- **Scratch Pad**  
RTF scratchpad, a small text window you can use for taking quick notes or for drag-and-drop text editing between notes.
- **Key Repeat**  
Key autorepeat plugin, repeatedly sends a keyboard shortcut to KeyNote, with optional delay. Can perform an auto-scroll function, or can be useful for inserting an exact number of characters.
- **Erisian Date**  
Discordian calendar. (See Erisian warez page for more...)
- **Test Plugin**  
A test plugin (distributed with KeyNote).



# Bibliography

- [1] Microsoft Corporation. PowerPoint Slide Presentation Software, PPT - Presentation Tool - <https://products.office.com/en/powerpoint>.
- [2] Jean-Pierre Joubert, Jean Greyling, and Charmain Cilliers. The Conversion from PowerPoint (PPT) to Compressed Scalable Vector Graphics (SVGZ). In *Proceedings of South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment*, SAICSIT '11, pages 123–132, New York, NY, USA, 2011. ACM.
- [3] Apple Inc. Keynote - Presentation Tool - <http://www.apple.com/lae/keynote/>.
- [4] Prezi Inc. Prezi Presentation Software - Presentation Tool - <https://prezi.com/>.
- [5] Reinout Roels and Beat Signer. MindXpres: An Extensible Content-Driven Cross-Media Presentation Platform. In *Proceedings of the 15th International Conference on Web Information System Engineering*, pages 215–230, Thessaloniki, Greece, October 2014.
- [6] Reinout Roels and Beat Signer. An Extensible Presentation Tool for Flexible Human-Information Interaction.
- [7] Saurabh Panjwani, Aakar Gupta, Navkar Samdaria, Edward Cutrell, and Kentaro Toyama. Collage: A Presentation Tool for School Teachers. In *Proceedings of the 4th ACM/IEEE International Conference on Information and Communication Technologies and Development*, ICTD '10, pages 30:1–30:10, New York, NY, USA, 2010. ACM.
- [8] Joe Buzzanga. Beyond Keywords: The Revolution in Search - [https://www.sla.org/wp-content/uploads/2015/06/2015\\_buzzanga.pdf](https://www.sla.org/wp-content/uploads/2015/06/2015_buzzanga.pdf).
- [9] WhatsApp Inc. WhatsApp Features - <http://www.whatsapp.com>.

- [10] Google Inc. Google Allo - A smart messaging app - <https://allo.google.com>.
- [11] Ramanathan Guha, Vineet Gupta, Vivek Raghunathan, and Ramakrishnan Srikant. User Modeling for a Personal Assistant. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 275–284, New York, NY, USA, 2015. ACM.
- [12] Apple Inc. iOS 10 - Siri - <https://allo.google.com>.
- [13] Apple Inc. Use Siri on your Mac - <https://support.apple.com/en-us/HT206993>.
- [14] Erica Sadun and Steve Sande. *Talking to Siri: Mastering the Language of Apple's Intelligent Assistant*. Que Publishing, March 2014.
- [15] Mehdi Assefi, Guangchi Liu, Mike P. Wittie, and Clemente Izurieta. Experimental Evaluation of Apple Siri and Google Speech Recognition. In *Proceedings of the 24th International Conference on Software Engineering and Data Engineering (SEDE)*, pages 133–140, San Diego, California, USA, October 2015.
- [16] Mehdi Assefi, Guangchi Liu, Mike P. Wittie, and Clemente Izurieta. Measuring the Impact of Network Performance on Cloud-Based Speech Recognition - An Empirical Study of Apple Siri and Google Speech Recognition. *International Journal for Computers & Their Applications*, 23(1):19–28, March 2016.
- [17] Microsoft Corporation. Cortana - Meet your personal assistant - Global - <https://www.microsoft.com/en/mobile/experiences/cortana/>.
- [18] Andrew Williams. Siri vs Google Now vs Cortana: Which is best? - <http://www.trustedreviews.com/opinions/siri-vs-google-now-vs-cortana>.
- [19] Adam Fourney and Susan T. Dumais. Automatic Identification and Contextual Reformulation of Implicit System-Related Queries. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 761–764, New York, NY, USA, 2016. ACM.
- [20] Amazon.com, Inc. Amazon Echo - Black - <https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00x4whp5e>.

- [21] Steve Dent. Smarter bots are coming to Facebook, Google and Amazon assistants - <https://www.engadget.com/2016/11/17/mindmeld-deep-domain-conversational-ai-platform/>.
- [22] BotList, Inc. Botlist - Various Smart Bots for Amazon Echo, Facebook, iOS, iMessage, Instagram, Messenger, Twitter, Slack, Skype, WeChat and more. - <https://botlist.co>.
- [23] L. Zhang, Y. Shi, and B. Chen. NALP: Navigating Assistant for Large Display Presentation Using Laser Pointer. In *Proceedings of the 1st International Conference on Advances in Computer-Human Interaction*, pages 39–44, February 2008.
- [24] Ha Trinh, Lazlo Ring, and Timothy Bickmore. DynamicDuo: Copresenting with Virtual Agents. In *Proceedings of the 33th Annual ACM Conference on Human Factors in Computing Systems, CHI '15*, pages 1739–1748, New York, NY, USA, 2015. ACM.
- [25] Samuel Gibbs. Baidu launches Duer digital assistant to take on Siri, Cortana and Google Now - <https://www.theguardian.com/technology/2015/sep/09/baidu-duer-digital-assistant-siri-cortana-google-now>. *The Guardian*, September 2015.
- [26] Ethan Selfridge and Michael Johnston. Interact: Tightly-coupling Multimodal Dialog with an Interactive Virtual Assistant. In *Proceedings of ACM on International Conference on Multimodal Interaction, ICMI '15*, pages 381–382, New York, NY, USA, 2015. ACM.
- [27] Karolina Kuligowska and Mirosława Lasek. Intelligent Agents in the Human-Computer Interaction. In *Proceedings of the 7th International Conference on Information Management–Human Computer Interaction*, Uniwersytet Gdański, Gdańsk, 2005. Fundacja Rozwoju Uniwersytetu Gdańskiego.
- [28] Benedikt Schmidt, Sebastian Benchea, Rüdiger Eichin, and Christian Meurisch. Fitness Tracker or Digital Personal Coach: How to Personalize Training. In *Adjunct Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, UbiComp/ISWC'15 Adjunct*, pages 1063–1067, New York, NY, USA, 2015. ACM.

- [29] Sheikh I. Ahamed, Munirul M. Haque, Karl Stamm, and Ahmed J Khan. Wellness Assistant: A Virtual Wellness Assistant Using Pervasive Computing. In *Proceedings of ACM Symposium on Applied Computing, SAC '07*, pages 782–787, New York, NY, USA, 2007. ACM.
- [30] Michal Gordon and Cynthia Breazeal. Designing a Virtual Assistant for In-car Child Entertainment. In *Proceedings of the 14th International Conference on Interaction Design and Children, IDC '15*, pages 359–362, New York, NY, USA, 2015. ACM.
- [31] P. Milhorat, S. Schlögl, G. Chollet, J. Boudy, A. Esposito, and G. Pelosi. Building the next generation of personal digital Assistants. In *Proceedings of the 1st International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pages 458–463, March 2014.
- [32] Bryan Lufkin. Siri Is Woefully Ill-Equipped to Help With Your Mental Health Problems - <http://gizmodo.com/siri-is-woefully-ill-equipped-to-treat-your-mental-heal-1764801513>.
- [33] Adam S Miner, Arnold Milstein, Stephen Schueller, and Eleni Linos. Smartphone-Based Conversational Agents and Responses to Questions About Mental Health, Interpersonal Violence, and Physical Health.
- [34] Lode Hoste and Beat Signer. SpeeG2: A Speech- and Gesture-based Interface for Efficient Controller-free Text Input. In *Proceedings of the 15th International Conference on Multimodal Interaction, ICMI '13*, pages 213–220, New York, NY, USA, 2013. ACM.
- [35] Tanay Pant. Building a Virtual Assistant for Raspberry Pi: The practical guide for constructing a voice-controlled virtual assistant. In *Building a Virtual Assistant for Raspberry Pi: The practical guide for constructing a voice-controlled virtual assistant*, pages 2–3,9. Apress, July 2016. Google-Books-ID: 9TjADAAAQBAJ.
- [36] Kamer Ali Yuksel, Sinan Buyukbas, and Serdar Hasan Adali. Designing Mobile Phones Using Silent Speech Input and Auditory Feedback. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '11*, pages 711–713, New York, NY, USA, 2011. ACM.
- [37] Abdelkareem Bedri, David Byrd, Peter Presti, Himanshu Sahni, Zehua Gue, and Thad Starner. Stick It in Your Ear: Building an In-ear Jaw Movement Sensor. In *Adjunct Proceedings of ACM International Joint*

- Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pages 1333–1338, New York, NY, USA, 2015. ACM.
- [38] Toyin Osunkoya and Johng-Chern Chern. Gesture-based Human Computer Interaction using Kinect for Windows Mouse Control and Powerpoint Presentation. In *Proceedings of the 46th Midwest Instruction and Computing Symposium (MICS 2013)*, volume 60628, Department of Mathematics and Computer Science, Chicago State University, Chicago, IL, April 2013.
- [39] Soonmo Kwon, Hongsuck Seo, and Hyo-Jeong So. A User-centered Design Approach for Developing New Input Modalities and Features in Presentation Tools. In *Proceedings of HCI Korea*, HCIK '16, pages 22–28, South Korea, 2016. Hanbit Media, Inc.
- [40] Beat Signer and Moira C. Norrie. PaperPoint: A Paper-based Presentation and Interactive Paper Prototyping Tool. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 57–64, New York, NY, USA, 2007. ACM.
- [41] Jawin - A Java/Win32 Interoperability Project - <http://jawinproject.sourceforge.net/jawin.html>.
- [42] Kraig Finstad. The System Usability Scale and Non-native English Speakers. *J. Usability Studies*, 1(4):185–188, August 2006.