Universität des Saarlandes
Fakultät für Mathematik und Informatik
Fachrichtung Informatik

Multimodal Fission with a Focus on Collaborative Human-Robot Interaction

Masterarbeit

vorgelegt von

Magdalena Theresa Kaiser

am 22.05.2017

Angefertigt und betreut unter der Leitung von
Christian Bürckert

Begutachtet von
Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
Prof. Dr. Josef van Genabith

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,______________________ ____________________________
(Datum/Date) (Unterschrift/Signature)

Erklärung

Ich erkläre hiermit, dass die vorliegende Arbeit mit der elektronischen Version übereinstimmt.

Statement

I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Saarbrücken,------------------------ --
(Datum/Date) (Unterschrift / Signature)

Abstract

Multimodal systems have been designed to enable a more natural interaction between humans and computers. The system's process of generating a multimodal presentation is called multimodal fission. This is achieved by selecting some combinations of the available modalities and devices. The aim of this thesis is to create a reusable, extendable and domain-independent framework for **M**ulti**M**odal **F**ission (MMF) with a special focus on the area of collaborative human-robot interaction. The framework receives a semantic predicate, providing the information to be presented in an abstract form, as input. It generates a plan containing the selected modalities and devices for each part of the output. When performing the modality and device selection, information about the respective user, previously generated output as well as the current interaction context is taken into account. The framework tackles these selections by solving two constraint optimization problems in which the different planning criteria are formulated as constraints that need to be optimized. The modalities which are available in the framework have been classified into several categories according to their functionalities. As an example, one category contains modalities which are able to generate object references. Referring multimodally to objects in the environment is an important ability for a robot when solving tasks collaboratively with a human. Therefore, the framework also addresses the generation of suitable verbal and multimodal references. The usefulness of these references in particular and of the created multimodal output in general has been verified in a user study.

Contents

1. Introduction

Communication among people is diverse. Information is not only contained in the words chosen by the speaker: The used intonation, facial expressions as well as gestures performed while talking and even the gaze direction can convey information. Nonverbal clues often help to interpret an utterance correctly and more quickly. In contrast to this, traditional interaction with computers is rather simple and limited. Keyboard and mouse are used for input and screens and speakers form the most important output channels. However, so-called *multimodal systems*, which enable a flexible and more natural interaction with computers that users are familiar with from interactions between humans, have been developed over the last two decades [6, 12]. Systems are multimodal if they allow the user to supply input via several modalities, like speech or gaze, and if they use a combination of modalities to present their output. Thus, modalities can be selected which are also used in the interaction between people.

Another advantage of multimodal systems is that these systems enhance robustness due to the combination of partial information sources [12]. For example, it is difficult to understand spoken words in a noisy environment. Yet, receiving redundant information through a combination of speech and gestures may help to understand the meaning even though not every word has been understood. Apart from that, information received over several modalities can be processed better and faster [49]. Furthermore, studies on multimodal interfaces have shown that 95% to 100% of the users prefer multimodal over unimodal interaction [39]. They also found that multimodal interfaces are slightly more efficient than traditional ones - using a multimodal interface speeds up task completion by 10%. Additionally, users made 36% less errors with a multimodal interface than with a unimodal interface. Moreover, multimodal systems can be adapted to the preferences, abilities and special needs of different users and to varying environments. On the input side, it is up to the user to select their preferred modalities. Yet, the system itself must decide which combination of channels to use to output a specific information to a certain user. The system can take information about the current user and the environment into account to generate a suitable output. This is particularly helpful for people with impairments. For example, a robot performing only nonverbal actions, like pointing or gaze, might not be very helpful for people with visual impairments or when operating in dark environments. Furthermore, when displaying something on a screen, visual impairment can be considered by using a bigger font, while people with auditory impairments or persons working in noisy environments might benefit from an increased speaker volume. Hence, using multimodal systems has many advantages.

Figure 1.1 illustrates two important concepts of multimodal systems, namely the *fusion* of inputs and the *fission* of outputs. Different input channels (spoken input, gestures, facial expressions and a GUI) are available in this example. The information from each channel needs to be extracted and combined to form a multimodal representation. This

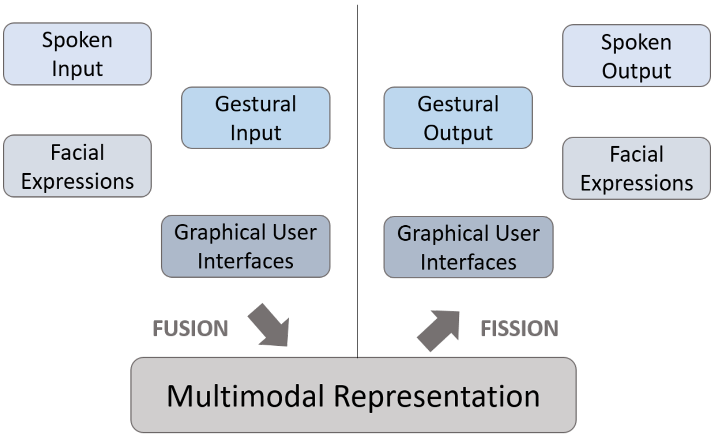


Figure 1.1.: Concept of Fusion and Fission

process is called fusion. The reverse process, which consists of splitting the multimodal representation and distributing it over the desired output channels, is called fission.

Multimodality used for the system's input as well as different approaches for the fusion task have been well studied (e.g., [2, 32, 38, 40]). However, not much research has been conducted about fission [6, 42, 50]. Therefore, this thesis addresses multimodal fission: The aim has been to develop a **M**ulti**M**odal **F**ission (MMF) framework.

The focus has been set on generating multimodal output for the area of collaborative human-robot interaction. Industrial robots traditionally work separately in cages so that they do not harm any people. However, robots are evolving into coworkers that solve tasks closely together with humans. Yet, human-robot collaboration tasks are diverse and are not limited to the industrial context. Cooperative robots can be seen as a step towards taking robots out of the factories and academic laboratories and integrating them into everyday life [25]. It is very likely that future robots will assist people in their homes (e.g., [24]). In particular, they are promising for the care of elderly people (e.g., [4, 15]). Even though human-robot collaboration tasks are diverse, what they have in common is that they benefit from the robot's ability to present information multimodally.

1.1. Research Questions

Five research questions have been investigated in this thesis:

1. **How to design a reusable, extendable and domain-independent multimodal fission framework?**

A framework for multimodal fission should be developed independently of any specific multimodal system, so that it can be connected and used together with various systems. Another requirement is to be able to extend the framework by new modalities and devices as well as new criteria, stating how to select suitable modalities and devices, in an easy manner. Furthermore, the framework should not be tailored to a specific domain. Each of the subsequent research questions considers these design requirements.

2. **How to define the framework's input and output?**

Since the framework should be independent of any specific multimodal system, an easy and well-defined input specification needs to be provided. The framework's output should be stated in such way that the connected devices can understand and process the output easily.

3. **How to generate suitable object references?**

Being able to refer to objects in the environment is an important task of a fission module. In particular, referring to objects in a natural and easily understandable manner is essential for the area of human-robot collaboration. Generating a suitable linguistic object description is of particular importance since speech is the main source of information.

4. **How to categorize modalities inside the framework?**

For a suitable categorization, the functionalities of each modality need to be considered. It should be possible to classify new modalities into existing categories and to be able to extend the framework by new categories, if necessary.

5. **How to select the most suitable modalities and devices for each part of the output?**

The modality and device selection is a major task inside a multimodal fission framework. The selection should take various factors into account, like the current user, information about the environment and previously generated output. The criteria considered in the planning process need to be exchangeable in an easy manner. Furthermore, it is beneficial to be able to state the importance of a criterion. In this way, a crucial criterion must be considered, whereas it is only desirable but not required to consider a less relevant one.

1.2. Thesis Structure

After having introduced the thesis topic and the research questions in this chapter, chapter 2 provides the required foundations on topics that are relevant for this thesis. It defines several terms regarding human-robot interaction, multimodality and multimodal fission. Apart from that, it presents the referring expressions that are used in this thesis and introduces the artificial language Lojban, since the framework's input adopts Lojban's predicate structure. In the end of chapter 2, the theory of constraint optimization problems is presented. After that, chapter 3 gives an overview of related work in the area of multimodal fission and concludes with a comparison of the presented works by criteria that are relevant for this thesis. Chapter 4 explains the concepts developed in this thesis in detail. It presents the input and the output representation of the framework and describes the usage of the output history. Furthermore, the generation of multimodal references in general and an algorithm for generating verbal references in particular will be explained. After that, the categorization of modalities inside the framework will be described. The chapter concludes with presenting the formulation of the modality and device selection as constraint optimization problems. Then chapter 5 describes how these concepts are put into practice, resulting in the implementation of the MMF framework. It gives an idea of the implemented framework, its main structure and procedures. After that, the results of the user study, which has been conducted to validate the suitability of the generated output, are presented in chapter 6. Finally, chapter 7 concludes this thesis and chapter 8 gives a brief outlook on future work.

2. Foundations

This chapter gives some background information about important aspects used in this thesis. At first, collaborative human-robot interaction will be defined. After that, the terminology for multimodal interaction will be introduced and different approaches for multimodal fission will be presented. Furthermore, the referring expressions that are relevant for this thesis will be introduced as well as the artificial language Lojban, since Lojban's predicate structure has been adopted in this thesis. This chapter concludes with defining constraint optimization problems, which are used for formally describing the framework's modality and device selection process.

2.1. Collaborative Human-Robot Interaction

The multimodal fission framework developed in this thesis focuses on the area of collaborative human-robot interaction. Thus, the term will be briefly explained in the following.

Robots are commonly viewed as tools or devices that perform tasks on command [16]. Such a robot has limited freedom to act and will perform poorly if its capabilities are not suitable for a certain task. Furthermore, industrial robots are usually kept in secured environments, like cages, in order to protect human workers. In contrast to this, so-called cooperative robots are designed to work hand in hand with humans. Thus, they must be aware of their surroundings and the actions of human workers to avoid any risk when working closely together. Required safety measures are defined, for example, in the EU Machinery Directive[1].

Definition 1. Collaborative Human-Robot Interaction [16]
A human and a robot work together as partners side by side, sharing the workplace and collaborating to perform tasks and to achieve common goals.

[1] EU Machinery Directive: `http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32006L0042&from=en` (last accessed: 19th of May 2017)

2.2. Multimodal Interaction

A lot of different terms are used in the literature of multimodal interaction, like *mode*, *modality*, *medium*, *device*, *channel* and *multimedia*. These terms are often used with diverse meanings and "the definitions have blurred" [36]. Therefore, this section will introduce the terms and their meanings used in this thesis.

In [52], the term *modality* is used to refer to the human senses: vision, audition, touch, taste and olfaction. The definition of a modality used in this thesis slightly differs from this view and focuses more on the structure of the presented information. It is based on the modality definition in [46]. Furthermore, since this thesis covers the output side of a multimodal system, modalities and devices will be defined regarding the output.

Definition 2. Output Modality [46]
An output modality is a way to convey information from a system to a user. It is defined by the information structure as it is perceived by the user, like speech, gaze, image or vibration.

The difference between human senses and information structure shows the following example: A warning can be transferred via a signal tone or verbally. In both cases, the information is perceived auditorily. Yet, the information structure differs: The tone is a special noise which is interpreted as a warning due to its sound, whereas in the other case, speech is used to express the warning, so the words need to be understood to retrieve their meaning. Therefore, a differentiation is made between a speech modality and an alarm sound modality. This modality definition also differentiates, for example, between spoken language, written language, braille and sign language. Spoken language is perceived auditorily, written language and sign language visually and braille tactilily. Furthermore, written language and sign language cannot be composed to one single modality, even though both types are perceived visually, since the information structure of written language is a concatenation of written letters, whereas sign language is expressed via gestures and facial expressions.

In [36], the terms *devices*, *device components* and *device services* are differentiated. For this thesis, it is sufficient to look at output devices in the following way:

Definition 3. Output Device
An output device is a physical component that is connected to a multimodal system and that is used to execute a system output. It is able to present information from one or several modalities to the user.

A 1:n relation as well as a n:1 relation may exist between modalities and devices. As stated in the definition, one device can output information from several modalities. On the other hand, the information from one modality may be presented by several devices. Figure 2.1 shows this relationship between some sample modalities and devices. For example, images and texts can both be outputted on a screen. Furthermore, in the case of a humanoid robot, pointing can be performed with either the left arm or the right arm.

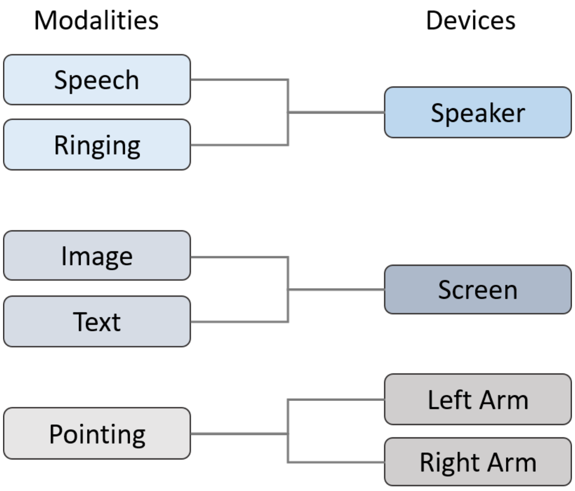


Figure 2.1.: Relationship between some sample Modalities and Devices

Definition 4. Multimodal System [41]
A multimodal system is an architecture that processes two or more modalities for input as well as two or more modalities for output, such as speech, touch, gestures, gaze as well as head and body movements. The modalities may coexist but can be used simultaneously or alternately.

Multimodal systems enable a flexible and more natural interaction that users are familiar with from interactions between humans [6]. Furthermore, multimodal systems often take information about the current context of the interaction into account. A very general definition of an interaction context is given in [11]:

Definition 5. Interaction Context [11]
“Interaction context includes any information that can be used to characterize the situation of an entity. An entity is a person or object that is considered relevant to the interaction between a user and an application, including the user and application themselves”.

Examples for context information are the prevailing lighting conditions and the background noise as well as preferences or abilities of the current user. Adapting the choice of modalities and devices to certain users can be especially helpful for people with certain impairments. For example, the multimodal presentation can be adapted to a person with strong auditory impairment by replacing spoken output with written text, by increasing the speakers volume and by using gestures to emphasize certain words. Yet, considering context information and providing various input and output modalities increases the system’s complexity.

Multimodal systems comprise *multimodal user interfaces* as well as *multimodal dialog systems*. This thesis considers the fission task for multimodal dialog systems. Such system has three main components, namely a *fusion module*, a *dialog manager* and a *fission module*. Furthermore, the system needs different recognizers for the input of several devices and different synthesizers for the output. The used input and output devices need to be connected to the multimodal dialog system. Figure 2.2 gives an

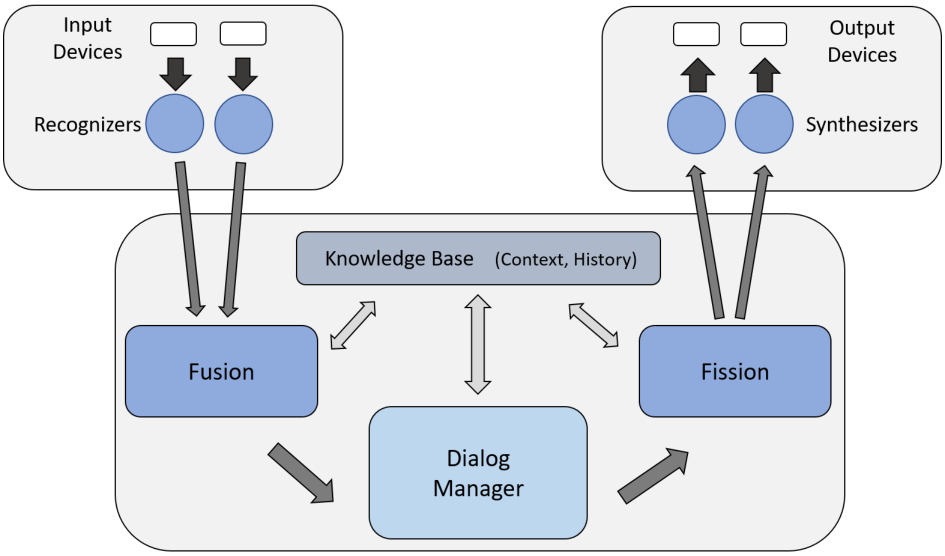


Figure 2.2.: Architecture of a Multimodal Dialog System (based on [12])

overview of the architecture of such a system. First, the system needs to recognize the user's input. Note that, in the case of proactive systems, an input is not necessary for the system to perform a communicative act, since the system may proactively start the conversation. The task of the fusion module is to combine and interpret the information retrieved from the different input recognizers. For the interpretation, the fusion module has access to information from the knowledge base. Then the dialog manager, which is the core component of the dialog system, receives the multimodal input representation from the fusion module and extracts the meaning of it. The dialog manager needs to identify the current dialog state. Moreover, it must decide the transition that is to perform next and the message the user should receive subsequently. It usually retrieves information from the knowledge base for these tasks. In reality, the knowledge base may consists of several models, like a context model, a user model and a discourse model which keeps track of the dialog history. Finally, the task of the fission module is to present the output to the user, using the most adequate combination of modalities. It receives an abstract information from the dialog manager as input and outputs a plan, specifying which concrete information should be presented and what modalities and devices are to be used at which points during the execution. In order to select the most suitable modalities, the fission module has also access to the knowledge base and may retrieve information about the current context and user. Then, each output device receives not only the information that it should output but also the desired time to start the execution of the output.

2.3. Multimodal Fission

Since a framework for multimodal fission has been developed in this thesis, further information about the fission task will be provided in this section.

Definition 6. Multimodal Fission
Multimodal fission is the process of creating a coordinated and coherent multimodal presentation out of an abstract, modality-independent message by selecting some combinations of the available modalities and devices.

The terms *multimodal fission* and *multimodal presentation planning* are often used synonymously. Yet, presentation planning seems to focus more strongly on device selection and on planning the representation of the output on the concrete devices, like layout planning for displays. Thus, the terms are not used interchangeably in this thesis.

The following information is retrieved from Foster's state of the art review on fission from 2002 [17]. Foster divides the tasks in fission into three categories:

- **Content Selection and Structuring**: The task of choosing the content that should be included in the presentation and of arranging its overall structure.
- **Modality Selection**: The task of choosing, among all available modalities, those modalities which are most suitable for realizing the particular output.
- **Output Coordination**: The task of coordinating the different output channels so that the output forms a coherent presentation.

In general, content selection and structuring is the task of the dialog manager: It needs to determine which content should be presented to the user. However, some systems consider this task in their fission module. Most often, such systems use a plan-based approach to achieve this task. In this approach, the fission module receives a high-level presentation goal and a set of generation parameters, such as the user's profile, a presentation objective or resource limitations, as input. The abstract goal is divided into subgoals by using planning operators. This process is performed until all goals consist of primitive tasks which can then be forwarded to specific output generators. Figure 2.3 visualizes this process.

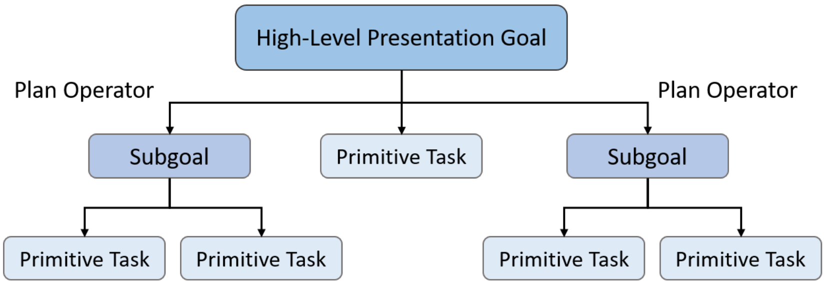


Figure 2.3.: Plan-based Approach for Content and Modality Selection

In order to perform modality selection, the second task mentioned by Foster, knowledge about the characteristics of the available output modalities and the information to be presented can be taken into account. Furthermore, user characteristics, like age or literacy level, the performed task, communicative goals of the presenter and any resource limitations, like the size of the display device, might be considered as well. Foster mentions four possible approaches to perform this task:

- **Composition**: Primarily used for graphic-generating systems. Different UI primitives are ranked and assembled together to form a coherent and more complex GUI.

- **Rule-based**: A predefined set of rules states how suitable certain modalities are for certain situations. These rules can contain knowledge of the domain, the task, user preferences or capabilities of devices. Further application specific rules can be added.

- **Plan-based**: A high-level presentation goal is decomposed into primitive generation task as described above and depicted in Figure 2.3. A plan-based approach performs the task of content selection and structuring. Modality selection happens as a side effect since the required knowledge about the modalities to use is encoded in the planning strategies.

- **Competing and cooperative agents**: A hierarchical system of agents is used to plan a presentation that should meet certain given requirements. The presentation can be improved by negotiating different solutions among the agents meeting various preferred constraints.

For the third task, output coordination, Foster defines three sub-tasks: *physical layouting* (necessary if several modalities use a screen for representing information), *temporal coordination* (necessary if several dynamic modalities like speech or animations are used) and the representation of *referring expressions*, which will be the topic of the next section.

2.4. Referring Expressions

This section gives an overview of the type of referring expressions relevant in this thesis. The main information is taken from Pfleger who gives a broad introduction to the different types of referring expressions [43] from a linguistic point of view.

Definition 7. Referring Expression [43]
A referring expression is a key linguistic phenomenon of verbal interaction that is used for the identification of objects in the real world. The object that is referred to is called *referent.*

An *Anaphora* (Greek: "carrying back") is one type of referring expressions that is used in this thesis.

Definition 8. Anaphoric Expression [43]
"An anaphoric expression denotes a reference to an entity mentioned in the previous discourse, where it is expected that the other person is able to identify the referenced entity."

Examples for anaphoric expressions are the following:

Example 1. Anaphora

- The pencil is next to the paper. Use *it* to write your name.
- Peter stopped the car. *He* thought *it* had a flat tire.

In this first sentence, "it" refers back to the entity "pencil". The second sentence has two anaphoric expressions: "He" refers back to a person named Peter and "it" refers to the car.

Expressions that are used to refer back to an entity that has been previously introduced in the discourse are also called *deictic expressions.* Further examples for deictic expressions include personal pronouns (e.g., I, you), adverbial expressions (e.g., here, now) and demonstrative pronouns (e.g., this, that).

Definition 9. Deictic Expression [43]
"A deictic expression is a referring expression that refers to some entity or concept of the physical, situational or discourse context."

In general, *deixis* (Greek: display, demonstration, or reference) is a form of reference that entirely depends on the context and cannot be interpreted without it. An example of a sentence with a deictic reference is:

Example 2. Deixis
Take this [*pointing to a pencil*] pencil.

Without having access to the visual information about the context - being able to see the pencil that is pointed at - the reference cannot be resolved.

In the following, two forms of deixis which are used in this thesis are explained a bit further:

- **Person Deixis** encodes the role of a referenced person in an utterance. This is mainly realized through the three grammatical categories of person. The first person is used by speakers to make a reference to themselves, whereas the second person is used to refer to one or several addressees. When talking about someone who is neither the speaker nor the addressee of the utterance, the third person is used.

- **Place Deixis** denotes spatial references that depend on the current position of the interacting persons or on other mentioned objects. Examples for place deixis are:

  Example 3. Place Deixis

  - Use the scissors to my left.
  - Use the pen next to the paper.
  - Place the book here.

  In order to be able to understand these expressions, the point of view that the speaker adopts needs to be known. This point of view is also called *frame of reference*. Often a set of primitives is used to describe frames of reference. For example, language specific labels, like “left”, “in the back” or “south” as well as fixed label coordinates can be used. Furthermore, the viewpoint of the observer (e.g., “to my left”) can be considered. Apart from that, the location of the referenced objects can be provided relative to another object, called *relatum* (e.g., “Use the pen next to the big red box“, where “the pen” is the object to refer to and “the big red box” is the relatum).

There are two relevant types of references, which consider multimodality:

- **Multimodal References** are references where two or more modalities are used together - not necessarily simultaneously but in the same time slot - to refer to an entity. Examples for multimodal references are:

  Example 4. Multimodal References

  - Give me this [*pointing to a book*] book.
  - Give me the blue, small book [*pointing to a book*].

  In both sentences, the references are also deictic references, since they cannot be resolved without seeing the pointing gesture. However, in the first sentence, the information of both modalities is needed to understand which object is referenced, whereas in the second sentence the pointing might be redundant if the linguistic description (blue, small) is sufficient to uniquely identify the book.

- **Cross-modal References** are references in which one modality refers to the content of another modality's presentation. Examples for cross-modal references are:

  Example 5. Cross-modal References

  – The upper left corner of the image is yellow.

  – Play the third song in the list.

  In both cases, an element presented on a screen is referenced (in the first case, part of a displayed image and a song from a list of displayed songs in the second case). Thus, cross-modal references do not refer to entities in the real world, but to entities which are presented by another modality.

2.5. Artificial Language Lojban

For the predicate input of the MMF framework, the predicate structure of the artificial, unambiguous language Lojban (pronounced [’loʒban]) has been adopted. Thus, in this section a short overview on Lojban is given.

Development on Lojban started in 1987 by the Logical Language group, which was founded for that purpose. Lojban has a predecessor language called Loglan, which has been invented in 1955. For over three decades, Lojban has been built by hundreds of contributors. The book “The Complete Lojban Language“ [9], published in 1997, serves as a reference grammar, which attempts to describe the whole Lojban language.

The necessity of dealing with ambiguity, polysemy, vague grammar rules, etc., makes the task of constructing a computational representation of natural languages quite hard [48]. Furthermore, in the communication between people, ambiguous sentences sometimes may lead to misunderstandings. Therefore, one aim in designing the artificial language Lojban was to overcome ambiguity. The following is an example for a highly ambiguous sentence:

Example 6. Ambiguous Sentence
John saw the man on the mountain with a telescope.

It is not clear whether a person named John is on the mountain, or whether another man is on the mountain or whether both persons are located on a mountain. Furthermore, John may have had the telescope through which he saw the man. Alternatively, John saw a man who had a telescope or the telescope could be located on the mountain and John saw the man on this particular mountain.

Apart from that, misunderstandings between people may also result from different cultural backgrounds. Therefore, Lojban was designed to be cultural neutral. The phonetic form of Lojban’s 1330 root words, also called *gismu*, has been created algorithmically out of the six most spoken languages in the world, namely Mandarin, English, Hindi, Spanish, Russian, and Arabic. Yet, the result is mainly a mixture of Mandarin and English. Lojban has an active community of speakers: It has been designed as a language usable for the communication between humans, with as much expressive power as a natural language. Furthermore, Lojban is based on predicate logic and has an unambiguous grammar with regular rules, which are free of exceptions. Thus, it can be used “to meet the computer halfway” [48]: Compared to a natural language, it is much easier to translate Lojban into a semantic representation. Lojban’s grammar can be parsed, like some programming languages, by using parsing expression grammars (PEG)[2]. Therefore, there have been proposals to use Lojban as an intermediate language in machine translation and knowledge representation [22, 48]. For example, a semantic parser has been developed which translates Lojban into predicate logic [3]. Furthermore, there are attempts to make use of Lojban for the Semantic Web, like [14] or [54].

[2] Parsing Expression grammars for Lojban: `https://mw.lojban.org/papri/PEG` (last accessed: 19th of May 2017)

In the following, Lojban's predicate structure will be explained in more detail. The information is mainly taken from [9] and from [37].

Verbs in English can express a certain relationship. For example, the English verb "to give" defines the relation between a donor, a recipient and a gift. In Lojban, such a relationship is expressed by a *bridi*, which is the basic building block of a Lojban sentence. Figure 2.4 shows an example for a bridi and its structure. The whole structure presented by the bridi is known as *predication*. A bridi consists of a logical predicate called *selbri*, which may be proceeded by several arguments called *sumti*. The order in which the arguments appear is called the selbri's *place structure*. The simplest kind of selbri consists of a single gismu (root word), whose place structure is explicitly stated in the Lojban dictionary[3]. More complex predicates can be expressed by combining several root words. The number of arguments that can belong to one selbri is limited to five. By convention, they are numerated from left to right with x_1 to x_5. Example 7 shows the place structure of the gismu "klama" (= "to go, to come").

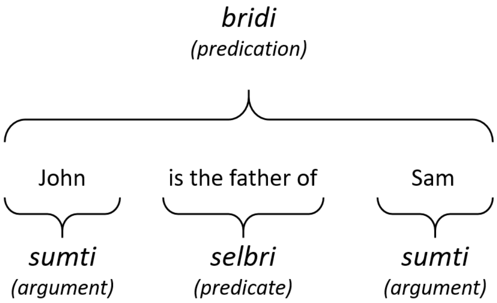


Figure 2.4.: Basic predicate structure in Lojban [9]

Example 7. Place Structure for "klama"

x_1	[*cu*]	klama	to x_2	from x_3	via x_4	using x_5
sumti	-	*selbri*	*sumti*	*sumti*	*sumti*	*sumti*
Moving entity	-	goes	destination	origin	route	means of transport

The gismu "klama" can have up to five arguments, stating the person or thing that is moving, the destination of the movement, as well as its origin, the taken route and the means of transport that is used. The word "cu" has no meaning and exists only to mark the beginning of the selbri within the bridi in order to separate it from a previous sumti. The primary task of constructing a Lojban sentence, after choosing the relationship itself, is deciding how the sumti places should be filled. Example 8 presents a sentence resulting when filling in the arguments in the predicate structure of Example 7.

Example 8. Filled in Sumti for "klama"

[*le prenu*]	[*cu*]	klama	[*le zdani*]	[*le briju*]	[*le zarci*]	[*le karce*]
The person	-	goes	to the house	from the office	via the market	using the car.

[3] English-Lojban Dictionary: `http://tiki.lojban.org/tiki/tiki-download_wiki_attachment.php?attId=711` (last accessed: 19th of May 2017)

It is also possible to put more than one sumti in front of the selbri. This is mainly done for style or for emphasis on the sumti. Furthermore, not all arguments need to be filled in if some information is not necessary or obvious. However, the information cannot be simply left out, since this would violate Lojban’s clearly defined place structure. For example, at position 3 an argument of type x_3, as defined for the corresponding predicate, is expected. Therefore, an irrelevant argument is filled with “zo’e” (= “null”) to keep the defined place structure. The following shows the sentence of Example 8 when only arguments x_1 and x_5 should be expressed:

Example 9. “zo’e” for Irrelevant Arguments

[*le prenu*]	[*cu*]	klama	[*zo’e*]	[*zo’e*]	[*zo’e*]	[*le karce*]
The person	-	goes	-	-	-	using the car.

In this example, only the moving person and the means of transport are relevant. If there are no sumti before the selbri, then it is understood that the x_1 sumti value is equivalent to “zo’e”. Any sumti after the selbri starts counting from x_2. If the irrelevant arguments are at the end of a sentence, “zo’e“ can be omitted:

Example 10. Omitting “zo’e“

[*mi*]	[*cu*]	klama
I	-	go.

Yet, using “zo’e” three times in a row, like in the sentence presented in Example 9, in a conversation would be very inconvenient and would have a negative impact on the usability of the language. Therefore, it is also allowed to change the sumti positions. However, this change must be indicated by using specific tags so that it is still comprehensible which argument can be found at which position. Example 11 shows such a position change. This sentence is equivalent to the one presented in Example 9, but sounds more natural.

Example 11. Switching Position of Sumti

[*le prenu*]	[*cu*]	klama	[*fu*]	[*le karce*]
The person	-	goes	-	using the car.

In this example, the tag “fu” indicates that the following sumti fills place x_5 and not place x_2 as one would expect when the standard place structure is used. Such tags exist for all other sumtis as well. If no tag is provided, then the standard place structure is used.

This is one major advantage of Lojban: For known relationships, the type of argument that comes at a certain position is known. Therefore, language understanding as well as language generation may be facilitated. Section 4.1.1 will describe how Lojban’s predicate structure is used inside the MMF framework.

2.6. Constraint Optimization Problems

The modality and device selection are formulated as constraint optimization problems in the MMF framework. This section will first introduce the term *constraint satisfaction problem*, which is an important component of a *constraint optimization problem.* After that, a definition of a constraint optimization problem is given. The information is taken from Part II, Chapter 5 of [47] and from Chapter 13 of [10].

Definition 10. Constraint Satisfaction Problem (CSP)
A CSP consists of the triple $\langle X, D, C \rangle$, where

- $X = \{X_1, X_2, ..., X_n\}$ is the set of planning variables
- $D = \{D_1, D_2, ..., D_n\}$ is the set of variable domains
- $C = \{C_1, C_2, ..., C_m\}$ is the set of constraints

Each planning variable X_i has a non-empty domain of possible values D_i. Furthermore, each constraint considers a subset of variables and specifies which combination of values are allowed for the subset. A state of the problem is created by an assignment of values to some or all variables. One example for a problem that can be formulated as a CSP is the graph coloring problem. It is the problem of coloring regions (planning variables) with a set of available colors (variable domains), in which neighboring regions are not allowed to have the same color (constraint). Other examples include some logic puzzles, like the 8 queens puzzle, where eight queens must be placed on a 8x8 chessboard in such way that no queen can threaten another one.

The simplest kind of CSPs contain discrete variables over finite domains. In general, solving CSPs for finite domains is a NP-complete problem. The number of complete assignments is exponential in the number of variables: $O(d^n)$, where d is the maximal domain size and n the number of variables.

For standard CSP, all constraints must be fulfilled in order to have a valid solution. However, many real-life problems, like resource allocation and scheduling problems, frequently involve two types of constraints: hard and soft constraints. This leads to Constraint Optimization Problems (COPs):

Definition 11. Constraint Optimization Problem (COP)
A COP consists of the quadruple $\langle X, D, C_h, C_s \rangle$, where

- $\langle X, D, C_h \rangle$ is a CSP; the elements of C_h are called hard constraints
- $C_s = \{f_1, f_2, ..., f_l\}$, is a set of real-valued functions over the scopes $s_1, s_2, ..., s_l$, where $s_i \subseteq X$; the elements of C_s are called soft constraints

A constraint optimization problem extends a constraint satisfaction problem by the set of functions C_s. Each function $f_i \in C_s$ is applied to a subset of the planning variables. A valid solution for a COP is defined as follows:

Definition 12. COP Solution
A solution to a COP is a valid assignment a to $\langle X, D, C_h \rangle$, which additionally maximizes (minimizes) the global cost function $f = \sum_{i=1}^{l} f_i(a)$, with $f_i : a[s_i] \mapsto \mathbb{R}$.

This means that the CSP is extended by a *global cost function*, also called *objective function*, which needs to be maximized (or minimized depending on the context) to receive a solution for a COP. Each function f_i is applied to the subset of planning variables in assignment a which is defined in its scope s_i. It evaluates the assigned variable values in the respective subset by returning a real value. The cost function yields a better result the higher the assigned function values of the soft constraints are. Thus, solving a COP results in fulfilling all hard constraints and maximizing (minimizing) the cost function for soft constraints.

The task of creating a lecture timetable is an example for a COP. Prohibiting two lectures taking place at the same time and in the same room is a hard constraint, whereas preventing a lecture being scheduled on Monday morning would be a soft constraint. Scheduling a lecture on Monday morning can have a cost of 2, while having the lecture in the afternoon might only have a cost of 1. Therefore, the used objective function will try to minimize the costs.

In general, there are several techniques for solving CSPs. The most frequently ones are variants of backtracking, constraint propagation and local search. COPs can be solved by using optimization search methods, like path-based search or local search. Local search algorithms often start with a random assignment which is iteratively improved by changing the values of a small number of variables. The aim is to increase the number of constraints that are satisfied by an assignment. Local search algorithms may find a solution of a problem, but they find not necessarily the optimal one since they can get stuck in a local minimum where no improving neighboring solutions are available.

3. Related Work

As mentioned earlier, multimodality used for the system's input is well studied, whereas not much research has been conducted about multimodal output [6, 42, 50]. One reason might be that most applications use only few different output modalities and thus, simple and direct output mechanisms are often sufficient [6]. Nevertheless, there have been a series of systems considering fission. One of the earliest of such systems was developed in 1992 at the DFKI. It is called WIP [53] and generates illustrated text customized for the users' needs and the situation. Furthermore, one important task in fission is the selection and combination of suitable modalities to generate a useful output. In the following, two concepts on how to effectively combine modalities will be presented. Furthermore, some more recent multimodal systems which consider the fission task will be presented. After that, their fission components will be compared and the differences will be discussed.

3.1. Characterization and Combination of Modalities

In 1995, Coutaz et al. proposed the CARE properties (**C**omplementarity, **A**ssignment, **R**edundancy and **E**quivalence) as a way to characterize and assess multimodal interaction [8]. These properties describe the relationship between modalities for reaching a certain communicative goal.

- Complementarity: Several modalities must be used together in the same temporal window. One modality is not enough to express the desired meaning.
- Assignment: Only one specific modality can be used to express the meaning. None of the others is suitable to be chosen.
- Redundancy: At least two modalities are used together in the same time window. Both have the same expressiveness.
- Equivalence: There are several modalities suitable to express a certain meaning. Any of them can be chosen.

For complementarity and redundancy, the used modalities need to be combined appropriately. The CARE properties form a basis used by various other systems. Most often, the CARE properties are considered for input modalities. Yet, the following is an example for a work that is based on the CARE properties but focuses on output modalities.

Vernier and Nigay proposed a framework for the combination and characterization of output modalities in 2000 [50]. They defined an *output modality* as a tuple consisting of a *physical device* and an *interaction language*. An interaction language defines a set of well-formed expressions that convey meaning. They introduced a *combination space*, that consists of *combination schemas*, which define how to combine several modalities and *combination aspects*, which determine what to combine. They consider temporal and spatial combination aspects, as well as syntactical and semantical ones. For example, for the temporal combination they use five combination schemas: *Sequences*, *anachronism*, *concomitance*, *coincidence* and *parallelism.* Three of the relations describe that the modality usage overlaps and occurs simultaneously (concomitance, coincidence, parallelism). On the other hand, anachronism and sequence are distinguished by the size of the temporal window between the usage of the two modalities. Furthermore, they present characterizations for atomic and composite modalities. For example, they distinguish between the following properties: local versus global, vague versus precise, static versus dynamic and linguistic versus non-linguistic.

In the MMF framework, modalities will be categorized depending on their characteristics and the aim is that their combination leads to a well suited output representation.

3.2. Multimodal Systems with Fission Components

In the following, the multimodal systems and their fission components will be presented in chronological order, starting with earlier systems and concluding with the more recent ones.

The SmartKom project [51], "one of the largest projects world-wide that examined multimodal interaction" [36], was concluded in 2003. During this project, a framework for multimodal dialog systems was developed, with which several prototype systems were realized. Speech, gaze and facial expressions of an animated character are used to create the output in these systems. For the communication between all system modules, a new language called M3L, which is based on XML, was invented. Moreover, the term *symmetric multimodality* was introduced to describe systems for which all modalities used for the input are also available for the output, and vice versa. In SmartKom, a plan-based approach is performed in the fission module. A presentation planner is used, which receives a modality-free representation of the system's intended communicative act as input. It applies 121 possible presentation strategies to decompose the presentation goal into primitive presentation tasks. By using presentation parameters that encode user preferences, the presentation planning process can be adapted to various application scenarios. One example scenario, which has been implemented in the SmartKom project, was an infotainment system. In this scenario, the TV schedule can be presented by the animated character.

In 2005, three years after Foster has summarized the state of the art in fission, Foster and White described their plan-based fission approach used in the COMIC project [18, 19]. The COMIC dialog system is used to realize an intelligent bathroom designer. The output is presented via a GUI and a virtual talking head, which is able to speak, do facial expressions and make deictic references by gaze shifts. In order to find an output which is adequate for the current situation and the current user, the user's preferences and the dialog history are taken into account. The user model is based on multi-attribute decision theory, which means that the overall value for a user is computed as the weighted sum of the values for some primitive features. The user model can either be created offline in advance or the system can start with a neutral user model which is updated during the interaction. The dialog history is used to create links between features of the current description and those in the preceding descriptions. Apart from that, it helps to avoid repetition. Moreover, they present their approach of interleaving output preparation and execution. Their system is able to produce a certain part of the output while still planning other parts. They state that their parallel planning process significantly reduces the time needed to produce an output. Besides, the synthesized speech is prepared in advance and the timing information from the synthesizer is used to create the schedule for the other modalities.

In 2006, Rousseau et al. presented a conceptual model for multimodal presentations called WWHT model [46]. This model includes the following four concepts: **W**hat information to present, **W**hich modalities to choose to present the information, **H**ow to present this information using these modalities and **T**hen - how to handle the evolution of the resulting presentation. Apart from the last concept, the others are related to the three main tasks in fission defined by Foster (see section 2.3). The last concept represents the problem of how to react if the interaction context changes during the presentation. This is mainly important for persistent presentations. Furthermore, they state that, for representing suitable output, the interaction context and the choice of the interaction components need to be considered. They distinguish three types of interaction components: *mode*, *modality* and *medium*. Modes correspond to human sensory systems (visual, auditory, tactile, etc.), a modality is defined by the perceived information structure (text, image, vibration, etc.) and a medium is a device used to produce the output (screen, speaker, vibrator, etc.). Besides, the modality selection process (Which-question) is called *allocation process* by the authors and consists of choosing a suitable set of modality-medium pairs for the representation. For this purpose, Rousseau et al. use a base of election rules. They define three different types of rules, namely *contextual*, *property* and *composition* rules. Contextual rules take the interaction context into account (e.g., noisy environment) and property rules refer to modality properties (e.g., linguistic or analog). Composition rules state, for example, whether a combination of redundant modalities should be used. Based on the WWHT model, they presented a platform for the design and the development of multimodal output systems called ELOQUENCE. This platform contains one tool for the output specification, one for the output simulation and a runtime kernel for the execution of the output. They state that the specification tool allows to reuse the output specifications, which are saved in MOXML (**M**ultimodal **O**utput e**X**tended **M**arkup **L**anguage) format, a data representation language based on XML invented by the authors themselves.

Hina et al. developed a multimodal system based on multi-agents in 2011 [26]. The aim of their system is to be adaptable to the interaction context and to be independent of the application domain. In their work, the interaction context consists of three parts: the user context, the environment context and the system context with information about the available computing resources. They use a machine-learning approach and case based reasoning. This means that a new scenario, also called *case*, is compared to all previous cases. In order to be able to do this, the different cases are always composed of the same components consisting of different context parameters which model the interaction context. Each context parameter consists of a name and a value (e.g., "noise level = 1" refers to a quiet working environment). The similarity between the new case and each of the previous cases is calculated. The modalities are selected which have been used in the previous case which has the highest similarity score when compared to the current case. In the beginning, the system has some test data as initial knowledge. Over time, new cases and the modalities chosen for them will be memorized. It is up to the user to decide whether a case should be memorized. Thus, the machine-learning component will gradually increase its knowledge on the used domain. The idea is that, over time, it will have learned which modalities are most suitable for specific situations. Furthermore, in order to decide which device should be chosen for a selected modality, they use a priority ranking of devices (e.g., for audio output, speakers have a higher priority than headphones).

In 2012, Honold et al. presented their Probabilistic Fission (ProFi) system [27], which is designed to reason on adaptive and multimodal output based on uncertain or ambiguous data. Their aim is to dynamically adapt the user interface by taking data from sensors, which deliver diverse and sometimes uncertain information, into account. They seem to be the first who use a probabilistic approach for the fission task. They propose to divide fission into an *early fission* and a *late fission* step similar to the concepts proposed by Foster: Early fission corresponds to content selection and structuring and late fission is covered by modality selection and output coordination. Furthermore, they present a reasoning algebra which can be used with an arbitrary context model. In all steps, context knowledge about the available devices, the users and the environment is considered. XML is used for the communication between all modules. In order to select the most suitable modalities, they identify all possible output configurations and their combinations first. In a second step, each configuration is rated by using a set of probabilistic rules. The combination with the highest reward is chosen. This second step is performed in parallel in order to fulfill real-time requirements. They also include a post-processing step, in which, for example, private messages are obfuscated if only public devices are available for presentation.

In the scope of the European project GUIDE, Costa and Duarte developed a multimodal user interface for elderly and differently impaired users in 2013 [7]. Their proposed framework has a particular focus on its adaptive multimodal fission component for TV-based applications. Their goal is to support interface adaption . In order to understand the applications' UI, the framework uses UIML as their interface markup language. Furthermore, they use a context model together with a user model, since they prioritize to adapt the choice of output representations to a specific user. In order to achieve this, the user interface is complemented with other modalities. This process is called *augmentation.* In addition, the interface is adjusted to the abilities of the users. For example, color, contrast and audio volume can be adjusted, but also wider spacing between buttons can be applied for people with, for example, Parkinson's disease. Costa and Duarte have conducted a user study with participants who have age-related disabilities. In the study two versions of a TV-based application have been compared, in which one version has been adapted to the special needs of the user. Results show that the adaption has been perceived and assessed positively. Furthermore, the users needed less time, made less mistakes and asked less frequently for help when using the adapted application.

In 2015, Neßelrath presented SiAM-dp [36], a platform for the development of multimodal dialog systems in *cyber-physical environments* (CPEs). The term CPE denotes "the connection of the cybernetic world of computer and communication with the real world". SiAM-dp's fission component receives the semantically represented output from the dialog manager. First, the system needs to determine, with the help of the device manager, which devices are currently available in the CPE. In the next step, the prioritized devices are determined. User preferences as well as the current environment context can be taken into account for this step. The result is a ranked list of devices. The data for presentation is distributed to the devices in the priority list based on device specific features, taking into account that not every device is suitable for presenting every output. In a last step, private data can be obfuscated if presented on a public display and the overall presentation of information needs to be synchronized.

3.3. Comparison of Fission Components

The fission components of the presented systems and the MMF framework presented in this thesis will be compared among each other according to criteria which are important for the MMF framework. Table 3.1 gives an overview of the comparison. Note that this table should not primarily be seen as a comparison of features but more as a basis for discussing differences in aspects which are relevant for the MMF framework. The following criteria are considered:

- **Fission Approach**: Which approach is used for the fission task?
- **Semantic Input Language**: In which form does the fission component receive the abstract information from the dialog manager?
- **Adaptability**: Is the produced output adaptable to
  - the **Interaction Context**: Are user preferences, information about the environment or any other information about the context taken into account?
  - the **Interaction History**: Are previous statements, previously referenced objects or previously made decisions about modality and device selection taken into account?
- **Extensibility**: Is the fission component extendable by
  - **new modalities and devices**: Can new modalities and devices be easily added?
  - **new planning criteria**: Can new planning criteria be easily added?
- **Reusability**: Can the Fission component be easily used for creating new applications?
- **Focus**:
  - **Output Centered**: Is the focus of the overall presented work on the fission component?
  - **Modality Selection Centered**: Does the presented fission component focus on the modality selection process?
- **HRI Suitability**: Can the fission component produce output that is suitable for human-robot interaction?

A “x” in the table denotes that the corresponding system fulfills a certain aspect, whereas a “?” denotes that the information provided by the literature is not enough to be able evaluate the corresponding aspect. A “-” means that the aspect is not relevant for the respective system. Futhermore, some abbreviations are used in the table: “ELOQ” is short for “ELOQUENCE”, referring to the system presented by Rousseau et al. [46] and “ML” denotes the multi-agent system presented by Hina et al. [26]. Their system is the only presented system which uses a machine-learning approach (abbreviated as “ml” fission approach in the table). This is a novel approach which is not included in

	Fission Approach	Semantic Input Language	Context Adaptability	History Adaptability	Extendable Modalities	Extendable Criteria	Reusability	Output Focus	Modality Focus	HRI Suitability
COMIC	plan	XML	x	x		?		x		
SmartKom	plan	M3L	x	?	x	?	x			x
ELOQ	rules	MOXML	x		x	x	x	x		x
ML	ml	?	x	x		-	x	x	x	x
ProFi	prob. rules	XML	x		x	x	x	x	x	
GUIDE	rules	UIML	x			x		x		
SiAM-dp	?	XML-based	x		x	?	x			x
MMF	constraints	Predicate	x	x	x	x	x	x	x	x

Table 3.1.: Comparison between Fission Components of presented Systems

the standard approaches for fission presented by Foster [17]. Two systems use a plan-based approach (COMIC, SmartKom) and three are rule-based (ELOQUENCE, ProFi, GUIDE), though ProFi uses probabilities (abbreviated as “prob. rules” in the table), which is also novel. From [36] the approach of SiAM-dp’s fission module can unfortunately not be retrieved. In plan-based approaches, modality selection happens as a side-effect, according to Foster [17] (see section 2.3 for further details). Furthermore, rule-based approaches often tend to keep the modality selection process rather simplistic. Since modality selection is an important part of the MMF framework, a novel approach has been favored, which is called *constrained-based* approach in this thesis. The selection of suitable modalities and devices is seen as a constraint optimization problem (for more details see section 4.6).

The ProFi fission system receives a semantic input represented in XML, whereas SmartKom’s fission component receives its input in the XML-related language M3L. For the ML system, no information is provided, whereas for the remaining systems (COMIC, ELOQUENCE, GUIDE, SiAM-dp) only the language used for communication throughout the fission component or throughout the whole system is mentioned. In all cases, XML or a XML-related language is used. Thus, it can be assumed that the fission component in these systems also receives their input represented in XML or a XML-related language. The fission framework presented in this thesis receives a semantic predicate, based on the predicate structure of the artificial language Lojban as input (see section 4.1.1 for further details).

All of the presented systems consider the interaction context in some manner. This shows that being adaptable to the interaction context is an important property of a fission component. The COMIC system and GUIDE particularly focus on adapting the system to user preferences. Taking into account information about the user is also a very important context information considered by the MMF framework.

As the table shows, the interaction history is clearly considered less often by the presented systems than the interaction context. In the MMF framework, references to objects are memorized in order to adapt the generated output to the conversational situation. For example, it is not necessary to point at an object again, if the focus is already on that object since it has been mentioned in the previous sentence. The COMIC system also makes highly use of the interaction history. In the ML system, modalities for a certain scenario are chosen based on previously seen scenarios. This can be considered as interaction history. Yet, it is not necessarily the interaction history of one specific user but the history of all previously seen interactions in a certain domain. In SmartKom, information about previously mentioned discourse objects is stored. However, this information is mainly used in order to understand references from the user (e.g., "take the third one") but does not seem to have an impact on the generated output.

It is an aim of the MMF framework to be extendable in an easy manner by adding new modalities and devices and by introducing new planning criteria. Both, the multimodal dialog system COMIC and the multimodal user interface GUIDE have their main focus on context adaptability. However, both are primarily tailored to the scenario they present. Therefore, the used modalities and devices are scenario specific and might not be interchanged easily. New modalities might be added for the ML system. Yet, it can be assumed that they cannot be added in an easy manner, since the system first needs new test data for the new modalities. Furthermore, for previously seen cases the choice of modalities might change when a new modality is available. Therefore, the old memorized cases might not be used anymore. As far as can be judged, it is possible to add new modalities and devices to the other presented systems without too much effort. Furthermore, it is rather easy to add new planning criteria to rule-based systems by extending the existing rule set. Unfortunately, the literature does not provide enough information to judge whether new criteria can be added easily in COMIC, SmartKom and SiAM-dp. The question is not relevant for the ML system, since those modalities which have already been used in a very similar case are chosen for the current scenario. Therefore, no real planning criteria are needed.

As stated earlier, COMIC and GUIDE seem both tailored to their concrete application scenario (presenting bathroom designs in the case of COMIC and making TV-based applications accessible for elderly people in the case of GUIDE). Therefore, their reusability is assessed as low. SiAM-dp has been designed as platform for the development of multimodal dialog systems and ELOQUENCE is a platform for the development of multimodal output systems. Therefore, reusability is given in both cases. Furthermore, the SmartKom system describes itself as a reusable dialog shell. ProFi's approach of using probabilistic rules and a reasoning algebra for modeling context information is reusable, as well as the machine-learning based approach for modality selection used in the ML system. The MMF framework is also reusable, since it is designed to be used together with different dialog managers. Furthermore, various applications can easily be created in the area of human-robot interaction.

SmartKom and SiAM-dp are suitable for the development of entire multimodal dialog systems. Thus, the focus of their work does not lie on the fission component. For the other presented systems suitable output generation is a central topic. ProFi and the ELOQUENCE platform are systems for solving the multimodal fission task. The framework presented in this thesis has the same aim.

Selecting the most suitable modalities is a key task inside the MMF framework. In ProFi, all possible output configurations and their combinations are identified. Then each configuration is evaluated with a rule set to identify the best output modalities. Therefore, ProFi also has a focus on the modality selection task. Machine-learning is used in the ML system in order to select the most suitable modalities based on previous seen scenarios. Thus, the modality selection plays a major part in the ML system.

Furthermore, the MMF framework has a special focus on the area of collaborative human-robot interaction. As far as known from the literature, none of the other presented systems has been used to develop an application in this area. GUIDE and ProFi are not suitable for this area since they focus on multimodal user interfaces. As stated above, the COMIC system strongly focuses on its application scenario and therefore is not suitable for designing a fission module for human-robot interaction. It might be possible to use any of the other systems to design applications in this area. In particular, ML and ELOQUENCE could be suitable due to their general structure.

In conclusion, COMIC and GUIDE have a strong focus on adaptability to the user and their interaction contexts. However, they are both tailored to their respective application scenario and are therefore not very reusable. Furthermore, ProFi and GUIDE are limited to the development of multimodal user interfaces. SmartKom and SiAM-dp can be used for the development of entire multimodal dialog systems. Therefore, both seem to focus more on other parts than fission. Furthermore, SiAM-dp particularly does not give much information about its fission component. It primarily performs the task of ranking the available output devices. Therefore, its main emphasis is put on presentation planning (as defined in section 2.3) instead of fission. SmartKom's fission module is "controlled by a presentation planner" [51] and it also seems to prioritize output coordination, in particular physical layouting and the consideration of device specific characteristics. The ELOQUENCE platform provides a graphical tool which is rather suitable for developing small exemplary systems. The fission component of the ML system focuses on modeling the interaction context. Furthermore, using machine-learning for the modality selection is an interesting approach with the disadvantage that, for each new application, new test data is required.

The framework developed in thesis is not tailored to be used as part of any specific multimodal system, but can be connected to different ones by only requiring a semantic predicate as input. Thus, extensibility and reusability are very important. Furthermore, the MMF framework takes the interaction context and the previously generated output into account to be adaptable regarding the user's needs and to improve the quality of the multimodal output. Modality selection is one major task of a fission component. A constrained-based approach is used to solve this task. The MMF framework has a special focus on human-robot interaction: It comes with some modalities and devices as well as some planning criteria for this area, which can be directly used out of the box, without being limited to them.

4. Concept

This chapter will provide an overview of the main concepts realized in the MMF framework. The MMF process is depicted in Figure 4.1. The framework receives a semantic predicate and a world model as input and outputs a plan which can be executed by a provided execution component if desired. The most important topics covered in the MMF framework include reference generation, modality categorization, the output history as well as modality and device selection.

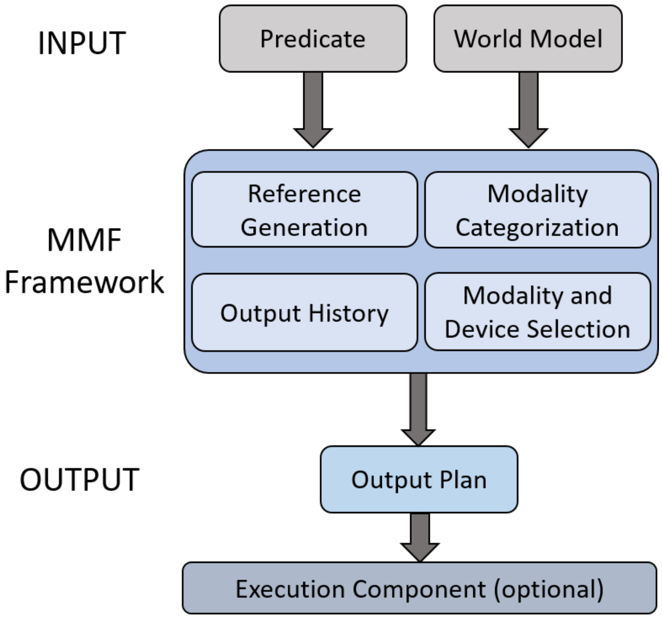


Figure 4.1.: Overview of the MMF Process

This chapter will first cover the input and the output representation. Then, the actual framework will be presented, starting with the output history. After that, reference generation in general and an algorithm for the creation of linguistic reference in particular, as well as one for assessing the quality of pointing gestures, will be explained in detail. Categories for classifying the different modalities are presented and, in the end, a detailed presentation of the modality and device selection process is given.

vecnu	x_1 (*seller*) sells x_2 (*goods*) to x_3 (*buyer*) for x_4 (*price*)
klama	x_1 goes to x_2 (*destination*) from x_3 (*origin*) via x_4 (*route*) using x_5 (*means*)
tavla	x_1 (*talker*) talks to x_2 (*audience*) about x_3 (*topic*) in language x_4
dunda	x_1 (*donor*) gives x_2 (*gift/present*) to x_3 (*recipient*)
blari'o	x_1 (*object/light source*) is blue-green
melbi	x_1 (*object/idea*) is beautiful to x_2 (*observer*) by standard x_3
barda	x_1 is big/large in property/dimension x_2 as compared with standard x_3

Table 4.1.: Some Example Predicate Structures stated in the English-Lojban Dictionary

4.1. Input Representation

There are two major inputs to the MMF framework: a semantic predicate and the world model. The semantic predicate is a high-level representation of the information to be presented, whereas information about the context as well as the available modalities and devices are contained in the world model. In the following, more detailed information about the world model and the predicate input will be provided. Another input specifies the recipient of the output, which can be directed to one or several people. The output language and information about the natural language generation format need to be passed to the framework as well. Further information about this can be found in section 5.2.

4.1.1. Semantic Predicate

The MMF framework requires a semantic input that encodes the information, which should be presented to the user, in an abstract form. This input needs to be provided by a potential dialog manager, like the dialog manager of SiAM-dp [36]. As already stated in section 2.5, the required semantic input of the MMF framework consists of a predicate that uses the structure of the predicates in the artificial language Lojban. There is an English-Lojban dictionary[1], which states the place structure for the Lojban predicates. In order to get an impression of its entries, Table 4.1 shows examples for the defined place structure of some predicates.

Lojban has been developed over more than three decades. During this time the developers of the language have put a lot of effort into the design of suitable predicate structures. Therefore, using this elaborated predicate structure is beneficial: Whenever a new predicate should be supported by the MMF framework, the predicate structure does not need to be defined from scratch, but the structure of the corresponding Lojban predicate can be looked up in the dictionary. This is always possible because Lojban is as expressive as a natural language. As a result, all predicates are built in the same way and adding new predicates can be done in an easy manner.

A language called Lojban++ is described in [23], which uses English vocabulary in Lojban structured sentences. The aim is to make it easier to learn and understand the language, while keeping the advantage of being more easily understood by a com-

[1] English-Lojban Dictionary: `http://tiki.lojban.org/tiki/tiki-download_wiki_attachment.php?attId=711` (last accessed: 19th of May 2017)

puter than natural languages. This idea has been adopted for the MMF framework: The predicates used in the framework follow Lojban's predicate structure and also keep the names of the selbri as predicate names, but they use English words as arguments. Example 12 shows a typical predicate input used in the MMF framework.

Example 12. Predicate Input
vecnu(robot1, vase3, user2, 40€)

In the example, the Lojban selbri "vecnu" (= "to sell") is used to uniquely identify the predicate. The predicate structure of "vecnu" can also be found in Table 4.1. Its arguments are filled with English words, specifically with the unique ids of the objects (*robot1*, *vase3*, *user2*) and a price quotation. These entities are in relation with each other as defined by the "vecnu" predicate. The structure of the predicate input is always the same: *predicate_name*(x_1, x_2, x_3, x_4, x_5), with up to five arguments depending on the predicate. If not all arguments are necessary, "z'oe", Lojban's indicator for irrelevant arguments, is used. Example 13 represents the same input as seen in Example 12, except for the price argument, which is obvious or irrelevant in Example 13 and should not be told to the user.

Example 13. Predicate Input with "z'oe"
vecnu(robot1, vase3, user2, [zoe])

In Lojban, it would be possible to leave argument x_4 entirely out, since it is the last argument of the predicate. Furthermore, it is possible to swap arguments in Lojban (see section 2.5). However, in order to facilitate parsing the input, "zoe" is always present to indicate left out arguments and the standard order of arguments is used as defined in the dictionary. Note that the apostrophe inside the Lojban word "zo'e" is left out. This is done for simplicity for all words taken from Lojban. Apart from that, if the Lojban predicate is formed by several words, these words are linked with an underscore when used inside the framework (e.g., "na_goi").

The predicates in Table 4.1 all represent declarative sentences. Apart from those sentences, the MMF framework supports phrases, questions, commands and negations. The different types of supported sentences are presented in more detail in the following.

Phrases

Phrases like "hello", "goodbye", "thanks", "please", "yes" and "no" are supported by the MMF framework. They are represented as predicates without arguments. Example 14 presents a phrase input, where "coi" is Lojban for "hello".

Example 14. Phrase Input
coi()

Negation

In Lojban, the negation operator "na'e" is used to negate single words, whereas "na" is used to negate a whole sentence. Example 15 presents two negated sentences.

Example 15. Negation in Lojban

-
mi	na	klama	le zarci
I	[false]	go	to the market.

-
mi	na'e cadzu	klama	le zarci.
I	other-than-by foot	go	to the market.

In the first sentence, "na" is used to negate the whole statement: "I do not go to the market". In the second sentence, "na'e" is used to negate "cadzu" (= "to walk"): "I go to the market but I am not going by foot."

Currently, only negations of whole sentences are supported by the MMF framework. Negations of individual words can be added easily in a future version. Example 16 shows a negated input to the MMF framework.

Example 16. Negated Predicate Input
[*na*] vecnu(robot1, vase3, [zoe], [zoe])

The sentence "Robot1 does not sell vase3" can be generated from the input presented in this example. Since the whole predicate should be negated, "na" is put in front of it.

Questions

There are two basic types of questions in Lojban: truth questions and "fill-in-the-blank" questions. Truth questions ask whether something is true or false, i.e. the answer is either "yes" or "no". Therefore, they can also be called yes-no questions. Example 17 presents a truth question in Lojban.

Example 17. Truth Question with "xu"

xu	do	klama	le zarci
[True or false?]	You	go	to the market.

When transformed into proper English, this question results in: "Are you going to the market?". The word "xu" can either be used to ask whether the entire statement is true by being positioned at the beginning of the statement, or it can be used to ask if a specific part of the sentence is true by following this part. For the MMF framework, only truth questions which consider the entire sentence are used. Example 18 shows an input representing a truth question.

Example 18. Truth Question Predicate Input
[xu] vecnu(robot1, vase3, [zoe], [zoe])"

The question "Does robot1 sell vase3?" can be generated from this input.

The other type of questions, which are "fill-in-the-blank" questions, is used in Lojban if some word or phrase is not known to the questioner and needs to be supplied by the responder. For example, if a sumti is not known, a question is formed with "ma".

Example 19. Sumti Questions with “ma”

ma	klama	le zarci
[What sumti?]	goes	to the market

do	klama	ma
You	go	to [what sumti?]

ma	klama	ma
[What sumti?]	goes	to [what sumti?]

The first question in Example 19 asks for the first sumti, the subject of the sentence: “Who is going to the market?”. In the second question, the second sumti, which is the location, is queried: “Where do you go?”. As demonstrated in the last question, “ma” can appear several times inside the predicate to query several sumti at once. The corresponding sentence is: “Who goes where?”. Sumti questions are the most common “fill-in-the-blank” questions. But there are also some further questions of this type. For example, “xo” is used to ask for numbers (“How many?”). However, only sumti questions are supported in the MMF framework:

Example 20. Sumti Questions Predicate Input

- vecnu([ma], vase3, [zoe], [zoe])
- vecnu(robot1, [ma], [zoe], [zoe])

Example 20 shows that “ma” is placed at the position of the queried argument in the predicate input, like it is done in Lojban. The first input in the example can be transformed into the question “Who sells vase3?” and the second input into “What does robot1 sell?”.

Commands

The last type of supported sentences consists of commands. Since orders are directed towards a listener, the subject is not necessary and is left out in English. Leaving out the subject, located almost always at position x_1 of the predicate, is not enough to indicate an order in Lojban: It would be interpreted as “zo’e”, which means that it is not relevant for the sentence. Therefore, the word “ko” is used to signalize the command. Example 21 shows a command in Lojban.

Example 21. Command with “ko”

ko	klama	le zarci
[order]	Go	to the market!

The word “ko” always precedes the selbri (“klama” in this example). The input to the MMF framework describing a command looks as follows:

Example 22. Command Predicate Input
[ko] vecnu([zoe], vase3, user1, [zoe])

As can be seen in Example 22, “ko” is positioned in front of the whole predicate and the x_1 argument is replaced with “zoe”. The resulting command, generated from this example input, is: “Sell vase3 to user1!”.

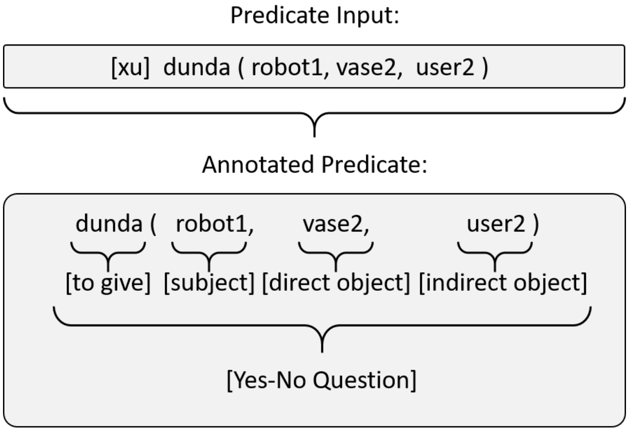
Predicate Processing

For each supported predicate in the MMF framework, a template stating its predicate structure is available. The input predicate is parsed and the information of its corresponding template is used to retrieve the predicate's meaning. Example 23 shows the English template for the predicate "dunda".

Example 23. English Template for "dunda"
dunda(x_1, x_2, x_3) :=
(ENG): [*subject*: x_1] [give] [*direct object*: x_2] [*indirect object*: to x_3]

The selbri "dunda" is translated into "to give". For each predicate argument, the respective sentence unit (subject, direct/indirect object) as well as the structure of the English sentence is given. Prepositions like "to" in the example, which usually precede certain arguments, are also inserted. The example only depicts the template for English, but templates for other languages can easily be added. Apart from that, special annotations required for certain natural language generation tools can be added as well. This will be further described in section 5.2.1. Example 24 illustrates the annotation of the input predicate.

Example 24. Predicate Annotation



A predicate input as well as the annotated predicate are depicted in this example. From the template in Example 23 the information about the basic sentence units are retrieved: The selbri "dunda" receives the translated verb in English as annotation, "robot1" is annotated with "subject" and so on. Furthermore, the meaning of [xu], [ma], [ko] and [zoe], which can be used in the predicate input, need to be resolved in the parsing process. In this example, the predicate input contains [xu], which indicates that the sentence need to be formulated as a yes-no question. Therefore, the whole predicate receives the annotation "yes-no question". In the MMF framework, it is the task of the speech modality to generate a sentence out of the annotated predicate. This will be described in section 4.5.1.

4.1.2. World Model

The *world model* represents all available information about the interaction context. Figure 4.2 provides an overview of its components.

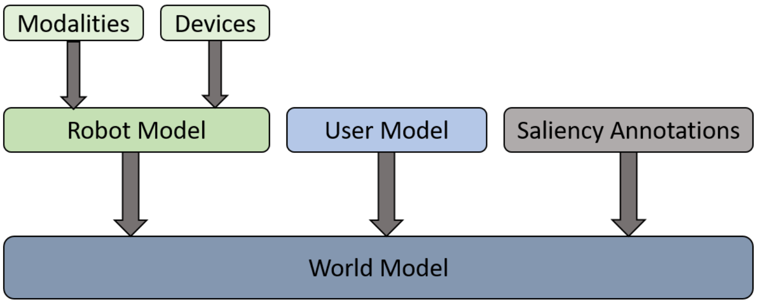


Figure 4.2.: Overview of the World Model Components

The user of the framework has to decide which modalities and which of the available output devices should be offered. The list of available devices limits the list of possible modalities. Since the framework focuses on human-robot interaction, this information becomes part of the *robot model*. Furthermore, the robot model includes some properties of the robot, like its current position. For now, the robot model depicts only one robot. Yet, a model including several robots is conceivable as well. Alongside the robot model, the world model contains a *user model*. The user model provides information, such as name, age or disabilities, about one or several users. This information can be used to adapt the output representation to a specific user. For example, modalities providing visual information cannot be selected for a blind user. Moreover, the world model encompasses information about the concrete objects in the environment, that are relevant for the interaction. If the locations of the objects are provided, information about the objects' proximity to each other is calculated and stored inside the model (see section 4.4.2 for further details about the proximity calculation).

All objects, users and robots defined in the world model require a unique id, called *worldobjectid*, which is internally used to identify the object, and a *worldobjecttype*, which assigns each object to a category. Additionally, position coordinates are required if certain modalities, like a pointing modality, are used. Furthermore, the properties of the objects are represented as key-value pairs, which can be nested if desired. Example 25 shows the stored information for a small, blue vase.

Example 25. Stored Object Information
$\langle worldobjectid : Vase1\rangle$,
$\langle worldobjecttype : Vase\rangle$,
$\langle position : \langle xPos : 3.0\rangle, \langle yPos : 5.0\rangle, \langle zPos : 2.0\rangle\rangle$,
$\langle color : blue\rangle$,
$\langle size : small\rangle$.

The positions of the objects and robots are given as a nested key-value pair stating the x-, y- and z-coordinates in a coordinate system in which, by default, the robot is at the center.

The last part of the world model consists of the so-called *saliency annotations* which need to be provided by the user of the framework. They state how salient the individual objects' properties are such that the object can be identified based on a description that uses these properties. A *saliency value* between 0.0 and 1.0 is assigned to each property of the world model, where 1.0 means very salient and 0.0 not salient at all. Example 26 shows three properties and their corresponding saliency value.

Example 26. Saliency Annotations
$\langle color : 1.0\rangle$
$\langle material : 0.8\rangle$
$\langle pos : 0.0\rangle$

Color is a very salient property, therefore it receives a saliency value of 1.0. The material of an object can usually be captured rather easily (saliency value: 0.8), whereas the object's absolute position in the internal coordinate system of the world is not a very helpful information for the user (saliency value = 0.0). In section 4.4.1, the usage of these annotations will be explained.

The world model is created in an initialization step before the main procedure of the framework is started. A world model needs to be available during the runtime of the framework. In order to have a consistent state of the internally used model, it is not allowed to change it during one execution cycle of the framework, beginning with receiving a new predicate input and ending with the generated multimodal output for this predicate. Yet, the world model can be updated before processing a new input predicate.

Furthermore, it can be extended by new models. For example, an *environment model*, in which information about the lighting condition or the current noise level is stored, can be added. Such information can be considered when selecting suitable modalities and devices for the output. For example, pointing gestures and certain objects' properties could not be visible in darkness or a noisy environment requires using a device with increased volume.

4.2. Output Representation

The output which is created by the MMF framework consists of several nested components. The basic components are triples, each of them referring to an output element. Each triple consists of the selected modality, the selected device and the output information provided in a specific output format, which can be understood by the selected device. Example 27 depicts three output triples.

Example 27. Output Triples
⟨*Speech Modality, Speaker1, "Hello"*⟩
⟨*Pointing Modality, Left Arm, Pos*(1.0, 1.0, 1.0)⟩
⟨*Image Modality, Screen2, "http://example.com/Vase.jpg"*⟩

In each triple, the first position is filled with the selected modality type (Speech, Pointing, Image), the second position with the name of the selected device (Speaker1, Left Arm, Screen2) and the third position with the respective output. In the first triple, a string representation is used as output format, from which a text-to-speech engine can create speech that will be outputted by the selected speaker (Speaker1). In the second example, the x-, y- and z-coordinate of the position at which the pointing device (Left Arm) is supposed to point are given. The output element of the last triple consists of a URL from which a resource can be loaded and displayed on the selected screen (Screen2).

Furthermore, a higher-level component can consist of several triples that refer to the same part of the output and, therefore, are executed concurrently. Example 28 shows such triples.

Example 28. Triples for Same Output Element
⟨*Speech Modality, Speaker1, "the green cup"*⟩
⟨*Pointing Modality, Left Arm, Pos*(2.0, 3.0, 2.0)⟩

What the triples presented in Example 28 have in common is that the generated output for both contains a reference to the same object (a green cup at position $x = 2.0$, $y = 3.0$, $z = 2.0$ in the internal coordinate system). Thus, they form a multimodal reference.

The final output of the system consists of a sequence of such composed triples. Figure 4.3 illustrates the structure of a multimodal output. Speech, pointing and gaze modalities have been selected in the presented example. The different colors indicate the selected devices (a speech device, a pointing device and a gaze device). The proceeding time steps are represented by the indices t_0 to t_4. The entry in each box represents the respective output information. As can be seen, the track for the speech modality has the output elements represented in string format, whereas for the pointing and the gaze modality the coordinates of the position of an output element are depicted. Output elements in the same time slot, formed by two time steps, will be executed concurrently. Most often, they form a multimodal reference to the same entity. For example, in the first time slot (between t_0 and t_1), the speech device will output the word "you" and the gaze device will turn its gaze towards the stated position. On this position, a user, who is linguistically referenced with "you", is presumably located. In

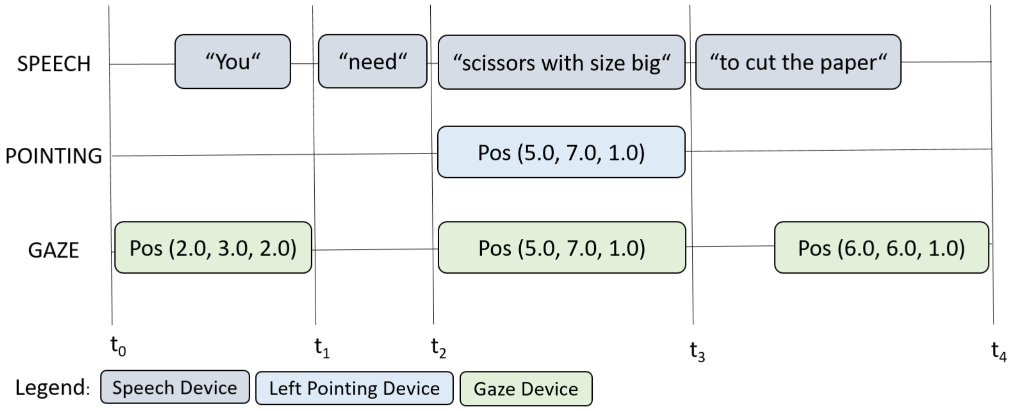


Figure 4.3.: Example for a Multimodal Output in the MMF Framework

the next time slot, only the speech device performs an output. After that, big scissors are simultaneously referenced linguistically as well as by a pointing gesture and by gaze. In the last time slot, the speech device outputs "to cut the paper", while "the paper" is additionally referenced via gaze.

As can also be seen in Figure 4.3, actions in the same time slot do not necessarily happen completely in parallel. In the last time step, the gaze device waits with the output until the speech device refers to the paper, which should be referenced in both modalities. Furthermore, the gaze is directed towards the addressed person before the actual speech output starts in the first time step. Referring to entities via gaze and speech, while gaze precedes the speech by arround 800-1000ms, makes the robot appear more "competent" and improves task completion in human-robot interaction [35]. Thus, defining the concrete starting times for each output device can be reasonable in some cases. Yet, the MMF framework defines which actions should be executed in parallel, but no concrete starting times are determined for now. Further information on how the starting times can be included in the framework is mentioned in chapter 8.

The output of the MMF framework is a plan which can be seen as a sequence of parallel actions. It is up to the user of the framework to make use of the information stated in the plan by whatever means. Optionally, an execution component is provided by the MMF framework, which can be connected to the available devices. It coordinates the output as prescribed in the output plan and forwards the specific outputs to the corresponding devices.

4.3. Output History

When deciding what to do next in a given dialog state, it is most often not sufficient for a dialog manager to consider only the current situation. In order to react properly in a given situation, knowledge about the *dialog history*, which is also called *interaction history*, needs to be taken into account. Keeping track of previously generated output can likewise be beneficial for an output module: It might improve the quality of the generated output. Therefore, the MMF framework stores some information about previously created outputs. The stored information is called the *output history*, since, in contrast to a dialog manager, an output system cannot keep track of the entire interaction between a user and the system but is only capable of storing the output it created. The MMF framework stores for each previously made reference to a known entity in the world (an object or a person) which modalities and devices have been selected to refer to the entity. The idea is that previously made decisions are considered in the current decision making process, in which the modalities and devices which should be used for creating references are selected. Example 29 shows the latest stored information for a vase in the output history.

Example 29. Latest Stored Information for *vase2*
⟨*Speech Modality, Speaker1,"the blue, small vase next to the table"*⟩
⟨*Pointing Modality, Right Arm, Pos*(1.0, −1.0, 0.5)⟩

The object with the *worldobjectid* "vase2" has been referenced in the previous sentence linguistically ("the blue, small vase next to the table") and by a pointing gesture towards it. If the object should now be referenced again, it is probably not necessary to perform another pointing gesture since the focus is already on the object. Furthermore, the linguistic description might be shortened. Depending on the situation, it might be sufficient to refer to "vase2" as "the vase". Further information on how the output history is used in the modality selection process will be provided in section 4.6.2.

4.4. Reference Generation

The MMF framework internally uses unique ids, the *worldobjectids*, to refer to objects defined in the world model. Depending on the modality type, references to entities are generated in various ways in the MMF framework. In any case, the *worldobjectid* is replaced by the output information prescribing the reference. For example, it is replaced by a linguistic object description when using a speech modality. Furthermore, coordinates of the object's position, given in the internal coordinate system, are used for creating a pointing gesture or gaze towards the referenced object. There might be other ways in which pointing references can be created, like stating the degree and direction in which the pointing device has to move. Another possibility might be to use object detection methods. However, both suggestions strongly depend on the available devices, whereas fixed coordinates can be used independently of them and are easier to handle. Nevertheless, using position coordinates also has its limitations. In a dynamic setting, the coordinates need to be updated. Furthermore, pointing gestures may be inaccurate when pointing to small objects which are farther away or if a lot of objects are in between the robot that performs the pointing gesture and the referenced object. Section 4.4.2 will present an algorithm that helps in deciding whether a pointing gesture is suitable in a given context.

The MMF framework generates multimodal references, as already seen in section 4.2 in Example 28. Creating cross-modal references is currently not actively supported, but can be included for suitable modalities. For example, a pointing gesture could be performed towards an image or part of an image displayed on a screen. Displaying images on a screen is already supported. To be able to perform a cross-modal pointing gesture, the image location on the screen needs to be known and recalculated in the internal coordinate system used to reference all physical present objects.

Section 2.4 has already given an overview of the referring expressions relevant for this thesis. In the following, short examples on anaphoric and deictic references, which can be generated by the MMF framework, are given.

Example 30. Anaphoric References

- The hammer is on the table. Use *it* for the task.
- Do you want to buy the big, blue box or the red bucket? *The box* costs 10€.

In Example 30, "it" refers back to the hammer mentioned in the first sentence. "It" can be used since it is obvious that the hammer is referenced, because no other object has been mentioned in the first sentence. Two objects, a big, blue box and a red bucket, are described in the second example. A backwards reference to the big, blue box is made by simply stating "the box". This is possible since the focus is on two objects of a different type.

Example 31. Deictic References

- Use *this [pointing at a hammer].*
- Use *this hammer [pointing at a hammer].*
- *I [robot]* am talking to *you [user].*

Example 31 presents three multimodal deictic references. In the first and the second sentence, "this" and "this hammer" are used to refer linguistically to a hammer, respectively. Yet, the reference can only be resolved by perceiving the pointing gesture. The third sentence gives an example for person deixis. The robot makes a reference to itself by denoting itself with "I" and a reference to the user it is currently talking to with "you". Place deixis is also used inside the framework. However, a relation to another object (e.g., "next to object x" or "in front of object y") or any other spatial information needs to be stated explicitly in the world model. It would be an interesting extension to create such spatial references automatically based on the position of the objects (see chapter 8 for further information).

The above presented linguistic references are special cases generated in very specific situations. In most cases, the MMF framework creates a linguistic reference based on salient properties of the referred object or person. This procedure will be explained in more detail in the following.

4.4.1. Linguistic Reference Generation - Attribute Selection Algorithm

Referencing one out of many similar objects in such a way that the referenced object can be identified unmistakably is not always easy. In order to do this, the MMF framework chooses suitable salient object attributes, like color or the object's type, in case that different types of objects are present. It is common to refer to objects by using such properties. This observation has also been shown in [20]: Gargett et al. have annotated referring expressions from a human instruction giving corpus (GIVE-2 corpus). The corpus was collected by asking one person to guide another one through a 3D Environment. Their result shows that the majority of the used referring expressions contains absolute properties of the referenced object, like color or shape (85.73% in German and 92.53% in English). Furthermore, the object's type was used frequently when different types of objects were present (53.66% in German and 58.51% in English).

Figure 4.4 shows a shelf with four different vases. These vases have different attributes. The most salient ones include color (blue, yellow-green), size (big, small), motif (stripes, dots, plain) and their position on the shelf (leftmost, center left, cnter right, rightmost). If one of these vases is to be referenced uniquely, it is natural to use such attributes (e.g., "the yellow-green vase") or a combination of several attributes (e.g., "the blue, dotted vase").

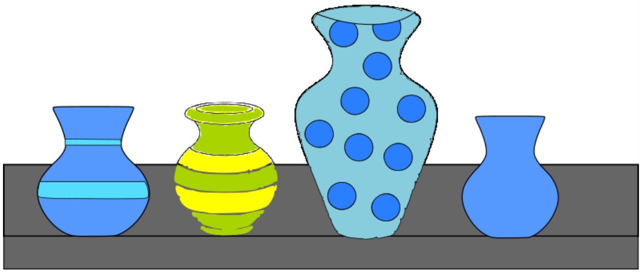
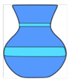
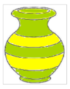
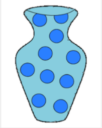
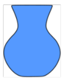


Figure 4.4.: Different Types of Vases

The **A**ttribute **S**election **A**lgorithm (ASA) has been developed in this thesis. Its aim is to automatically extract the most suitable, salient object's attributes which differentiate the object from all other objects of the same type and to uniquely identify the object. Example 32 shows for each vase the possible stored information in the framework.

Example 32. Stored Vase Attributes and the Attributes' Saliency Annotation

0.0	⟨worldobjectid: vase1⟩	⟨worldobjectid: vase2⟩	⟨worldobjectid: vase3⟩	⟨worldobjectid: vase4⟩
1.0	⟨worldobjecttype: vase⟩	⟨worldobjecttype: vase⟩	⟨worldobjecttype: vase⟩	⟨worldobjecttype: vase⟩
1.0	⟨color: blue⟩	⟨color: yellow, green⟩	⟨color: blue⟩	⟨color: blue⟩
0.9	⟨size: small⟩	⟨size: small⟩	⟨size: big⟩	⟨size: small⟩
0.9	⟨motif: stripes⟩	⟨motif: stripes⟩	⟨motif: dots⟩	⟨motif: plain⟩
0.7	⟨position: leftmost⟩	⟨position: center left⟩	⟨position: center right⟩	⟨position: rightmost⟩
0.0	⟨price: 15€⟩		⟨price: 34€⟩	

The numbers in front of the attributes of vase1 denote the chosen saliency values for each of the used attributes. Each attribute in the world model receives such an annotation provided by the user of the framework in advance. In general, the annotations should be generic (e.g., color can generally be seen as a very salient property) and not be tailored to reference specific objects. The reason for this is that the annotations should only provide the algorithm with basic knowledge about which attributes are useful to describe an object in general. However, the annotations should not be seen as a handcrafted way to determine which concrete attributes are to use to reference a specific object. Nevertheless, it can make sense to adapt certain saliency values depending on the current user. For example, if the user is colorblind, the color's saliency value should be lowered. Thus, it is beneficial to have different saliency assessments for different users. Furthermore, it makes sense, under certain circumstances, to adapt the annotations slightly to the specific domain or scenario (e.g., a low saliency value for color makes sense if only black and white pictures are presented in the scenario or a motif might not always be as salient as in the presented vase example).

Since the *worldobjectid* is only used internally, it receives a saliency value of 0.0, denoting that it is not salient at all. The type of an object and its color are seen as highly salient. Thus, they receive a saliency value of 1.0. Furthermore, the object's size and the printed motif are also salient, both receive the value 0.9 in this example. In comparison to that, the position of the object denoted with a statement containing “left” or “right” might be confusing if the vases can be observed from different perspectives. Thus, the position receives a saliency value of 0.7. Since no price tag is visible, the price is not a salient attribute and receives a saliency value of 0.0. Furthermore, an attribute may have several values (e.g., the color attribute for vase2) or not each attribute is given for each object (e.g., no price information is available for vase2 and vase4).

The algorithm considers only objects of the same type, since objects of different type can be easily differentiated by their type. It consists of two main steps:

1. Finding all possible sets of salient attributes that uniquely identify the referent
2. Selecting the most suitable one from these sets according to some criteria

Possible criteria may include the number of attributes in the set and the overall saliency value. If an object can be described sufficiently by few attributes, using additional ones is redundant. For example, referring to vase1 as the “the small, blue vase with stripes, located leftmost and next to a yellow-green vase” is probably a needlessly detailed description. However, using several attributes, possibly redundant, increases the saliency and therefore a person might distinguish an object faster and more easily. How the two steps are realized and which criteria have been implemented in the MMF framework will be presented in detail in section 5.4.

In some cases, no unique identifier can be found. For example, assume there are five vases that are only distinguishable by their color (two red and three blue ones). One of the red vases should be referenced. Thus, only a partial identifier, consisting of the color red, can be found. At least, this identifier differentiates the referent from the three blue vases. The algorithm indicates that only a partial identifier has been found. This information can be used later, for example, to combine the linguistic reference with a pointing gesture such that the object is still uniquely referable. If only a partial identifier has been found, the second step of the algorithm is omitted and the set with all distinguishing attributes is used to reference the object as good as possible. An algorithm, similar to the ASA, is described in [45]. Their algorithm does not differentiate the attributes based on their saliency, but focuses on providing the best discriminative properties, which is especially useful if only partial identifiers are available. A property is more discriminative than another one if more objects have this property and if they have more distinct values for this property.

Generating referring expressions is a research field of its own and there exist various approaches (e.g., logic-based [1], search-based [29], graph-based [31]). Most approaches try to generate human-like referring expressions. According to [30], humans often overspecify by including information in a referring expression that is not necessary to make it distinguishing. Even though referring expressions created by humans may not be optimal, it is also the aim of the ASA to select salient attributes which may be used by humans as well. The reason for this is that the focus of this thesis is on human-robot collaboration: The robot should reference objects in a way its human counterpart is used to from the communication between humans.

The Attribute Selection Algorithm applies a simple search-based approach for solving this task. The algorithm is highly configurable and adaptable. All possible combinations which can be used to reference the desired object are retrieved in the first step. The final selection of attributes can be made in the second step by using different criteria. Several different criteria are currently offered by the MMF framework for this task, which will be presented in section 5.4. Further ones can be included in an easy manner. Furthermore, different thresholds can be set from outside. Apart from that, the algorithm is defined in its own module: It can be exchanged as a whole with minimal effort and can also be used independently from the framework. A future version may use machine-learning techniques to automatically determine the saliency of the attributes and to free the user of the framework from this task. Furthermore, this has the advantage to overcome different saliency assessments made by different users.

4.4.2. Assessing Pointing References - Cone Intersection Algorithm

Pointing references are often very helpful to detect the referenced object more quickly. Furthermore, when a pointing gesture accompanies a linguistic reference, a complicated linguistic description of the referent can potentially be shortened. When a pointing gesture or rather a deictic reference containing a pointing gesture is performed, it is important that the addressee can resolve the reference. This is only possible if the addressee can visually detect the referent (with no obstacles limiting the sight) and if the pointing gesture is precise enough. The precision of a pointing gesture depends on the size of the referent as well as on the distance from which the pointing is performed. Furthermore, other objects close to the referent can make it more difficult to recognize which object has been referenced. Therefore, an algorithm will be presented in the following which detects the objects that are between the robot that performs the pointing gesture and the referent. The gathered information can later be used in the modality planning process to determine if a pointing gesture is helpful and precise enough or whether it is not possible to recognize the referenced object because of similar objects which are close.

All objects in the world model are classified into three different categories, depending on their size: small, medium, large. This categorization may vary depending on the domain and the context. In general, examples for small objects are a pen or a bottle, whereas a chair or a bookshelf are seen as medium-sized. Objects like a house are considered as large. The algorithm constructs a circle around the object's center. The radius of the circle depends on the category an object belongs to. Based on experimental findings, the radius is calculated in the following way:

Large Objects:	$r = 0.075 + \mathit{distance} \cdot 0.15$
Medium Sized Objects:	$r = 0.05 + \mathit{distance} \cdot 0.1$
Small Objects:	$r = 0.025 + \mathit{distance} \cdot 0.05$

The variable *distance*, given in centimeters, refers to the distance between the robot's position and the object's position. If no size information is provided, the object is assumed to be small. Furthermore, it is possible to explicitly state a radius for a certain object in the world model, which will then be used instead of the calculated one. The circle around an object forms the base of a cone which has its origin at the robot's position and whose height is the distance from the robot's position to the object's position. Such cones are created for each object in the world model. Figure 4.5 shows two intersecting cones for two vases. In the figure, a pointing gesture is performed towards the blue vase. Since its cone intersects the cone of the green vase, the green vase is added to the blue vase's list of objects which are between the robot and the blue vase. The idea is that intersecting cones indicate that the referenced object cannot be distinguished easily from the objects whose cones intersect its own.

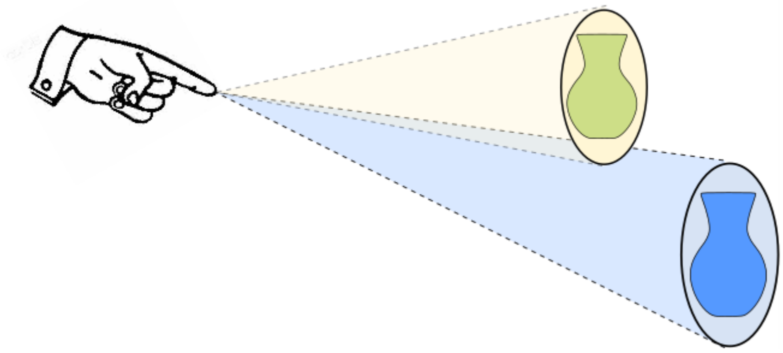


Figure 4.5.: Two Intersecting Cones

In the following, it will be explained how the intersection of two cones is determined. The idea is taken from [5]. Figure 4.6 depicts the relevant geometry. The following conditions need to be fulfilled for two cones to be intersecting:

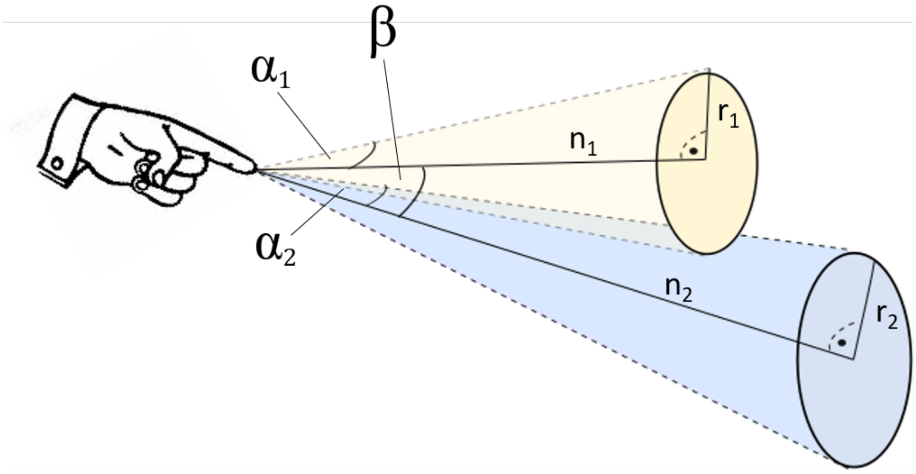


Figure 4.6.: Geometry of Intersecting Cones

$$\begin{aligned} n_1 \cdot n_2 &\geq 0 \\ \alpha_1 + \alpha_2 &\geq \beta \end{aligned} \tag{4.1}$$

This means two cones intersect if the dot product between their normal vectors n_1 and n_2 is greater than or equal to zero and if the sum of the two apex angles α_1 and

α_2, which span from the respective normal vector to the cone wall is bigger than the angle between their normal vectors β. Since the normal vectors are simply the distance between the robot's position and the object's position, its dot product can be directly calculated. Furthermore, the radius has already been determined beforehand. By using this information and basic trigonometry, the apex angles α_1 and α_2 of both rectangular triangles, which are spanned by the respective normal vector, the radius and the cone wall, can be calculated as follows:

$$\alpha_i = \arctan\left(\frac{r_i}{n_i}\right);\ i = \{1, 2\} \tag{4.2}$$

For calculating the angle β between the normal vectors, the dot product between the normal vectors is required since the following holds:

$$n_1 \cdot n_2 = ||n_1|| \cdot ||n_2|| \cdot \cos(\beta) \tag{4.3}$$

In this equation, $||n_1||$ denotes the length of the normal vector n_1. Thus, β can be calculated as follows:

$$\beta = \arccos\left(n_1 \cdot n_2\right) \tag{4.4}$$

Now all components, necessary to check whether the conditions for cone intersection from Equation 4.1 are fulfilled, are provided. If the dot product of two normal vectors is zero, then the two vectors are orthogonal. If the dot product is positive (greater than zero), then $\cos(\beta)$ will also be positive since the vector lengths are always positive values (see Equation 4.3). If $\cos(\beta)$ is positive, then β is either less than 90° or more than 270°. However, since the angle between the normal vectors is used, it can be assumed that it is not greater than 180°. Intuitively, if the angle between the normal vectors is at most 90°, the cones are inclined towards each other and it is more likely that they intersect. Furthermore, the cones are inclined towards each other if β is smaller than the sum of α_1 and α_2, which can be seen in Figure 4.6. This calculation is performed for all objects in the world model, such that finally each object has a list of objects whose cones intersect with the object's cone. In other words: If a pointing reference is performed towards an object, all objects are known which are between the robot that performs the pointing gesture and the referenced object.

4.5. Modality Categorization

In the framework, modalities are categorized into three different types: *structure-forming*, *object-referencing* and *predicate-referencing* modalities. Figure 4.7 shows the currently supported modalities together with their categories. The speech modality is special as it belongs to two categories, namely structure-forming and object-referencing. The gaze, pointing and image modalities also belong to the object-referencing modalities, whereas the waving and the nodding/headshaking modality fall into the predicate-referencing category. In the following, the categorization into these three types will be explained in more detail.

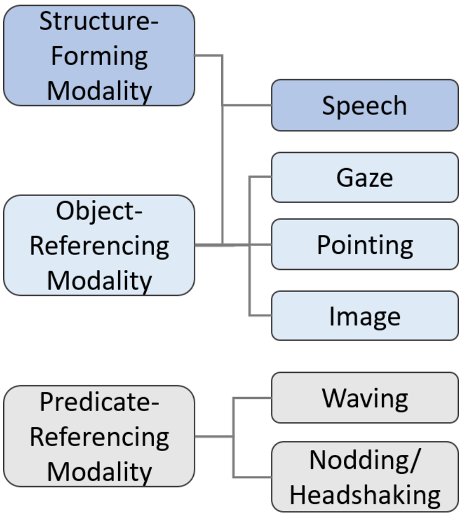
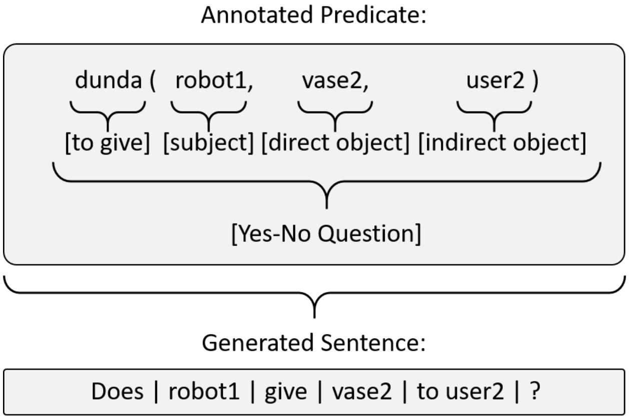

Structure-Forming Modality
Speech
Gaze
Object-Referencing Modality
Pointing
Image
Predicate-Referencing Modality
Waving
Nodding/ Headshaking

Figure 4.7.: Categorization of Modalities in the MMF Framework

4.5.1. Structure-Forming Modalities

The task of a structure-forming modality is to determine the structure of the output and, thus, the order of the output elements. It can be seen as the dominant modality since the order of the output elements is a necessary information required by all other modalities to form the output. In the MMF framework, the speech modality is the only structure-forming modality. Modalities like written language, sign language, braille or even morse code are other possible structure-forming modalities. At least one structure-forming output modality is required by the MMF framework to produce non-trivial output. In the framework, it is the task of the speech modality to create a sentence out of the annotated predicate input. Example 33 shows the annotated predicate dunda (= "to give") and the generated sentence.

Example 33. From Annotated Predicate to Sentence


Annotated Predicate:
dunda (robot1, vase2, user2)
[to give] [subject] [direct object] [indirect object]
[Yes-No Question]
Generated Sentence:
Does | robot1 | give | vase2 | to user2 | ?

The annotation step has already been described in section 4.1.1. The predicate arguments are annotated with their function in the resulting clause. Furthermore, the annotation contains the meaning of “dunda” and the type of the sentence that should be produced (a yes-no question in this case). Based on these information, the resulting question is generated by the speech modality. The language generation tool, used for this task, will be presented in section 5.2.1. The vertical lines divide the generated sentence into the different output elements. The given order of the elements can now be used by other modalities.

4.5.2. Object-Referencing Modalities

Currently, four object-referencing modalities are available in the MMF framework. They are characterized by their ability to refer to objects in the world model. As already stated in section 4.4, these modalities refer to objects in different ways, depending on their type, by replacing the internally used *worldobjectid* by the output expressed in their specific output formats. The speech modality generates a linguistic description using salient object properties to reference an object verbally. Both the pointing modality and the gaze modality use the object’s position to indicate a reference to this object, while the image modality provides an image URL which can be used to reference the image. If the input to the MMF framework contains a *worldobjectid*, at least one object-referencing modality needs to be selected to represent the required reference. Object-referencing modalities are very important since they can be combined to refer to objects of interest in the environment multimodally. Humans often combine gaze, pointing and speech when referring to objects. Thus, using object-referencing modalities is particularly helpful in a collaborative human-robot setting: The robot appears more human-like and the people can act like they would do in an interaction with another person.

4.5.3. Predicate-Referencing Modalities

Predicate-referencing modalities describe modalities whose output is determined by either one element of the predicate or the predicate as a whole. The two predicate-referencing modalities in Figure 4.7, namely the waving and the nodding/headshaking modality, refer to whole predicates. For example, the waving modality is activated by predicates expressing greetings or farewells. The nodding/headshaking modality is activated if approval respectively disapproval is expressed with the predicate. However, it could also be possible to not only reference a whole predicate but one or several arguments. For example, in order to emphasize that an object is very big, a respective gesture can be performed. Predicate-referencing modalities are used to intensify and accentuate the effect of a statement. Waving can amplify a welcome greeting, while nodding may express agreement more strongly and head-shaking disagreement, respectively.

Example 34 shows the input predicate “na_goi()”(= “no”). It is a predicate without arguments, a so-called phrase.

Example 34. Predicate-Referencing Example
Input Predicate: na_goi()
Speech Modality: “No”
Nodding/Headshaking Modality: *HEADSHAKING*

In this example, the verbal utterance “No” is accompanied by a head-shaking gesture. It could be argued that a speech modality is also a predicate-referencing modality when no object is referenced. For predicate-referencing modalities, like the headshaking modality, a mapping from the generated output command (*HEADSHAKING*) to the final performed action (turning the head from left to right) needs to be performed. A text-to-speech engine is also required to map the string representation of the speech modality to synthesized speech which can be outputted by a speaker. Yet, the speech modality is not considered to be in the predicate-referencing category, since a speech modality is a dominant modality which does not focus on supporting other multimodal output like predicate-referencing modalities do.

Furthermore, a predicate-referencing modality is not an object-referencing modality, since no object is referenced - at most a statement about an object or a property of the object might be emphasized. It can be argued that an object-referencing modality could be seen as predicate-referencing since the referenced object is also part of the predicate’s arguments. However, the output created by a predicate-referencing modality can be seen as optional and is not necessary to gain a complete output. On the other hand, at least one object-referencing modality is required if a *worldobjectid* appears in the predicate input such that the object can be referenced and the output is complete. Therefore, both modality categories differ.

The presented categories have been chosen to categorize the modalities currently supported by the MMF framework. When adding new modalities, further categorization can be considered in the future.

4.6. Modality and Device Selection

A key task of a fission module is the selection of suitable modalities and devices to represent each output element. In this thesis, the focus is on modality selection. Each modality has a list of devices which can output the type of information represented by the modality. First, the most suitable device is selected for each potentially used modality. Afterwards, the modalities which should be used for each output element are selected. Thus, modality and device selection are tackled as two separate planning problems. A constrained-based approach is used to solve both problems. It contains two types of constraints: hard and soft constraints. Thus, the problems are stated as Constraint Optimization Problems (COPs) (see section 2.6 for further details). An optimal solution for the modality and device selection in the MMF framework is one which does not violate any hard constraints and which maximizes the value of an objective function for the soft constraints. Figure 4.8 shows the general cycle that is performed during the planning process. Starting with a random assignment a, the used optimization algorithm creates a new assignment a in each cycle. This assignment is then evaluated by the objective function. Examples for termination conditions are a time limit or the fact that a solution has not improved over several cycles. In the MMF framework, a local search optimization algorithm is applied.

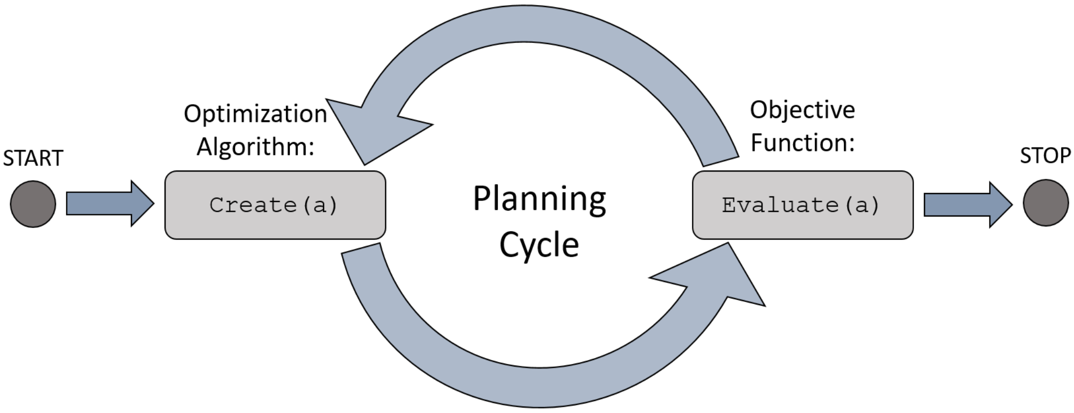


Figure 4.8.: General Planning Cycle

The following sections describe how the modality and device selection problems are formulated and solved.

4.6.1. Device Selection

Since the MMF framework focuses on modality selection, device selection is currently kept rather simple. The device selection precedes the modality selection process. There are two reasons for this order: If the modalities would be selected first, it would be problematic if the device selection concludes that a selected modality does not have a suitable device which can generate the output. Furthermore, using this order, technical information about the selected devices can be considered within the modality selection process. For example, one such information could be the approximated time a device needs to output a certain element. Definition 13 shows how the device selection planning problem is formulated as a constraint optimization problem.

Definition 13. Extended COP Formulation for Device Selection

- Set of Output Elements: O
- Set of available Modalities: M
- Set of available Devices: D
- Planning Variables: $X = \{dev_1, dev_2, ..., dev_n\}$
- Variable Domains: $VD = \{Dev_1, Dev_2, ..., Dev_n\}$, with $Dev_i = \{d \in D. (d \Uparrow m) \wedge t_i = \langle m, o \rangle\}$
- Planning Entities: $E = \{t_1, t_2, ..., t_n\}$, with $t_i = \langle m, o \rangle \ \wedge \ m \in M \ \wedge \ o \in O \wedge (m \Uparrow o)$
- Hard Constraints: $C_h = \{C_1, ..., C_k\}$
- Soft Constraints: $C_s = \{f_1, ..., f_l\}$

There is a set of output elements O, a set containing the available modalities M and a set with the available devices D. For the planning variable dev_i a concrete device from the set D should be selected. The definitions presented in section 2.6, have been extended by *planning entities*. The planning entities in the device selection are tuples t_i consisting of a modality m and an output element o. A modality and an output element can only form a tuple, if the respective modality can represent this output element. This is denoted by $(m \Uparrow o)$. For each planning entity, at least one planning variable need to be selected. In the device selection, exactly one variable dev_i is assigned to each entity t_i. The domain Dev_i determines that only these devices $d \in D$ can be chosen for the variable dev_i which can present the information expressed by the modality m in the respective tuple t_i, denoted by $(d \Uparrow m)$. The tuples used in the device selection could also be defined as the planning variables. However, using planning entities seems to be more intuitive, since it can be expressed that one device belongs to each tuple. Furthermore, there is a set of hard and soft constraints C_h and C_s. As already stated in section 2.6, the soft constraints are denoted as functions f_i applied to a subset of the variables. The sum over all function values should be maximized (or minimized). Example 35 shows a concrete problem formulation for the device selection.

Example 35. Problem Formulation for Device Selection

- Set of Output Elements $O = \{$“user”, “sell”, “vase3”$\}$
- Set of available Modalities $M = \{speech, pointing, gaze\}$
- Set of available Devices $D = \{speaker_1, speaker_2, arm_1, arm_2, arm_3\}$
- Planning Variables: $X = \{dev_1,\ dev_2,\ dev_3,\ dev_4,\ dev_5\}$
- Variable Domains: $VD = \{Dev_1, Dev_2, Dev_3, Dev_4, Dev_5\}$, with

  - $Dev_1 = Dev_3 = Dev_4 = \{speaker_1, speaker_2\}$
  - $Dev_2 = Dev_5 = \{arm_1, arm_2, arm_3\}$

- Planning Entities: $E = \{$
  t_1: ⟨speech,“user1”⟩,
  t_2: ⟨pointing,“user1”⟩,
  t_3: ⟨speech,“sell”⟩,
  t_4: ⟨speech,“vase3”⟩,
  t_5: ⟨pointing, “vase3”⟩$\}$

- $C_h = \emptyset$

- $C_s = \{f_1, f_2\}$, with

$$f_1 = \begin{cases} 0 & \text{if } \exists dev_i = speaker_1 \in a; \text{ with } i \in \{1, ..., 5\} \\ -1 & \text{otherwise} \end{cases}$$

$$f_2 = \begin{cases} 0 & \text{if } \nexists dev_i = arm_3 \in a; \text{ with } i \in \{1, ..., 5\} \\ -1 & \text{otherwise} \end{cases}$$

  where a is the current assignment.

The output elements are: “user1”, “sell”, “vase3”. Two modalities are available, namely a speech modality and a pointing modality. Furthermore, two robots are available which have the following devices: The first has one speaker and two arms and the second has one speaker and one arm. For tuples containing the speech modality, a valid planning variable assignment consists of one of the speakers (Dev_1, Dev_3, Dev_4), whereas tuples with the pointing modality require one of the robot’s arms as device (Dev_2, Dev_5). Furthermore, the tuple ⟨pointing, “sell”⟩ does not exist, since a pointing modality cannot represent this kind of information. The example does not provide a hard constraint, but two soft constraints. One states that $speaker_1$ should be selected at least once and the other one expresses that it is better to avoid using arm_3. The soft constraints are represented as functions, where f_i: $a[s_i] \mapsto \mathbb{R}$. This means that each function f_i is applied to the variables in its scope s_i in an assignment a. In the example above, these scopes are: $s_1 = \{dev_1, dev_3, dev_4\}$ and $s_2 = \{dev_2, dev_5\}$. The result of the function is a real value assessing the quality of the current assignment from the perspective of the respective soft constraint. As already stated above, the objective function $f = \sum_{i=0}^{l} f_i$ should be maximized for having a good result. In the example above, f_1 and f_2 map to 0 if the respective soft constraint is fulfilled and to -1 if it is violated. Example 36 depicts three example assignments and the respective results of the objective function.

Example 36. Evaluation of Assignments

- Assignments:
  - $a_1 = \{dev_1 = speaker_1, dev_3 = speaker_2, dev_4 = speaker_2,$
    $dev_2 = arm_1, dev_5 = arm_2\}$
  - $a_2 = \{dev_1 = speaker_1, dev_3 = speaker_1, dev_4 = speaker_1,$
    $dev_2 = arm_3, dev_5 = arm_2\}$

  - $a_3 = \{dev_1 = speaker_2, dev_3 = speaker_2, dev_4 = speaker_2, dev_2 = arm_1, dev_5 = arm_3\}$

- Evaluations:
  - $f(a_1) = f_1(a_1) + f_2(a_1) = 0 + 0 = 0$
  - $f(a_2) = f_1(a_2) + f_2(a_2) = 0 + (-1) = -1$
  - $f(a_3) = f_1(a_3) + f_2(a_3) = (-1) + (-1) = -2$

Assignment a_1 receives the highest overall evaluation, since both soft constraints are satisfied, whereas constraint 2 is violated in a_2 since arm_3 is assigned to dev_2. The assignment a_3 has the lowest evaluation since both constraints are violated. Thus, assignment a_1 would be optimal regarding the soft constraints. Depending on the modality type, different criteria for the device selection, formulated as hard and soft constraints, might be reasonable. The concrete criteria implemented in the MMF framework will be described in section 5.6.1.

4.6.2. Modality Selection

The aim of the modality planning process is to select for each output element the most suitable combination of modalities. Furthermore, if an output element contains a *worldobjectid* and if a speech modality is part of the selected combination, the linguistic object description which should be used to refer to the respective object, is selected as well. In section 4.4.1, the Attribute Selection Algorithm (ASA) for creating a linguistic object description based on salient attributes has been presented. Using the output of the algorithm is the default behavior. Yet, in some cases shorter references can be made. For example, if a combination of speech and pointing modality is chosen, it might be sufficient if the speech modality expresses “this” and the object’s type, alongside the generated pointing gesture. Furthermore, as already mentioned in section 4.3 regarding the usage of the output history, if the focus is already on an object, because it has been referenced in the previous sentence, then it can be sufficient to refer to it again by saying “it” or name the object’s type. Such linguistic variations are considered in the planning process as well. Definition 14 presents the modality selection formulated as a constraint optimization problem.

Definition 14. Extended COP Formulation for Modality Selection

- Set of Output Elements: O
- Set of available Modalities: M
- Set of available Linguistic Descriptors: LD
- Planning Variables: $X = \{mc_1, mc_2, ..., mc_n, ld_1, ld_2, ..., ld_n\}$
- Variable Domains: $VD = \{PM_1, PM_2, ..., PM_n, LD_1, LD_2, ..., LD_n\}$, with
  - $PM_i \subseteq PM = \{\emptyset, \{m_1\}, \{m_2\}, ..., \{m_p\}, \{m_1, m_2\}, \{m_1, m_3\}, ..., \{m_1, m_p\}, ..., \{m_1, m_2, ..., m_p\}\} \;\wedge\; \nexists pm \in PM_i : m \in pm \;\wedge\; \neg(m \Uparrow o_i)$

  - $LD_i \subseteq \begin{cases} LD & \text{if } \textit{worldobjectid} \in o_i \\ \emptyset & \text{otherwise} \end{cases}$

- Planning Entities: $E = \{o_1, o_2, ..., o_n\}$
- Hard Constraints: $C_h = \{C_1, ..., C_k\}$
- Soft Constraints: $C_s = \{f_1, ..., f_l\}$

As for the device selection, there are a set of output elements O and a set of modalities M. Furthermore, a set of linguistic descriptors LD is given. The planning variables consist of the modality combinations (mc_i) and the linguistic descriptions (ld_i) that should be selected. The planning entities for the modality selection consist of the output elements (o_i). The variable domains for the mc_i variables are subsets of the powerset PM of all available modalities. Each set in the powerset represents a possible value of a planning variable. The domain of the planning variable mc_i is exactly this subset PM_i of the powerset PM that contains only sets with modalities that can represent the corresponding planning entity o_i. This is represented by the notation $(m \Uparrow o_i)$. The ld_i variables are taken from the set of available linguistic descriptions LD if their corresponding output element o_i contains a *worldobjectid*. Otherwise their domain is empty since no reference to an object in the world should be performed and thus, no linguistic description is necessary. Furthermore, the hard and soft constraints are defined as for the device selection in section 4.6.1.

Example 37 shows the problem formulation for three available modalities, namely speech, pointing and gaze for the same three output elements which have already been used in Example 35 to illustrate the device selection.

Example 37. Problem Formulation for Modality Selection

- Set of Output Elements O = {“user”, “sell”, “vase3”}
- Set of available Modalities $M = \{speech, pointing, gaze\}$
- Set of available Linguistic Descriptors LD = {“attributive_identifier”, “this”, “this_type”, “the_type”, “it”}
- Planning Variables: $X = \{mc_1, mc_2, mc_3, ld_1, ld_2, ld_3\}$
- Variable Domains: $VD = \{PM_1, PM_2, PM_3, LD_1, LD_2, LD_3\}$, with
  - $PM_1 = PM_3$ = {∅, {speech}, {pointing}, {gaze}, {speech, pointing}, {speech, gaze}, {pointing, gaze}, {speech, pointing, gaze}}
  - PM_2 = {∅, {speech}}
  - $LD_1 = LD_3$ = {“attributive_identifier”, “this”, “this_type”, “the_type”, “it”}
  - $LD_2 = \emptyset$
- Planning Entities: E = {o_1: “user1”, o_2: “sell”, o_3: “vase3”}

The planning variables are assigned to the respective output element, such that, for example, mc_1 and ld_1 both refer to o_1. Elements of the ld set encode how the object should be described. The element "attributive_identifier" denotes the result found by the ASA. As an example, the element "this_type" means that "this" in combination with the object's type should be used to reference the object. The planning variable ld_2 has an empty domain, since o_2 = "sell" does not refer to a world object. Apart from that, the domain of mc_2 consists only of the speech modality and the empty set, since for its corresponding output element o_2 any set containing the pointing and the gaze modality cannot be used since these modalities cannot represent the output element. Concrete soft and hard constraints are omitted in this example. How soft constraints are organized in the MMF framework for the modality selection will be described in more detail in the following.

Soft Constraints for the Modality Selection

The MMF framework provides a large number of different soft constraints for the modality selection process. The constraints are categorized according to what they try to attain. The following main types of criteria are realized:

- Technical Criteria: Examples for these criteria are, "Use the modalities which can represent the output in the shortest time" or "Use the modalities whose corresponding devices can execute the output with the lowest power consumption". This kind of data must be queried from the concrete devices at runtime.

- Context-Aware Criteria: The output history can be used to receive an output which is adjusted to the current conversational situation. For example, after the robot points to an object when it is mentioned first, it is not necessary to point to it again in the following sentences. Instead, it is sufficient to refer to the object via speech. Moreover, by using information retrieved from the user model, the output can be adapted to the current user. For example, impairments, language skills or preferred output modalities can be taken into account. Furthermore, information about the environment, like noise level or lighting conditions, can be considered. Apart from that, results from the Cone Intersection Algorithm (see section 4.4.2) can be used: If objects of the same type are very close to each other, a pointing reference might not suffice to uniquely identify the object.

- Human-Likeness Criteria: This category contains all criteria which have the aim to let the robot act in a more human-like way while outputting the information. For example, a general criterion would be to look at people rather than pointing to them. Besides, looking at the person to talk to or looking at things while pointing at them might also lead to a more human-like behavior. Furthermore, speech output is the foundation of the conversation while the other modalities clarify and emphasize certain things. Additionally, considering context-aware criteria, like the output history and the user model mentioned above, also improves human-like behavior.

As for the device selection, the soft constraints are represented as functions f_i evaluating an assignment by returning a real value. In addition, *categories* of soft constraints are defined for the modality selection. One category comprises a number of constraints which have a similar purpose. For example, all constraints considering the output history are part of one category. A category is represented by a superordinated function F_i that also has a scope S_i of affected variables and assesses an assignment a with a real value, like each function for an individual soft constraint f_i. The weighted sum over all category functions should be maximized for the modality selection:

$$max\ (F(a)) = max\ \left(\sum_{i=1}^{n} w_i \cdot F_i(a)\right), \quad with\ F_i : a[S_i] \mapsto \mathbb{R} \tag{4.5}$$

In Formula 4.5, n denotes the number of categories, w_i the weight assigned to each category and $F_i(a)$ is the function of category i applied to the current assignment a. By default, the weights are set to 1. If one or several categories should dominate the others, because they are considered to be more important, different weights can be set from outside. Furthermore, the soft constraints inside a category can also be weighted according to their importance. The number of different soft constraints per category can vary. However, this means that even if two categories have the same weight, they might not be treated as equally important. Therefore, the following formula may be used if it is desired that the categories are comparable and they all influence the final calculation equally:

$$max\ (F(a)) = max\left(\sum_{i=1}^{n}\left(\frac{F_i(a)}{m_i}\right)\right) \tag{4.6}$$

In Formula 4.6, the result of each $F_i(a)$ is divided by a number (m_i) describing the maximal value by which the corresponding category can decrease or increase the final sum. Note that this can only be used if all constraints of a category either increase the function value if they are fulfilled or if all decrease it if they are violated.

Example 38. Comparable Categories Example

- Category 1: 3 Soft Constraints; 2 fulfilled: $f_1 = 1$, $f_2 = 1$; 1 violated: $f_3 = 0$ $\rightarrow F_1 = f_1 + f_2 + f_3 = 2$
- Category 2: 5 Soft Constraints; 2 fulfilled: $f_1 = 1$, $f_5 = 1$; 3 violated: $f_2 = 0$, $f_3 = 0$, $f_4 = 0 \rightarrow F_1 = f_1 + f_2 + f_3 + f_4 + f_5 = 2$
- $F = \left(\frac{F_1}{max_1} + \frac{F_2}{max_2}\right) \cdot C = \left(\frac{2}{3} + \frac{2}{5}\right) \cdot 15 = \frac{2}{3} \cdot 15 + \frac{2}{5} \cdot 15 = 10 + 6 = 16$

Example 38 shows two different categories, with three and five soft constraints, respectively. For the current assignment, two out of three constraints are fulfilled for category 1, whereas for category 2 only two of five are fulfilled. If a constraint is satisfied in the example, the function value is increased by 1. If the function values of the two categories would just be added, both would contribute +2 to the final value. Therefore, the two categories are not treated as equally important and the information that more constraints are fulfilled for the first category than for the second one would be lost. By dividing the function values by their maximal value, and by multiplying with the constant $C = 15$ in this case, the information is kept: Now category 1 contributes +10 and category 2 contributes +6 to the final value of F. Note that the constant C is only used in this example to receive integer values.

Currently, only the formula of Equation 4.5 is implemented. The reason is that several implemented categories contain default constraints, whereas others describe special cases of these constraints. If a special case is encountered, the more specific constraint should outweigh the general one. This can not be achieved when applying Equation 4.6. Yet, both equations are ready to be used in the MMF framework and making categories comparable may be important for future use. The implemented categories and soft constraints as well as the used hard constraints will be presented in section 5.6.2.

The constrained-based approach allows for a lot of flexibility and ease of use at the same time. Constraints that must not be violated are formulated as hard constraints while multiple preferences can be stated as soft constraints, which can be categorized. This has the advantage that constraints prescribing certain criteria, like adapting the output to a certain user or supporting human-likeness, can be prioritized easily by assigning a weight to the respective category. The available categories can be used out of the box. New constraints can be classified into the different provided categories and can extend them. Moreover, if the existing categories are not sufficient or not suitable for certain domains, entire new categories can be designed and added according to the new requirements. Then the function values of the new categories become part of the weighted sum. Furthermore, even using a weighted sum as objective function can be replaced by another procedure: It is up to the user of the framework to decide how the results from each individual category should be combined.

5. Implementation

It has been decided to implement a framework for realizing the multimodal fission task. In contrast to a library, which is a set of functions that can be called by the user, a framework has an abstract design and provides certain functionalities which can be changed selectively by user-written code. Then, the framework's code can call the user code. In this way, users can extend the framework to adapt it to their needs.

The MMF framework is written in Java and is available as an open-source project on github: `https://github.com/magkai/MultimodalFissionFramework`. The implementation fulfills the following requirements:

- It is independent of any available dialog manager and it is therefore possible to use the framework together with various dialog managers
- The resulting software product is extendable by new modalities and devices as well as new planning criteria
- Several modalities and devices as well as multiple planning criteria are provided, which can be directly used out of the box
- The software product is highly configurable in order to be adaptable to different (and complex) application scenarios

This chapter demonstrates how the concepts, presented in the previous chapter, are implemented. First, an overview of the implemented procedure is given. After that, the concrete input and output representations, used in the MMF framework, are illustrated. Then some implementation details about the Attribute Selection Algorithm, which extracts salient attributes for verbal references, are provided. Furthermore, how modalities and devices are represented in the MMF framework and how the modality and device selection is implemented is described. The chapter concludes with a description of the three example scenarios, which have been implemented for demonstrating the usage of the framework.

5.1. Procedure Overview

An overview of the procedure implemented in the MMF framework is given in Figure 5.1. The individual steps are the following:

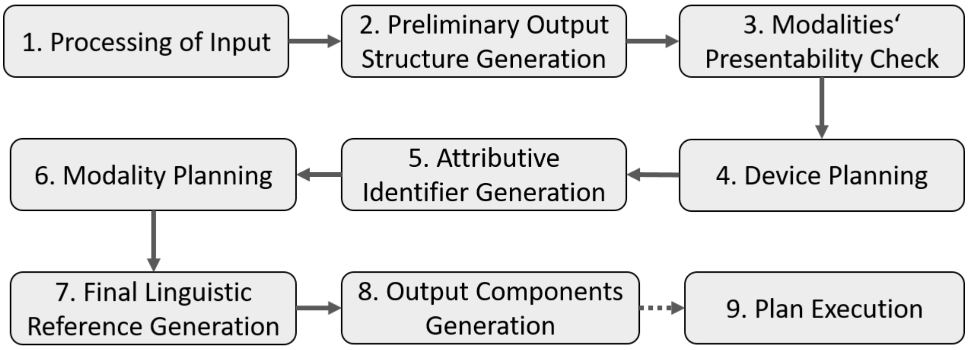

1. Processing of Input
2. Preliminary Output Structure Generation
3. Modalities' Presentability Check
4. Device Planning
5. Attributive Identifier Generation
6. Modality Planning
7. Final Linguistic Reference Generation
8. Output Components Generation
9. Plan Execution

Figure 5.1.: Main Procedure in the MMF framework

1. The internally used world model is created in an initialization step before the actual start of the framework (see section 5.2.2). Furthermore, the predicate input is parsed and annotated (see section 5.2.1).

2. The speech modality generates a preliminary output structure based on the annotated predicates which will then be used by all other modalities.

3. The modalities check how well they can present each part of the output by assigning a value between 0.0 (not presentable at all) to 1.0 (perfectly presentable). In case they can present the output, they store information on how to present it. The final speech output of the framework will be determined during the planning process. Therefore, the speech modality can later update its stored information.

4. For each combination, consisting of a modality and each of its presentable output elements, a suitable device is selected (see section 5.6.1).

5. The Attribute Identifier Algorithm is used to generate a linguistic reference based on salient attributes (see section 5.4).

6. A set of modalities is selected for each output element, while taking several different criteria into account. Furthermore, the most suitable linguistic reference based on the context can be selected. The attributive identifier created beforehand is used as default (see section 5.6.2).

7. The final linguistic references are created based on the planning result. The speech modality can now update its stored information accordingly.

8. The output components are generated based on the planning result and the resulting plan is returned (see section 5.3).

9. Additionally, an execution component is available which can execute the final plan concurrently (see section 5.3).

5.2. MMF Input

Listing 5.1 shows the initialization of the `Controller` class, the entry point of the framework.

```
Controller c = new Controller(predicate, LanguageFormat.ENG_SIMPLENLG,
                               talkingToUserList);
```

Listing 5.1: Controller Initialization

The controller receives the semantic predicate (`predicate`) which represents information about what to output in an abstract form. The predicate can either be provided as a string or by using the `Predicate` class provided by the framework. The next input provide information about the output language and the used Natural Language Generation (NLG) tool. In this case, `LanguageFormat.ENG_SIMPLENLG` is used, which means that English sentences generated with SimpleNLG[1] will be produced. SimpleNLG is a Java API for Natural Language Generation and will be described further in section 5.2.1. The last input consists of a list containing the *worldobjectids* of the current human interaction partners for whom the output is created for (`talkingToUserList`). This information helps the framework to tailor its output to the corresponding users, their preferences and special needs.

5.2.1. Semantic Predicate

The MMF framework illustrates the predicate usage by supporting 30 built-in sample predicates. Eight of them have no arguments and represent phrases, like “hello”, “thank you” or “no”. Extending the list of supported predicates is easily possible.

SimpleNLG is used for creating a sentence out of the respective predicate. SimpleNLG [21] is an open-source Java library providing a surface realization engine for the English language. The library allows to build and combine sentences, as well as inflectional morphological operations and linearization. According to one of its authors, “SimpleNLG has always been inferior to other realisers (such as KPML[2] or OpenCCG[3]) in terms of functionality, but then its focus has always been on usability” [44]. Indeed, SimpleNLG has been chosen for this thesis since it is very easy to use and it is sufficient to realize simple natural sentences with it.

At least one template is defined for each supported predicate, which states its predicate structure. As already described in section 4.1.1, annotations can be extracted from the templates. It is possible to define several templates for each predicate in order to be able to support different languages or various NLG tools. Currently, three templates are available for each predicate, whereas one of them provides information required by the SimpleNLG tool. Example 39 shows the three provided templates for the predicate “vecnu”.

[1] SimpleNLG: `https://github.com/simplenlg` (last accessed: 19th of May 2017)

[2] KPML: `http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/README.html` (last accessed: 19th of May 2017)

[3] OpenCCG: `http://openccg.sourceforge.net/` (last accessed: 19th of May 2017)

Example 39. Templates for “vecnu”

vecnu(x_1, x_2, x_3, x_4) :=
(ENG): [x_1] [sell] [x_2] [to x_3] [for x_4]
(DEU): [x_1] [verkaufen] [x_2] [an x_3] [für x_4]
(ENG_SIMPLENLG): [subject/WHO_SUBJECT: x_1] [verb: sell]
[directObject/WHAT_OBJECT: x_2] [indirectObject: to x_3]
[complement: for x_4]

The first and the second template provide the structure of the sentence in English and in German respectively. They are independent of any NLG tool. The last template contains information about the basic sentence units which are subject, verb, direct/indirect object and complement. A complement in SimpleNLG is “anything that comes after the verb” [34], which can be, for example, adjective, adverbial or prepositional phrases. Furthermore, the example depicts the question annotations “WHO_SUBJECT” and “WHAT_OBJECT”. If a question should be generated in which the subject is queried, a question with “who”, asking for the respective person, is needed for this predicate. In the example, a person rather than an object can sell something. Thus, the resulting question would be: “*Who* sells ...?”. The same holds for the direct object, that states the object to be sold. Therefore, the resulting question would be: “*What* does person x_1 sell?”.

In the following, the classes prescribing the predicate will be presented. Figure 5.2 shows the UML class diagrams of the relevant classes. If the input predicate is provided in string format, then it first needs to be converted into the internally used `Predicate` class. Each `Predicate` has a name (`predicateName`) and may contain a list of modifiers (`predicateModifiers`). These modifiers precede the predicate and effect it as a whole, like [xu], which defines a yes-no question, or [ko], which defines a command (see section 4.1.1 for further examples). Furthermore, the `Predicate` can have various `PredicateAnnotation` instances. The annotations, extracted from the template, are used to provide SimpleNLG with the knowledge whether a negated sentence, certain type of question or a command should be generated. Additionally, a `NO_NLG` option is available, which can be used for phrases consisting of one word for which no Natural Language Generation is needed.

Moreover, the `Predicate` can have several `PredicateElement` instances representing its arguments. `PredicateElement` is an interface which has one class implementing it, namely `StringPredicateElement`, which represent the corresponding argument in string format. However, the interface is provided to be flexible and to allow further concrete representations if desired. If the argument, represented by the `StringPredicateElement`, refers to an object in the world model, its corresponding *worldobjectid* and *worldobjecttype* are saved inside the `StringPredicateElement`. Furthermore, a `StringPredicateElement` can have a `PredicateElementAnnotation`. These correspond to the basic sentence units which are also extracted from the template. Each `StringPredicateElement` can only have one such annotation. SimpleNLG can use these annotations together with the annotations for the entire predicate to create a simple sentence from the predicate input. It is the task of the speech modality to call the respective SimpleNLG functions for the sentence generation.

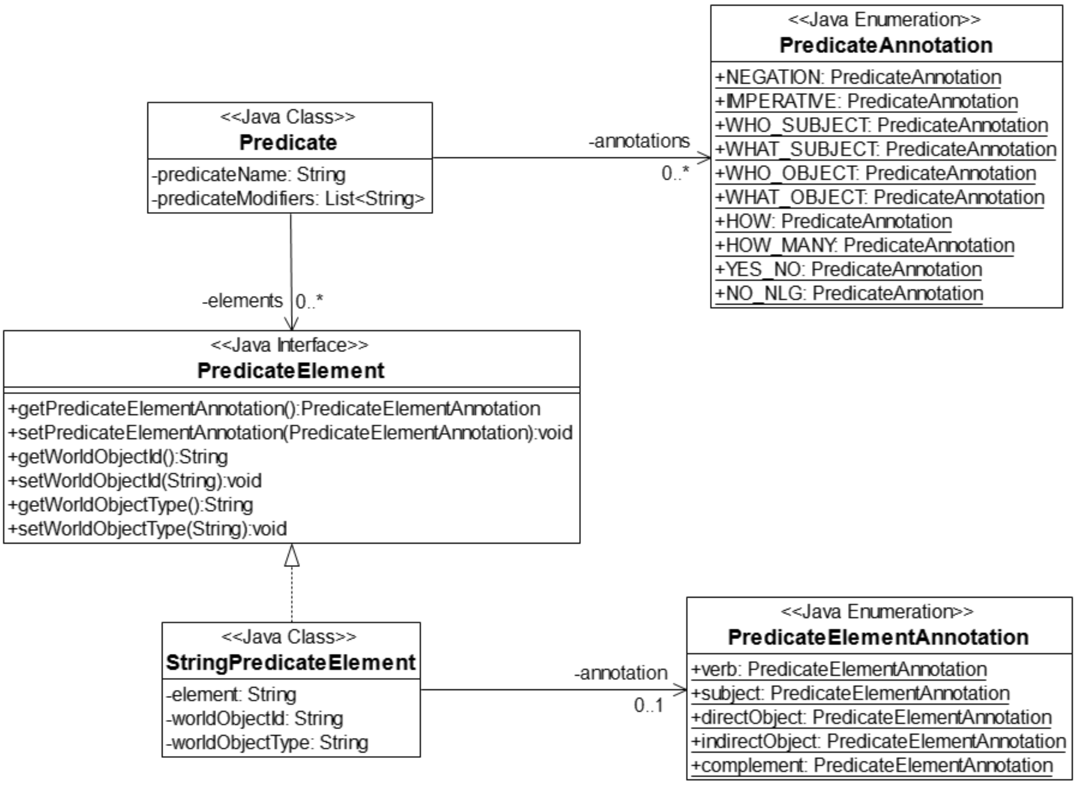

<<Java Class>>
Predicate
-predicateName: String
-predicateModifiers: List<String>
-annotations
0..*
<<Java Enumeration>>
PredicateAnnotation
+NEGATION: PredicateAnnotation
+IMPERATIVE: PredicateAnnotation
+WHO_SUBJECT: PredicateAnnotation
+WHAT_SUBJECT: PredicateAnnotation
+WHO_OBJECT: PredicateAnnotation
+WHAT_OBJECT: PredicateAnnotation
+HOW: PredicateAnnotation
+HOW_MANY: PredicateAnnotation
+YES_NO: PredicateAnnotation
+NO_NLG: PredicateAnnotation
-elements
0..*
<<Java Interface>>
PredicateElement
+getPredicateElementAnnotation():PredicateElementAnnotation
+setPredicateElementAnnotation(PredicateElementAnnotation):void
+getWorldObjectId():String
+setWorldObjectId(String):void
+getWorldObjectType():String
+setWorldObjectType(String):void
<<Java Class>>
StringPredicateElement
-element: String
-worldObjectId: String
-worldObjectType: String
-annotation
0..1
<<Java Enumeration>>
PredicateElementAnnotation
+verb: PredicateElementAnnotation
+subject: PredicateElementAnnotation
+directObject: PredicateElementAnnotation
+indirectObject: PredicateElementAnnotation
+complement: PredicateElementAnnotation

Figure 5.2.: UML Class Diagrams representing Information about the Semantic Predicate

5.2.2. World Model

Currently, the information stored in the world model of the MMF framework can be retrieved from an ontology written in the Web Ontology Language (OWL)[4] or from the database entries of a document-orientated MongoDB[5] database. In section 5.7, the three implemented example scenarios will be described. One of them uses an OWL ontology, whereas the other two have MongoDB databases as their sources for retrieving the world model information. The world model and its submodels, as presented in section 4.2, primarily consists of JSON objects. JSON[6] is a data-interchange format that is completely language independent. JSON objects are unordered sets of key-value pairs. Thus, they are very suitable to store the properties of the objects to be defined in the world model, since each object property consists of an attribute name and attribute values.

OWL ontologies are widely used for knowledge representation. Therefore, they are supported by the MMF framework. Additionally, it has been decided to support MongoDB, since document-orientated databases, like MongoDB, use JSON documents to store records. Therefore, it is very easy to extract the information from the database and store it in the internally used JSON representation. Furthermore, in contrast to simple key-value stores, document-orientated databases are able to store nested key-value pairs. Thus, nested properties, like the position property which is further divided

[4] OWL: `https://www.w3.org/TR/owl-ref/` (last accessed: 19th of May 2017)
[5] MongoDB: `https://www.mongodb.com/` (last accessed: 19th of May 2017)
[6] JSON: `http://www.json.org/` (last accessed: 19th of May 2017)

into its x-, y- and z-coordinate, can be represented.

The MMF framework requires access to the ontology respectively the database in order to retrieve the data and store it as JSON objects. This process is handled by the `WorldModelFactory` class. Listing 5.2 shows the pseudocode of the method, provided by the `WorldModelFactory` class, that needs to be called by the user code to create or respectively update the internal world model.

```
FUNCTION createUpdateWorldModel(modelType, resourcePath)
    CASE modelType OF
        OWL: owlModelRetrieval(resource);
        MONGODB: mongoDBModelRetrieval(resource);
    ELSE
        OUTPUT ‘‘Model type not supported yet.’’;
    ENDCASE
ENDFUNCTION
```

Listing 5.2: Creating or Updating the Internal World Model

Both currently supported `modelTypes`, “OWL” and “MONGODB” are depicted. For each `modelType`, a corresponding method which retrieves the data from that model and stores it into JSON objects is required (in this case the methods `owlModelRetrieval` and `mongoDBModelRetrieval` are provided). These methods require the path to the corresponding model (`resourcePath`) as input. The switch-case structure allows to add new `modelTypes` and its corresponding retrieval-methods in an easy manner.

As already stated above, the data retrieval from the MongoDB database is fairly easy. The owl model is parsed with OWL API[7] to retrieve the relevant data. “OWL API is an open-source Java API and reference implementation for creating, manipulating and serializing OWL ontologies” [28]. An ontology is viewed as a set of axioms and annotations and provides a high level of abstraction which is above RDF or triple-based representations. A reasoner interface is available which provides access to functionality relating to the process of reasoning with OWL ontologies, like consistency checking and the computation of class or property hierarchies. A simple structural reasoner provided by the API, which implements the interface, is used in the MMF framework to retrieve the classes and their individuals as well as their object and data properties from the ontology. Using this simple reasoner has been sufficient for extracting the information of the ontology used by one of the example scenarios provided alongside the framework (see section 5.7). For more complicated ontologies, the reasoner can be replaced by a more sophisticated one.

The retrieved JSON objects are stored in the `WorldModel` class and its subclasses. Figure 5.3 shows the corresponding class diagrams. All properties of objects in the environment are stored in the list of `worldProperties`. Furthermore, the `WorldModel` contains the `saliencyAnnotation`, which are also provided as a JSON object. Apart from that, the `WorldModel` includes the `coneIntersectionMap` which stores for each object the entities located between the robot and that object. Currently, the `WorldModel` contains one `RobotModel` and several `UserModel` instances. The `WorldModelFactory`

[7] OWL API: `http://owlapi.sourceforge.net/` (last accessed: 19th of May 2017)

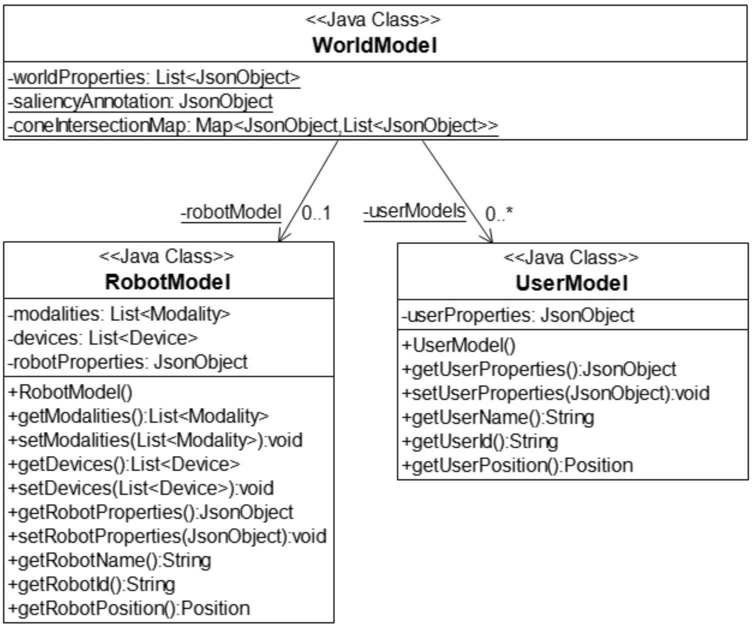

<<Java Class>>
WorldModel
-worldProperties: List<JsonObject>
-saliencyAnnotation: JsonObject
-coneIntersectionMap: Map<JsonObject,List<JsonObject>>
-robotModel
0..1
-userModels
0..*
<<Java Class>>
RobotModel
-modalities: List<Modality>
-devices: List<Device>
-robotProperties: JsonObject
+RobotModel()
+getModalities():List<Modality>
+setModalities(List<Modality>):void
+getDevices():List<Device>
+setDevices(List<Device>):void
+getRobotProperties():JsonObject
+setRobotProperties(JsonObject):void
+getRobotName():String
+getRobotId():String
+getRobotPosition():Position
<<Java Class>>
UserModel
-userProperties: JsonObject
+UserModel()
+getUserProperties():JsonObject
+setUserProperties(JsonObject):void
+getUserName():String
+getUserId():String
+getUserPosition():Position

Figure 5.3.: UML Class Diagrams of the `World Model` and its Subclasses

fills these models with the required information. Properties regarding the robot and the users are additionally stored in the `robotProperties` of the `RobotModel` class and in the `userProperties` of the `UserModel` class, respectively. In the current implementation, no `ContextModel` class is provided, since the object properties are directly stored in the `WorldModel`. Furthermore, the `RobotModel` contains a list of available modalities and devices. There are several methods for accessing specific properties like the robot's/user's name, id and position. These properties are frequently used inside the framework and in this way they can be accessed more conveniently.

5.3. MMF Output

Figure 5.4 shows the relevant class diagrams for the output. The basic output unit is the `PlanComponent` (*PC*), consisting of a modality, an output and a device. Note that currently a modality is not required for executing an output using the device models of the MMF framework. Yet, it is part of the plan returned by the MMF framework, since a different output execution might require the modality information. Additionally, the *PC* can use the `executionTime` variable for storing the approximate time needed for outputting this element. The corresponding physical device can be queried for this information. A `ComposedPlanComponent` (*CPC*) has one or several *PC* instances which will be executed concurrently. An approximate `executionTime` can also be provided for this component. The output of the MMF framework consists of a list of *CPCs*, in which the order of the components determines the execution order. Furthermore, the framework provides the `PlanExecutor` as execution component, which receives the planned output and triggers the execution of the plan components on the concrete devices. It is up to the user of the framework to use this `PlanExecutor` or their own execution mechanism.

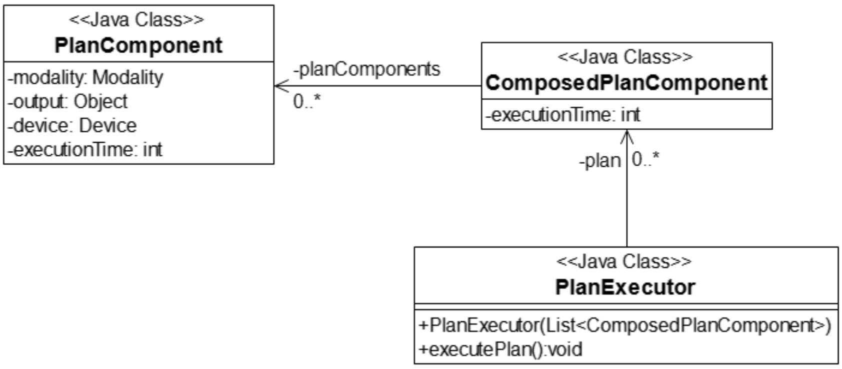


Figure 5.4.: Output UML Class Diagrams

The provided `executePlan()` method is presented as pseudocode in Listing 5.3. The method receives the final plan, which is the list of *CPCs*, as input. The *CPCs* are executed consecutively, while the individual *PCs* belonging to one *CPC* are executed concurrently by their own threads. This allows the corresponding devices to perform their actions at the same time. Since the devices are implemented as `Runnables`, they can be given to a thread. The `Device` class will be presented in section 5.5.2.

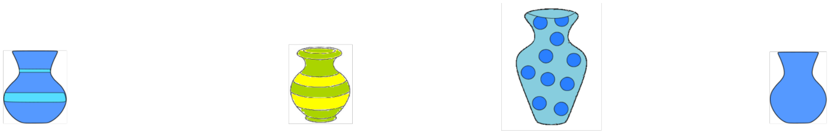
```
FUNCTION executePlan(plan)
    #execute the PCs of one CPC on the corresponding devices in parallel
    FOR(i = 0 TO LEN(plan)-1)
        composedPlanComponent = plan[i]
        planComponents = composedPlanComponent.PlanComponents();
        deviceThreads[] = new Thread[LEN(planComponents)];
        #retrieve the corresponding device and output
        #each device is a thread to execute the corresponding output
        FOR(j = 0 TO LEN(planComponents)-1)
            device = planComponents[j].Device();
            output = planComponents[j].Output();
            device.Output = output;
            deviceThreads[j] = new Thread(device);
        ENDFOR
        FOR(k = 0 TO LEN(planComponents)-1)
            deviceThreads[k].start();
        ENDFOR
        FOR(k = 0 TO LEN(planComponents)-1)
            deviceThreads[k].join();
        ENDFOR
    ENDFOR
ENDFUNCTION
```

Listing 5.3: Plan Execution

5.4. Implementation of Attribute Selection Algorithm

The idea behind the Attribute Selection Algorithm (ASA) has already been presented in section 4.4.1: The algorithm extracts a suitable set of salient attributes to uniquely reference an object out of a set of similar objects. In the following, the algorithm will be described in detail, by means of the vase example already used in section 4.4.1. Example 40 depicts the four vases, their stored attributes and saliency values.

Example 40. Stored Vase Attributes and the Attributes' Saliency Annotation



0.0	⟨worldobjectid: vase1⟩	⟨worldobjectid: vase2⟩	⟨worldobjectid: vase3⟩	⟨worldobjectid: vase4⟩
1.0	⟨worldobjecttype: vase⟩	⟨worldobjecttype: vase⟩	⟨worldobjecttype: vase⟩	⟨worldobjecttype: vase⟩
1.0	⟨color: blue⟩	⟨color: yellow, green⟩	⟨color: blue⟩	⟨color: blue⟩
0.9	⟨size: small⟩	⟨size: small⟩	⟨size: big⟩	⟨size: small⟩
0.9	⟨motif: stripes⟩	⟨motif: stripes⟩	⟨motif: dots⟩	⟨motif: plain⟩
0.7	⟨position: leftmost⟩	⟨position: center left⟩	⟨position: center right⟩	⟨position: rightmost⟩
0.0	⟨price: 15€⟩		⟨price: 34€⟩	

The algorithm's procedure will be described by using the example of extracting suitable attributes for referencing the vase with *worldobjectid* "vase1", called referent in the following.

In the first step of the algorithm, all possible combinations of salient attributes need to be found. This step is divided into two parts. In the first part, the attribute values of the referent are compared to the attribute values of all other objects of the same type consecutively. Listing 5.4 presents this part of the algorithm as pseudocode.

```
 1 FUNCTION computeSingleDiscriminatingIdentifiers(referent, similarObjects,
 2                                                 saliency, threshold)
 3     #the attributes of each similarObject will be compared to the
 4     #attributes of the referent
 5     FOR(i = 0 TO LEN(similarObjects)-1)
 6         FOR(j = 0 TO LEN(referent.Attributes)-1)
 7             referentKey = referent.Attributes[j].Key();
 8             #look up the saliency annotation of attribute
 9             IF(referentKey IN saliency.Keys())
10                 saliencyValue = saliency[referentKey];
11                 #prune attributes with saliency lower than threshold
12                 IF(saliencyValue < threshold)
13                     CONTINUE;
14                 ENDIF
15             ELSE
16                 CONTINUE;
17             ENDIF
18             referentValues = referent.Attributes[referentKey];
19             #check if similarObject has current attribute
20             IF(referentKey IN similarObjects[i].Attribute.Keys)
21                 similarObjectValues =
22                     similarObjects[i].Attributes[referentKey];
23                 #check if attribute values differ -> if so, store
24                 #referent's differentiating attribute values
25                 IF( NOT (referentValues IN similarObjectValues AND
26                          similarObjectValues IN referentValues))
27                     singleIdentifiers.ADD(referentKey, referentValues);
28                 ENDIF
29
30             #if similarObject does not have particular attribute ->
31             #can be seen as differentiating
32             ELSE
33                 singleIdentifiers.ADD(referentKey, referentValues);
34             ENDIF
35         ENDFOR
36         #accumulate identifiers
37         identifiers.ADD(singleIdentifiers);
38     ENDFOR
39     RETURN identifiers;
40 ENDFUNCTION
```

Listing 5.4: Finding Discriminating Identifier Sets

The function receives the referent, all objects of the referent's type (`similarObjects`), the saliency annotations (`saliency`) and a threshold for saliency (`threshold`) as input. The referent's attributes are compared to the attributes of each `similarObject` consecutively. Only those attributes are considered further which have a saliency value above certain threshold, the others are pruned (line 12 and 13). By default, the threshold is 0.5. In the example, the *worldobjectid* and the price are not considered further since their saliency values are below 0.5. If a certain attribute is not available for one of the objects, the attribute can be seen as differentiating this object from the referent (line 33). For example, if the price tag was considered further, the price of vase1 would be a differentiation from vase2, for which the price is not known. Lines 25 and 26 check if the referent's attribute values differ from the ones of the current `similarObject`. The attribute values of two objects are only considered as equal if they share all values. For example, if a yellow vase was compared with vase2, which has yellow and green stated as color attribute values, the two vases would be seen as different in color. All attributes which distinguish the current `similarObject` from the referent are stored in `singleIdentifiers` (line 27 and line 33). The output of the function consists of a set (`identifiers`) storing for each comparison between referent and element of the `similarObjects` the set of distinguishing attributes (`singleIdentifiers`). The results for the vase example for this step can be seen in Figure 5.5. The number of the set indicates the comparison of the referent with a certain vase (e.g., Set 2: comparison between referent (vase1) and vase2). The attributes with the corresponding values for vase1 are depicted. When comparing vase1 with vase2, only the color and the position differentiate the two vases (Set 2). The comparison with vase3 yields the size, the motif and the position (Set 3) and the comparison with vase4 the motif and the position (Set 4) as differentiating attributes.

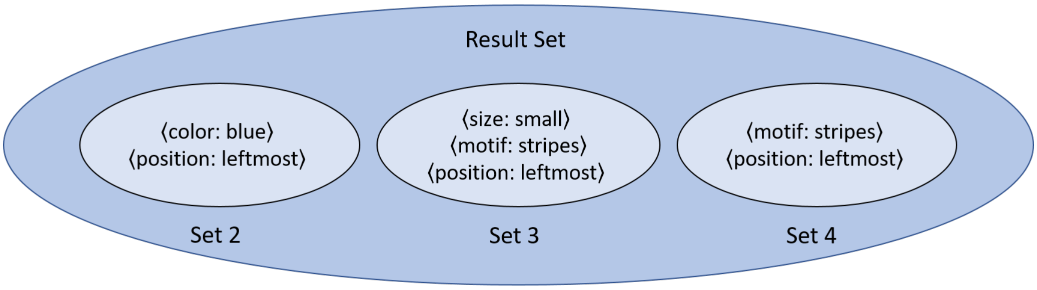


Figure 5.5.: Attribute Sets resulting from the Comparison with the Referent

In part two of the first step, all possible combinations of attributes which uniquely identify the referent are built from these sets. This means that each resulting combination needs to contain at least one attribute from each set. Obviously, duplicate attributes are not allowed in the built combinations. Furthermore, the order of the attributes does not matter. First, the powerset of all found distinguishing properties is calculated. However, the powerset can contain nonvalid combinations. For example, the combinations ⟨blue, small⟩ and ⟨small, stripes⟩ are not valid, since both combinations are not sufficient to uniquely describe the referent: There are two blue and small vases (vase1 and vase4) and two small vases with stripes (vase1 and vase2). Thus, the nonvalid combinations need to be removed from the powerset. This procedure is depicted in Listing 5.5. The function receives the `identifiers`, which is a set of sets that has been returned by the pseudocode presented in Listing 5.4, and the calculated `powerSet` as input. A entry in the `powerSet` is only valid if each individual set in the `identifiers` contains at least one element of this `powerSet` entry. For example, the entry ⟨blue, small⟩ has no element which is contained in Set 4 of Figure 5.5. Therefore, it is removed in line 7.

```
FUNCTION removeNonValidCombinations(identifiers, powerSet)
    #a powerset entry is valid, if it contains only elements
    #present in each individual identifier
    FOR(i = 0 TO LEN(powerSet)-1)
        FOR(j = 0 TO LEN(identifiers)-1)
            IF(NO element OF powerSet[i] IN identifiers[j])
                REMOVE powerSet[i];
                BREAK;
            ENDIF
        ENDFOR
    ENDFOR
    RETURN powerSet
ENDFUNCTION
```

Listing 5.5: Removal of Nonvalid Combinations

Here is a list of all allowed combinations for the vase example:

- ⟨leftmost⟩
- ⟨leftmost, blue⟩
- ⟨leftmost, small⟩
- ⟨leftmost, stripes⟩
- ⟨blue, stripes⟩
- ⟨leftmost, blue, small⟩
- ⟨leftmost, blue, stripes⟩
- ⟨leftmost, small, stripes⟩
- ⟨blue, small, stripes⟩
- ⟨leftmost, blue, small, stripes⟩

The position attribute is the only single attribute which can be used to uniquely refer to vase1. The remaining attributes can only be used in combination with others.

The second step of the algorithm consists of selecting the final combination of attributes, which should be used to reference the object, based on certain criteria. The MMF framework currently provides four different criteria and it is up to the user of the framework to decide which criteria to use or to implement a new one tailored to their specific needs. In the MMF framework, two main factors are considered for the selection: the number of attributes in a combination and the overall saliency value of the attributes contained in the combinations. One provided criterion only considers the number of attributes in the combination and selects the shortest combination. In the vase example, the attribute "leftmost" is the shortest description to uniquely identify the referent. Yet, this attribute has a saliency value of 0.7. Therefore, it might be better in some cases to use several, possibly redundant attributes with a higher overall saliency value to make sure that the object can be identified correctly by the user. Therefore, another provided criteria only considers saliency and thus the largest list with attributes is chosen. However, this might result in a long and unnecessary complicated description. In the vase example, all four attributes would be used (⟨leftmost, blue, small, stripes⟩). Therefore, combining the aim for high saliency and shortness may yield better results.

The third offered criterion selects the shortest list of attributes with a saliency value above a certain threshold. If there are several combinations of the same length above the threshold, then the most salient combination is chosen. A sigmoid function, which has monotonically increasing output values between 0 and 1 for positive real value inputs, is used to keep the accumulated saliency values between 0.0 and 1.0.

The following sigmoid function is applied:

$$\frac{x}{\sqrt{1+x^2}} \tag{5.1}$$

For example, the combination ⟨leftmost, blue⟩ has a combined saliency value of

$$\frac{0.7+1.0}{\sqrt{1+(0.7+1.0)^2}} \approx 0.86 \tag{5.2}$$

Assume a threshold of 0.8 has been chosen by the user of the framework for the third criterion. The sigmoid function is also applied on the threshold, which results in a value of about 0.62. The combinations with their saliency values (sv) are depicted in the following, in descending order of their sv:

- ⟨blue, leftmost, small, stripes⟩, sv: 0.96
- ⟨blue, small, stripes⟩, sv: 0.94
- ⟨blue, leftmost, stripes⟩, sv: 0.93
- ⟨blue, leftmost, small⟩, sv: 0.93
- ⟨leftmost, small, stripes⟩, sv: 0.93
- ⟨blue, stripes⟩, sv: 0.88
- ⟨blue, leftmost⟩, sv: 0.86
- ⟨leftmost, stripes⟩, sv: 0.85
- ⟨leftmost, small⟩, sv: 0.85
- ⟨leftmost⟩, sv: 0.57

Obviously, combinations with several attributes have a higher saliency value. The single attribute ⟨leftmost⟩ cannot be used since its saliency value of 0.57 is lower than the chosen threshold of 0.62. All combinations with two elements have a saliency value above the threshold. Therefore, it is sufficient to use one of them. The combination ⟨blue, stripes⟩ will be used, because it has the highest saliency value (0.88) among the combinations with two elements.

The last offered criterion sorts all combinations according to their length and to their saliency value separately. Depending on their position in each list, they receive a rank for this list, where a smaller number indicates a better result. The two different rankings are depicted in the following:

Saliency Ranking:

- ⟨blue, leftmost, small, stripes⟩, rank: 1
- ⟨blue, small, stripes⟩, rank: 2
- ⟨blue, leftmost, stripes⟩, rank: 3
- ⟨blue, leftmost, small⟩, rank: 3
- ⟨leftmost, small, stripes⟩, rank: 3
- ⟨blue, stripes⟩, rank: 4
- ⟨blue, leftmost⟩, rank: 5
- ⟨leftmost, stripes⟩, rank: 6
- ⟨leftmost, small⟩, rank: 6
- ⟨leftmost⟩, rank: 7

Shortness Ranking:

- ⟨leftmost⟩, rank: 1
- ⟨leftmost, blue⟩, rank: 2
- ⟨leftmost, small⟩, rank: 2
- ⟨leftmost, stripes⟩, rank: 2
- ⟨blue, stripes⟩, rank: 2
- ⟨leftmost, blue, small⟩, rank: 3
- ⟨leftmost, blue, stripes⟩, rank: 3
- ⟨leftmost, small, stripes⟩, rank: 3
- ⟨blue, small, stripes⟩, rank: 3
- ⟨leftmost, blue, small, stripes⟩, rank: 4

The combination should be chosen which has a good rank in both categories. As can be seen, both categories are complementary, which means that a higher saliency value is reached when using a longer list of attributes and vice versa. Therefore, a combination is chosen which ranks in the middle for both categories. The final rank of a combination is calculated in the following way:

$$\frac{salRank}{nbrOfSalRanks} + \frac{shortRank}{nbrOfShortRanks} + \left| \frac{salRank}{nbrOfSalRanks} - \frac{shortRank}{nbrOfShortRanks} \right| \tag{5.3}$$

The variable *salRank* denotes the rank a combination has in the saliency ranking and *shortRank* denotes the rank in the shortness ranking. By $nbrOfSalRanks$ the overall amount of ranks in the saliency ranking (7 in this example) is denoted and $nbrOfShortRanks$ corresponds to the amount of ranks in the shortness ranking (4 in this example). The rank of a combination is divided by that number to receive the relative position of the combination. This is done in order to make the two rankings, which differ in their number of ranks, more comparable. The resulting relative positions are summed up. Furthermore, the absolute value of the difference between the relative ranks is added as well to penalize a combination which has a very good rank in one ranking but a worse one in the other. The combination with the smallest overall rank is chosen in the end.

In this example, ⟨blue, stripes⟩ is the result when applying this criterion. The same result has been found when using criterion 3 with a threshold of 0.8, as presented before. Criterion 3 is simpler than this one, but yields quite good results in practice if a suitable threshold is chosen. Criterion 4 aims to find a combination which maximizes both, saliency and shortness. The result might be further improved by varying the calculation of the final rank.

5.5. Modality and Device Representation

Currently, six modalities are supported by the MMF framework, namely speech, pointing, gaze, image, waving and nodding/headshaking. Since the MMF framework focuses on human-robot interaction, the currently supported devices mainly consist of components of a humanoid robot, namely its arms, its head, its eyes and its speakers. Furthermore, a screen is supported, which can either be part of the robot (like in the case of a Baxter robot[8]) or external. As already stated in section 2.2, one modality may have several devices and one device can execute output from several modalities. Figure 5.6 depicts this relation for the modalities and devices used in the framework. The generated speech can be outputted by the robot's speakers and images can be displayed on the screen. Gaze can be expressed by the robot's eyes and a pointing as well as a waving gesture can be performed with the robot's arms. Furthermore, the robot's head can perform a nodding or head-shaking gesture. In general, each modality generates output in a specific output format which can be understood by the corresponding device representation.

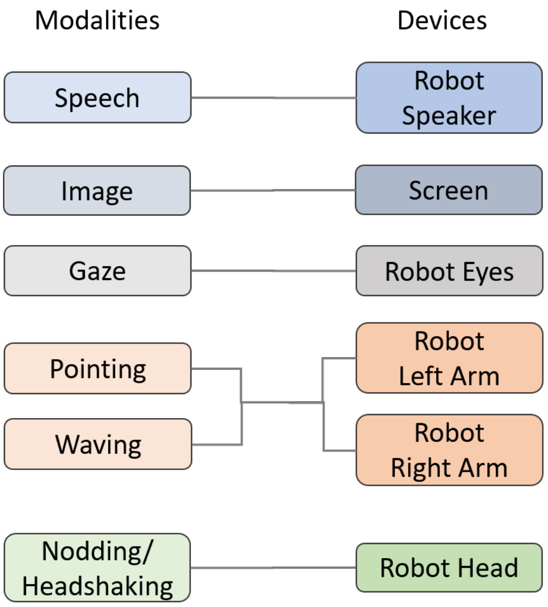


Figure 5.6.: Relationship between Modalities and Devices supported by the MMF Framework

5.5.1. Modality Representation

How modalities and their functionalities are represented in the MMF framework will be described in the following. An overview of the relevant class diagrams can be found in Figure 5.7. The `Modality` is an abstract class containing functionalities relevant for all modalities. Each individual modality inherits from this abstract class. This has the advantage that a list of modalities can easily be iterated without knowing the modalities' exact types. The `Modality` class contains a list of devices (`deviceList`) which can output the information represented by the specific modality. Furthermore, each

[8] Baxter Robot: `http://www.rethinkrobotics.com/de/baxter/` (last accessed: 19th of May 2017)

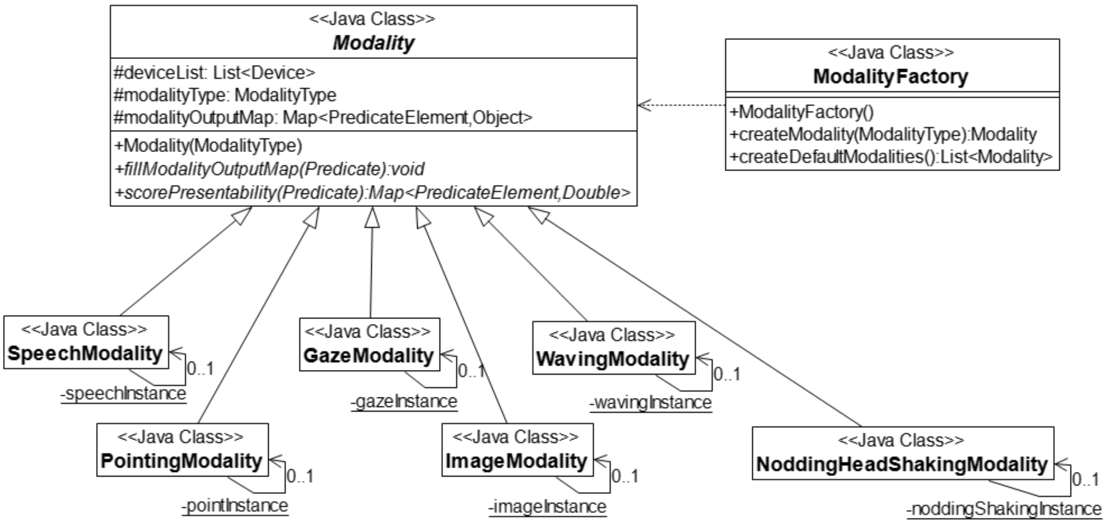


Figure 5.7.: UML Class Diagrams of `ModalityFactory` and abstract `Modality` including its inheriting Modalities

`Modality` has a `modalityType` (e.g., "POINTING") and a `modalityOutputMap`. This map contains each `PredicateElement` that can be represented by the corresponding modality and the respective output for this element. The output has type `Object` since the `Modality` class is generic and in this way each concrete modality can fill this map using its own output format (e.g., a string representation is used for the `SpeechModality` and a position specification is used by the `PointingModality`). The `Modality` class has two abstract methods, namely `fillModalityOutputMap()` and `scorePresentability()`, which both need to be implemented by the individual modalities. Note that only these two methods are depicted in the `Modality` class diagram since they are the most important ones. Further methods include, for example, the adding and removing of devices. The task of `fillModalityOutputMap()` is to add the required information to the `modalityOutputMap`, whereas the `scorePresentability()` method fills a map, stating how presentable each `PredicateElement` is by the corresponding modality by assigning a value between 0.0 and 1.0. The `ModalityFactory` class can be be called by the user of the framework to create a certain modality instance (`createModality()`). Therefore, the respective `ModalityType` needs to be provided. Alternatively, a set of default modalities, currently consisting of speech, pointing and gaze modalities, can be created (`createDefaultModalities()`). A modality describes a certain general concept which is reflected by its information structure. Therefore, having one modality instance of each `ModalityType` is sufficient. Thus, modalities are realized as Singletons.

Modality Categorization

Apart from inheriting from the abstract `Modality` class, all individual modalities also inherit from one or several interfaces, depending on the modality category they belong to. Figure 5.8 and Figure 5.9 depict the respective interfaces. The `StructureFormingModality` interface provides the `generateOutputStructure()` method which needs to be implemented by the `SpeechModality`, the available structure-forming modality in the MMF framework. This method receives the annotated input predicate and creates

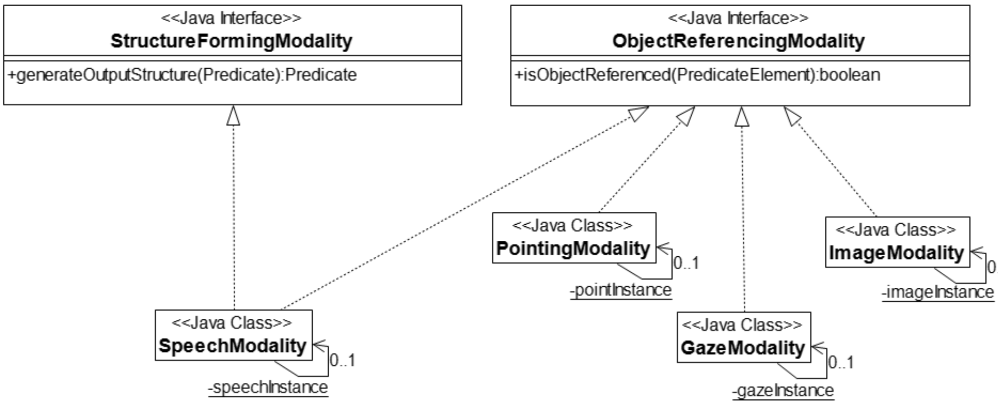


Figure 5.8.: UML Class Diagrams of Structure-Forming and Object-Referencing Interfaces and their inheriting Classes

a modified predicate as output. The SpeechModality uses SimpleNLG in this method to create a preliminary output sentence based on the annotations. Then, it decomposes the sentence into its individual parts and stores them as new PredicateElements in the correct order defined by the created English sentence and returns the modified Predicate with its new PredicateElements.

The ObjectReferencingModality interface provides the isObjectReferenced() method. This method receives a PredicateElement as input and returns true if this PredicateElement contains a *worldobjectid*, which means that an object reference should be performed.

Figure 5.9 presents the PredicateReferencingModality interface. It provides two

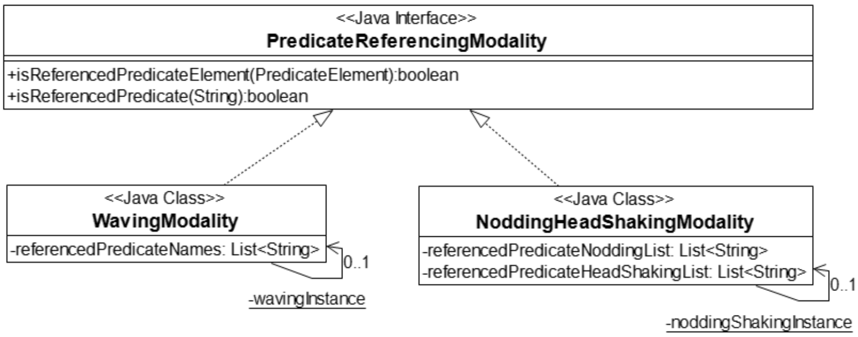

<<Java Interface>>
PredicateReferencingModality
+isReferencedPredicateElement(PredicateElement):boolean
+isReferencedPredicate(String):boolean
<<Java Class>>
WavingModality
-referencedPredicateNames: List<String>
0..1
-wavingInstance
<<Java Class>>
NoddingHeadShakingModality
-referencedPredicateNoddingList: List<String>
-referencedPredicateHeadShakingList: List<String>
0..1
-noddingShakingInstance

Figure 5.9.: UML Class Diagrams of Predicate-Referencing Interface and the inheriting Classes

methods, namely isReferencedPredicateElement() and isReferencedPredicate(). The isReferencedPredicateElement() method receives a PredicateElement as input and returns true if this PredicateElement should be referenced. The isReferencedPredicate() method receives the predicate name in string format as input and outputs true if the whole predicate with this name should be referenced. The WavingModality contains a list of predicate names that can be referenced by this modality. The NoddingHeadShakingModality contains one list of predicate names which can be highlighted by nodding and another list for those where head-shaking can be applied for highlighting.

5.5.2. Device Representation

This section will describe how devices are represented in the MMF framework. Figure 5.10 depicts the device hierarchy for two concrete devices. Each device represented in the MMF framework inherits from the abstract Device class. A Device can have a name (deviceName), a position (devicePosition) and its direction can be stated as a direction vector (directionVector). Furthermore, each device has a unique id (deviceId). Apart from getter and setter methods not depicted in this class diagram, the Device class contains several abstract methods. Three methods address the output which the device should produce (getOutput(), setOutput(), clearOutput()) and another one provides an estimation of the time the device will need to output certain object (getDurationEstimation()).

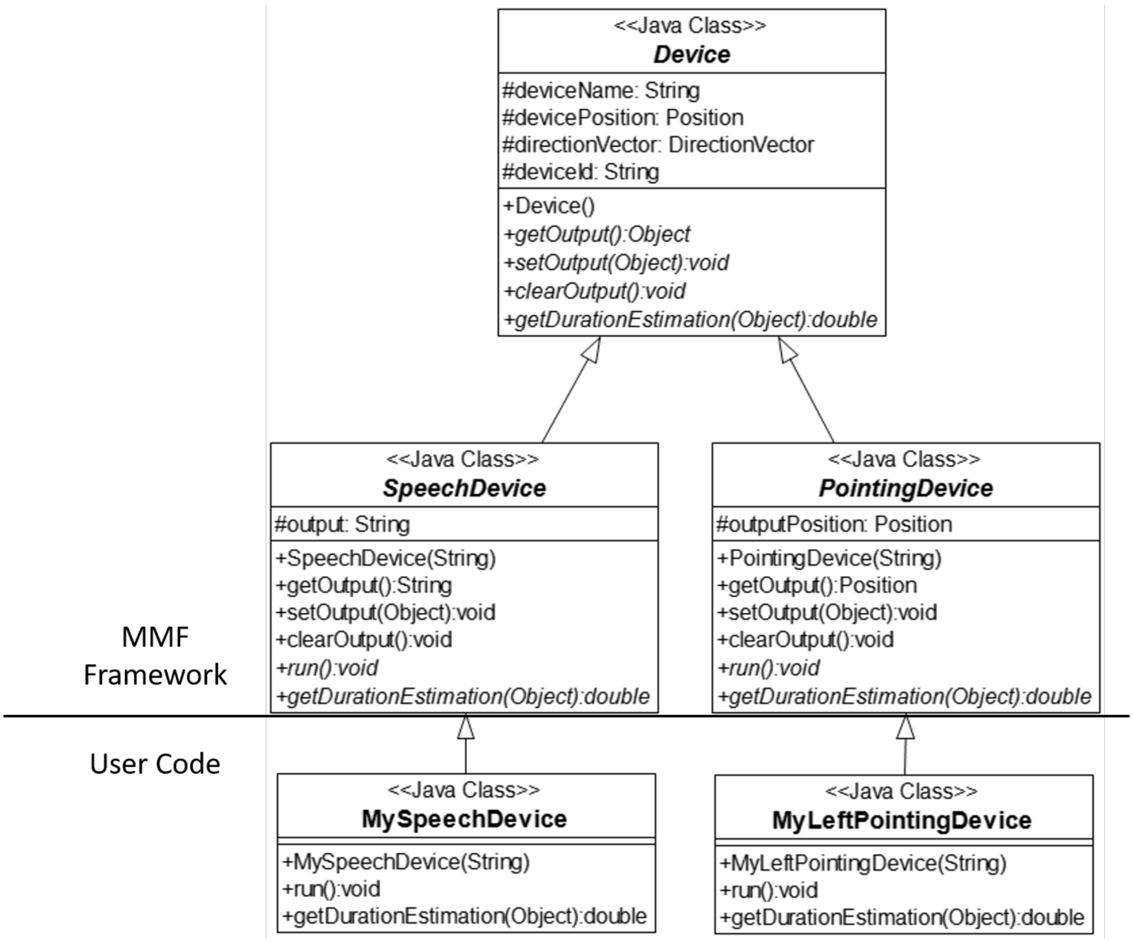


Figure 5.10.: UML Class Diagrams of Device and its Subclasses

The abstract classes SpeechDevice and PointingDevice, taken as representatives for all more specific devices, implement the output methods (getOutput(), setOutput(), clearOutput()). For a SpeechDevice, the output is represented as a String, whereas for the PointingDevice it is a Position object. If the PlanExecutor (see section 5.3) is used, it sets the output for the concrete device. As already stated in section 5.3, the specific devices, like the SpeechDevice and the PointingDevice, are implemented as Runnables and can be executed by threads. The respective run() method forms the transition from framework code to user code: This method stays abstract in the framework and needs to be implemented by a class representing the physical available device, which is provided by the user of the framework. The `MySpeechDevice` and `MyLeftPointingDevice` classes are examples for the concrete devices in this case. Their `run()` methods require access to the physical device they represent to trigger the execution of the desired output on this physical device. Furthermore, since they model the underlying devices, their properties are used in the device selection process. Having access to the physical devices enables them to additionally provide an implementation of the `getDurationEstimation()` method, if needed, which also remains abstract in the framework code.

Thus, the physical devices, depicted in Figure 5.6 at the beginning of this section, underlie a hierarchy from the representation of the concrete, physical devices up to a very abstract device representation. The idea behind this hierarchy is to increase the ease of use: The user of the framework must only implement the `run()` method to execute the output on a new available device (e.g., `MyLeftPointingDevice`) in the case that this type of device is already supported (e.g. `PointingDevice`). If it is not supported yet, they need to go up one step in the hierarchy and implement a new abstract device type, which is fairly easy: The main decision that need to be made is to determine the required output format.

5.6. Modality and Device Selection

It has been decided to use the constraint satisfaction solver OptaPlanner[9] for the modality and device selection. OptaPlanner [13] is an open-source software written in Java and designed to solve optimization problems in an easy and highly configurable way. It provides several optimization heuristics and metaheuristics combined with elaborated score calculations. OptaPlanner is very suitable to realize the concept of the modality and device selection described in section 4.6: The COP formulation can easily be transferred to an implementation using OptaPlanner. As already stated in section 4.6, the standard COP definition has been extended by assigning planning variables to *planning entities*, a concept that has been adopted from OptaPlanner. Furthermore, OptaPlanner supports hard and soft constraints and provides so-called *scorers*, which are used in the MMF framework to compose the soft constraints belonging to one category. Thus, a scorer in OptaPlanner corresponds to a category as presented in section 4.6.2.

An XML file is used to configure OptaPlanner. In this file, the path to the problem specifications, represented as Java classes, is provided. Apart from that, the score and the optimization algorithm can be configured. Currently, a simple Java score calculator together with hard and soft scores and a *First Fit* algorithm as optimization heuristic are used. This configuration can easily be modified and adapted to the user's specific requirements by modifying the XML configuration.

In the following, more details about the implementation of the device and modality selection are presented.

5.6.1. Device Selection

Figure 5.11 shows the relevant class diagrams for the device selection. A `PhraseModalityComponent` represents the planning entity, consisting of tuples of an output element (`PredicateElement`) and a presentable modality (`Modality`), as defined in section 4.6.1. The `device` is the planning variable for which a suitable value need to be found. The method `getPossibleDeviceList()` of the `PhraseModalityComponent` limits the number of possible devices to those which can present the information

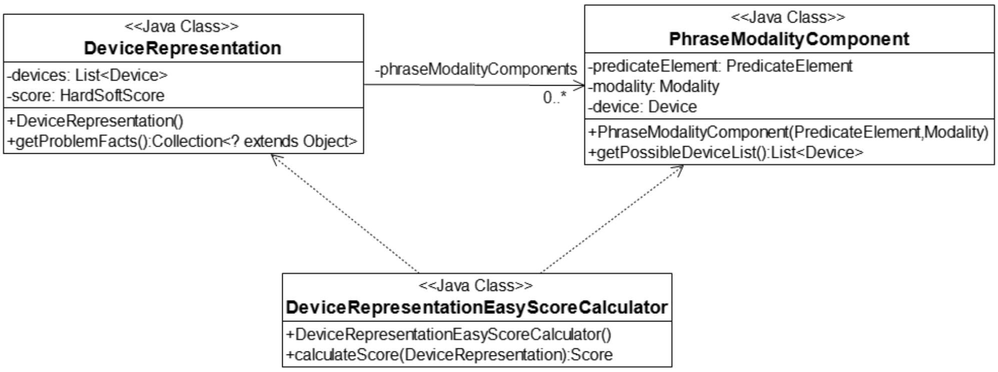

<<Java Class>>
DeviceRepresentation
-devices: List<Device>
-score: HardSoftScore
+DeviceRepresentation()
+getProblemFacts():Collection<? extends Object>
-phraseModalityComponents
0..*
<<Java Class>>
PhraseModalityComponent
-predicateElement: PredicateElement
-modality: Modality
-device: Device
+PhraseModalityComponent(PredicateElement,Modality)
+getPossibleDeviceList():List<Device>
<<Java Class>>
DeviceRepresentationEasyScoreCalculator
+DeviceRepresentationEasyScoreCalculator()
+calculateScore(DeviceRepresentation):Score

Figure 5.11.: UML Class Diagrams for Device Selection

[9] OptaPlanner: `http://www.optaplanner.org/` (last accessed: 19th of May 2017)

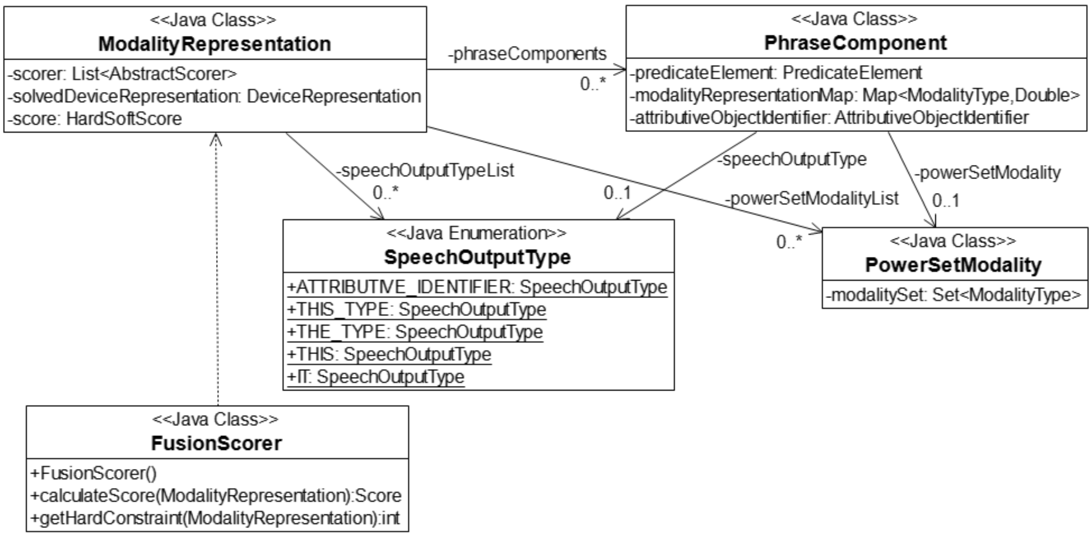
of the respective component's modality. The `DeviceRepresentation` class contains the current assignment and obtains the planning solution in the end. It has a list of the `PhraseModalityComponent` instances as well as a list of devices and the currently reached score. Its `getProblemFacts()` method provide OptaPlanner with the necessary knowledge about the planning problem. The `DeviceRepresentationEasyScoreCalculator` is the class in which the score of the current assignment is calculated using the method `calculateScore()`. It contains the hard and soft constraints for the device selection.

The MMF framework currently provides one hard and two soft constraints for the device selection. The hard constraint ensures that all variables receive a value, which means that a device is assigned to all tuples. One exception to this is the case that a modality has no devices which can represent its information. In this case, no device can be selected. The first soft constraint considers tuples consisting of an object-referencing modality that is no speech modality and a `PredicateElement` containing a *worldobject-id*. For this kind of tuples, the soft constraint states the preference of selecting a device located physically close to the object which should be referenced. Thus, devices close to the respective object obtain a higher soft score. The second soft constraint considers modalities like speech or those predicate-referencing modalities, which generate gestures mostly used to accompany speech acts. For such modalities, a device is preferred which is located near the user with whom the dialog system is interacting at the moment.

These are some basic criteria for a simple device selection which can be extended by more sophisticated ones which, for example, also include detailed device characteristics.

5.6.2. Modality Selection

Figure 5.12 shows the relevant class diagrams for the modality selection. The `PhraseComponent` class represents the planning entity consisting of a `PredicateElement` for



<<Java Class>>
ModalityRepresentation
-scorer: List<AbstractScorer>
-solvedDeviceRepresentation: DeviceRepresentation
-score: HardSoftScore
-phraseComponents
0..*
<<Java Class>>
PhraseComponent
-predicateElement: PredicateElement
-modalityRepresentationMap: Map<ModalityType,Double>
-attributiveObjectIdentifier: AttributiveObjectIdentifier
-speechOutputTypeList
0..*
-speechOutputType
0..1
-powerSetModalityList
0..*
-powerSetModality
0..1
<<Java Enumeration>>
SpeechOutputType
+ATTRIBUTIVE_IDENTIFIER: SpeechOutputType
+THIS_TYPE: SpeechOutputType
+THE_TYPE: SpeechOutputType
+THIS: SpeechOutputType
+IT: SpeechOutputType
<<Java Class>>
PowerSetModality
-modalitySet: Set<ModalityType>
<<Java Class>>
FusionScorer
+FusionScorer()
+calculateScore(ModalityRepresentation):Score
+getHardConstraint(ModalityRepresentation):int

Figure 5.12.: UML Class Diagrams for Modality Selection

which a set of suitable modalities (`PowerSetModality`) and a suitable `SpeechOutputType` should be selected. Furthermore, the `PhraseComponent` receives the result of the Attribute Selection Algorithm (`attributiveObjectIdentifier`), if available for the respective `PredicateElement`, and a map containing the assessment on how well each modality can represent the element (`modalityRepresentationMap`). The planning variables have their own classes: The `PowerSetModality` contains a set of the modality powerset, whereas the `SpeechOutputType` contains the respective types encoding the different available linguistic object descriptions. The `ModalityRepresentation` class represents a current assignment and contains the final solution of the planning problem. It has lists of `PhraseComponent`, `PowerSetModality` and `SpeechOutputType` instances and a `score` variable containing the score for the current assignment. Furthermore, it contains the result of the device selection (`solvedDeviceRepresentation`) to be able to use information of the selected devices in the modality planning process. Apart from that, it receives a list of `AbstractScorer`, which represent the different categories. This will be explained in more detail later. The `FusionScorer` is the main scorer in the MMF framework. In its `calculateScore()` method, the soft score calculation of all individual scorers is triggered and the weighted sum over all resulting scores as defined in section 4.6.2 is calculated. The hard score is calculated by the `getHardConstraint()` method. Currently, there are several hard constraints for the modality selection:

- Impossible modality combinations for a certain `PredicateElement` are excluded. For example, a combination containing the pointing modality is not allowed for a verb.
- Using the linguistic description “this” or “this_type” is only allowed in combination with the pointing modality.
- Selecting the empty set from the powerset is not valid if there exist at least one modality which can represent the corresponding `PredicateElement`.

In the following, the available individual scorers, which are used for calculating the soft score, will be presented.

Provided Scorer

Figure 5.13 shows the available scorers. Currently, six different individual scorers for calculating the soft score are provided by the framework. They all inherit from the abstract class `AbstractScorer`, which has two variables, namely `weight` and `maximalReducedScore`. The `weight` expresses the influence the respective scorer has on the overall soft score calculation and `maximalReducedScore` states how much each scorer can reduce the soft score at most. Note that all provided soft constraints decrease the soft score if they are not fulfilled. The `maximalReducedScore` variable can be used to make scorers, or the respective categories they present, equally important (see section 4.6.2 for further details). Figure 5.13 illustrates some details of the `GeneralHumanLikenessScorer` as representative for the other scorers. It has several variables containing the individual score for each soft constraint (e.g., `speechScore`, `pointingScore`), calculated in separate methods, like `calculateSpeechScore()`. Thus, a hierarchy of soft score calculation results: For each soft constraint the score is calculated separately. Then,

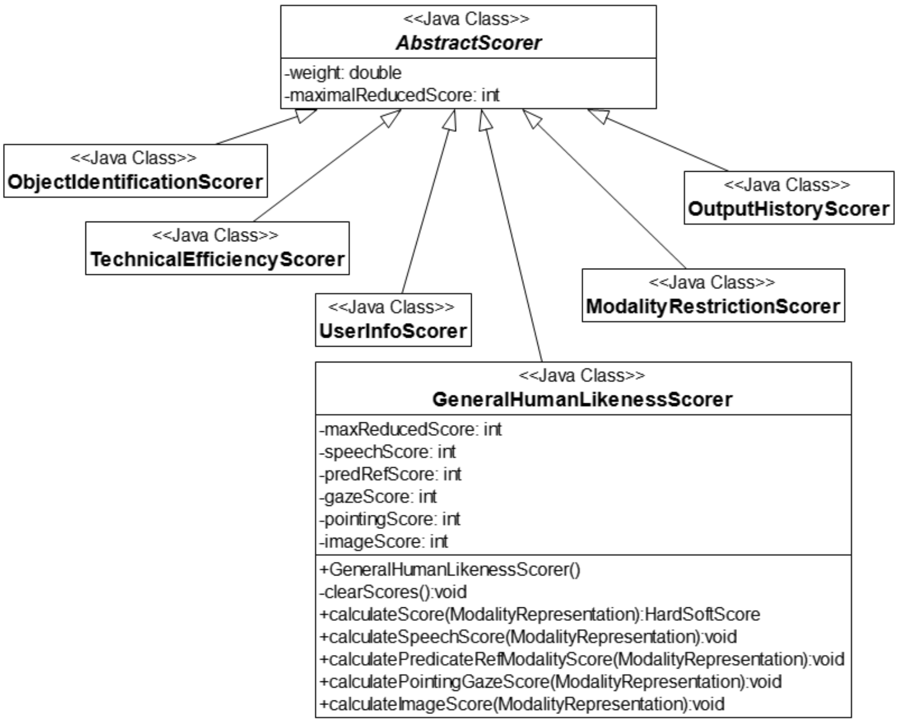

<<Java Class>>
AbstractScorer
-weight: double
-maximalReducedScore: int
<<Java Class>>
ObjectIdentificationScorer
<<Java Class>>
TechnicalEfficiencyScorer
<<Java Class>>
UserInfoScorer
<<Java Class>>
OutputHistoryScorer
<<Java Class>>
ModalityRestrictionScorer
<<Java Class>>
GeneralHumanLikenessScorer
-maxReducedScore: int
-speechScore: int
-predRefScore: int
-gazeScore: int
-pointingScore: int
-imageScore: int
+GeneralHumanLikenessScorer()
-clearScores():void
+calculateScore(ModalityRepresentation):HardSoftScore
+calculateSpeechScore(ModalityRepresentation):void
+calculatePredicateRefModalityScore(ModalityRepresentation):void
+calculatePointingGazeScore(ModalityRepresentation):void
+calculateImageScore(ModalityRepresentation):void

Figure 5.13.: Overview of the Scorers' UML Class Diagrams

each scorer calculates in its `calculateScore()` method the category's score out of the individual scores. This method can also use weights to allow one soft constraint to have more influence than another one. Finally, the `FusionScorer`, which has been presented in Figure 5.12, calculates the weighted sum over the resulting scores of each used scorer. It is up to the user of the framework to decide which of the available scorers should be used and how the weights should be set. Not all scorers are intended to be used all together and in some cases not even all constraints inside a scorer should be used at the same time. Furthermore, most of the available scorers in the MMF framework are tailored for the usage in the area of human-robot interaction.

The aim of the `GeneralHumanLikenessScorer` is to define some basic constraints which should result in a more human-like behavior of the robot. An informal description of the intentions behind the implemented constraints for this scorer are the following:

- Speech is the main source of information. Therefore, it is preferred to use speech whenever possible.
- Prefer looking at objects and persons while talking about them.
- Pointing at objects is very helpful to identify the object.
- Prefer looking at a certain user when referring to them instead of pointing at them.
- Prefer pointing at things rather than just looking at them, because pointing is more salient than looking.

- Waving, nodding and head-shaking enable more human-like behavior.
- It is helpful to display the image the robot is talking about if it is not displayed yet.

Note that the General Human-Likeness scorer does not impose any constraints regarding the linguistic object description.

Another scorer, the `ObjectIdentificationScorer`, fulfills the task of selecting an appropriate linguistic description and fitting pointing actions for identifying objects. The soft constraints contained in this scorer are two-folded. First, it is considered whether an attributive identifier has been found:

- It is preferred to use an attributive identifier for referencing objects if available.
- If no attributive identifier has been found, prefer using "the_type" and pointing for clarification.
- If only a partial identifier has been found, prefer using the partial attributive identifier in combination with pointing. Maybe pointing can make the reference unique.

Second, a list of objects which are on the way from the robot to referent has been calculated by the Cone Intersection Algorithm (see section 4.4.2). This list is used by the scorer to assess the usefulness of a pointing gesture and its accompanied linguistic descriptions:

- Using pointing is not helpful if the referent is very close to objects of the same type.
- Using "this" or "this_type" is not helpful if the referent is very close to objects of the same type.

Assume that a reference to an object which is very close to another one of the same type should be performed. Furthermore, assume that both, the `GeneralHumanLikenessScorer` and the `ObjectIdentificationScorer`, are used. Both scorers have conflicting soft constraints in this case: The `GeneralHumanLikenessScorer` states that pointing to objects is helpful to identify them, whereas the `ObjectIdentificationScorer` assesses pointing as not helpful in case that objects of the same type are very close to each other. Therefore, the two constraints cannot be both fulfilled and one outperforms the other. The constraint dominates which stronger decreases the score when not being fulfilled. If both constraints decrease the score by the same amount, the used constraint is selected randomly. In the example, it is preferable that the more specific constraint, stated in the `ObjectIdentificationScorer`, dominates the more general one in order to properly react on the given situation that two similar objects are very close to each other.

The `OutputHistoryScorer` makes use of the output history in order to be able to take previously made object references into account. This scorer considers the following soft constraints:

- If an object has been referenced in the previous sentence, pointing is not needed since the focus is already on the object.
- If only one object has been referenced in the previous sentence, “it” can be used to refer to the object once again.
- It is preferable to use “the_type” to refer to an object again if no other object of the same type appears in the sentence and if the referent appeared in the previous sentence as the only object or only together with objects of different types.
- If an image is already being displayed on a device, there is no need to display it twice.

The next presented scorer, the `UserInfoScorer`, considers information about users and their preferences for the modality selection. This scorer needs to be adapted to the current user. The provided soft constraints in this scorer can be seen as examples and are not intended to be all used together:

- If the current user has a preferred modality, it is good to use this modality whenever possible.
- If the user has any impairments, certain modalities should be avoided:
  - If the user has a strong visual impairment, gaze and pointing might not be very helpful.
  - If the user has an auditory impairment, speech could be difficult to understand, at least at the standard volume.
- If the user does not have very good language skills in the language used for communication, avoid using speech as the only modality.
- If the user likes a verbose output, prefer using the speech modality and the attributive identifier to reference objects whenever possible.

A further scorer, called `TechnicalEfficiencyScorer`, has been designed to consider criteria like the fastness of the output. Such criteria require knowledge about the used devices, which can be provided since for each modality a corresponding device has already been selected. For each modality combination, the respective devices receive the potential output and estimate its duration. For the speech modality the currently selected linguistic description, in case of an object reference, is used. The duration information is device dependent and needs to be provided by the user of the framework. The slowest output determines the speed of the selected combination. The soft constraint assigns a negative score if the average speed for outputting the current `PredicateElement` is below a certain threshold.

The last available scorer, the `ModalityRestrictionScorer`, is used for restricting certain modality usage. The provided soft constraints of this scorer are some examples for testing such restrictions and are not intended to be used together:

- Prefer using as many different modalities as possible.

- Prefer certain modality whenever possible.
- The usage of a specific modality should not exceed certain limit in a sentence.
- Several consecutive pointing gestures should be avoided.

The available scorers can be used out of the box and new ones can be added easily. A combination of the `GeneralHumanLikenessScorer`, the `ObjectIdentificationScorer` and the `OutputHistoryScorer` has been used in the three example scenarios, which will be presented in more detail in the following section. These example scenarios show that this combination of scorers yields good results in the area of human-robot interaction. The user study, which has been conducted for evaluating the created fission framework, also supports this claim. The results of the study can be found in section 6.4.

5.7. Example Scenarios

In order to understand more easily which components need to be supplied when using the MMF framework, three example scenarios, which are provided alongside the framework, have been implemented. Usually, the framework receives the predicates from a dialog manager. Since it is not within the scope of this thesis to connect the framework to an existing dialog manager, the predicates used in the examples are directly given to the framework as input. A short overview of the represented scenarios and the sample configurations is given in the following.

5.7.1. Craft Task Scenario

In this scenario, a robot should assist the user with a craft task. Various tools are available, like a hammer, scissors and sponges, which are located near the robot and the user. The robot tells the user which tools to use next. Information about the objects, the robot and the respective user are stored in a MongoDB database. Furthermore, speech, pointing, gaze and nodding/headshaking modalities are used in the scenario. One speech, two pointing, one gaze and one nodding/headshaking device models are available for the respective modalities. The device models execute code that trigger a connected Nao robot[10] to perform the generated framework output. Thus, the physical devices, to which the device models map, include a speaker, the robot's arms and its head. In the provided example, eight predicates are given as input out of which sentences and phrases are generated. No scorer is set in the example. Therefore, the `GeneralHumanLikenessScorer` is used by default. The default settings are also used for the Attribute Selection Algorithm. They consist of using a possibly short combination of attributes with a saliency value above a threshold of 0.8.

5.7.2. Vase Selling Scenario

In the vase selling scenario, a robot presents vases to customers. There is a shelf with five vases varying in color, material, size, origin and position. The robot can greet the customer, can provide information about the different vases, including displaying images of vases currently not available, or it can negotiate prices. The information about the entities in the environment are stored in an OWL ontology. In this scenario, speech, pointing, gaze, image and waving modalities are available. The respective devices include two speech, three pointing, one gaze, one image and one waving device. For this example scenario, the device models are not mapped to physical devices. Therefore, each device outputs the respective information in text form on the computer screen and the image device is additionally able to display images on the screen. However, physical devices can easily be added to this scenario. A variety of scorers are used, namely the `UserInfoScorer`, the `GeneralHumanLikenessScorer`, the `OutputHistoryScorer`, and the `ObjectIdentificationScorer`. Furthermore, for the final selection of attributes in the Attribute Selection Algorithm, the shortest combination with a saliency value above 0.9 should be used. Phrases, questions and statements are generated for 12 example predicate inputs in this example.

[10] Nao robot: `https://www.ald.softbankrobotics.com/en/cool-robots/nao` (last accessed: 19th of May 2017)

5.7.3. Messy Working Place Scenario

This scenario represents the situation in which an employee is sitting in front of their messy working place. There are cups, plates, scissors, paper and various pens lying on the desk. The robot helps the person to locate the required objects in their chaos or helps them with tidying by requesting them to put an objects at a certain position or recommends them to use specific objects for certain tasks. This setting has been used for the user study, about which detailed information can be found in section 6. A MongoDB database is used to store all required information. The number of available modalities is limited to speech, pointing and gaze, for which respective device models have been created. These are mapped to the corresponding components of a Nao robot, which is able to execute the output. The default configurations for the Attribute Selection Algorithm and three scorers are used, namely, the `GeneralHumanLikenessScorer`, the `OutputHistoryScorer`, and the `ObjectIdentificationScorer`. Furthermore, a total of 14 input predicates is provided, from which questions, statements and commands are generated.

6. User Study

A user study has been conducted with the goal to answer the following questions:

- Does the multimodal output generated by the MMF framework enable a better understanding compared to using speech-only output?
- Are the generated object references helpful to identify the correct object easily?
- How human-like does the output of the framework appear to the robot's interaction partners?

6.1. Study Design

An office environment has been chosen for the user study. Some information about the scenario has already been given in section 5.7.3. The study participant sits at a desk with 33 objects on it, including different types of pens, scissors, paper, cups and plates. All objects have been given labels with the unique ids which are used inside the framework to identify these objects. A Nao robot is placed on the opposite site of the participant. This robot has been chosen for the study, because it is small enough to be placed on the table and is able to speak, point to objects and look at them. Thus, the speech, pointing and gaze modalities, which are used in this study, can be expressed. An overview of the setup can be seen in Figure 6.1.

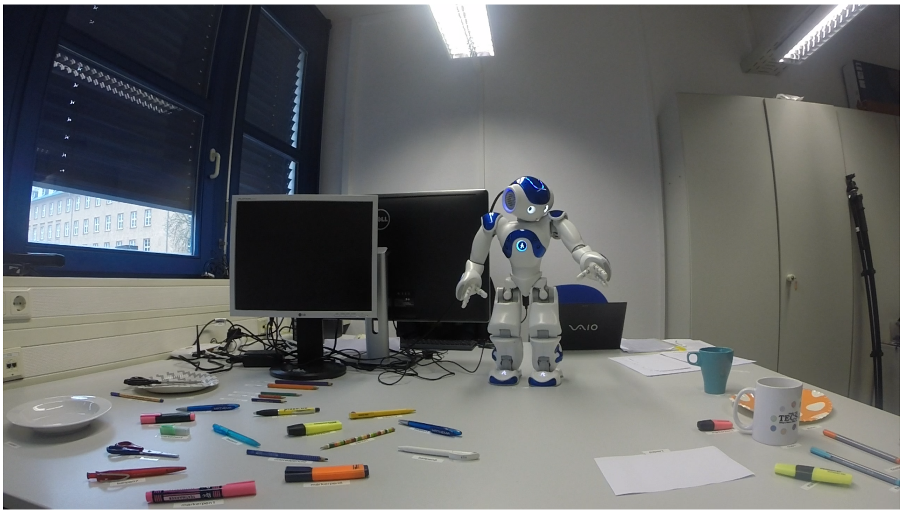


Figure 6.1.: Overview of the Study's Setup

The study has been divided into two parts. In the first part, the individual study participants have been shown a short sentence on the screen containing photos of objects lying on the desk. Examples can be seen in Figure 6.2. The participants have been

Figure 6.2.: Example Sentences

asked to read the sentence and to substitute the images with a description of the object, in such a way that it is clear which object they refer to. However, they were not allowed to use the object's id written on the labels. The goal of this was to make the participants refer to the objects as they would do in a normal conversation. In the second part of the study, several input predicates have been processed after each other by the MMF framework and the corresponding outputs have been executed by the Nao robot. The generated sentences consist of questions to the participants and of requests for performing certain tasks with the objects on the desk. If the study participants have not been able to understand Nao, it has been possible to repeat the output up to three times.

The participants have been divided into four different groups: two *speech-only* groups (A12, A21) and two *multimodal* groups (B12, B21). People in a multimodal group were allowed to use pointing gestures to refer to objects in the first part of the study, whereas the members of the speech-only groups were asked to only give a verbal description of the objects. However, members of both groups were allowed to look at the object they are talking about, because it would have been difficult to prevent such a natural behavior in the speech-only group. In the second part of the study, Nao used speech, pointing and gaze for the participants in the multimodal groups to refer to objects on the desk, whereas people in the speech-only groups only heard its voice. Additionally, Nao stayed in its position for both groups. Table 6.1 and Table 6.2 show the predicate inputs to the framework and the sentences the framework generated out of this input using speech, gaze and pointing. These sentences are executed by Nao for the multimodal groups in the second part of the study, where group B21 had to react to the

Predicate Input	Generated Sentence
[*xu*] **lamji**(*markerpen3, cup1,* [*zoe*], [*zoe*])	Is markerpen *(pointing, gaze)* with size small, with color pink next to *(pointing, gaze)* cup with color white?
[*xu*] **viska**(*you, coloredpencil1,* [*zoe*])	Do you see *(pointing, gaze)* colored pencil with color purple, with size small?
[*ko*] **punji**([*zoe*], *coloredpencil1, into cup1*)	Put it *(gaze)* into cup with color white.
[*ko*] **punji**([*zoe*], *ballpen2, into cup1*)	Put ballpen *(pointing, gaze)* with approximate position on your far left, with label mathema *(gaze)* into the cup.
[*xu*] **zmanei**(*you, plate1, plate3*, [*zoe*], [*zoe*])	Do you prefer *(pointing, gaze)* plate with color orange over *(pointing, gaze)* plate with color white?
[*ko*] **punji**([*zoe*], *markerpen4, into cup2*)	Put *(pointing, gaze)* markerpen with color green into *(pointing, gaze)* cup with color blue.
[*ko*] **pilno**([*zoe*], *scissors1, to cut paper1*)	Use *(pointing, gaze)* scissors with size big to cut *(pointing, gaze)* the paper.

Table 6.1.: Predicate Set 1

Predicate Input	Generated Sentence
[*xu*] **bramau**(*scissors2, scissors1*, [*zoe*], [*zoe*])	Is *(pointing, gaze)* scissors with color blue and red bigger than *(pointing, gaze)* scissors with color black?
[*xu*] **skari**(*markerPen2, yellow*, [*zoe*], [*zoe*])	Does *(pointing, gaze)* markerpen with label stabilo, with size big have color yellow?
[*ko*] **pilno**([*zoe*], m*arkerPen2, to draw a line on paper1*)	Use it *(gaze)* to (gaze) draw a line on the paper.
[*ko*] **pilno**([*zoe*], *coloredpencil2, to draw a line on paper1*)	Use *(pointing, gaze)* colored pencil with color blue to draw a line on the paper.
lamji([*ma*], *plate2*, [*zoe*], [*zoe*])	What is next to *(pointing, gaze)* plate with color grey white?
[*ko*] **pilno**([*zoe*], *markerPen1, to draw a line on paper1*)	Use markerpen *(pointing, gaze)* with color pink, with label Rex textmarker *(pointing, gaze)* to draw a line on the paper.
[*ko*] **dunda**([*zoe*], *ballpen3, person1*)	Give person with name Magdalena ballpen *(pointing, gaze)* with color yellow, with label pizza.

Table 6.2.: Predicate Set 2

seven sentences of *predicate set 1* (Table 6.1) and B12 on the ones of *predicate set 2* (Table 6.2). Group A12 (A21) receives the same sentences as B12 (B21) in the second part. However, Nao does not perform multimodal actions but uses only speech. Thus, the second number in the group name, "2" in the case of A1**2** and B1**2**, refers to the used predicate set (predicate set **2**) in the second part. The same is true for the groups A21 and B21. Group A**1**2 and B**1**2 had to read these seven sentences presented with pictures on the screen which correspond to the generated sentences of predicate set **1**, indicated by the first number in their group names, "1" in this case. These sentences on the screen correspond to the generated ones presented to the other groups, A21 and B21, in the second part of the study. Hence, the references created by participants in the groups A12 and B12 in the first part of the study can then be compared to the sentences generated by the framework and presented to the participants in groups A21 and B21 in the second part, and vice versa.

6.2. Participants

52 participants were recruited for the study (24 females and 28 males) - their age ranging from 19 to 39 (*mean*= 25.69, *standard derivation* = 3.93) - from the Saarland University campus. There were 33 Computer Scientists/students in related fields (Media Informatics, Bioinformatics, etc.) and 19 students studying an unrelated subject. The participants were randomly assigned to the individual groups while making sure that the groups were balanced in their size (6 females and 7 males in each group). Among the participants, 17 have worked with a robot before but not in a context similar to this study.

6.3. Measures

In order to find out whether the generated multimodality is helpful, the performance of the participants in the multimodal groups in the second part of the study has been compared with the one of the speech-only groups. For this purpose, the study included three objective measures, namely the number of tries, the number of incorrectly accomplished tasks and the number of not completed tasks after three failed tries. The results will be reported in section 6.4.2. At the end of the study, the participants have been asked to fill in a questionnaire in order to reflect on the study. In the questionnaire, the participants should state how much they agree, or disagree with several statements. A five-level Likert scale [33] has been used, in which 1 indicated total disagreement and 5 total agreement. Additionally, several free text questions have been asked in order to get a more precise idea of the persons' opinions. There have been two different questionnaires: one for the speech-only groups and one for the multimodal groups, which differ in a few specific questions. For instance, the multimodal groups have been asked how helpful pointing and the robot's gaze have been in order to retrieve the correct object, whereas the speech-only groups have been asked what Nao could do, apart from talking, to enable a better and faster understanding. Note that the participants from the speech-only groups were not aware that there have been multimodal groups. There were also some general questions for both groups, like how difficult was it to perform the tasks and how much they have liked the interaction with Nao. There have been 13 questions in total for the multimodal groups and 11 for the speech-only groups. The analysis of the questionnaires regarding the helpfulness of multimodality can also be found in section 6.4.2. The entire questionnaires can be found in the appendix in section A.2.

The aim of the first part of the study was to collect data on how humans describe certain objects. The sentences used in the first part for the groups A12 and B12 are based on the same predicates that are used as input for the framework in the second part for the two other groups A21 and B21 and vice versa. As a result, it is possible to compare the human utterances with the output produced by the framework. It should be evaluated whether using salient properties is an adequate way of producing a suitable object description. The results of this comparison are reported in section 6.4.3. Furthermore, two questions regarding the content of the speech output and the used attributes have been asked in the questionnaires. In this way, the subjective assessments of the participants have also been included. Additionally, questions considering how human-like the generated output appeared have also been included (see section 6.4.4).

In order to retrieve the objective measures and the human object description, video material has been recorded throughout the study. Unfortunately, 7 recordings are not available for the evaluation due to technical errors. In total, 7h 7min 49s of video material has been evaluated.

6.4. Results of the Study

In the following, the results of the study will be presented. The first section will briefly discuss problems occurred during the study which can be traced back to the usage of a Nao robot. After that, each section presents the results relevant for answering one of the central questions that has been asked in the beginning. Detailed material of the evaluation of the user study can be found in the appendix in section A.3.

6.4.1. Nao Specific Problems

Before the study had been conducted, it has been assumed that the speech generated by Nao's built-in text-to-speech engine might be difficult to understand when not being used to it. However, this might not necessarily be seen as a problem but even as an advantage: It could make the task a bit more difficult and thus, if the used multimodality is helpful, this result might emerge more clearly. The question whether there have been any problems in understanding Nao has been included into the questionnaire in order to be able to consider this information when evaluating the answers to the other questions. As an example, the attributes provided to describe an object might also not be helpful if they are not understood acoustically. For the study, Nao's standard volume has been slightly increased to a value of 70 and Nao's standard talking speed has been decreased to 70. In the following, the problems the participants had when interaction with Nao will be presented. However, since they are Nao specific problems and are not helpful in answering the goal questions of the study, they will not be evaluated further. Nevertheless, they may give some insights for further studies conducted with a Nao robot.

33 participants (more than 60%) stated that they found Nao's voice and pronunciation difficult to understand or at least had problems in understanding some words. Some of them recommended to let Nao talk more slowly and to use a different text-to-speech engine in order to receive a clearer voice. However, one person mentioned that the talking speed has already been quite slow for him. Other people suggested introducing more pauses in the sentence, although this could sound more unnatural. Some people have already said that there have been pauses between some words that has been too long and irritating. Furthermore, four participants in the multimodal groups mentioned that they could not understand Nao well due to the noise it made while moving. Therefore, they suggested increasing the volume of its speech.

6.4.2. Quality and Usefulness of Multimodal Output

This section evaluates the answers regarding the central question of the study whether the multimodal output generated by the MMF framework enables a better understanding compared to only having a speech output. On the one hand, the usefulness of the pointing and the gaze, respectively, will be evaluated by considering the assessment of the participants in the questionnaires. On the other hand, it will be investigated to which extent the objective measures contribute to answer this question.

Assessment of Multimodality in the Questionnaires

In the following, the distribution of answers for the two questions, whether the participants liked interacting with Nao and whether the tasks given by Nao were hard to understand, will be investigated. Figure 6.3 presents the answers of the speech-only group A12 and the multimodal group B12, while Figure 6.4 presents the results of the groups A21 and B21, respectively. In group A12, six people totally agreed (giving 5 out of 5 points) and five agreed (giving 4 points) with the statement "I liked interacting with Nao", whereas all people in group B12 agreed (giving 4 or 5 points), 10 of them totally (5 points). Since the number of participants in each group is small, the differences in the distributions of answers between the groups A12 and B12 could have also happened by chance. In order to find out whether this difference is statistically significant, a *t-test* has been conducted. This statistical hypothesis test is used to verify or reject the null hypothesis, which states that the means of the two distributions are equal. If the means are equal, it can be concluded that the differences are not statistically significant and can be traced back to chance. In this case, a t-test with significance level $\alpha = 0.10$ has been performed. The significance level is the probability of rejecting the null hypothesis in the case it is true. Thus, when using $\alpha = 0.10$, there is a risk of 10% concluding that a difference exists when there is no actual difference. The results of the t-test are the following: A12: $mean = 4.23$, $variance = 0.86$, B12: $mean = 4.77$, $variance = 0.19$, $t\text{-}value = -1.89$, $p\text{-}value = 0.075$. Since the calculated p-value is smaller than the chosen significance level α, the null hypothesis can be rejected and the difference in the means of both groups is statistically significant with 90% probability.

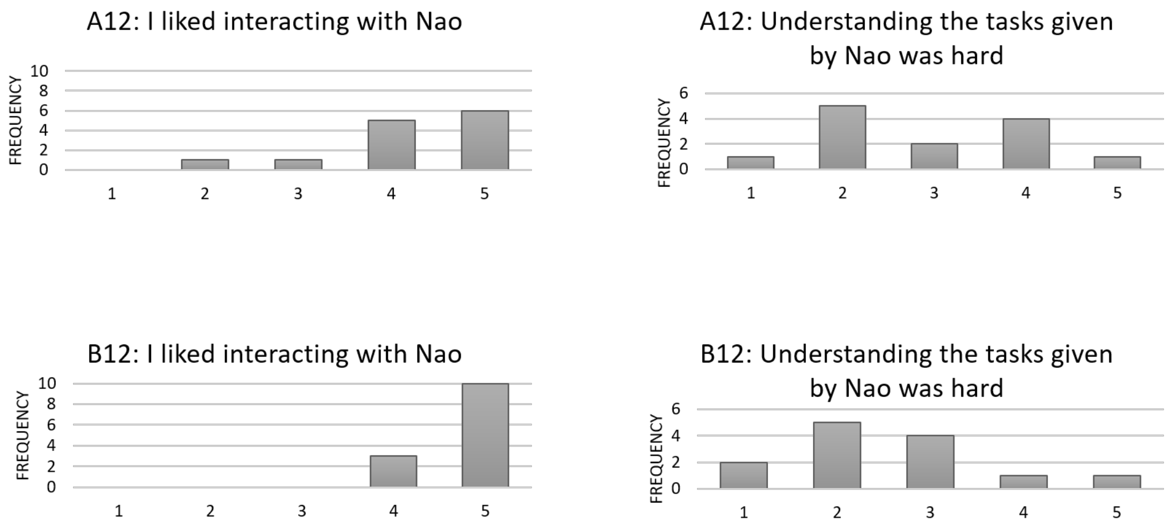


Figure 6.3.: Distributions of Answers for two Questions for the Speech-only Group A12 and for the Multimodal Group B12

Conducting another t-test revealed that the differences in the means between group A12 and group B12 for the statement "Understanding the tasks given by Nao was hard" is not statistically significant. Nevertheless, the figures show small differences: Five people from group A12 considered understanding the tasks as hard (giving 4 or 5 points), whereas in group B12, only two people had this opinion. Furthermore, the differences

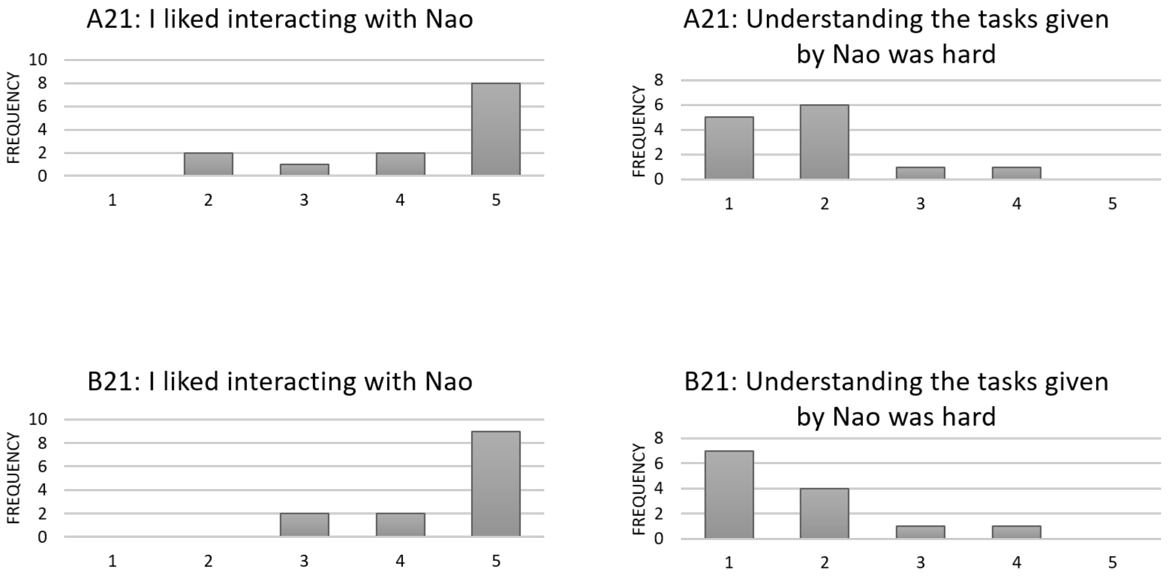


Figure 6.4.: Distributions of Answers for two Questions for the Speech-only Group A21 and for the Multimodal Group B21

in the means between group A21 and group B21 have not been found as statistically significant for both questions. Again, slight differences can be seen in the figures: Eight people of group A21 totally agreed that they liked interacting with Nao. Yet, two people from this group rated the statement with only 2 points. In contrast, nine people from group B21 enjoyed working with Nao and nobody gave less than 3 points. Furthermore, five participants in group A21 totally disagreed with understanding the tasks given by Nao has been hard, whereas in group B21 even seven people considered them as very easy.

It is interesting to note that participants in the A21/B21 groups found that the tasks are easier to understand than people in the groups A12/B12: Only two people agreed in A21/B21 (giving 4 points), compared to seven who agreed in the A12/B12 groups (giving 4 or 5 points). It seems that the sentences generated from *predicate set 2* have been more difficult to understand.

Furthermore, regarding the question whether participants had any problems in understanding the tasks, several people in the speech-only groups have been overstrained in listening and finding the correct object at the same time. For example, one person said: “Simply grasping what he meant while looking at these many objects was a little difficult.”, whereas one person in the multimodal group said: “Sometimes the objects looked the same, but the pointing helped”. Another person of a multimodal group stated that he had only “acoustical problems”, whereas the gestures and tasks were clearly understandable for him. Another opinion was that the pointing helped to find the object faster but that they have not been very precise.

The multimodal groups have been asked how helpful Nao’s pointing gestures and its gaze have been in order to understand which object has been referenced. There have been no major differences between both multimodal groups, which means that the respective sentences they heard in this scenario had little influence. The joint results for both groups can be seen in Figure 6.5. Ten people found the pointing gestures as absolutely helpful (giving 5 out of 5 points), nine people found them helpful (giving 4 points), five people assessed the helpfulness with 3 points and two with 2 points. Thus, more than 70% said that pointing was helpful. Furthermore, nine people found gaze absolutely helpful for finding the correct object (giving 5 points), 10 people found it also helpful (giving 4), four people gave 3 points (among them one person stating that she has not actually considered the gaze), and three people assigned 1 or 2 points. Again, more than 70% agreed that gaze was helpful. The figures suggest that pointing is seen slightly more helpful than gaze. When people found that pointing or gaze has not been useful, they have additionally been asked whether they assessed pointing or gaze as distracting. Two people said that pointing has been absolutely distracting for them (giving 5 points). One of them also stated that gaze has been absolutely distracting (giving 5 points), whereas the other person considered gaze, in contrast to pointing, as absolutely helpful. There has been a third person assessing both gaze and pointing as distracting (giving 4 points). One of these persons stated that she was rather focusing on Nao's speech and therefore did not paid too much attention to its gestures. This could also be a possible explanation for the assessment of the other two people.

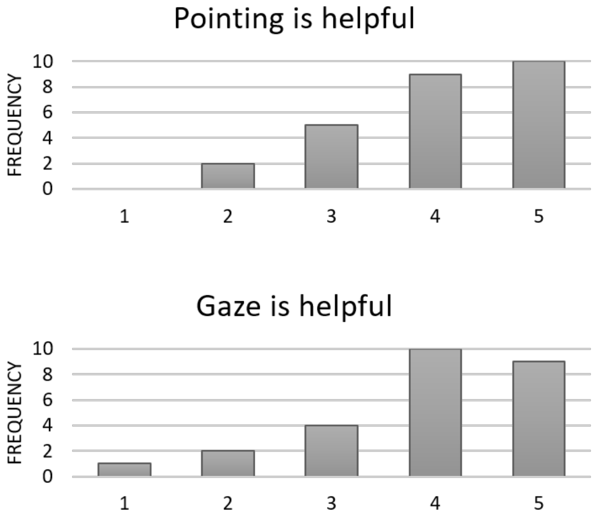


Figure 6.5.: Assessment of Helpfulness of Pointing and Gaze in both Multimodal Groups

The participants in the speech-only groups have been asked what Nao could do, apart from talking, such that it would be easier to understand which object has been referenced. They have not been informed about the existence of multimodal groups. Nevertheless, around 65% of the participants in both groups (nine people in A12 and eight in A21) suggested pointing at the object. Four people also mentioned looking at the object. For example, one person said: "Pointing at the object, maybe even looking in the direction of the object". Another one stated: "I think if Nao knows about the object and turns his head while pointing, it would be more intuitive". One person wanted to have descriptions of the things next to the referenced object. Furthermore, several people stated that positional information with respect to nearby objects and viewer centered position information, like “to my left or right”, would be helpful. Another idea was to put objects to different sides and refer to a left and a right pile. Apart from that, there have been some further ideas involving other modalities: displaying a picture of the respective object on the screen, making gestures (e.g., for drawing) or using the hands to show the dimensions of an object.

Several people in the speech-only groups stated that Nao’s motionlessness has irritated them. For example, one person said that interacting with a robot that does not move or show any reactions (like a human would do) is strange.

Objective Measures for the Assessment of Multimodality

In the following, the number of tries, the number of tasks which were not completed after three tries and the number of incorrectly performed task in the different groups will be investigated. The total number of tries for all sentences and for all participants of group A12 was 140, whereas group B12 only needed 120 tries in total. Group A21 had a total of 117 tries and group B21 of 103 tries. Thus, it can be seen that the multimodal groups needed less tries. Additionally, as already observed previously, *predicate set 2* seems to be more difficult, since participants in A12 and B12 needed more total tries than their colleagues in the two other groups.

Furthermore, the mean of tries for every single sentence has been compared among the groups. When performing a t-test with significance level $\alpha = 0.05$, the differences in the means of four sentences are significant. The exact test results for these four sentences can be seen in Table 6.3. The means differ the most between group A12 and B12 for sentence 3 (“Use it to draw a line on the paper.”). Therefore, Figure 6.6 presents the number of tries needed in both groups for this sentence. In group A12, eight people needed three tries, two persons needed two tries and one person needed one try, whereas in group B12 the result is the other way around: three people needed one try, two persons needed two tries and only one person had three tries. The “it” used in sentence 3 might be seen as slightly confusing, because most people potentially have not expected backward references. However, with Nao looking at the object might have made it clearer for the participants of group B12 and thus, they needed less tries.

	Group A12		Group B12			
sentence	mean	var	mean	var	t-value	p-value
3	2.64	0.45	1.36	0.45	4.43	0.000026
7	2.63	0.45	1.55	0.67	3.41	0.0030

	Group A21		Group B21			
sentence	mean	var	mean	var	t-value	p-value
2	1	0	1.63	0.65	-2.61	0.026
3	1.58	0.45	1.09	0.09	2.31	0.035

Table 6.3.: Statistically Significant T-test Results for Comparison of the Means of the Number of Tries for four Sentences between the Multimodal and the Speech-only Groups

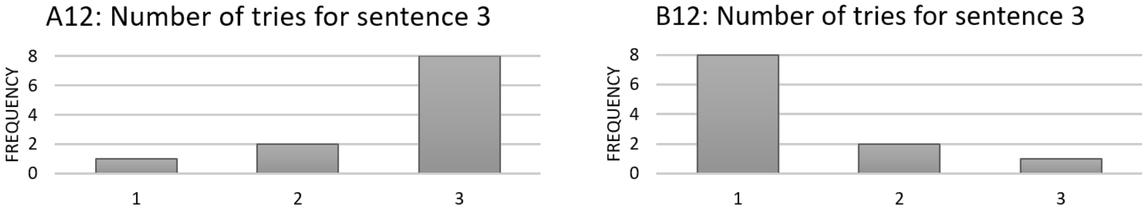


Figure 6.6.: Number of Tries in Groups A12 and B12 for Understanding Sentence 3

The distribution of the number of tries between the groups A12 and B12 for sentence 7 looks similar. For the two other groups, the differences are less significant (smaller p-value). However, the speech-only group A21 performed better than group B21 for sentence 2 ("Do you see colored pencil with color purple with size small?"). Yet, the video material shows that most of the people having problems with this sentence in group B21 have identified the correct pen but have not understood the "Do you see" at the beginning of the sentence. Therefore, the problem seems to be an acoustical one and is likely unrelated to the usage of multimodality.

Regarding the number of not completed tasks after three tries, the differences between the groups A21 and B21 are marginal. The multimodal group B21 performed slightly better. In contrast, the other two groups had a lot more uncompleted tasks, which once again indicates that *predicate set 2* has been more difficult. Therefore, the differences between the groups become more obvious. In total, group A12 had 18 uncompleted tasks, whereas group B12 had 11. Figure 6.7 shows the distribution of uncompleted tasks among participants for both groups. Only one person in group A12 had managed to perform all tasks with at most three tries, whereas in group B12 at least four people were able to do this. A t-test with significance level $\alpha = 0.1$ reveals that the difference in the means of the number of uncompleted tasks is significant between group A12 and group B12. On average, participants of the multimodal group were not able to complete one task, whereas people in the speech-only group could not complete on average 1.63 tasks. Furthermore, the sentences which mostly resulted in uncompleted tasks are also those for which the participants needed the most tries on average. In group A12, sentences 3, 5 and 7 caused the most problems and in group B12 this is true for the sentences 5 and 7. Yet, only two people were not able to understand sentence 7 in group B12, whereas five people from group A12 had difficulties with it. Furthermore, six people could not successfully complete the task described in sentence 3 in the speech-only group, whereas only one person had a problem with this sentence in the multimodal group.

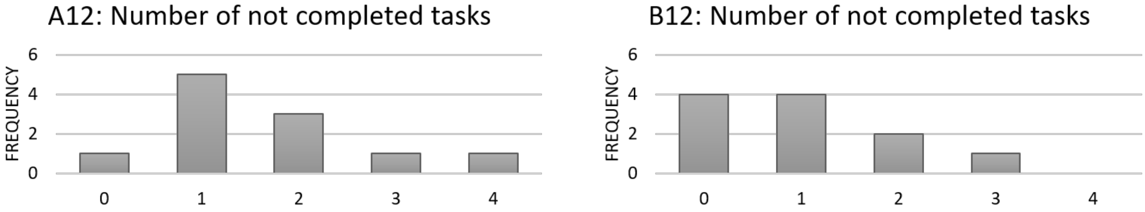


Figure 6.7.: Number of Uncompleted Tasks in Groups A12 and B12

The last objective measure which will be evaluated is the number of incorrectly performed tasks. In group A12, fewer tasks have been performed incorrectly than in any other group. One explanation could be that they asked for the sentence to be repeated more often and therefore did not make so many mistakes. It is interesting to note that in groups A21 and B21, more tasks have been executed incorrectly, even though it has been supposed based on the previous findings that these tasks have been easier to understand. In group A21, seven tasks in total have been performed incorrectly, five of them due to sentence 4 ("Put ballpen with approximate position on your far left, with label mathema, into the cup."). The participants in group A21 had problems with identifying the correct pen, which has not been easy since two pens with the same label have been on the desk. This problem did not occur in the multimodal group: combining the verbal description "on the far left" with a pointing gesture seemed to help a lot to identify the correct pen. In group B21, four tasks were performed incorrectly. In two cases, the green colored pencil was chosen instead of the green markerpen. This might have happened that due to an inexact pointing gesture.

In conclusion, a great majority considered pointing and gaze as helpful in the questionnaires. Only very few people had problems with them and considered them as distracting. Most people in the speech-only groups mentioned that pointing would have simplified the tasks. Furthermore, multimodality slightly increased the positive assessment of the interaction, whereas Nao's motionlessness seems to be irritating otherwise. Several tries were often needed in order to understand the content of a sentence, which is mainly caused due to the difficulty in understanding Nao's generated speech. Nevertheless, comparing the number of tries and the numbers of uncompleted tasks between the groups shows that multimodality can reduce these numbers, especially for difficult sentences. The comparison of the number of incorrectly performed tasks also shows that pointing may help in performing the tasks correctly. However, it also indicates that careful listening is required since speech is the main source of information which cannot be replaced by only relying on not necessarily precise pointing gestures.

6.4.3. Object Referencing

This section answers the central question, asked in the beginning, whether the generated object references are helpful enough for identifying the referenced objects easily. In order to do this, the attributes used in the generated descriptions are compared to the ones chosen by humans in the first part of the study. Furthermore, the questionnaires are analyzed regarding the participants' assessment of the attributes and the generated speech output in general.

Comparison with Human Attribute Choice

For the evaluation of the first part of the study, the attributes used by humans to refer to the objects have been categorized. Table 6.4 shows the different categories and their meaning. Examples for the *History* category would be: "it", "the same paper", "the cup about which I talked before". Examples for relative locations are: "the blue pen next to the purple pen" or "scissors on the plate". Furthermore, "The blue pen which is to my left" or "the cup in front of you" are examples for viewer-centered locations and "the

Category	Meaning
C	**C**olor
S	**S**ize
L	**L**abel
M	**M**otif
Mat	**Mat**erial
T	**T**ype
H	**H**istory (indication for repeated reference)
relLoc	**rel**ative **Loc**ation to other objects
vLoc	**v**iewer-centered **Loc**ation
absLoc	**abs**olute **Loc**ation in the room
Comb	**Comb**ination of two objects

Table 6.4.: Categories of Attributes

pen pointing to the window" or "the blue pen in the middle of the desk" are examples for absolute locations. In the study scenario, a number of people named the grey-white paper plate "Plate with the scissors". This has been categorized as *Combination*. Apart from the categories defined in the table, there has been a small number of other properties that some participants have used to describe an object, like "the damaged pencil" or "the soup plate". Since these properties are very specific, they were not further considered. Apart from that, some object descriptions have not been considered in the evaluation since they were not sufficient to be able to distinguish the referred object from all others (e.g. "the pink markerpen" was not a sufficient description since there have been three pink markerpens on the desk). Object descriptions given in combination with a pointing gesture will be ignored in the evaluation provided in the following paragraph, but will later be considered separately.

Figure 6.11 shows the chosen attribute categories for three objects, namely the cup in sentence 1 and scissors in sentence 7 of *predicate set 1* and the plate in sentence 5 of *predicate set 2*. Furthermore, it shows how many participants have chosen each category

Cup in S. 1, PS. 1

Category	Times
C	7
L	3
M	3
C+M	1
C+L	1

Nao: C

Scissors in S. 7, PS. 1

Category	Times
rel.Loc	6
C	5
S	1
v.Loc	1
C+rel.Loc	4
C+S	2

Nao: S

Plate in S. 5, PS. 2

Category	Times
C	10
Comb	3
C+Comb	5
Comb+S	1
C+Mat	1
C+rel.Loc	1

Nao: C

Figure 6.11.: Examples for Chosen Attributes

as well as the category chosen by the MMF framework and outputted by Nao. The first table from the left indicates that most participants (7 out of 15) used color to refer to the cup, but also the label and the motif are used several times (both 3 times), whereas a combination is used only by two people. The framework also used the color attribute to refer to the cup. In the case of the scissors, the output of the MMF framework differs from the choice the majority of the participants made. The location on the plate (used by 6 out of 19 people), the color (used by 5 people) and a combination of both (used by 4 people) seem to be more salient than the size of the scissors generated as output by the framework. The third table presents the resulting distribution for referencing the plate. In this case, the category of the attributes used by most participants (10 out of 21) agrees with the one found in the generated output of the framework. However, for most of the people, the scissors, or rather the combination of scissors and plate, have been more salient than the plate itself. The generated sentence by the framework is "What is next to plate with color grey white?" (see Figure 6.2, Sentence 5), whereas most people asked "What is next to the black scissors?" and "What is next to the black scissors on the plate".

In general, the 14 sentences contained 24 objects which should be referenced. For 17 objects, the categories of attributes chosen by the framework and the categories chosen by the majority of the participants match. Apart from the different categories chosen for the scissors, which have already been presented, the other differences are discussed in the following. In two cases, the majority of the participants used a description which was not precise enough to differentiate the requested object from all others. In three other cases, the History category has been chosen by the framework, because the respective object has already been referenced in the previous sentence, whereas the majority of the participants used the same attributes once again. However, due to the manner of the task, the participants probably regarded each sentence as an independent entity. More people might tend to refer back to previously referenced objects by saying "it" or something similar in a normal conversation. It is noticeable that, in general, color seems to be the most salient attribute which is used by the majority of the participants whenever possible. However, when referring to a yellow pen that has the label "pizza.de" written in big letters on it, the label seems to be by far the most salient attribute in this case. Therefore, most people (15 out of 19) used the label instead of the color to refer to the pen. Since labels are rated less salient than color by the framework and color-only is not considered since there is another yellow pen nearby, the framework uses both, the color and the label, to describe the pen. However, this shows that the salience of attributes can vary depending on the object (e.g., due to the size of the label or whether a certain material can be determined). Currently, the saliency annotations used in the framework does not consider different salience among objects but gives a estimation on how salient a property might be in general. Nevertheless, this approach seems to be sufficient in most cases. Furthermore, considering saliency values for each property individually might result in hardcoding the optimal properties of each object.

It has been assumed before conducting the study that people tend to be more verbose compared to the generated output. However, the results regarding the length cannot support this claim. The number of different attribute categories for one object generated by the framework agrees most of the time with the number that the majority of people has used. In three cases, the length differs. In one case, the output of the framework is shorter since the History category is used. In two other cases, the framework uses two attributes, whereas the majority of the people used only one. Both cases have already been discussed: To refer to the ballpen with label “pizza.de”, the framework combines its color and its label, whereas the participants mainly use the label only. Apart from that, the yellow markerpen in sentence 2 of *predicate set 2* (see Figure 6.2) is defined only via the label by most people, which is not enough to differentiate it from all other markerpens. In general, the generated output does not exceed an attribute length of two for the example scenario, whereas 10 times people also combined three different attributes.

In the following, the pointing behavior in the multimodal groups will be evaluated. Five out of 11 people used pointing in group B12. Three of them pointed very frequently. In group B21, six out of 10 people used pointing. Among them, two pointed very frequently and two frequently. Additionally, female participants performed pointing less often (four female and seven male pointing participants) and used it less frequently than the male participants. Furthermore, there have been 55 pointing gestures in total among both groups. Most times (29 times), people said “this” + the object’s type while pointing. Another 15 times “the/this” + color + type is used. Sometimes, also size or motif are part of the accompanied utterance. Pointing happens mostly at the same time while saying “this”. Furthermore, since in sentence 2 of *predicate set 2* (see Figure 6.2) the color attribute cannot be used, five out of nine people decided to perform a pointing gesture instead of giving a more complex description of the object. In general, it seems that pointing is used more frequently when there are several objects in the sentence, especially when having objects of the same type. For example: “Do you prefer this (*pointing*) plate or that one (*pointing*)?”.

Pointing is used less frequently than in the multimodal output generated by the framework. It cannot be said for sure whether people in a natural conversation would act slightly different. The participants were told in the beginning that they are allowed to point to an object. Some might have forgotten this information during the study, others might not have felt comfortable gesturing while being videotaped. Nevertheless, the number of pointing gestures planned by the framework would probably still exceed the number of human pointing gestures. Pointing at objects is scored highly inside the framework. Another difference is that the generated verbal output is not reduced when a pointing gesture is performed simultaneously, whereas most participants decided to use only the type of the object and no further attributes together with the pointing. This means that the framework’s speech output does not distinguish between having a multimodal or a speech-only output and thus, the pointing gesture is a way to increase the redundancy of the statement. The idea is that this may improve the understanding of the statement. Nao’s arms are not very long and in the scenario it was not moving around. Therefore, the pointing gestures were not intended to be precise but to show the rough direction where the object is positioned. That is why additional attributes are needed in the case of the robot, but not necessarily when humans are pointing: The participants could reach most objects easily and most people put their index finger very close to the object while pointing so that misunderstandings were hardly possible. Nevertheless, 15 times a color attribute was additionally chosen.

The evaluation of the human object references additionally revealed that people often use the relative position to other objects as well as a viewer-centered position. Furthermore, when an object is described relative to another one, it seems that the other object is described in more detail than the actual object (e.g., “the scissors which are lying on the white plate with the grey stripes”). It makes sense to refer to neighboring objects when these have more salient properties. Currently, the framework mainly focuses on absolute properties. Enabling a proper support of these position specifications in the framework might be challenging, especially regarding correct perspective and possible occlusion. Apart from that, it has been noticed that people often use ellipses when contrasting two objects of the same type (e.g., “the blue and red scissors are bigger than the black one”).

Assessment of Object References in the Questionnaires

The questionnaire contained two questions regarding the speech output and the used attributes in particular. The first was whether the participants agree with the following statement: “The content of Nao’s spoken explanation was helpful in order to understand which object it referenced.”. Twenty-eight of the 52 participants totally agreed with this statement (giving 5 out of 5 points) and 19 also agreed with it (giving 4 points). Only five people rated the spoken explanation with 2 or 3 points. Thus, more than 90% of the participants found the explanation helpful. Furthermore, the participants should state whether they agree with “The object’s attributes (e.g., color of the object) Nao used to refer to a certain object were helpful to identify the object.”. Forty-three people (more than 80%) entirely agreed with this statement (giving 5 out of 5 points), eight people gave 3 or 4 points. Only one person, which was colorblind, gave 2 points. Since a lot of color attributes, including pink and green, are used by Nao, it is not surprising that the attributes were not that helpful for this participant. This problem can be overcome by adapting the choice of attributes. This is possible due to the fact that characteristics of the user can be considered in the framework’s planning process. As a result, a vast majority considered the attributes used for referencing as helpful.

Nevertheless, some people criticized the complicated and unnatural sentence structure (e.g., “pen with color red” instead of “the red pen”). Currently, the sentences containing the descriptive attributes are built in the following way: “object’s type (+ ’with’ type of attribute + value of attribute)*”. The Kleene star “ * ” indicates that something is repeated arbitrarily but finitely often. Using “(value of attribute)* + object’s type” as the structure of the sentence would sound more natural in most cases. However, this structure does not work in every case. For example, “person with name Peter” might sound strange, but is still a correct sentence, whereas “Peter person” is not. Therefore, a more sophisticated Natural Language Generation tool would be required to build sentences that sounds natural and are correct at the same time.

Apart from that, a few people were confused when Nao said “the cup” when referring to the white cup once again. As in the first part of the study, the participants probably regarded the sentences as independent entities and, therefore, the reference was not clear enough. It would be presumably clearer in an actual conversation. Nevertheless, most people used the correct cup. It is interesting to note that using “it” to refer to a pen described in the previous sentence caused less confusion. A possible reason for that is that “the cup” could also be an insufficient specification, in contrast to using “it”.

Furthermore, several people stated that, in the cases where they had not understood what Nao had said, hearing the same sentence again was not always helpful. This statement is supported by the number of repetitions: more than 60% of the people who had not understood Nao after the second try were also not able to understand the utterance in a third try. Therefore, it has been suggested to reformulate the sentence when it is repeated. Obviously, it goes beyond the possibilities of a fission framework to find the critical parts in the explanation which have not been understood and to substitute them specifically. However, it might be possible to vary the used attributes in some manner.

A person stated regarding the combination of pointing and speech that sometimes further explanations were not necessary, because the pointing together with, for example, the type of the pen was enough. In other cases, the explanations were necessary. This comment directly refers back to the comparison between the object references generated by the framework and the ones used by humans: It has been determined that most people pointed without using additional attributes, since their pointing gestures are very accurate. However, in the case of the generated output, another person stated that despite the pointing gesture, the whole verbal description was needed to find the correct object. Therefore, it would be interesting to investigate how many attributes are needed and how salient they need to be to identify the referenced object easily, depending on the accuracy of the pointing gesture.

In conclusion, the results show that the object descriptions generated by the framework are suitable for detecting the referenced object. The majority of the participants considered the used attributes as helpful. Furthermore, the categories of attributes used in the generated output mostly agree with the ones humans used in their description. Yet, the sentence structure is seen as rather unnatural and could be improved. Furthermore, the attributes choice can be varied for repetition and the speech output can be adapted depending on the pointing accuracy.

6.4.4. Human-Likeness of the Generated Output

The last central question to investigate is the question on how human-like the generated output appears. One person said that the verbal object description was accurate, though not like a human would describe the object. Other people stated, as mentioned previously, that the sentence structure sounds unnatural (e.g. “x with size big” instead of “the big x”). Therefore, it cannot be said that the generated speech appears human-like. This is not surprising, since the language generation has been kept rather simple in this thesis. Nevertheless, in section 6.4.3, the comparison with the human object description has revealed that the framework mostly uses the same attribute categories as humans would do in order to describe a certain object. Furthermore, it could be observed during the first part of the study that some people like to vary attributes used to describe an object once again in the following sentences. Thus, the use of variations might entail more human-likeness. It would be an interesting feature to vary attributes for an object when the object appears several times in an interaction.

Apart from that, the multimodal groups have been asked in the questionnaire whether they would point/look at the objects at same moments as Nao did. Figure 6.12 shows the distribution of answers. It can be seen that 12 people totally agreed with this statement (giving 5 out of 5 points) and another 10 people also agreed with it (4 points). Only four people rated the statement with 2 or 3 points. Thus, the vast majority of participants found nothing unusual nor uncanny with the timing for the pointing gestures and the gaze.

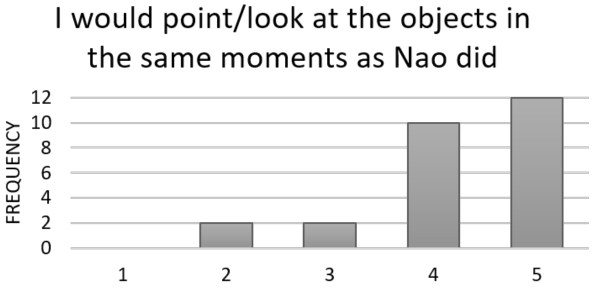


Figure 6.12.: Assessment of the Human-Likeness of the used Multimodality by Groups B12 and B21

The participants made several improvement proposals regarding more human-like behavior. As already mentioned earlier, people suggested a sentence variation when they had not understood the output and requested a repeat. A human would also not always repeat the same sentence in the exact same manner if it was obvious that it has not been understood previously. Another person said some sort of context-awareness would be great. This person refers to the fact that Nao's pointing gestures and verbal explanations seem to be redundant in some cases. As mentioned earlier, the number of attributes which are required to describe an object properly depends on the precision of the pointing. The length of the robots arms, whether it can move around and the robot's distance to the object that should be referenced may influence the precision. Nevertheless, even though the human pointing gestures performed in the first part of the study have been very precise, a large number of people have additionally chosen a color attribute. This seems to indicate that it is natural to provide information that is partly redundant. One participant suggested to use "human interaction sentences", like "Don't panic, just look at your right". Another proposal was to use eye gaze to follow the user's hands in order to let Nao seem more vivid. However, both suggestions go beyond the work of a fission framework and the scope of this thesis.

7. Conclusion

A framework for multimodal fission has been developed in this thesis. The framework receives a semantic predicate, stating the information to present in an abstract form, as input and generates a multimodal output representation by taking various planning criteria into account. Using the predicate structure defined in the artificial language Lojban has the advantage that all predicates are built in the same way and it is easy to add new predicates: It is not necessary to define an order of the predicate's arguments by oneself, but the structure of the corresponding predicate in Lojban can be looked up in the English-Lojban dictionary[1] instead. This is always possible since Lojban is as expressive as natural languages. Thus, using these predicates allows an easy and well-defined specification of the input.

Knowledge about the entities in the environment can currently be automatically retrieved from an OWL ontology or from a MongoDB database. This is done in an initialization step before processing the predicate input. If data from other sources, possibly stored in different formats, should be included, the method for retrieving the desired information can be added by the user. The retrieved information is stored internally as JSON objects which enable a flexible usage.

The system's output is a plan that determines the order of triples consisting of the selected modality, the selected device and the output element, respectively. Each output is given in a format that can be understood by the selected device. Furthermore, devices are modeled in an abstraction hierarchy inside the framework so that the user of the framework has minimal effort to connect a physical device to the framework.

Modalities are classified, depending on their functionalities, into different categories inside the framework. For now, three categories have evolved: the structure-forming modalities, which create the basic structure of the output, the object-referencing modalities, which enable references to entities in the environment, and the predicate-referencing modalities, which can highlight and intensify parts of the predicate input or the entire predicate. If new modalities are needed, they can be classified into these categories in an easy manner, since each category provides its own interface. If the existing categories are not sufficient, it is possible to add new ones.

As mentioned above, object-referencing modalities are used to refer to objects in the environment. Currently, speech, pointing, gaze and image modalities can create such references. The user study revealed that the multimodal output generated when using a combination of speech, pointing and gaze modalities is suitable for referencing objects

[1] English-Lojban Dictionary: `http://tiki.lojban.org/tiki/tiki-download_wiki_attachment.php?attId=711` (last accessed: 19th of May 2017)

in an easy, understandable manner, similar to how people would reference these objects. An algorithm has been implemented which extracts salient object attributes to generate a verbal reference to the respective object. The results of the study show that using salient attributes is natural when referencing objects. Furthermore, the categories of attributes used by the framework for referencing objects do not differ significantly from the ones used by humans.

The modality and the device selection is a key part of the MMF framework. The selections are formulated as constraint optimization problems. This allows the differentiation between hard constraints, which must be satisfied, and soft constraint, which reflect preferences and which are not required for a solution to be valid. Yet, an optimal solution tries to maximize the number of fulfilled soft constraints. The constraint satisfaction solver OptaPlanner, which is well suited to express the constraint optimization formulation, is used. OptaPlanner itself is highly configurable. Therefore, configurations like the used optimization algorithm can be adapted with minimal effort. The framework has its main focus on modality selection rather than on device selection. Therefore, some basic criteria for the device selection have been provided, whereas various example criteria classified into different categories have been defined for the modality selection. The available categories consider, for example, previously generated multimodal references, some general criteria for enabling more human-like behavior and information about the current user, like disabilities or their language level. A weight can be assigned to each category to state its importance. Then the objective function calculates the weighted sum over each category's function value. It is up to the user to decide which combination of categories to use for the modality selection. Furthermore, it might be necessary to change some criteria for different application scenarios. The flexible structure allows extending the framework by new constraints as well as new categories in an easy manner.

The results of the user study show that the combinations of modalities selected by the framework are suitable in most situations and enable a better understanding compared to using speech-only output. This shows that the provided criteria can be applied successfully in the area of human-robot interaction.

Hence, a framework for multimodal fission has been developed that is highly configurable and extendable. The existing modalities and devices as well as the provided planning criteria are reusable and reflect the framework's focus on the area of collaborative human-robot interaction. Yet, the design allows adding new modalities and devices as well as new planning criteria easily. Furthermore, the framework is neither tailored to a specific domain nor is it limited to the usage with a specific dialog manager.

8. Future Work

This chapter gives an outlook on future work. The usefulness of the framework has already been proven by conducting a user study. In the next step, the fission framework can be connected to an existing dialog manager or to an entire dialog system, like SiAM-dp [36]. An adapter needs to be defined which can translate the output of the dialog manager into the required predicate input for the framework. Extending the framework by new modalities and devices or by additional planning criteria is also a reasonable next step. Further improvements and extensions of the framework may consist of the following:

- **Including a better Natural Language Generation tool:** It has been decided to use SimpleNLG due to its ease of use. Yet, a future version of the framework may replace SimpleNLG by a tool that creates more well-sounding sentences.
- **Creating a more sophisticated output schedule:** Even though the majority of the participants in the user study have stated that the timing of pointing and gaze have been perfectly fine, the output plan can be enhanced by adding the concrete starting time for each part of the output. This could be achieved by preparing the synthesized speech in advance and using the timing information from the synthesizer to create the schedule for the other modalities as demonstrated in [18, 51].
- **Improving the generated object references further.** The following ideas evolved from the results of the user study:
  - **Refining the multimodal references:** The user study revealed that sometimes, when using speech and pointing to reference an object, not all attributes in the verbal object description are needed to identify the correct object. Hence, pointing is used redundantly in most cases: It is combined with a fully specified verbal object description. There are exceptions since, for instance, a pointing gesture is required when referring to an object by only using “this”. Yet, it would be interesting to adapt the number of attributes depending on the precision of the pointing gesture.
  - **Varying attributes for repeated sentences:** Some people found that repeating a sentence in the user study has not always been helpful, if they had not understood the sentence at the first try. Humans usually reformulate their sentences when they are not understood. Yet, it goes beyond the tasks of a fission framework to be able to gather which part of an utterance has not been understood. Nevertheless, exchanging some of the used attributes when repeating a sentence may help.
  - **Support several position specifications for objects:** If a position information should be used, it is currently necessary to add a corresponding

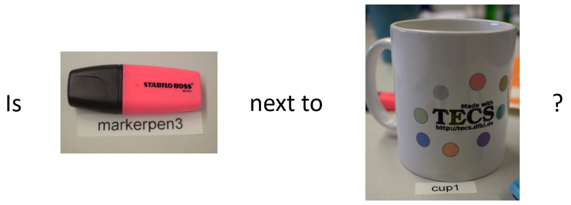
attribute, like “on the left” to the object’s list of attributes inside the world model. An extension could automatically divide the environment into different areas, like “front” or “on the left”, with the robot’s position or the position of the current user as point of reference. Then, all objects in the respective area automatically have the respective position specification. Furthermore, static objects, like a window or a table, can be used as landmarks. Apart from that, the results of the user study show that people often define an object’s position relative to other objects. This makes sense if the object that should be referenced does not have many salient attributes compared to neighboring objects.

- **Supporting multi-user and multi-robot scenarios:** An output can already be directed to several users since the framework receives a list of users for whom the output should be generated. Furthermore, several different user models can be stated in the framework. However, only information of one user is considered in the planning process for now. Apart from that, when directing output to several users, gaze needs to be produced that alternates between the addressed persons in a suitable way. Additionally, the framework can be extended to support the simultaneous usage of several robots.

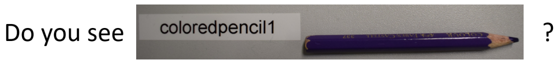
A. Appendix - User Study Material

A.1. User Study Inputs: Images, Predicates, Generated Sentences

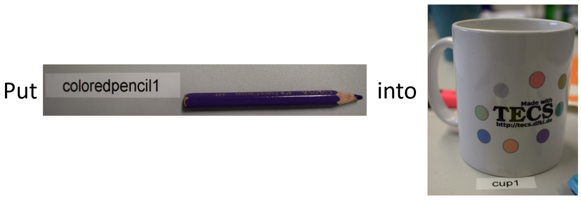
A.1.1. Inputs for Predicate Set 1



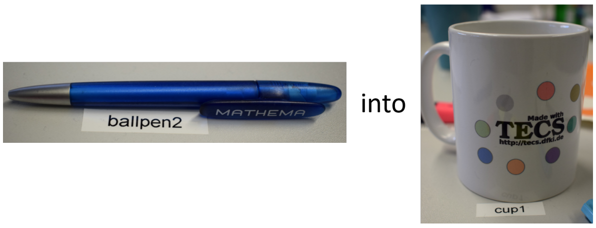
- [*xu*] **lamji**(markerpen3, cup1, [*zoe*], *zoe*])
- Is markerpen *(pointing, gaze)* with size small, with color pink next to *(pointing, gaze)* cup with color white?



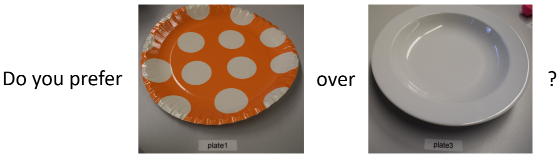
- [*xu*] **viska**(you, coloredpencil1, [*zoe*])
- Do you see *(pointing, gaze)* colored pencil with color purple, with size small?



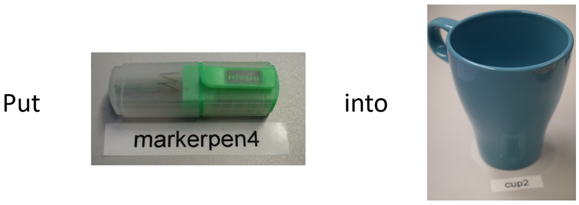
- [*ko*] **punji**([*zoe*], coloredpencil1, into cup1)
- Put it *(gaze)* into cup with color white.



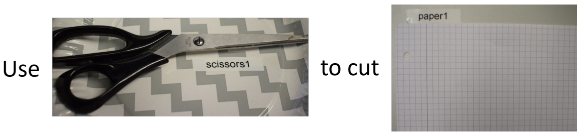
- [*ko*] **punji**([*zoe*], ballpen2, into cup1)
- Put ballpen *(pointing, gaze)* with approximate position on your far left, with label mathema *(gaze)* into the cup.



- [*xu*] **zmanei**(you, plate1, plate3, [*zoe*], [*zoe*])
- Do you prefer *(pointing, gaze)* plate with color orange over *(pointing, gaze)* plate with color white?

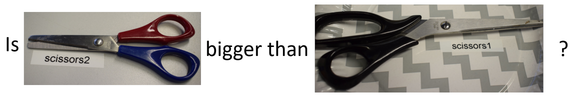


- [*ko*] **punji**([*zoe*], markerpen4, into cup2)
- Put *(pointing, gaze)* markerpen with color green into *(pointing, gaze)* cup with color blue.

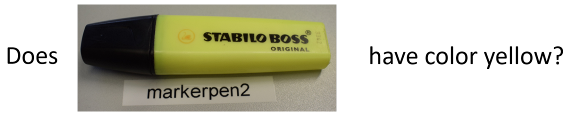


- [*ko*] **pilno**([*zoe*], scissors1, to cut paper1)
- Use *(pointing, gaze)* scissors with size big to cut *(pointing, gaze)* the paper.

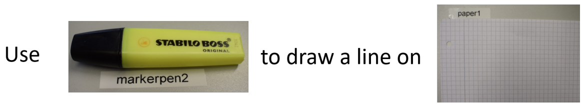
A.1.2. Inputs for Predicate Set 2



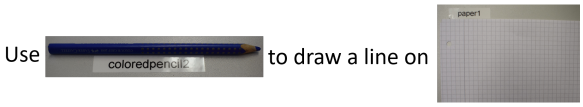
- [*xu*] **bramau**(scissors2, scissors1, [*zoe*], [*zoe*])
- Is *(pointing, gaze)* scissors with color blue and red bigger than *(pointing, gaze)* scissors with color black?



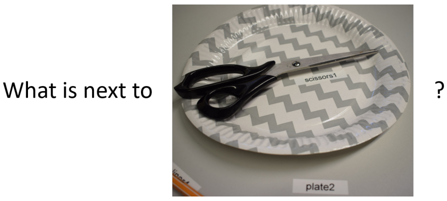
- [*xu*] **skari**(markerPen2, yellow, [*zoe*], [*zoe*])
- Does *(pointing, gaze)* markerpen with label stabilo, with size big have color yellow?



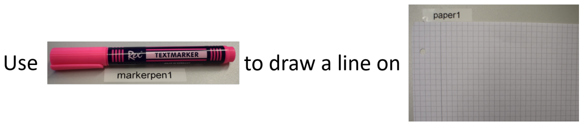
- [*ko*] **pilno**([*zoe*], markerPen2, to draw a line on paper1)
- Use it *(gaze)* to (gaze) draw a line on the paper.



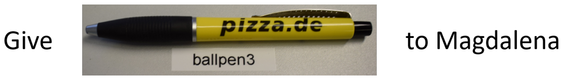
- [*ko*] **pilno**([*zoe*], coloredpencil2, to draw a line on paper1)
- Use *(pointing, gaze)* colored pencil with color blue to draw a line on the paper.



- [*ko*] **lamji**([*ma*], plate2, [*zoe*], [*zoe*])
- What is next to *(pointing, gaze)* plate with color grey white?

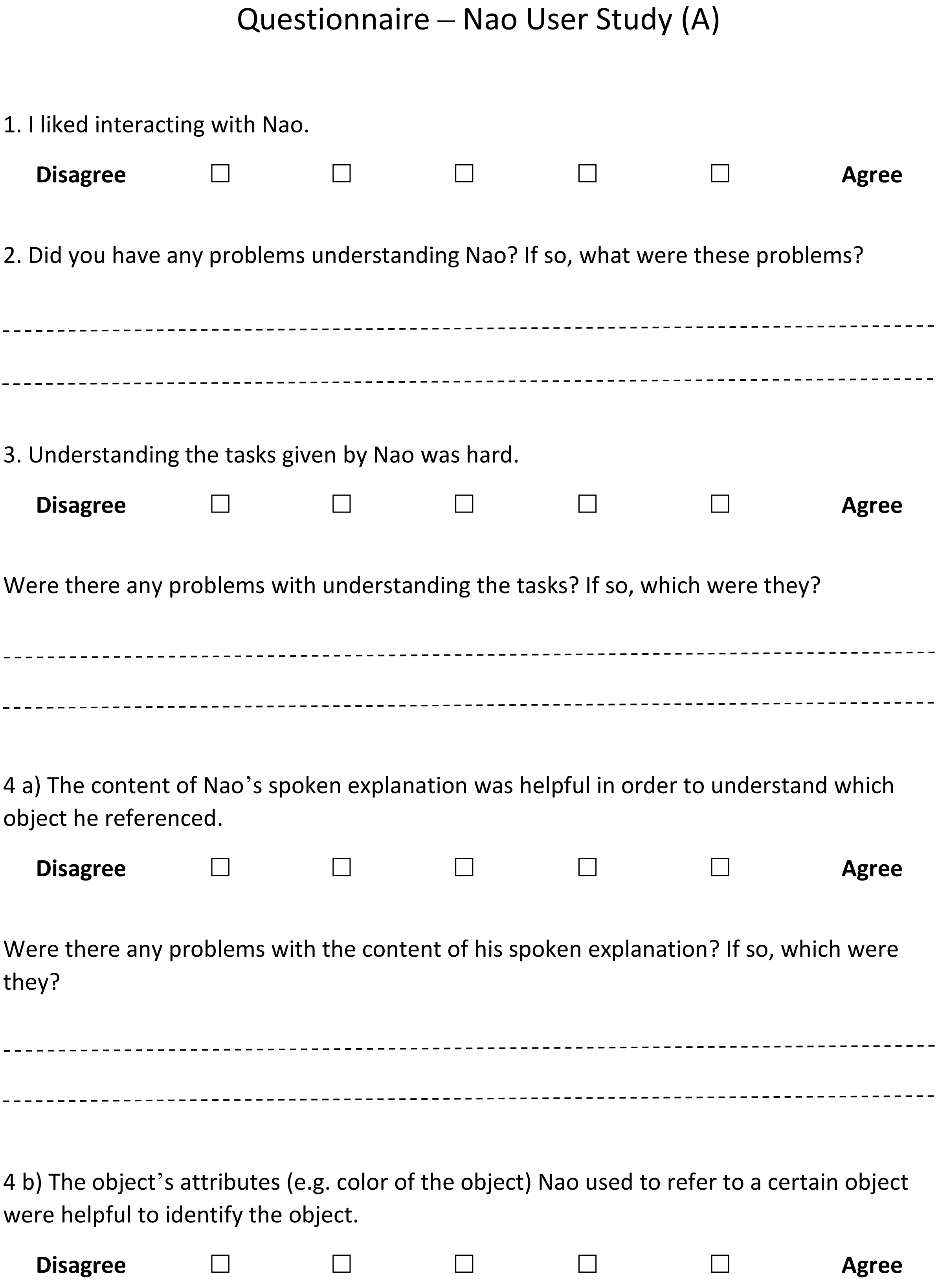


- [*ko*] **pilno**([*zoe*], markerPen1, to draw a line on paper1)
- Use markerpen *(pointing, gaze)* with color pink, with label Rex textmarker *(pointing, gaze)* to draw a line on the paper.



- [*ko*] **dunda**([*zoe*], ballpen3, person1)
- Give person with name Magdalena ballpen *(pointing, gaze)* with color yellow, with label pizza.

A.2. Study Questionnaires



Questionnaire – Nao User Study (A)

1. I liked interacting with Nao.

Disagree □ □ □ □ □ **Agree**

2. Did you have any problems understanding Nao? If so, what were these problems?

3. Understanding the tasks given by Nao was hard.

Disagree □ □ □ □ □ **Agree**

Were there any problems with understanding the tasks? If so, which were they?

4 a) The content of Nao’s spoken explanation was helpful in order to understand which object he referenced.

Disagree □ □ □ □ □ **Agree**

Were there any problems with the content of his spoken explanation? If so, which were they?

4 b) The object’s attributes (e.g. color of the object) Nao used to refer to a certain object were helpful to identify the object.

Disagree □ □ □ □ □ **Agree**

5. Besides talking, what could Nao do to help you to better understand/understand more quickly which object is referenced?

----

6. Was there something in Nao's behavior that irritated you? If so, what was it?

----

7. Do you have any proposals for improvement?

----

8. Your course of study and the semester you are in/your profession:

----

9. Your age: --------- Your gender: male □ female □

10. Have you interacted with a robot before? If so, in which context?

Questionnaire – Nao User Study (B)

1. I liked interacting with Nao.

Disagree □ □ □ □ □ **Agree**

2. Did you have any problems understanding Nao? If so, what were these problems?

3. Understanding the tasks given by Nao was hard.

Disagree □ □ □ □ □ **Agree**

Were there any problems with understanding the tasks? If so, which were they?

4 a) The content of Nao's spoken explanation was helpful in order to understand which object he referenced.

Disagree □ □ □ □ □ **Agree**

Were there any problems with the content of his spoken explanation? If so, which were they?

4 b) The object's attributes (e.g. color of the object) Nao used to refer to a certain object were helpful to identify the object.

Disagree □ □ □ □ □ **Agree**

5 a) Nao's pointing gestures were helpful in order to understand which object he referenced.

Disagree □ □ □ □ □ **Agree**

If you do not agree with the previous statement: Would you say that Nao’s pointing gestures were distracting? Otherwise, skip this question.

Disagree □ □ □ □ □ **Agree**

5 b) Nao’s viewing direction was helpful in order to understand which object he referenced.

Disagree □ □ □ □ □ **Agree**

If you do not agree with the previous statement: Would you say that Nao’s viewing direction was distracting? Otherwise, skip this question.

Disagree □ □ □ □ □ **Agree**

5 c) I would point/look at the objects I am referencing in the same moments as Nao did.

Disagree □ □ □ □ □ **Agree**

6. Was there something in Nao’s behavior that irritated you? If so, what was it?

----

7. Do you have any proposals for improvement?

----

8. Your course of study and the semester you are in/your profession:

----

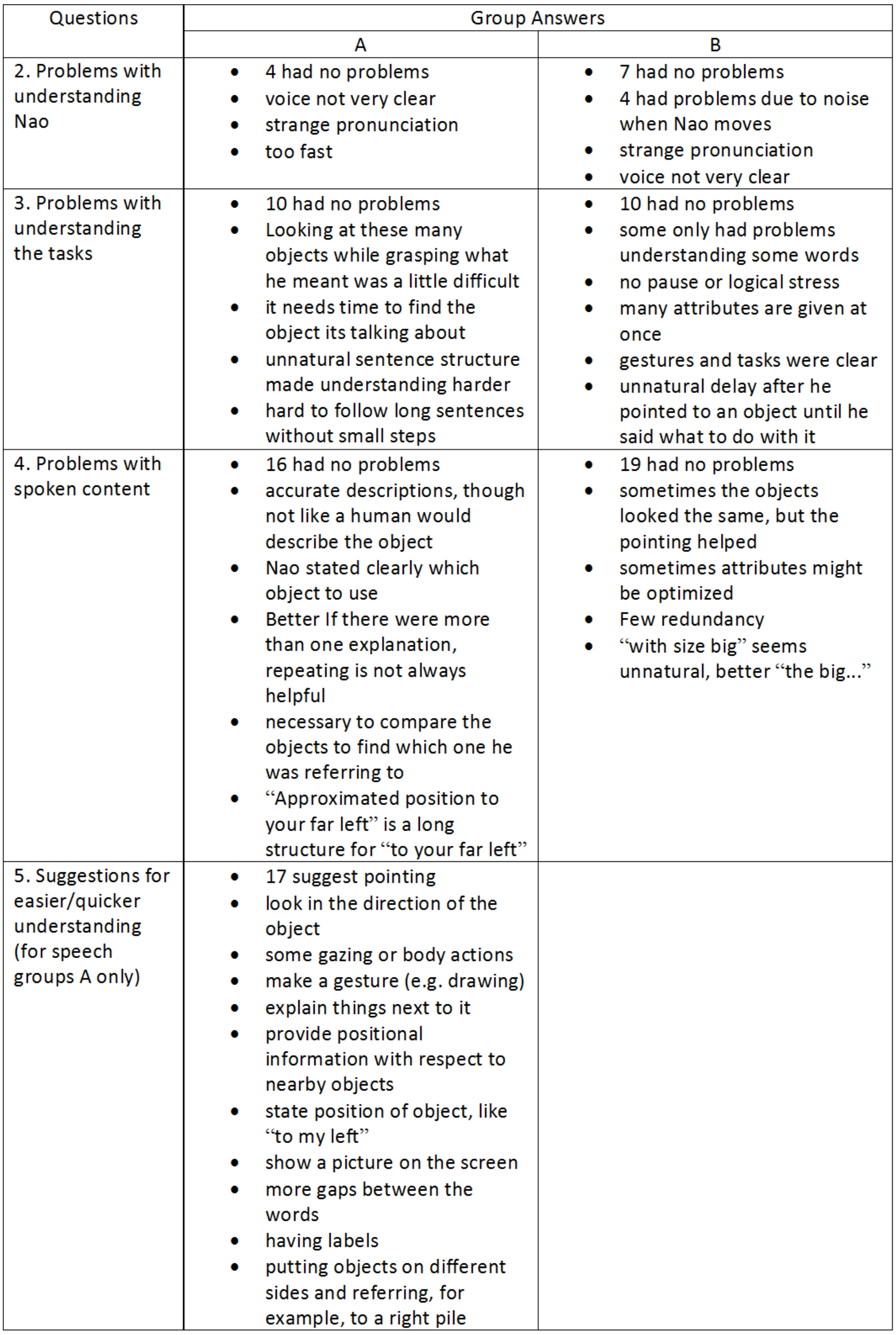
9. Your age: --------- Your gender: male □ female □

10. Have you interacted with a robot before? If so, in which context?

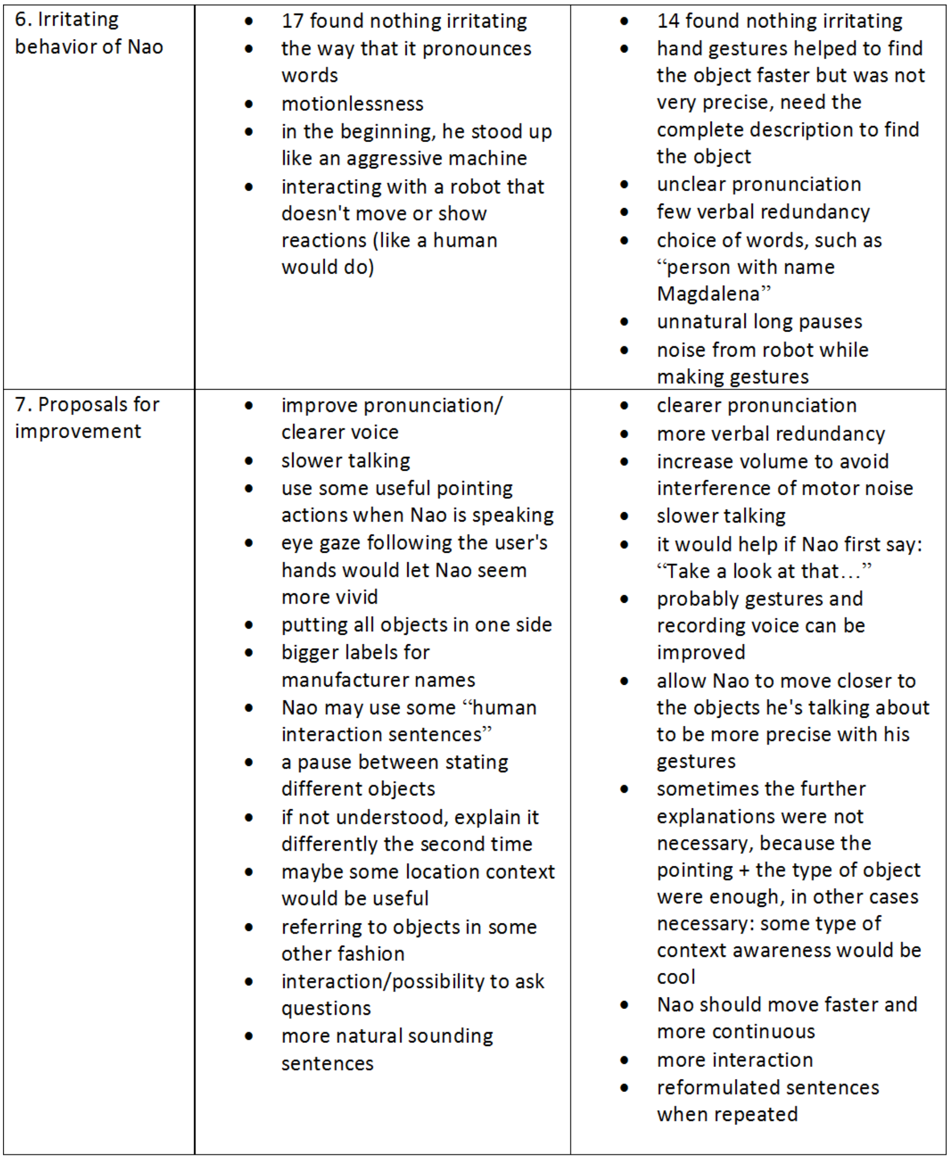
A.3. User Study Results

A.3.1. Questionnaire Evaluation

Free Text Answers



Questions	Group Answers	
	A	B
2. Problems with understanding Nao	• 4 had no problems • voice not very clear • strange pronunciation • too fast	• 7 had no problems • 4 had problems due to noise when Nao moves • strange pronunciation • voice not very clear
3. Problems with understanding the tasks	• 10 had no problems • Looking at these many objects while grasping what he meant was a little difficult • it needs time to find the object its talking about • unnatural sentence structure made understanding harder • hard to follow long sentences without small steps	• 10 had no problems • some only had problems understanding some words • no pause or logical stress • many attributes are given at once • gestures and tasks were clear • unnatural delay after he pointed to an object until he said what to do with it
4. Problems with spoken content	• 16 had no problems • accurate descriptions, though not like a human would describe the object • Nao stated clearly which object to use • Better If there were more than one explanation, repeating is not always helpful • necessary to compare the objects to find which one he was referring to • “Approximated position to your far left” is a long structure for “to your far left”	• 19 had no problems • sometimes the objects looked the same, but the pointing helped • sometimes attributes might be optimized • Few redundancy • “with size big” seems unnatural, better “the big...”
5. Suggestions for easier/quicker understanding (for speech groups A only)	• 17 suggest pointing • look in the direction of the object • some gazing or body actions • make a gesture (e.g. drawing) • explain things next to it • provide positional information with respect to nearby objects • state position of object, like “to my left” • show a picture on the screen • more gaps between the words • having labels • putting objects on different sides and referring, for example, to a right pile	

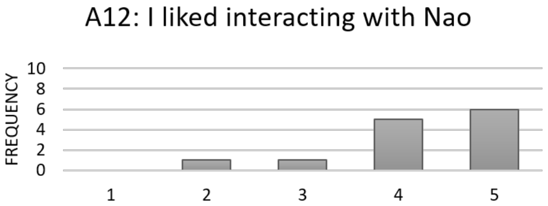


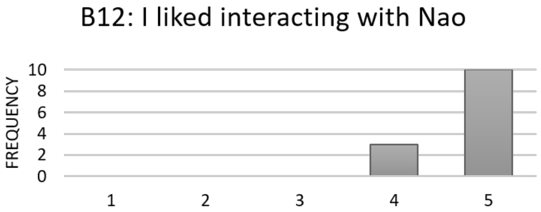
6. Irritating behavior of Nao	17 found nothing irritating the way that it pronounces words motionlessness in the beginning, he stood up like an aggressive machine interacting with a robot that doesn't move or show reactions (like a human would do)	14 found nothing irritating hand gestures helped to find the object faster but was not very precise, need the complete description to find the object unclear pronunciation few verbal redundancy choice of words, such as "person with name Magdalena" unnatural long pauses noise from robot while making gestures
7. Proposals for improvement	improve pronunciation/ clearer voice slower talking use some useful pointing actions when Nao is speaking eye gaze following the user's hands would let Nao seem more vivid putting all objects in one side bigger labels for manufacturer names Nao may use some "human interaction sentences" a pause between stating different objects if not understood, explain it differently the second time maybe some location context would be useful referring to objects in some other fashion interaction/possibility to ask questions more natural sounding sentences	clearer pronunciation more verbal redundancy increase volume to avoid interference of motor noise slower talking it would help if Nao first say: "Take a look at that…" probably gestures and recording voice can be improved allow Nao to move closer to the objects he's talking about to be more precise with his gestures sometimes the further explanations were not necessary, because the pointing + the type of object were enough, in other cases necessary: some type of context awareness would be cool Nao should move faster and more continuous more interaction reformulated sentences when repeated

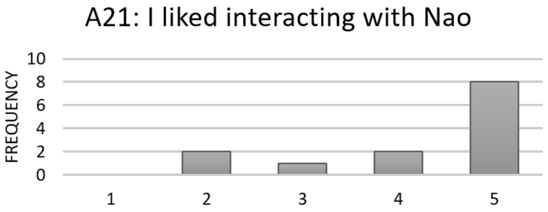
Means and Frequencies of Answers

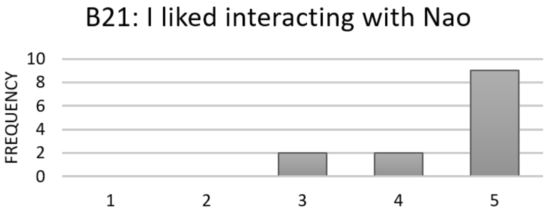
Question	Group Means			
	A12	B12	A21	B21
1. Liked Interaction	4.23	4.77	4.23	4.54
3. Hard Understanding	2.92	2.54	1.85	1.69
4a) Helpful Spoken Content	4.23	4.77	4.15	4.46
4b) Helpful Used Attributes	4.77	4.92	4.54	4.62
5a) Helpful Pointing	–	4	–	4.08
5b) Helpful Gaze	–	4	–	3.58
5c) Natural Gesture Timing	–	4.45	–	4.15
9. Participant's Age	25.61	27.77	23.69	25.69

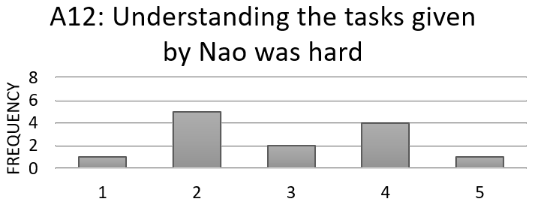
Table A.1.: Resulting Means of Answers in Questionnaires

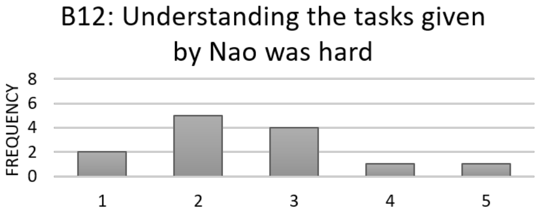


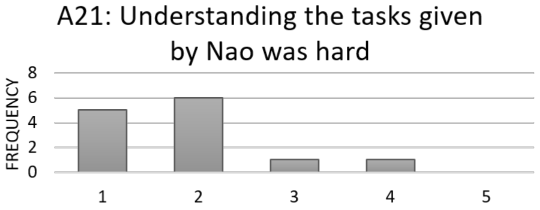


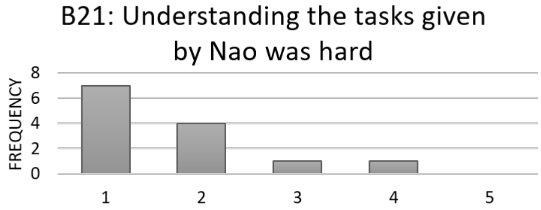


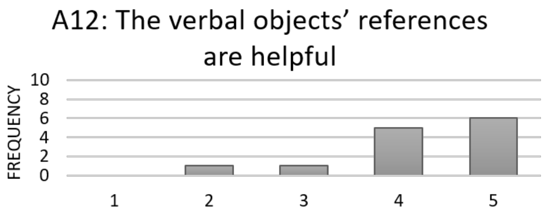


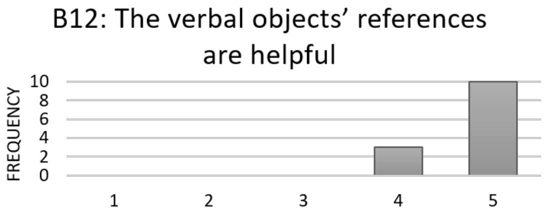


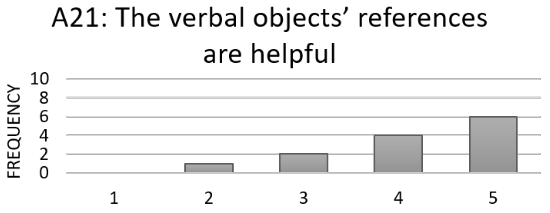


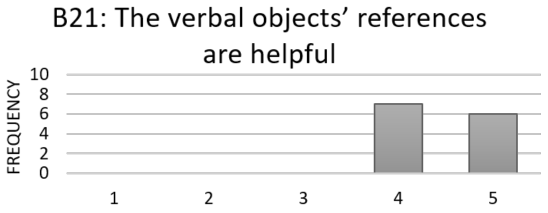


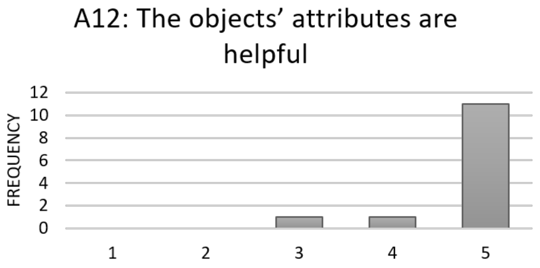


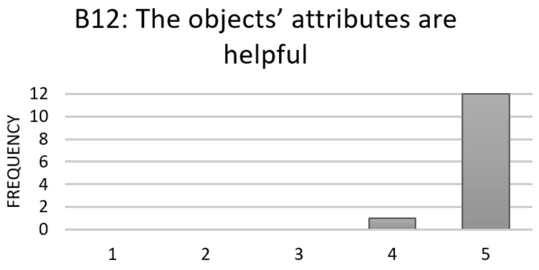

A12: The verbal objects' references are helpful

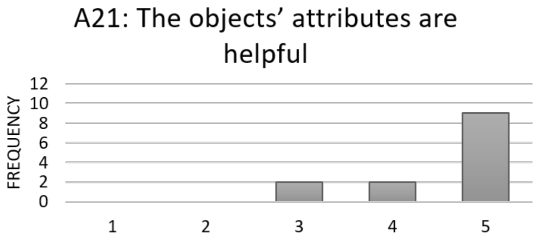

B12: The verbal objects' references are helpful

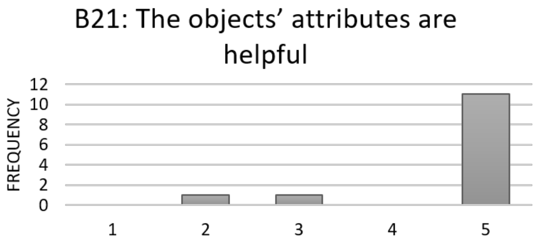

A21: The verbal objects' references are helpful

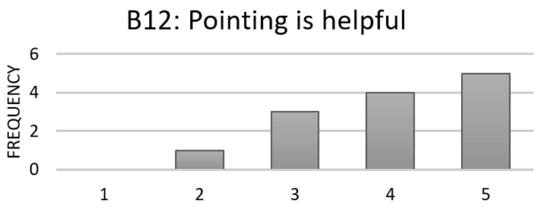

B21: The verbal objects' references are helpful

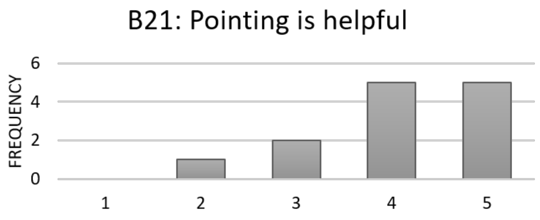

A12: The objects' attributes are helpful

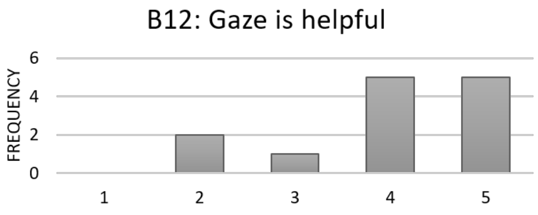

B12: The objects' attributes are helpful

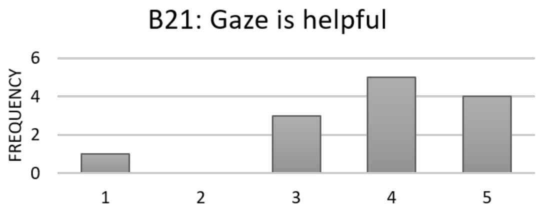

A21: The objects' attributes are helpful

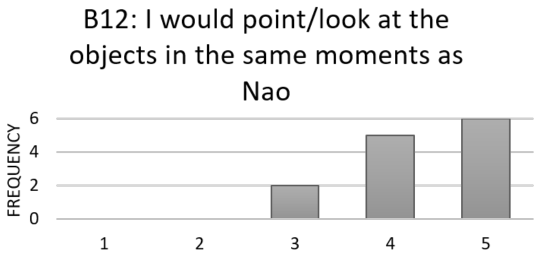

B21: The objects' attributes are helpful

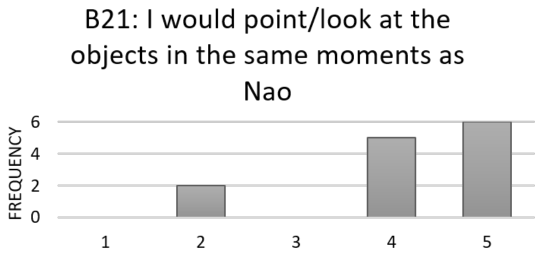

B12: Pointing is helpful
FREQUENCY
6
4
2
0
1
2
3
4
5


B21: Pointing is helpful
FREQUENCY
6
4
2
0
1
2
3
4
5


B12: Gaze is helpful
FREQUENCY
6
4
2
0
1
2
3
4
5

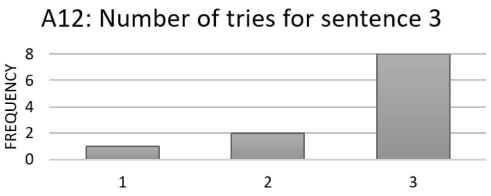

B21: Gaze is helpful
FREQUENCY
6
4
2
0
1
2
3
4
5

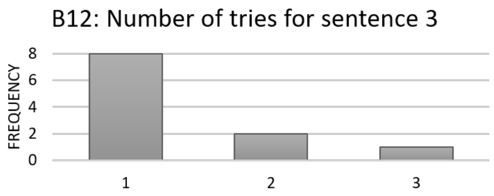

B12: I would point/look at the objects in the same moments as Nao
FREQUENCY
6
4
2
0
1
2
3
4
5

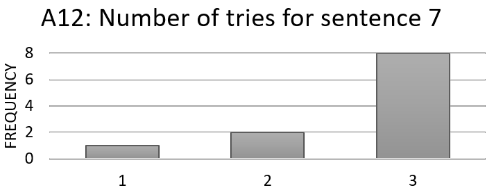

B21: I would point/look at the objects in the same moments as Nao
FREQUENCY
6
4
2
0
1
2
3
4
5

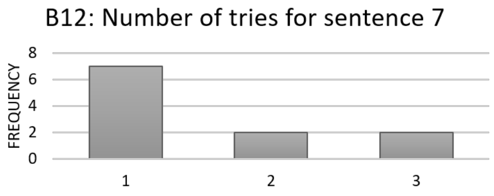
A.3.2. Objective Measures Results

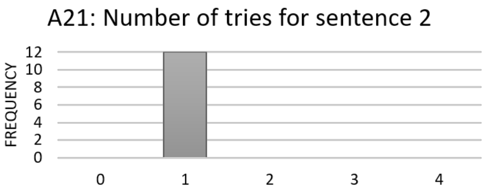
Number of tries for sentences with statistically significant differences in group means:

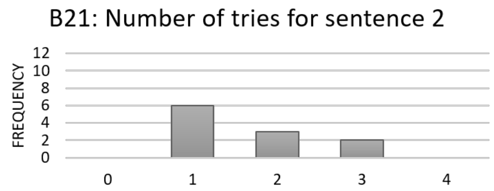

A12: Number of tries for sentence 3

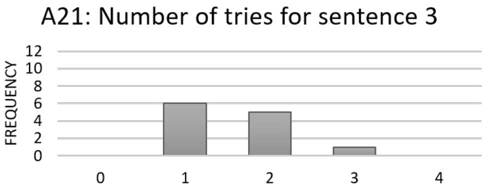

B12: Number of tries for sentence 3

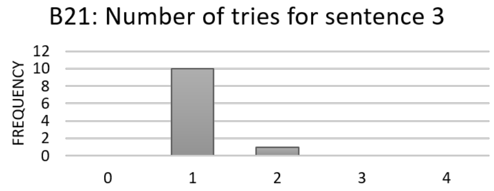

A12: Number of tries for sentence 7

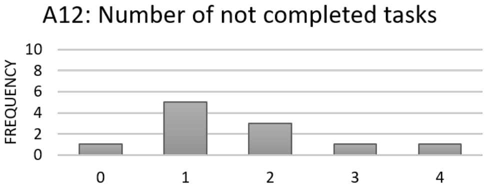

B12: Number of tries for sentence 7

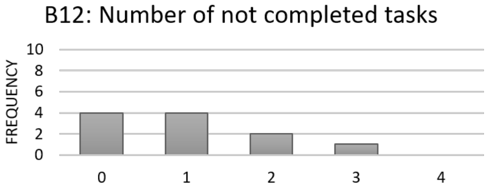

A21: Number of tries for sentence 2

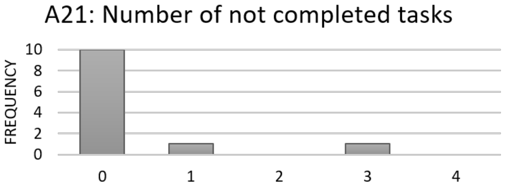

B21: Number of tries for sentence 2

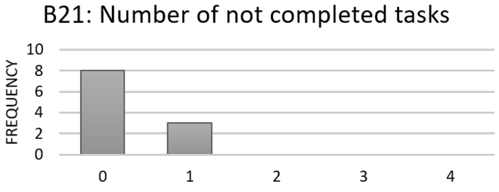

A21: Number of tries for sentence 3


B21: Number of tries for sentence 3


A12: Number of not completed tasks


B12: Number of not completed tasks


A21: Number of not completed tasks


B21: Number of not completed tasks

A.3.3. Object Referencing Results

Category	Meaning
C	**C**olor
S	**S**ize
L	**L**abel
M	**M**otif
Mat	**Mat**erial
T	**T**ype
H	**H**istory (indication for repeated reference)
relLoc	**rel**ative **Loc**ation to other objects
vLoc	**v**iewer-centered **Loc**ation
absLoc	**abs**olute **Loc**ation in the room
shape	**shape**
Comb	**Comb**ination of two objects
other	**other**, very specific attributes

Table A.2.: Categories of Attributes

Markerpen in S. 1, PS. 1

Category	Times
C+S	10
C+L	1
C+rel.Loc	1
C+v.Loc	1
S+rel.Loc	1
C+S+L	1

framework: C+S

Cup in S. 1, PS. 1

Category	Times
C	7
L	3
M	3
C+M	1
C+L	1

framework: C

Pencil in S. 2, PS. 1

Category	Times
C+S	15
C+v.Loc	2
C+other	1
C+S+v.loc	1
C+S+other	1

framework: C+S

Pencil in S. 3, PS. 1

Category	Times
H	1
C+S	15
C+v.loc	2
C+other	1

framework: H

Cup in S. 3, PS. 1

Category	Times
C	7
M	3
L	4
H	2
v.Loc	1
C+S	1
C+M	1
C+v.Loc	1

framework: C

Ballpen in S. 4, PS. 1

Category	Times
C+rel.Loc	5
C+v.Loc	5
L+v.Loc	1
L+rel.Loc	1
L+other	1
rel.Loc+abs.Loc	1
C+L+rel.Loc	3
C+L+abs.Loc	1

framework: L+v.Loc

Cup in S. 4, PS. 1

Category	Times
C	7
L	5
M	3
H	3
rel.loc	1

framework: H

Plate in S. 5, PS. 1

Category	Times
C	8
Mat	1
M	2
C+M	3
C+Mat	2
M+Mat	1
C+ v.Loc	1
C+M+Mat	1

framework: C

Plate in S. 5, PS. 1

Category	Times
C	8
Mat	3
v.Loc	1
other	2
C+Mat	1
C+other	2

framework: C

Markerpen in S. 6, PS. 1

Category	Times
C	10
S	2
C+S	5
C+rel.Loc	1
C+S+v.Loc	1

framework: C

Cup in S. 6, PS. 1

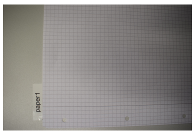
Category	Times
C	20
C+v.Loc	1

framework: C

Scissors in S. 7, PS. 1

Category	Times
rel.Loc	6
C	5
S	1
v.Loc	1
C+rel.Loc	4
C+S	2

framework: S



Paper in S. 7, PS. 1

Category	Times
T	18
v.Loc	1
C+rel.Loc	1

framework: T

Scissors in S. 1, PS. 2

Category	Times
C	16
S	2
T	1

framework: C

Scissors in S. 1, PS. 2

Category	Times
C	18
S	1

framework: C

Markerpen in S. 2, PS. 2

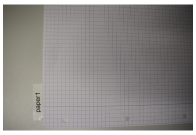
Category	Times
L	8
C	2
rel.Loc	1
S+L	1
C+v.Loc	1

framework: S+L

Markerpen in S. 3, PS. 2

Category	Times
C	11
L	2
H	1
C+L	3
C+v.Loc	1

framework: H



Paper in S. 3, PS. 2

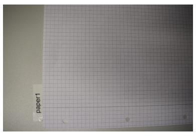
Category	Times
T	17
M	2

framework: T

Pencil in S. 4, PS. 2

Category	Times
C	19
M+C	1

framework: C



Paper in S. 4, PS. 2

Category	Times
T	17
H	2
M	1

framework: T

Plate in S. 5, PS. 2

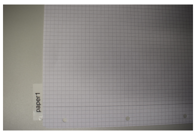
Category	Times
C	10
Comb	3
C+ Comb	5
C+Mat	1
C+rel.Loc	1
Comb+S	1

framework: C

Markerpen in S. 6, PS. 2

Category	Times
C+L	5
C+S	4
C+M	1
C+v.Loc	2
C+shape	1
C+shape+S	1

framework: C+L



Paper in S. 6, PS. 2

Category	Times
T	19
M	1
H	1

framework: T

Ballpen in S. 7, PS. 2

Category	Times
L	15
C	3
C+L	3

framework: C+L

Bibliography

[1] Carlos Areces, Alexander Koller, and Kristina Striegnitz. Referring expressions as formulas of description logic. In Proceedings of the fifth international natural language generation conference, pages 42–49. Association for Computational Linguistics, 2008.

[2] Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. Multimodal fusion for multimedia analysis: a survey. Multimedia systems, 16(6):345–379, 2010.

[3] Martin Bays. Tersmu - a semantic parser for the engineered human language lojban. `https://gitlab.com/zugz/tersmu`. Online; accessed: 19-May-2017.

[4] Elizabeth Broadbent, Rebecca Stafford, and Bruce MacDonald. Acceptance of healthcare robots for the older population: review and future directions. International journal of social robotics, 1(4):319–330, 2009.

[5] Matlab Central. Matlab answers: Determine cone intersection. `https://de.mathworks.com/matlabcentral/answers/62400-how-to-determine-if-two-cones-with-vertices-at-origin-intersect`. Online; accessed: 21-May-2017.

[6] David Costa and Carlos Duarte. Adapting multimodal fission to user's abilities. In Universal Access in Human-Computer Interaction. Design for All and eInclusion, pages 347–356. Springer, 2011.

[7] David Costa and Carlos Duarte. Improving interaction with tv-based applications through adaptive multimodal fission. Emerging Research and Trends in Interactivity and the Human-Computer Interface, page 54, 2013.

[8] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, and Richard M Young. Four easy pieces for assessing the usability of multimodal interaction: the care properties. In Human—Computer Interaction, pages 115–120. Springer, 1995.

[9] John Woldemar Cowan. The complete Lojban language. Logical Language Group, 1997.

[10] Rina Dechter. Constraint processing. Morgan Kaufmann, 2003.

[11] Anind K Dey, Gregory D Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-computer interaction, 16(2):97–166, 2001.

[12] Bruno Dumas, Denis Lalanne, and Sharon Oviatt. Multimodal interfaces: A survey of principles, models and frameworks. In Human machine interaction, pages 3–26. Springer, 2009.

[13] Geoffrey De Smet et al. OptaPlanner User Guide. Red Hat and the community. OptaPlanner is an open source constraint satisfaction solver in Java.

[14] Brian Eubanks. Jorne project. `http://jorne.org/`. Online; accessed: 17-May-2017.

[15] David Fischinger, Peter Einramhof, Walter Wohlkinger, K Papoutsakis, Peter Mayer, Paul Panek, T Koertner, S Hofmann, Antonis Argyros, Markus Vincze, et al. Hobbit-the mutual care robot. In Workshop-Proc. of ASROB, 2013.

[16] Terrence Fong, Charles Thorpe, and Charles Baur. Collaboration, dialogue, human-robot interaction. In Robotics Research, pages 255–266. Springer, 2003.

[17] Mary Ellen Foster. State of the art review: Multimodal fission. COMIC project Deliverable, 6(09), 2002.

[18] Mary Ellen Foster. Interleaved preparation and output in the comic fission module. In Proceedings of the Workshop on Software, pages 34–46. Association for Computational Linguistics, 2005.

[19] Mary Ellen Foster and Michael White. Assessing the impact of adaptive generation in the comic multimodal dialogue system. In Proceedings of the IJCAI 2005 Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2005.

[20] Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. The give-2 corpus of giving instructions in virtual environments. In LREC, 2010.

[21] Albert Gatt and Ehud Reiter. Simplenlg: A realisation engine for practical applications. In Proceedings of the 12th European Workshop on Natural Language Generation, pages 90–93. Association for Computational Linguistics, 2009.

[22] Ben Goertzel. Potential computational linguistics resources for lojban. `http://www.goertzel.org/new_research/lojban_AI.pdf`, 2005. Online; accessed: 21-May-2017.

[23] Ben Goertzel. Lojban++: an interlingua for communication between humans and agis. In International Conference on Artificial General Intelligence, pages 21–30. Springer, 2013.

[24] Birgit Graf, Ulrich Reiser, Martin Hägele, Kathrin Mauz, and Peter Klein. Robotic home assistant care-o-bot® 3-product vision and innovation platform. In Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on, pages 139–144. IEEE, 2009.

[25] Erico Guizzo and Travis Deyle. Robotics trends for 2012. IEEE Robotics & Automation Magazine, 19(1):119–123, 2012.

[26] Manolo Dulva Hina, Amar Ramdane-Cherif, Chakib Tadj, and Nicole Levy. A multi-agent based multimodal system adaptive to the user's interaction context. INTECH Open Access Publisher, 2011.

[27] Frank Honold, Felix Schüssel, and Michael Weber. Adaptive probabilistic fission for multimodal systems. In Proceedings of the 24th Australian Computer-Human Interaction Conference, pages 222–231. ACM, 2012.

[28] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. Semantic Web, 2(1):11–21, 2011.

[29] John D Kelleher and Geert-Jan M Kruijff. Incremental generation of spatial referring expressions in situated dialog. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pages 1041–1048. Association for Computational Linguistics, 2006.

[30] Ruud Koolen, Albert Gatt, Martijn Goudbeek, and Emiel Krahmer. Factors causing overspecification in definite descriptions. Journal of Pragmatics, 43(13):3231–3250, 2011.

[31] Emiel Krahmer, Sebastiaan Van Erk, and André Verleg. Graph-based generation of referring expressions. Computational Linguistics, 29(1):53–72, 2003.

[32] Dana Lahat, Tülay Adali, and Christian Jutten. Multimodal data fusion: an overview of methods, challenges, and prospects. Proceedings of the IEEE, 103(9):1449–1477, 2015.

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

[34] Saad Mahamood. Simplenlg wiki: Section vii – what are complements. `https://github.com/simplenlg/simplenlg/wiki/Section-VII-%E2%80%93-What-are-complements`. Online; accessed: 17-May-2017.

[35] Bilge Mutlu, Allison Terrell, and Chien-Ming Huang. Coordination mechanisms in human-robot collaboration. In Proceedings of the Workshop on Collaborative Manipulation, 8th ACM/IEEE International Conference on Human-Robot Interaction. Citeseer, 2013.

[36] Robert Neßelrath. SiAM-dp: an open development platform for massively multimodal dialogue systems in cyber-physical environments. PhD thesis, Saarland University, 2016.

[37] Nick Nicholas and John Woldemar Cowan. What Is Lojban? Logical Language Group, 2003.

[38] Laurence Nigay and Joëlle Coutaz. A design space for multimodal systems: concurrent processing and data fusion. In Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems, pages 172–178. ACM, 1993.

[39] Sharon Oviatt. Multimodal interactive maps: Designing for human performance. Human-computer interaction, 12(1):93–129, 1997.

[40] Sharon Oviatt. Ten myths of multimodal interaction. Communications of the ACM, 42(11):74–81, 1999.

[41] Sharon Oviatt and Philip R Cohen. The paradigm shift to multimodality in contemporary computer interfaces. Synthesis Lectures On Human-Centered Informatics, 8(3):1–243, 2015.

[42] Didier Perroud, Leonardo Angelini, Omar Abou Khaled, and Elena Mugellini. Context-based generation of multimodal feedbacks for natural interaction in smart environments. In Proceedings of the Second International Conference on Ambient Computing, Applications, Services and Technologies, number C, pages 19–25. Citeseer, 2012.

[43] Norbert Pfleger. Context-based multimodal interpretation: an integrated approach to multimodal fusion and discourse processing. PhD thesis, Saarland University, 2008.

[44] Ehud Reiter. The story of simplenlg. `https://ehudreiter.com/2016/12/15/the-story-of-simplenlg/`. Online; accessed: 17-May-2017.

[45] Raquel Ros, Séverin Lemaignan, E Akin Sisbot, Rachid Alami, Jasmin Steinwender, Katharina Hamann, and Felix Warneken. Which one? grounding the referent based on efficient human-robot interaction. In 19th International Symposium in Robot and Human Interactive Communication, pages 570–575. IEEE, 2010.

[46] Cyril Rousseau, Yacine Bellik, Frédéric Vernier, and Didier Bazalgette. A framework for the intelligent multimodal presentation of information. Signal Processing, 86(12):3696–3713, 2006.

[47] Stuart J Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2002.

[48] Rob Speer and Catherine Havasi. Meeting the computer halfway: Language processing in the artificial language lojban. In Proceedings of MIT Student Oxygen Conference, MIT, 2004.

[49] Virginie Van Wassenhove, Ken W Grant, and David Poeppel. Visual speech speeds up the neural processing of auditory speech. Proceedings of the National Academy of Sciences of the United States of America, 102(4):1181–1186, 2005.

[50] Frederic Vernier and Laurence Nigay. A framework for the combination and characterization of output modalities. In Interactive Systems Design, Specification, and Verification, pages 35–50. Springer, 2000.

[51] Wolfgang Wahlster. Smartkom: Symmetric multimodality in an adaptive and reusable dialogue shell. In Proceedings of the human computer interaction status conference, volume 3, pages 47–62. Berlin (Germany): DLR, 2003.

[52] Wolfgang Wahlster. SmartKom: foundations of multimodal dialogue systems, volume 12. Springer, 2006.

[53] Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, and Thomas Rist. WIP: The coordinated generation of multimodal presentations from a common representation. Springer, 1992.

[54] Brandon Wirick. Lojban as a tool for encoding prose on the semantic web. Master's thesis, California Polytechnic State University San Luis Obispo, 2005.