

Controlling an Intelligent Operating Room with Spoken Language



Bachelor Thesis

by

Nadine Gerstenlauer

Reviewer: Prof. Dr. Dr.-Ing. Wolfgang Minker
Supervisor: M.Sc. Juliana Miehle

Institute of Communications Engineering
University of Ulm
29 May 2017

Abstract

With the emergence of new technologies, the surgical working environment becomes increasingly complex and comprises many medical devices which have to be taken care of. However, the operating personnel cannot be extended infinitely, which is why new strategies are needed for keeping the working environment manageable. Existing research projects in the field of intelligent medical environments mostly concentrate on workflow modelling or single smart features rather than building up a complete Intelligent Environment. In this work, we present an intelligent spoken language operation assistant, providing the surgeon assistance in many different situations before and during an ongoing procedure. The speech interface enables the surgeon to concentrate on the surgery and control the technical environment at the same time, without taking care of how to interact with the system. Furthermore, the implemented system observes the context of the surgery and controls several devices autonomously at the appropriate time during the procedure.

I certify that I have prepared this Bachelor Thesis by my own without any inadmissible outside help.

Ulm, 29 May 2017

(Nadine Gerstenlauer)

Contents

List of Figures	ix
List of Tables	xi
1. Introduction	1
2. Background	3
2.1. Spoken Language Dialogue Systems	3
2.1.1. General architecture of Spoken Language Dialogue Systems	3
2.1.2. The OwlSpeak Dialogue Manager	5
2.2. Intelligent Environments	8
3. Related Work	9
4. The IDACO Spoken Language Operation Assistant	11
4.1. System Architecture	11
4.2. Functionalities	13
4.2.1. Pre-procedural Part	14
4.2.2. Procedural Part	18
4.3. Implementation	23
4.3.1. Interaction flexibility and subdialogues	23
4.3.2. Adapting the device settings	25
4.3.3. Writing to the database	26
4.3.4. Changing variable values	26
4.3.5. Tracking surgery orders	27
4.3.6. Monitoring the course of the procedure	27
4.3.7. Adapting the expected usage of material	28
4.3.8. Emergency mode	29
4.4. Reusability	29
5. Conclusion	31
Appendix A. IDACO Database	34
Appendix B. Surgery Schedule for a Laparoscopic Cholecystectomy	37

List of Figures

2.1. The modular architecture of a Spoken Language Dialogue System [19] . .	3
2.2. The general architecture of OwlSpeak following the model-view-presenter paradigm [30]	5
2.3. Scheme of the Spoken Dialogue Ontology [30]	6
4.1. Overall system architecture	11
4.2. Use case for the pre-procedural dialogue part	14
4.3. Interaction possibilities for the pre-procedural dialogue part	16
4.4. Interaction possibilities for the procedural dialogue part	19
4.5. Automatic and manual device control during surgery	22

List of Tables

4.1. Database table for a session	12
4.2. Database tables needed for adjusting to the surgeon's presetsings	13

1. Introduction

Using it from our early childhood on, speech is the most natural way of communication for human beings. Since the advent of computer technology in our everyday lives, it was only a matter of time, until the first computer applications could be controlled by speech. By now, spoken dialogue systems can be found in almost every part of our lives. Especially in mobile environments, speech control provides a significant simplification when using an application, as no hands or eyes are needed any more [5].

Until now, the whole topic was only scarcely researched in medical applications, even though the benefits seem obvious. In surgery, the operating room (OR) is the most value-creating factor. However, being the most cost-intensive and complex one as well, any effort in research related to this topic might be justified [18].

With the emergence of new technologies, the surgical working environment becomes increasingly complex and comprises many medical devices which have to be taken care of. However, the operating personnel cannot be extended infinitely, which is why new strategies are needed for keeping the working environment manageable. The *Operating Room of the Future (ORF)* is a keyword frequently used in this context [7]. It describes the application of new technologies such as computer-enhanced systems to create an intelligent OR [23] that facilitates work and reduces the staff needed to assist during a surgical intervention. This again reduces personnel cost and might even lessen the rate of incidents caused by human error.

Regarding Intelligent Environments, most research projects focus on multimodality, as they aim to combine the advantages of natural speech communication with those of graphical and gesture-based user interfaces [12]. This may ease the goal to integrate pervasive components seamlessly into the environment in order to obtain a natural interaction [10]. However, multimodal interfaces do not seem to be practical in surgical environments. The surgeon as well as the whole surgery team must not be disturbed by the usage of complex interfaces. Therefore, the human-computer-interface has to be designed simple and intuitive [6]. During a procedure, the surgeon needs his hands to operate on the patient and his eyes for being aware of what he is doing. Thus, any graphical and gesture-based systems are not well suited for this purpose.

Being hands-free and eyes-free, speech turns out to be the modality of choice. Additionally, speech can be used omnidirectional, making it a rather comfortable solution [5], especially in the operating room. The surgeon can concentrate on the surgery and control the technical environment at the same time, without taking care of how to interact with the system. Enabling the operating surgeon to control devices inherent to the OR by himself reduces the staff needed to assist during a surgical intervention.

The goal set by Feussner et al. [6] is to have an intelligent system that takes care of the surgical environment not only by explicit commands. In order to increase productivity

1. Introduction

and reduce the workload for the operating staff, it ought to act active-cooperatively and support the surgeon autonomously during the procedure.

Enabling an intelligent operating assistance system to control surgical devices automatically bears several challenges. The system needs to know when to perform which action on which device and when to stay in the background. Therefore, it has to be aware of the whole context of the surgery, i.e. especially the current point of the procedure as well as all past and future actions.

Developers have to find a reliable method for tracking the course of the surgery, which makes it possible to detect unscheduled events during the procedure. It has to be clear how the system is supposed to react in tenuous situations. For this purpose, standardised surgeries need to be modelled in detail, so that the system is able to compare the actual course of the procedure to the schedule [8]. However, this requires a close cooperation between medical science experts and system developers.

A first step towards the ORF was done by researchers in Munich, Germany. The intelligent operating room at the *Klinikum rechts der Isar, TU München* is equipped with numerous sensors enabling automated real-time data acquisition during minimally invasive procedures. The collected information can be analysed based on predefined workflow models. Furthermore, all devices contained in the OR are connected to a central network. Having controlled everything by hand so far, the devices can be regulated using a tablet PC now [27].

However, a graphical interface is not the best alternative for the surgical environment, as already indicated. Therefore, we developed IDACO (German acronym for 'Intelligentes digitales Assistenzsystem für die chirurgische OP-Umgebung'), a speech-based assistance system for the intelligent operating room. It is capable of receiving a multitude of speech commands and replying accordingly. The system assists during the preparation for the surgery by providing relevant information. It is possible to adjust the devices contained in the operating room to the surgeon's preferred settings. Furthermore, IDACO escorts the operating staff through the entire surgical intervention. It observes the course of the procedure and signals a deviation from the schedule. The system even performs some actions automatically. All functionalities are described in detail in this work.

As Spoken Language Dialogue Systems and Intelligent Environments might be unfamiliar, we give some relevant background in Chapter 2. In Chapter 3, we compare our system to some existing approaches in intelligent medical environments. Afterwards, the IDACO spoken language operation assistant is presented in Chapter 4. Finally, we conclude in Chapter 5.¹

¹As our system was developed for an operating room in the German city of Munich, all given dialogue fragments in this work are in German. However, everything is additionally explained in the text for better understanding.

2. Background

In this chapter, we give some relevant background for the work at hand. As the topics might be unfamiliar, we provide some information about Spoken Language Dialogue Systems, before outlining the basics about Intelligent Environments.

2.1. Spoken Language Dialogue Systems

A Spoken Language Dialogue System (SLDS) enables the user to use an application via speech. Input as well as output of a SLDS are speech audio signals.

In the following, we describe the general architecture of SLDS. Afterwards, the OwlSpeak dialogue manager is presented, which is the key technology that was used for developing our system.

2.1.1. General architecture of Spoken Language Dialogue Systems

A Spoken Language Dialogue System comprises five major components, namely speech recognition, semantic analysis, dialogue management, text generation and speech synthesis. Each of these modules has a specific task. Schmitt and Minker [25] describe the architecture of an SLDS as a pipeline where each component processes the output of the previous one. The modular architecture is illustrated in Figure 2.1. The single components are briefly described in the following.

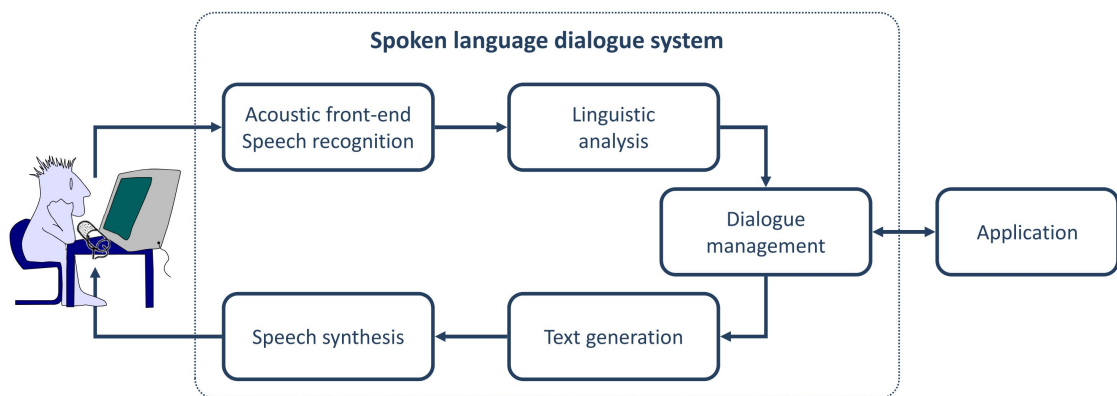


Figure 2.1.: The modular architecture of a Spoken Language Dialogue System [19]

2. Background

Speech Recognition

The user's speech input is first captured by the automatic speech recognizer (ASR). The audio signal is preprocessed by means of reducing noise and isolating the relevant information, called feature extraction. The cleaned speech signal is linguistically decoded and transformed into the most likely sequence of words.

Linguistic Analysis

The output of the speech recognition module is a sequence of words as text, lacking any semantic. The task of the linguistic analysis is to interpret this sequence of words semantically by using a semantic representation language, i.e. a grammar. The text representation is annotated with the most likely meaning creating a semantic representation, which is passed on to the following module, the dialogue manager.

Dialogue Management

The dialogue management is the core component of a SLDS. The dialogue manager processes the output of the linguistic analysis module which means it interprets the semantic meaning taking into account the context of the current dialogue.

The dialogue manager serves as the interface to the underlying application and handles any information that may be missing for the system to execute the current task. It first consults the dialogue history which stores all previous interaction. If the information cannot be found there, the dialogue manager has to enquire it of the user.

Being responsible for clarifying ambiguities, reprompting system actions if the user does not respond in time, asking for confirmation and much more, the dialogue manager needs to ensure that the user and the system always have a common ground. It is in charge of managing the dialogue flow and thus has to decide which actions to take in which order with respect to the task at hand. The output of the dialogue management module is an abstract system reaction.

Text Generation

The text generation module processes the abstract system reaction which has to be transformed into a sequence of words. The result is the actual system utterance.

Speech Synthesis

The text-to-speech (TTS) module transforms the system utterance underlying in textual form into an audio signal. Therefore, the sequence of words is either completely synthesized or divided into small parts and then constructed with prerecorded audio snippets. Either way, the resulting signal has to be smoothed and reworked. The result is finally a spoken system response.

2.1.2. The OwlSpeak Dialogue Manager

OwlSpeak is an ontology-based Dialogue Management System developed by Heinroth et al. [11] in particular for the use in Intelligent Environments. Therefore, its implementation as Java Servlet facilitates the communication with speech devices. The architecture is based on the often-used model-view-presenter paradigm in order to strictly separate data management, dialogue logic and dialogue interface [30]. The general architecture is illustrated in figure 2.2. The single components are briefly outlined below, based on the description of Ultes and Minker [30].

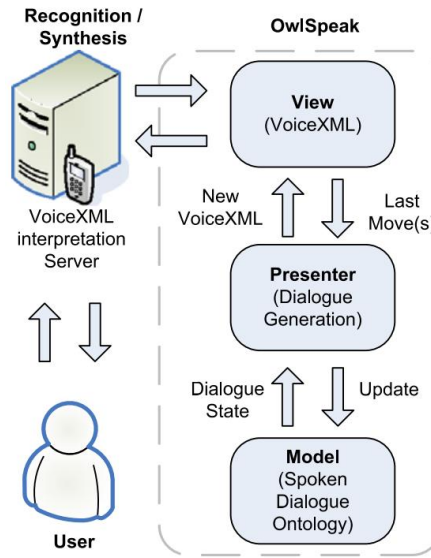


Figure 2.2.: The general architecture of OwlSpeak following the model-view-presenter paradigm [30]

Spoken Dialogue Ontology (model)

The model of OwlSpeak is designed using the Web Ontology Language (OWL) [21]. OWL is a Semantic Web language for representing rich and complex knowledge about entities, classes of entities and relations between them. This hierarchical structure is known as ontology and can be used as a skeletal foundation for a knowledge base [29]. A Spoken Dialogue Ontology (SDO) is composed of a static description of the dialogue, the *Speech* part, and the current dialogue *State* [30]. The schematic description can be seen in Figure 2.3.

The modules of the static *Speech* part are described as followed:

Utterance The *Utterance* represents the system’s output that is synthesized to a speech signal. There is one *Utterance* for each system turn containing one or more sentences.

2. Background

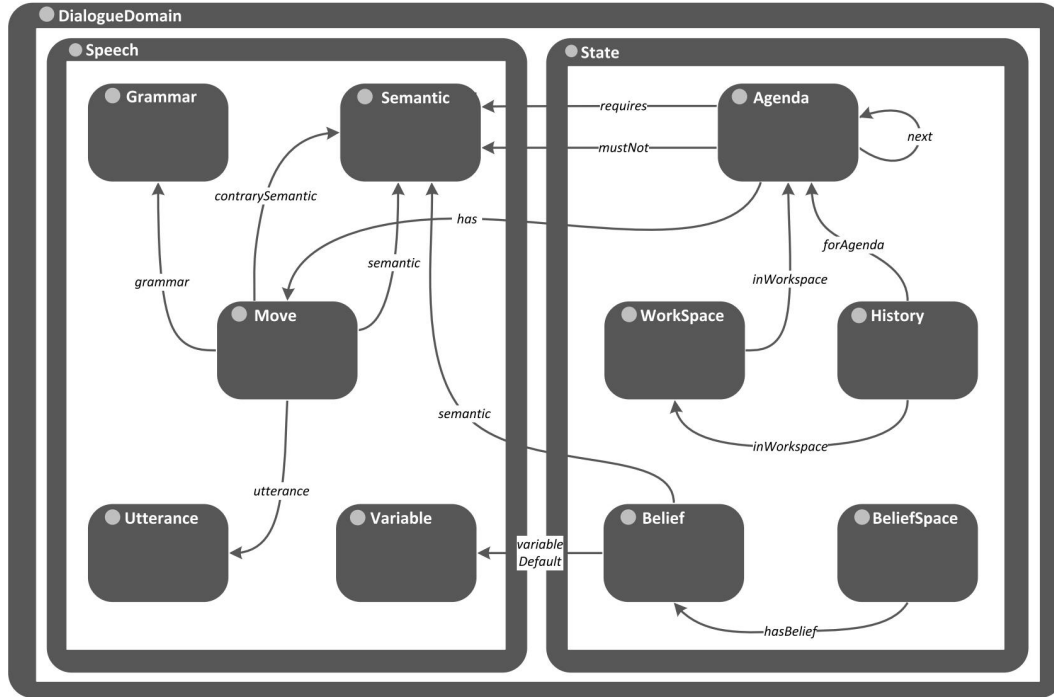


Figure 2.3.: Scheme of the Spoken Dialogue Ontology [30]

Grammar *Grammars* are used for the semantic representation of the user’s speech input. They define what the user can say and which input the system is able to understand accordingly.

Semantic A *Semantic* describes the meaning of what was said either by the system or by the user, representing single information snippets.

Variable *Variables* are used in order to store information provided during the dialogue that has several possible values. They can also be used for system internal values.

Move A *Move* represents one atomic dialogue step that can be initiated either by the user (i.e. a grammar move) or by the system (i.e. an utterance move). Each user action is semantically represented by a grammar move. When a *Move* is performed, it is possible to set or unset a corresponding *semantic* or so-called *contrarySemantic*.

Since utterances, grammars and semantics are completely separated from the moves, it is possible to reuse them for other moves.

In order to have a persistent dialogue that can be continued after being interrupted, it is necessary to store the current *State* of the dialogue in the dynamic part of the SDO. In the following we give a brief description of the important concepts.

Belief In contrast to the static concepts of *Semantics* and *Variables*, a *Belief* is a semantic or a variable which is valid in the current dialogue state. The static concepts are transferred into the dynamic part when a move is performed. The semantic of a move as well as variables with their corresponding values are then transformed into beliefs.

BeliefSpace All belief individuals which are valid in the current dialogue state are integrated in the *BeliefSpace*.

Agenda An *Agenda* represents a system action. In each turn the system selects and executes one agenda containing zero or one utterance moves and several grammar moves. Each agenda can have preconditions that have to be fulfilled. Therefore, the relations *requires* and *mustNot* define the *Semantics* that must or must not exist as a *Belief* in the *BeliefSpace*. Additionally, an agenda may require certain *Variable* values. There can be multiple agendas that may be executed in the next system turn. The *next* relation indicates that these agendas have to be written into the workspace after the agenda has been executed.

Workspace The *Workspace* contains all agendas that are allowed to be executed in the next system turn.

Voice document (view)

The view part of OwlSpeak is based on VoiceXML. This is the component that communicates with the speech recognizer and the speech synthesizer. In each system turn, the Java Servlet representing our Dialogue Manager dynamically creates a new VXML-Document based on the current agenda.

Therefore, the system prompt is provided by the utterance move. Furthermore, one big grammar is formed by putting together all grammar moves. In order to map the user utterance back to its corresponding move, the move names are used as grammar tags. An additional submit-tag is needed for finally passing the input to the voice browser back to OwlSpeak.

Dialogue control (presenter)

The presenter part contains the whole dialogue control logic of OwlSpeak. In order to start the dialogue, an agenda is needed which is marked as master agenda by using a specific flag of the Agenda class. All agendas that can be executed at the beginning of the dialogue are defined by the *next* relation of the master agenda.

All agendas in the workspace have to be checked regarding their preconditions, which was already described with the concept of agendas in the model part (see Section 2.1.2). There may be more than one agenda whose preconditions are fulfilled, which is why there is an additional priority score defining which agenda rather to execute.

Based on the agenda at hand, finally a view is created and passes new user input back

2. Background

to the presenter. The corresponding ontology is determined. Its belief space then has to be updated by deleting old beliefs and also creating new ones. Analogically, OwlSpeak removes the current agenda from the workspace and writes all agendas in the *next* relation to it.

Since the next agenda has to be chosen now, the process starts anew.

2.2. Intelligent Environments

Intelligent Environments (IE) are computing systems embedded in physical spaces. In order to increase the interactivity and to enhance the user experience, they are equipped with diverse pervasive computing technologies, bringing computation to the physical world [1].

To provide situational information about the environment and the user, the system is equipped with a diversity of sensors. The information they provide additionally may be relevant for the interaction with the user. After being processed, this information might be considered in future interaction for both triggering and parametrizing behaviours [26]. A system that is capable of automatically adapting to its changing environment is called a context-aware Intelligent Environment [9].

For being able to interact with the user, an Intelligent Environment needs input and output computing devices [26]. Most research projects focus on multimodal interaction approaches, as they aim to combine the advantages of natural speech communication with those of graphical and gesture-based user interfaces [12]. This may ease the goal to integrate pervasive components seamlessly into the environment in order to obtain a natural interaction [10].

3. Related Work

Like the prototype of Haya et al. [10], IE research projects concentrate predominantly on Intelligent Home Environments. In spite of numerous benefits (see Chapter 1), the whole topic was only scarcely researched in medical applications until now. Most research projects related to the topic of the *Operating Room of the Future* are focused on visualisation and intraoperative imaging, telesurgery and robotic surgery and educational features [3]. Some existing projects about context-awareness and spoken language assistance systems in the field of medical science are described in this chapter.

Existing general workflow implementations are not suited for the use in healthcare applications. These are rather specific and require real-time patient monitoring and detection of adverse events [28]. Katić et al. [14] describe a method for formalizing laparoscopic surgeries. Their system takes input provided by intraoperative sensor analysis and interprets it to deduce the current phase of the surgery. Kranzfelder et al. [15] use the same method to infer the workflow of laparoscopic cholecystectomy in real-time. However, each instrument needs to be handed back to the scrub nurse immediately after application, as the sensors cannot define the endpoint of the current phase otherwise. Furthermore, sensor data is not perfectly precise. Thus, this method requires a second data source, e.g. labelled videos of the procedure [15]. As stand-alone method, sensor data does not provide a reliable workflow analysis.

A proven method for defining the workflow in endoscopic surgeries is by the history and combination of used tools [22]. There are several approaches for tracking devices based on mechanical, optical, acoustic, or electromagnetic technologies [4].

Pardoy [22] suggests recording the whole procedure and labelling the videos afterwards, in order to track the usage of laparoscopic instruments. However, many factors can disturb vision-based solutions, such as the crowded environment, the small field of operation, the similar clothes of the operating staff and the changing lighting conditions. Furthermore, the workload is increased as the videos have to be labelled manually using a dedicated software.

Kranzfelder et al. [17] use bar code recognition in order to detect and register each laparoscopic instrument when inserting and removing it through the trocar. This method provides a high data capacity. Additionally, bar code information can be encrypted and read easily. However, technological improvements are needed, as instruments cannot be detected if they are moved too fast.

A similar approach is the radio-frequency identification (RFID)-based instrument detection system, which can be used for real-time recognition of the instrument in use during laparoscopic interventions [16]. As RF communication is very interference-prone, RFID-based instrument detection is not suited as stand-alone technology without further research.

3. Related Work

Real-time data retrieval plays an important role when discussing context awareness. Agarwal et al. [2] developed a prototype system focusing on this topic. It retrieves information from multiple sources in the OR (RFID amongst other things) and analyses these data streams in order to detect medically significant events. Thus, the surgical team is supported by the interpretation of events in the current context. However, the results are finally documented in an electronic encounter record without triggering any further system actions.

To the best of our knowledge, there exists only one speech-enabled operation assistant which is capable of handling devices from multiple manufacturers. The HERMES Control Center built by Computer Motion Inc. [24] connects all devices in the OR over a central network. Therefore, the scrub nurse can handle instruments over a central touch screen instead of touching numerous control panels. Furthermore, HERMES provides the surgeon speech control over several devices. Feedback about the equipment is then overlaid on the endoscopic video screen. Thus, the surgeon does not have to look at an additional screen. The system provides audio feedback, if one of the devices loses power or is disconnected from the network. However, HERMES implements an audio-visual interface. Even though the use of voice commands was well received among surgeons and the OR staff, pure speech interaction is not supported.

Finally, the existing applications mostly concentrate on workflow modelling or single smart features rather than building up a complete Intelligent Environment. Therefore, our goal was the development of an intelligent spoken language operation assistant putting together several functionalities which provide the surgeon assistance in many different situations before and during an ongoing procedure. Our system is described in the next chapter.

4. The IDACO Spoken Language Operation Assistant

IDACO is a speech-based intelligent assistance system for the surgical environment. It is capable of receiving a multitude of speech commands and replying accordingly. The system assists during the preparation for the surgery and escorts the operating staff through the entire surgical intervention.

In this chapter, we first outline the system architecture. A detailed description of its functionalities and the implementation is given afterwards. Finally, we analyse the potential reusability of our system.

4.1. System Architecture

As depicted in Section 2.1.2, using OwlSpeak, the whole dialogue management is based on an ontology. It describes the entire dialogue domain including entities and relations in between, i.e. the ontology stores each topic the user and the system can address and how this information is linked up.

In our case, there are many dialogue parts where the ontology is not sufficient as single knowledge source. A lot of information has to be obtained from external sources. Therefore, the system is additionally connected to a database. The overall architecture of our system is shown in Figure 4.1.

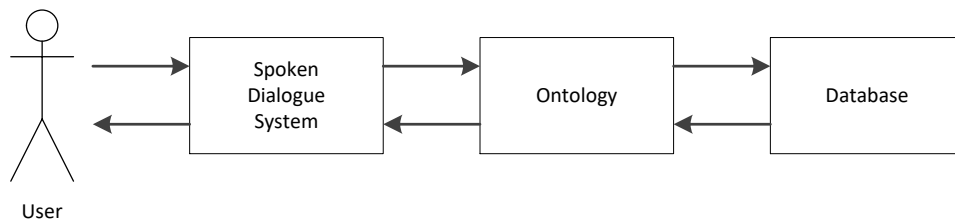


Figure 4.1.: Overall system architecture

The spoken user input is first processed by the SLDS. Each input is written to the ontology afterwards. In case of needing additional information from the database, the dialogue manager requests the missing values from the database and writes the results to the corresponding ontology variables. The system reaction is created using all necessary information contained in the ontology. The abstract system reaction is further processed by the SLDS and finally, the user receives a spoken language system reaction.

4. The IDACO Spoken Language Operation Assistant

IDACO_session
session_id
session_datetime
session_case_number
session_type_id
patient_id
primary_surgeon_id
secondary_surgeon_id
session_comment

Table 4.1.: Database table for a session

As described in Section 2.1.2, the Dialogue Manager OwlSpeak, which is used here, was developed as Java Servlet. Consequently, our system is also implemented in Java and the database connection is established using JDBC¹. Some details about the communication between the ontology and the database can be read up in Section 4.3.3.

In its full extent, the SQL database that is used in this work is illustrated in Appendix A. Some information contained in the database is only sample data. However, it is an image of the database which is in fact used at the *Klinikum rechts der Isar* in Munich. For actual use, IDACO shall be connected to the hospital information system.

The database contains the session record describing which surgery has to be executed on which patient. A specific flag indicates the current session.

From the session record, the patient files can be concluded. They include personal data as well as medical information such as pre-diseases, pre-medication and current laboratory data. All medical information related to the patient is given in a database table, like for example `IDACO_patient_pre_medication_list`. Therefore, an additional table is needed containing all possible values for the corresponding information. In the case of medication this is the table `IDACO_medications` providing all possible patient pre-medication. In the database, such relations are represented by primary and foreign keys.

The central point of the database with regards to our system is the table `IDACO_session`, shown in Table 4.1.

It contains all information that is relevant for the surgery at hand: which kind of procedure is done, which patient has to be operated and who is the leading and the second surgeon.

The whole surgery team is defined in the table `IDACO_session_team`. It contains each member of the staff who attends the procedure as well as if the person is currently present in the operating room. For each member of the staff there are again tables keeping information like name, role and experience.

In our database, there is also a lot of information about the devices in the operating room. This is necessary for automatically controlling several devices during the surgery, which is further described in Section 4.2.2. Therefore, the database contains a list of all devices

¹ Available: <http://www.oracle.com/technetwork/java/javase/jdbc/index.html>

IDACO_device_status	IDACO_device_status_setup
device_status_id	device_status_setup_id
device_list_id	device_list_id
session_id	setup_id
device_status_param_key	device_status_param_key
device_status_param_value	device_status_param_value

Table 4.2.: Database tables needed for adjusting to the surgeon's presets

and the corresponding device parameters. For each session and each device in use this includes the values of all device parameters.

Regarding the device settings, IDACO ought to adjust to the primary surgeon's preferences (see Section 4.2.1). Thus, there is a database table `IDACO_staff_setup` storing several personal setups for each surgeon. A setup belongs to a certain surgery type. Since there is a flag indicating which setups are currently active, each surgeon is allowed to save multiple setups for each surgery type. A setup contains all devices needed for the underlying procedure as well as the corresponding parameters.

Both tables for the device status and the device setup are shown in Table 4.2. In the following, we briefly describe how they relate to each other.

It might be evident that the tables for the device settings and the corresponding personal presets are very much alike. Both contain the device ID (i.e. `device_list_id`), the device parameters identified by the parameter key and the corresponding parameter value. The table `IDACO_device_status` additionally has to be aware of the underlying session. Finally, each setup and each session belong to a certain kind of procedure. This enables automatically adjusting the relevant parameters contained in `IDACO_device_status` to the surgeon's preferred values contained in `IDACO_device_status` before and during the procedure. The implementation can be read up in Section 4.2.1 and 4.2.2.

4.2. Functionalities

In this section, we present all functionalities inherent to our system.

IDACO can be divided into two major parts. During the first one, the pre-procedural part, the system assists the surgery team preparing for the forthcoming operation by providing relevant information. Afterwards, IDACO escorts them throughout the whole procedure and provides assistance where necessary.

The main functionalities inherent to the system are:

- Reading the most important data from the database: surgery type, session team, general patient data, pre-diseases, medical treatment and laboratory data
- Saving preferred device settings for each surgeon, reading and changing the presets as well as transmitting the parameters to the devices (OR table, room light, insufflator, suction and irrigation unit)

4. The IDACO Spoken Language Operation Assistant

- Starting the insufflator, increasing the gas insufflation, turning off and on the light, tilting the table automatically
- Tracking the usage of several material (trocars and different types of clips), warning if the usage differs from the schedule
- Emergency mode

Each of these functions is specified in the following sections.

4.2.1. Pre-procedural Part

After launching the IDACO spoken language operation assistant, the system first confirms its successful start. Then, IDACO waits for the user to greet. After IDACO replied the greeting, the interaction is started. Figure 4.2 shows the information the user can request from the system.

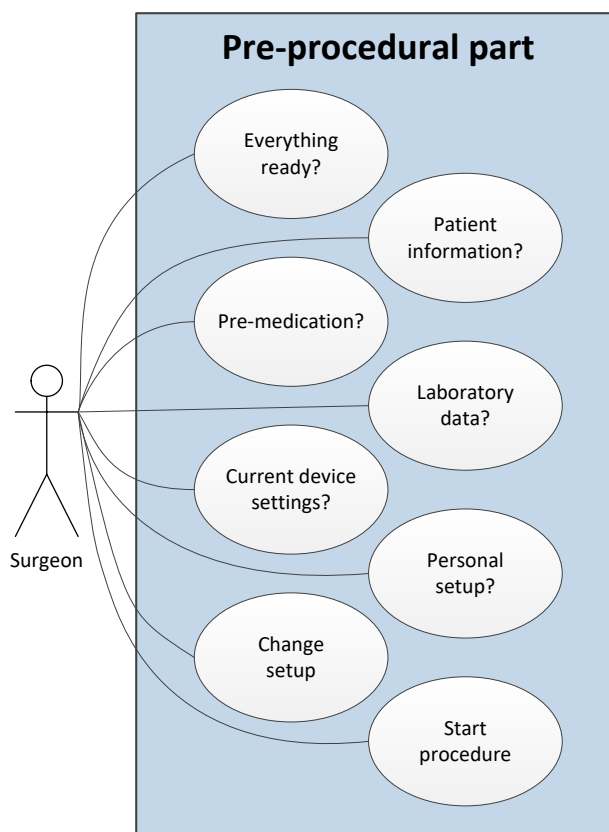


Figure 4.2.: Use case for the pre-procedural dialogue part

Flexible preparation for the procedure

Like already indicated, the user has the full control of how to start the interaction. Depending on the information he requests first, the dialogue initiative might switch to the system.

The variety of imaginable interaction sequences in this mixed-initiative dialogue part are illustrated in Figure 4.3. Each system prompt is depicted by a blue rectangle. Arrows with black labelling without square brackets represent user input leading to the next system turn. The green cylinders illustrate the database. Incoming arrows with green labelling represent writing information to the database, outgoing ones reading respectively. The white trapeziums describe setting a variable to a specific value, which is used for controlling the dialogue flow in the future interaction. Incoming arrows with black labelling surrounded by square brackets indicate that the corresponding variable value is required (e.g. [saidPatientData]) or must not be set (e.g. [¬saidPatientData]).

If the surgeon asks for example if everything is already prepared for the procedure, IDACO simply provides the corresponding information. If not everyone of the session team is present in the operating room yet, the user is informed about that and can either only wait for the staff or ask further questions in the meantime:

User: Ist denn schon alles vorbereitet?
 IDACO: Das OP-Team ist noch nicht komplett.
IDACO wartet erneut auf eine Nutzereingabe.

If the team is already complete, IDACO shares general information about the patient and the session team and directly poses a counter question:

User: Ist denn schon alles vorbereitet?
 IDACO: Es ist fast alles vorbereitet. Der Patient Max Mustermann ist gelagert, abgewaschen und abgedeckt. Doktor Kranzfelder und Schwester Steril stehen am Tisch bereit. Die Anästhesie wird von Doktor Hasenfuss durchgeführt. Soll ich den Tisch, den Thermoflator und die Lichtquelle auf Ihre gewünschten Voreinstellungen setzen?

The user is allowed to ask this question several times. Since our system is connected to the database, it always uses the current value to prove the presence of the surgery team. IDACO notices if the missing member of the staff enters the operating room and answers accordingly the next time it is asked if everything is already prepared for the procedure. Additional data can be read from the database via speech, including the patient's medication before the surgery and the current laboratory data. As it is not the most important patient data, both can be enquired but are optional to ask for, meaning that the system does not explicitly offer to provide it.

4. The IDACO Spoken Language Operation Assistant

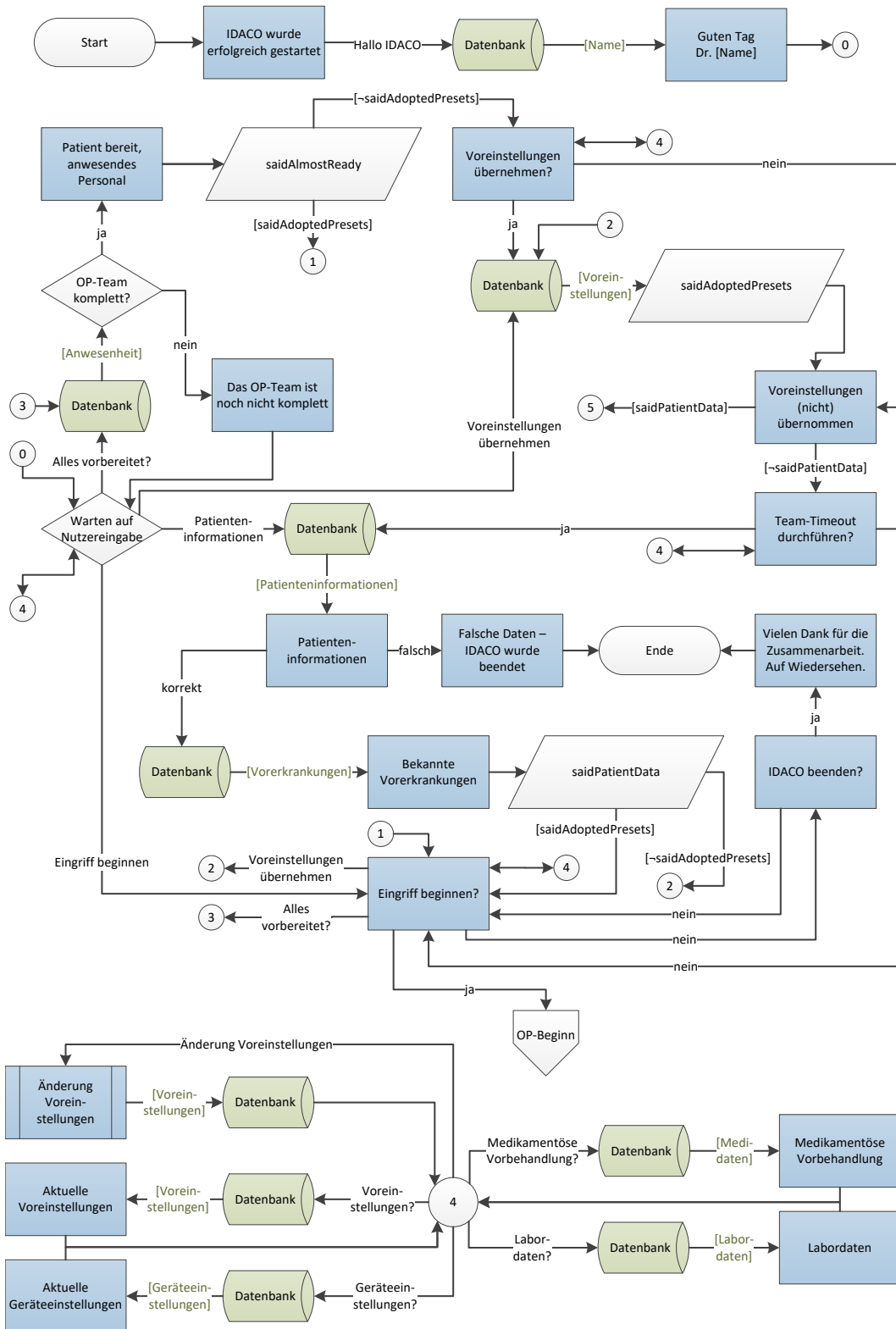


Figure 4.3.: Interaction possibilities for the pre-procedural dialogue part

If there is no information desired before the surgery begins, the information-providing pre-procedural dialogue part can be skipped completely. Consequently, the system is informed to start the procedure immediately. However, this is not recommended. During the team time-out, IDACO provides important information about the patient and the surgery that should be checked by the staff. If there is any wrong information, IDACO tells to review all information and thereupon ends the dialogue:

IDACO: Operiert wird heute Max Mustermann, 41 Jahre, männlich. Eingriff: Cholezystektomie. Bitte bestätigen.
User: Nein, das ist falsch.
IDACO: Offenbar liegt ein Fehler vor. Bitte überprüfen Sie die Patientendaten.
IDACO wird beendet.

After the team checked the patient and surgery information, the system has to be started anew. The use of the preparation part may prevent errors like having prepared the patient for the wrong surgery.

The first dialogue part is designed in such a way that IDACO is capable of providing as much information as possible without pushing the user into a fixed dialogue structure making the interaction inflexible and unnatural. The aim was to give the user the possibility to ask anything before switching the dialogue strategy by posing counter questions. Thus, the system reaction always depends on the topics that have been addressed in earlier user or system utterances. The user is able to lead the system in another direction at almost every point of the conversation, allowing a more natural human-machine-interaction. Additionally, he gets to know the system options and is free to decide which information to retrieve in which order and which to refuse.

Personal device settings

Each surgeon has the possibility to store his personally preferred settings for the devices involved in the surgery. Kranzfelder et al. [18] compare this to the automobile industry, where the car adjusts to each driver's individual settings. IDACO allows presettings for the following devices: OR table, room light, insufflator and the suction and irrigation unit. These are the ones needed for the laparoscopic cholecystectomy. Of course, this list can be expanded for other surgery types and needed devices.

Depending on the dialogue flow during the pre-procedural part, as described in the previous section, the surgeon can ask the system to adopt the devices in use to his presettings or IDACO asks him if he wants to do so.

The user can ask for the current device settings or for his personal presettings stored in the database. IDACO then reads out the current values.

If he wants to change his personal set-up for a specific device, he can also do this by speech. IDACO then requests the different parameter values one after another. The following is a sample dialogue for changing the setup for the insufflator:

User: Ich möchte gerne die Voreinstellungen für den Gasinsufflator ändern.

4. The IDACO Spoken Language Operation Assistant

IDACO: Bitte geben Sie den gewünschten Zieldruck des Thermoflators an.
User: Stell ihn doch bitte auf 15 mmHg.
IDACO: Geben Sie nun bitte den Maximalfluss an.
User: 17.
IDACO: Doktor Feußner, soll ich die Voreinstellungen für den Thermoflator speichern?
Andernfalls werden die neuen Werte verworfen.
User: Ja, bitte.
IDACO: Ihre Änderungen wurden gespeichert.

These presettings can be used in the future.

4.2.2. Procedural Part

After having obtained information and handled the device settings in the first dialogue section, the operational part is started by confirming the system's question if the surgery should begin. The following part diverges a lot from the pre-procedural one. While we wanted to design a preferably natural and flexible dialogue in the first part, the aim of the second part is to model a complex surgery structure as exactly as possible. This is a quite difficult task as the surgical domain is rather sensitive and does not allow for big errors.

A summary of all interaction possibilities is illustrated in Figure 4.4. Colours and shapes in this figure are used the same way as in Figure 4.3. All inherent functionalities are described in detail in the following sections.

Tracking the course of the procedure

In order to enable proactive and intelligent system behaviour, IDACO has to keep up with the current status as well as all previous actions at any time during the surgery [20]. In contrast to most existing systems (see Chapter 3), IDACO observes the entire procedure by listening to everything the operating surgeon says. The system tracks how often a tool or some kind of assistance was already demanded by the surgeon during the ongoing procedure. Since standardized procedures can be characterized by a specific sequence of instruments [17], the workflow and hence the current part of the operation can be derived from the history of used tools at any point of the surgical intervention [22].

Emergency mode

Sometimes during a procedure, the surgeon might have to do something out of schedule possibly confusing the system. Therefore, it is permitted to activate the so-called emergency mode at each arbitrary point of the operation. In this mode, IDACO does not listen any more and hence does not track the procedure until it is explicitly told to end it. Meanwhile, the user is free to do anything he wants without being disturbed by the system not being able to properly follow the conversation any more [20].

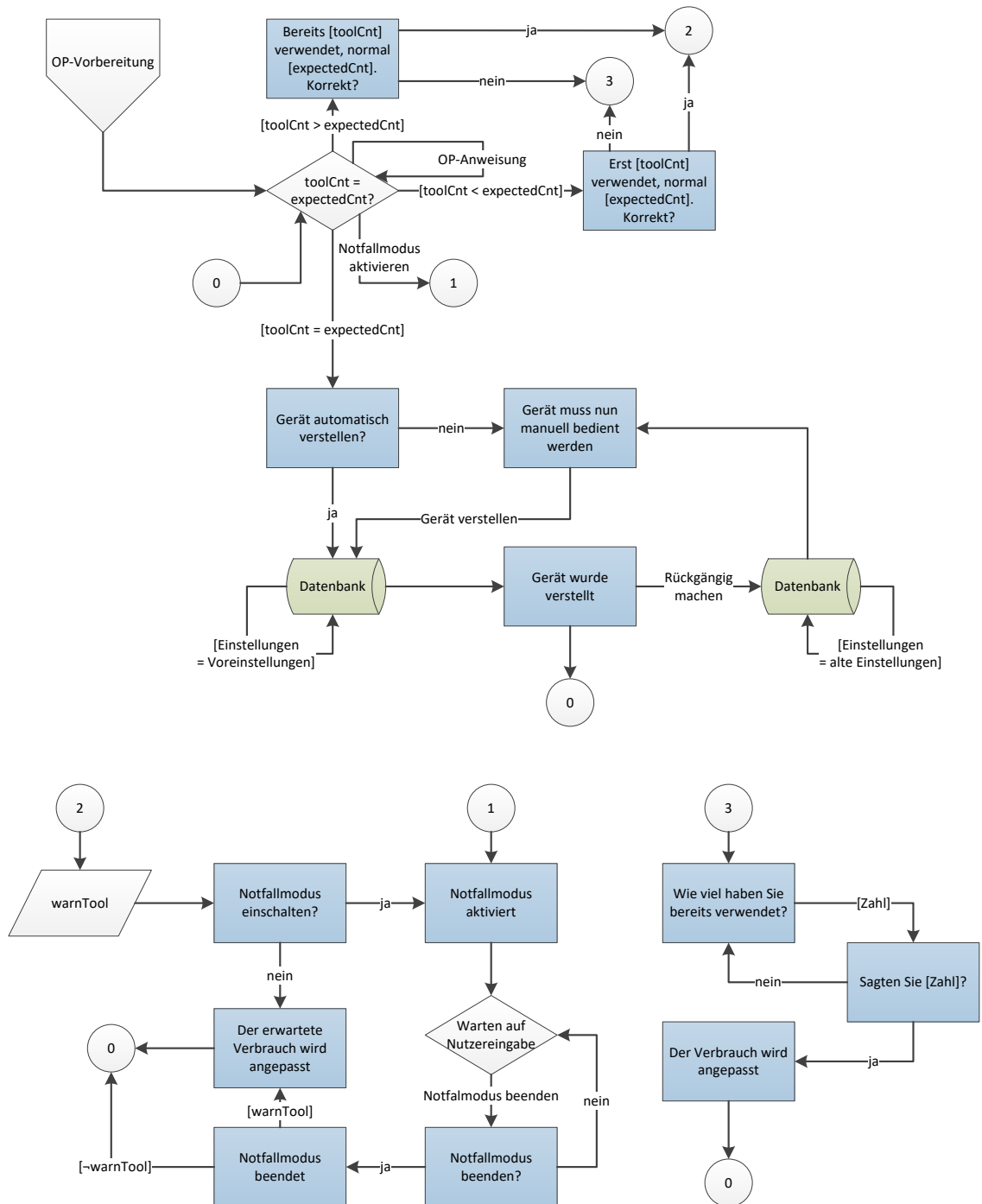


Figure 4.4.: Interaction possibilities for the procedural dialogue part

4. The IDACO Spoken Language Operation Assistant

IDACO might not wait for the explicit request for the emergency mode, but act proactively and offer its start in delicate situations. This is further described in the next section.

Handling deviation from surgery schedule

To understand what is meant by deviation from the surgery schedule, we first have to define what a correct surgery course is. We assume that a regular surgery is a fixed course of events that can be divided into multiple parts. In each part, the different steps the surgeon executes are observed (i.e. the system tracks every tool or assistance the surgeon asks for during the ongoing surgery) and compared to the given surgery plan [6]. However, the aim is not simply to collect and record intraoperative data, but to use this data in order to develop a situation-aware system which is able to recognize whether the ongoing operation still follows the plan or is deviating from the schedule [17]. As already described in Section 4.2.2, keeping up with the current situation in our approach means being aware of the history of used instruments.

IDACO recognizes a deviation from the surgery schedule if the user requests a tool or asks for assistance more often than expected in the current part of the surgery or if he tries to start the next phase before having given all commands scheduled for the current section. It reacts proactively by warning the user in such a way that it tells him what number of times he should have given this specific command and comparing it to the actual amount counted [13]. The surgeon is asked whether everything was understood correctly. If there was a misunderstanding, the user can correct the number the system counted and the procedure takes its normal course again. An exemplary dialogue fragment for this situation can be seen in the following:

User: Noch einen Trokar.
IDACO: Achtung, Sie haben nun bereits 5 Trokare verwendet. Der übliche Verbrauch bis zur Präparation des Calot-Dreiecks während diesem Eingriff liegt bei 4. Ist das korrekt?
User: Nein, da hast du mich offenbar falsch verstanden.
IDACO: Wie viele Trokare haben Sie bisher verwendet?
User: Erst 3.
IDACO: Sagten Sie 3?
User: Ja, das ist richtig.
IDACO: Der Verbrauch wird angepasst.

If the system is right about the deviation from the schedule, IDACO suspects a critical incident and suggests starting the previously mentioned emergency mode (see Section 4.2.2). After the emergency mode is deactivated or if the user refused it in the first place, IDACO adopts the amount of expected assistance and tool usage to the number it has recognized up to the warning. After that, the surgery can continue as usual.

Subsequently, a sample dialogue is shown for using too much clips when clipping the ductus cysticus. The user confirms starting the emergency mode. When the surgery is

under control again, he deactivates the emergency mode and the system adjusts to the situation by expecting a higher amount of clips in the future interaction.

Nutzer: Clip, bitte.

IDACO: Achtung, Sie haben nun bereits 7 Clips verwendet. Der übliche Verbrauch bis zum Clippen des Ductus Cysticus während diesem Eingriff liegt bei 6. Ist das korrekt?

Nutzer: Genau.

IDACO: Soll ich den Notfallmodus aktivieren?

Nutzer: Ja.

IDACO: Notfallmodus aktiviert.

IDACO ist nun in einem passiven Zustand, in dem die Operation nicht beobachtet wird.

Nutzer: [...]

IDACO, schalte bitte den Notfallmodus aus.

IDACO: Soll ich den Notfallmodus beenden?

Nutzer: Ja, bitte.

IDACO: Notfallmodus beendet.

Der registrierte Verbrauch von Clips wird angepasst.

Der Eingriff wird wieder von IDACO verfolgt.

In order to decide whether the procedure still goes according to plan, we made a distinction between different types of tools and material. For several tools, it does not matter if they are used more than according to the operation schedule. For example, if the surgeon asks for the knife twice does not need to be of importance. Maybe the cut was a little bit too short in the first place and has to be corrected by using the knife once more. IDACO surely does not need to intervene in this case and would annoy the operation team with unnecessary warnings.

For other tools, however, a deviation could be an indicator of something going wrong. Thus, the system warns about unusual usage of material that should not be needed in a great extent. For the laparoscopic cholecystectomy this concerns trocars and different types of clips (i.e. Backhaus, Alice, Pean and absorbable clips). If some of these are used more or less often than expected, the previously described subdialogues are started.

Intelligent device control

In state-of-the-art operating rooms nowadays, a lot of personnel has to be present at all times including additional persons to the operating surgeon and his medical assistants. The technical environment comprises many medical devices which have to be taken care of. Until now, most of the adjustments have to be done by hand during the surgery and could rather be executed automatically by an intelligent assistance system.

The goal set by Feussner et al. [6] is an intelligent system that handles these things not only by explicit commands. It should act active-cooperatively and thus support the surgeon autonomously during the procedure. In the sense of Intelligent Environments, this could be seen as the most important feature of our system.

4. The IDACO Spoken Language Operation Assistant

For the laparoscopic cholecystectomy we detected the following actions to be fitted for IDACO executing them automatically:

- Starting the insufflator
- Increasing the gas insufflation
- Turning off the light and tilting the table to a specific angle
- Turning on the light again
- Bringing the table back to a straight position and raising it 25 centimetres at the end of the procedure

Figure 4.5 summarizes the possible dialogue flows relating to device control. Colours and shapes in this figure are used the same way as in Figure 4.3. In the following, each part is briefly described.

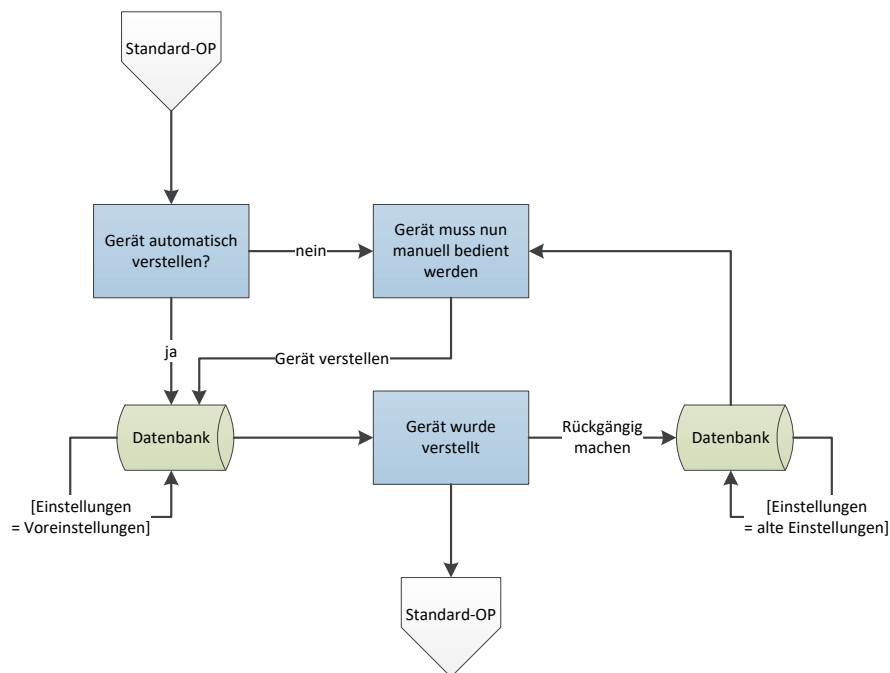


Figure 4.5.: Automatic and manual device control during surgery

As might be familiar from Section 4.2.2, IDACO is always aware of the situation and knows when all these actions should be executed. Hence, it asks at the right point of time during the surgery whether the system should take control of the corresponding device.

IDACO: Soll ich die Gasinsufflation auf den Maximalwert steigern?

User: Ja.

IDACO: Die Gasinsufflation ist nun auf dem Maximum.

If the surgeon does not want to change the settings automatically, e.g. if he needs more time to perform the current task, he can do it later manually. IDACO then waits for the appropriate command:

IDACO: Soll ich die Gasinsufflation auf den Maximalwert steigern?
 User: Nein.
 IDACO: Das Gas muss nun manuell hochgeschaltet werden.
IDACO wartet, bis der Benutzer die Anweisung gibt, den Gasfluss zu erhöhen.
 User: Volle Lotte.
 IDACO: Die Gasinsufflation ist nun auf dem Maximum.

Unfortunately, most state-of-the-art speech recognizers are still far from perfect. Thus, misunderstandings can occur which might, depending on the current situation, lead to more or less serious errors. In order to reduce the resulting cost of error, in our system, the user is able to undo any automatic action immediately, which was suggested by Shafer et al. [26]. The surgeon then has to give the command to execute the respective action eventually, which is analogue to having denied the automatic action in the first place.

4.3. Implementation

In the previous section, we described all functionalities inherent to IDACO. Now we take a look on how they were realized using OwlSpeak.

4.3.1. Interaction flexibility and subdialogues

Trying to reach a high amount of flexibility, the implementation of the first dialogue part turned out to be quite complex. To handle the multitude of interaction possibilities we need to use several variables. One important variable is `varPreOpPart`, which is set in each regular agenda (i.e. no subdialogue, see below) and used to remember the current point of the conversation in case of starting a subdialogue such as demanding additional information or asking for changing the personal settings.

As an example, the following code fragment shows the implementation of IDACO asking whether to do the team time-out. The utterance `moveSysReqTeamTimeOut` is performed while setting `varPreOpPart` to `reqTeamTimeOut`:

```
moveSysReqTeamTimeOut.setVariableOperator("SET("
    + varPreOpPart + "= reqTeamTimeOut)");
```

In order to start a certain subdialogue, the variable `varInfo` or `varChange` is set, depending on whether the user retrieves information or wants to change his presets. When ending the subdialogue, the system resets the corresponding variable to a default value. In case of the surgeon asking for the patient's medication information, which is represented

4. The IDACO Spoken Language Operation Assistant

by the grammar move `moveUserMedication`, the variable `varInfo` is set to `medication`. The corresponding agenda `agendaInfoMedication` requires this variable value accordingly. Thus, it can only be started if the right user move was performed. After IDACO provided the information given in the utterance move `moveSysMedication`, `varInfo` is reset to the default value `noInfo`.

```
moveUserMedication.setVariableOperator("SET(" + varInfo + "= medication)");
agendaInfoMedication.setVariableOperator("",
    "REQUIRES(%var_Info% == medication)");
moveSysMedication.setVariableOperator("SET(" + varInfo + "= noInfo)");
```

The most complex part of the implementation was handling the variety of possible dialogue sequences (see Figure 4.3) and determining which system prompt has to be uttered next. For this purpose, IDACO has to keep in mind the whole dialogue history. Therefore, we created a variable for each topic, storing whether it was already addressed or not. For example, if the user asks for information about the patient (i.e. the grammar move `moveUserPatientInfo` is performed), `varSaidPatientData` is set to the temporary value `req` for the start of this dialogue part:

```
moveUserPatientInfo.setVariableOperator("SET("
    + varSaidPatientData + "= req)");
```

First, general patient information is given. The user has to verify the correctness of the data and afterwards, IDACO tells him about the patient's prediseases. After the corresponding utterance move `moveSysInfoDiseases` is performed, `varSaidPatientData` is set to 1 and `varPreOpPart` is updated:

```
moveSysInfoDiseases.setVariableOperator("SET("
    + varSaidPatientData + "= 1; "
    + varPreOpPart + "= infoDiseases)");
```

Setting `varSaidPatientData` to 1 ensures that IDACO does not offer giving this information in the future. This request, which belongs to the agenda `agendaReqTeamTimeOut`, therefore requires the variable value 0 for `varSaidPatientData`, which can be seen in the code fragment below. This example also shows the complexity of the requirements for each agenda, enabling a flexible dialogue.

```
agendaReqTeamTimeOut.setVariableOperator("", "REQUIRES("
    + "((%var_Info% == noInfo) && (%var_Change% == noChange)) && "
    + "(((%var_PreOpPart% == adoptedPresets) && "
    + "((%var_SaidAdoptedPresets% == 1) && "
    + "(%var_SaidPatientData% == 0))) || "
    + "((%var_PreOpPart% == reqTeamTimeOut) && "
    + "(%var_SaidPatientData% == 0)))"
    + ")");
```

The agenda can only be started, if there is no subdialogue at the moment, i.e. `varInfo` and `varChange` must have their default values, `noInfo` and `noChange` respectively. There are two possible scenarios, when `agendaReqTeamTimeOut` is started.

Firstly, if the system just adapted the devices to the surgeon's presettings, `varPreOpPart` has the value `adoptedPresets` and `varSaidAdoptedPresets` is 1. If the system additionally did not give any patient information in the previous interaction (i.e. the variable `varSaidPatientData` is 0), `agendaReqTeamTimeOut` is started and IDACO suggests to do the team time-out.

The second scenario for starting `agendaReqTeamTimeOut` is that the system already offered to do the team time-out and thus the variable `varPreOpPart` has the value `reqTeamTimeOut`. If the user started a subdialogue immediately, IDACO did not give the patient information yet and `varSaidPatientData` is still 0. Ending the subdialogue and returning to the normal course of the dialogue, the system repeats the agenda and its question.

In the procedural dialogue part, subdialogues are implemented analogically for a deviation from the surgery schedule. After prompting a warning, the system offers the surgeon different possibilities to continue the dialogue.

4.3.2. Adapting the device settings

In order to adjust the system to the surgeon's preferred device settings, the variables storing the device settings have to be set to the values of the corresponding variables for the user's presettings. This is realized in the following code fragment:

```
moveUserConfirmPresets.setVariableOperator("SET("
+ varDB_dev_Thermoflator_TargetPressure
+ "= %varDB_setup_Insufflator_TargetPressure%; "
+ varDB_dev_RoomLight_TargetIntensity
+ "= %varDB_setup_Lichtquelle_TargetIntensity%; "
+ varDB_dev_Endomat_TargetFlow
+ "= %varDB_setup_Saug-Spuel-Einheit_TargetValue-F%; "
+ varDB_dev_Table_Height + "= %varDB_setup_Tisch_TableHeight%; "
+ varDB_dev_Thermoflator_TargetFlow + "= 0; "
+ varDB_dev_Table_Tilt + "= 0)");
```

The variables storing the target pressure of the gas insufflator, the target intensity of the room light, the target flow of the suction and irrigation unit and the table height are set to the values of the corresponding variables for the user's presettings. In the beginning, the target flow of the insufflator and the table tilt are set to 0 instead of their corresponding presettings, because they have to be adjusted later during the surgery.

The intelligent device control during the procedure is implemented likewise. The system communicates with the database and writes the needed values to the actual device settings. Depending on the task at hand, IDACO uses fixed values (e.g. the variable value 0 for turning the light off) or predefined values stored for the operating surgeon.

4.3.3. Writing to the database

The device settings and presettings are stored in variables of the ontology, which are then passed on to the database (see Section 4.1). Therefore, the dialogue manager had to be extended.

When IDACO is used, OwlSpeak synchronizes the database with chosen ontology variables at the end of each system turn. This is only done for the device settings and presettings, as all other database information must not be overwritten.

4.3.4. Changing variable values

If the surgeon wants to change his personal device presettings, every value he dictates has to be stored in a temporary variable, in case he decides not to save it.

As an example, the following shows how this is implemented for changing the presettings for the target pressure of the gas insufflator to any value *n*. When the grammar `moveUserPresetThermoflatorTargetPressureToN` is performed, *n* is temporarily stored in `varTempPresetThermoflatorTargetPressure`:

```
moveUserPresetThermoflatorTargetPressureToN.setVariableOperator("SET("
    + varTempPresetThermoflatorTargetPressure + "=" + n + ")");
```

Changing all parameters for one device at once, the next user move is to change the presettings for the target flow of the insufflator. This is done analogically.

If the user finally denies taking the changes, the old values ought to be kept. Therefore, we are not supposed to overwrite them directly. In this case, the temporary values are overwritten with a default value:

```
moveUserDenyPresetChangeThermoflator.setVariableOperator("SET("
    + varTempPresetThermoflatorTargetPressure + "= noTempChange; "
    + varTempPresetThermoflatorTargetFlow + "= noTempChange)");
```

If the user confirms the changes, the temporary variable values have to be written to the actual setup variables:

```
moveUserConfirmPresetChangeThermoflator.setVariableOperator("SET("
    + varPresetThermoflatorTargetPressure
    + "= %var_TempPresetThermoflatorTargetPressure%; "
    + varPresetThermoflatorTargetFlow
    + "= %var_TempPresetThermoflatorTargetFlow%)");
```

At the end of the system turn, OwlSpeak finally writes these values to the database, as described in the previous section.

However, this method is not only used for changing the device presettings. If the system misunderstands some surgery orders, the user might correct the amount of used material in order to ensure a persistent dialogue (see Section 4.2.2). This correction subdialogue is implemented analogically to the one described in this section.

4.3.5. Tracking surgery orders

In OwlSpeak, tracking the tool usage is done via variables for all kinds of instruments and assistance actions. These variables are incremented for each user utterance corresponding to its specific purpose.

The following code fragment shows the creation of a variable `varToolCnt` for counting the usage of a specific tool:

```
Variable varToolCnt = factory.createVariable("var_tool_cnt_" + shortName);
varToolCnt.setDefaultValue("", "0");

Move moveUserTool = factory.createMove("move_user_Tool" + shortName);
moveUserTool.setVariableOperator("INCREMENT(" + varToolCnt + ")");
```

At the beginning, its value is set to 0. For the surgery order to ask for this tool, a user move is created. The so-called variable operator says that each time the move is done, the corresponding variable has to be incremented. This is how IDACO tracks all commands given by the surgeon.

OwlSpeak did not support incrementing variables initially, as there was no possibility to set variables while using arithmetic expressions. Even though the implementation was done in Java, the VariableOperator does not understand usual Java syntax like `varToolCnt++`, as the dialogue manager cannot process this internally. Therefore, it was necessary to add this functionality to OwlSpeak by introducing the new VariableOperator `INCREMENT` for counting tools (i.e. `varToolCnt++` in Java).

4.3.6. Monitoring the course of the procedure

We exemplary modelled the course of a laparoscopic cholecystectomy. As can be seen in the surgery plan in Appendix B, the procedure can be divided into ten major parts. Each of them is implemented in such a way that one part can only be started if the preceding part has been finished correctly.

In terms of implementation this means that each surgery phase starts with a new agenda having several preconditions that have to be fulfilled. There are two possibilities to realize this in OwlSpeak.

The first possibility is to set semantic requirements which means that a specific topic has to be addressed already in the dialogue up to now. Since a surgery is a rather straight process, this option does not seem to be the best fitted.

The other possibility is to demand that some variables have a specific value. As described previously, this method was applied for the first dialogue part as well. Having variables storing all surgery orders, this option was also the right choice for the second part of our system. Katić et al. [14] describe an approach concluding that single triggering events are not specific enough for initiating the next phase of the surgery. Therefore, we used multiple preconditions for our system.

The following example shows how this is implemented for the second phase of the surgery, which is creating the aeroperitoneum.

```
// (2) Erzeugen des Pneumoperitoneums
agendaOp2.setVariableOperator("", "REQUIRES("
+ "(%var_tool_cnt_Skalpell% >= 1) && "
+ "(%var_tool_cnt_Backhaus% == %var_ExpectedBackhaus1%) && "
+ "(%var_tool_cnt_Veress% == 0))");
```

In the first phase the surgeon has to ask at least once for the knife and a specific number of times for a Backhaus clip (for a normal course of the procedure, the variable `var_ExpectedBackhaus1` has the value 2, see Section 4.3.7). Since the Veress needle is first used in the following phase, it must not be mentioned yet, i.e. the variable `var_tool_cnt_Veress` is 0. If all conditions are fulfilled, the creation of the aeroperitoneum can begin, which means the agenda `agendaOp2` is started.

If the surgeon uses more Backhaus clips than expected or if he tries to start the next phase before having used the expected amount of Backhaus clips in the current one (i.e. `%var_tool_cnt_Backhaus% >> %var_ExpectedBackhaus1%` or `%var_tool_cnt_Backhaus% << %var_ExpectedBackhaus1%` resp.), the system utters a warning. Thus, a corresponding subdialogue is started (see Section 4.3.1).

4.3.7. Adapting the expected usage of material

In case the surgery deviates from the schedule, IDACO has to remember the extraordinary number of used material in the future. For some material (see Section 4.2.2), the system keeps variables describing the expected amount of usage at different points of the procedure. These variables then have to be adopted to the current usage.

The following code fragment shows how the expected usage of trocars is adjusted:

```
moveSysAdaptRequest2.setVariableOperator("SET("
+ varWarn + "= false; "
+ varEmergency + "= normal; "
+ varExpectedTrokar2 + "= %var_tool_cnt_Trokar%; "
+ varExpectedTrokar2b + "= (%var_tool_cnt_Trokar% ++ 1); "
+ varExpectedTrokar3 + "= (%var_tool_cnt_Trokar% ++ 3); "
+ varExpectedTrokar8c + "= (%var_tool_cnt_Trokar% ++ 4)");
```

Since the variables are adjusted at the end of the warning subdialogue, `varWarn + "= false"` and `varEmergency + "= normal"` indicate that it is no warning situation or an emergency any more.

For the current point of the procedure, the expected amount of used trocars (i.e. `varExpectedTrokar2`) is set to the actually counted number of trocars in use (i.e. `%var_tool_cnt_Trokar%`). The next time, the amount of used trocars is checked, the system expects one trocar more than at the current point of the procedure. Thus, the corresponding variable `varExpectedTrokar2b` is set to the current number counted plus one. This is done analogically for all future points of the surgery, when the trocar usage is checked.

The possibility to use arithmetic expressions in VariableOperators initially worked for single equations. In order to support the usage of multiple arithmetic expressions for one single move, the corresponding part of OwlSpeak had to be expanded accordingly.

4.3.8. Emergency mode

In general, if the system does not understand what the user says or if it is not able to assign the utterance to any grammar, the current agenda is repeated several times until a matching grammar is found. Since most agendas have a system move, it results in IDACO repeating the system prompt over and over again. Therefore, the emergency mode was designed as an agenda without any system move, so that the user does not notice the repeating of the agenda. There is only one possible user move, which is the user giving the command to deactivate the emergency mode.

This shall ensure that IDACO really supports the surgery team rather than annoying it with wrong system behaviour as a consequence of the lacking ability to catch all possible user utterances.

4.4. Reusability

As mentioned in Section 4.2, the system as a whole consists of two major parts.

The first part before the start of the procedure is designed in a rather general and flexible manner. Therefore, it is possible to use it not only for a specific surgery, but also for various operation types. Only the variables storing the device settings and the personal device setup have to be adjusted to the devices needed for the specific operation type, which can easily be done using an additional database table storing which devices to use for which kind of operation. If there is a new surgeon, additionally a personal setup table has to be created.

The second part of our prototype is a rather exact model of the surgery plan for a laparoscopic cholecystectomy. IDACO observes the entire procedure in detail, so that it is able to interfere if necessary (see Section 4.2.2). Thus, the general structure of the procedural part is unsuitable for later reuse and has to be implemented precisely for each different type of surgery.

However, this part contains some additional features which do not refer to any surgery specific data. The adoption of the counting variables to the actual tool usage when IDACO misunderstood the surgeon could be used for other surgeries as well. Especially the emergency mode as single component also seems to be fitted for later reuse.

5. Conclusion

In this work, we used the OwlSpeak dialogue manager in order to enable speech-based interaction with an existing intelligent operating room at the *Klinikum rechts der Isar, TU München*.

Existing research projects in the field of intelligent medical environments mostly concentrate on workflow modelling or single smart features rather than building up a complete Intelligent Environment. To the best of our knowledge, IDACO is the first intelligent spoken language operation assistant putting together several functionalities which provide the surgeon assistance in many different situations before and during an ongoing procedure.

Using a speech interface, the surgeon can concentrate on the surgery and control the technical environment at the same time, without taking care of how to interact with the system. IDACO assists during the preparation for the surgery and escorts the operating staff through the entire surgery. We modelled exemplary a laparoscopic cholecystectomy. The workflow and hence the current part of the operation are derived from the history of used tools at any point of the surgical intervention. Therefore, the system listens to each of the surgeon's instructions. The observed course of the procedure is compared to the surgery schedule. In case of a deviation from the schedule, IDACO reacts proactively. As the surgery schedule allows for few deviation, we introduced the possibility to set IDACO in a passive mode. This might be useful in order to operate without being disturbed in delicate situations.

Furthermore, the system is connected to a database enabling interaction with the devices contained in the OR. It is possible to adjust the devices in the OR to the surgeon's preferred settings. Being aware of the context of the surgery, IDACO controls several devices autonomously at the appropriate time during the procedure, which facilitates work and reduces the staff needed to assist during a surgical intervention. This again reduces personnel cost and might even lessen the rate of incidents caused by human error.

The system was presented to specialists from the *Klinikum rechts der Isar, TU München* and received positive feedback. The speech interface and the multitude of interaction possibilities were well received. Our system is designed not to annoy the operating staff with inappropriate behaviour and unnecessary system prompts during the surgery. Therefore, IDACO stays in the background if the procedure goes as scheduled. Despite our expectations, passive system behaviour makes the operating staff insecure. They prefer any kind of feedback about what the system understands. It was suggested to equip the scrub nurse with a tablet PC showing each user utterance the system receives. Thus, the system behaviour can be observed and compared to the schedule manually.

The highly specific field of applications makes it hard for non-specialists to decide about the best system behaviour. The evaluation made clear that a close cooperation between

5. Conclusion

system developers and medical scientists is inevitable in projects like this.

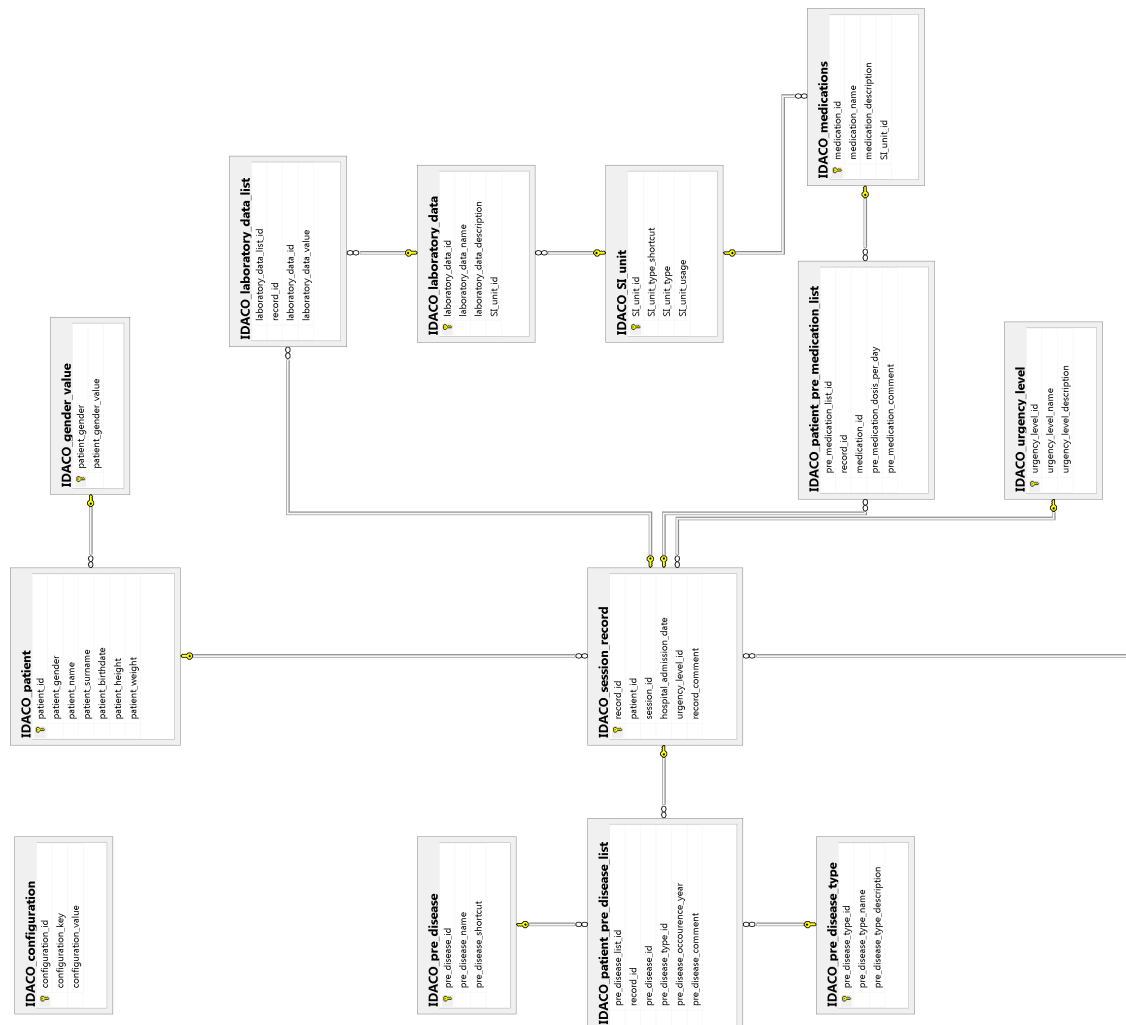
As described in Section 4.4, the pre-operational part of our system can be reused for different kinds of surgical interventions. However, the procedural part is a rather exact model of a complex surgery structure. The multitude of existing surgical procedures makes it impossible to implement each one individually. Thus, the goal is to find a generic method for modelling the surgery schedule.

In order to test our system, we used a VoiceXML interpreter and ASR and TTS system by Voxeo¹. Unfortunately, like most ASR and TTS systems, it is not suited for German and particularly medical applications. It supports a rather limited vocabulary and makes some system utterances hard to understand. Thus, better ASR and TTS modules especially developed for German medical applications would be desirable for future work.

With our system, we demonstrated the power of Spoken Language Dialogue Systems and context-aware intelligent systems in surgical environments. Even though we made a good attempt, there is still much work to be done to finally create the *Operating Room of the Future*.

¹Available: <https://evolution.voxeo.com/>

A. IDACO Database



B. Surgery Schedule for a Laparoscopic Cholecystectomy

Anweisung (Typische Formulierung und mögliche Alternativen)	Adressat	Aktion
---	----------	--------

I Anlage der Hautinzision, Einbringen des ersten Trokars		
Messer, bitte (Skalpell!) (Schnitt!)	OP-Sr.	OP-Sr. reicht das Skalpell
Backhaus (Klemme) (Backhauser)	OP-Sr.	OP-Sr. reicht die erste Backhaus-Klemme an
Zweite Backhaus (Backhaus) (Zweite) (Nächste)	OP-Sr.	OP-Sr. reicht die zweite Backhaus-Klemme an

II Erzeugen des Pneumoperitoneums		
Veress-Nadel (Gas) (Nadel)	OP-Sr.	OP-Sr. reicht die Veress-Nadel an
Gas einschalten (Insufflator Start) (Gas)	Springer	Springer startet die Gasinsufflation
Trokar	OP-Sr.	OP-Sr. reicht den ersten Arbeitstrokar
Gasanschluss	OP-Sr.	OP-Sr. reicht den Gasschlauch an

III Einführen der Optik und Setzen der weiteren Trokare		
Optik	OP-Sr.	OP-Sr. reicht das Teleskop an
Gas hochschalten (Gas hoch) (Volle Lotte)	Springer	Springer steigert die Gasinsufflation auf den Maximalwert
Probiernadel	OP-Sr.	OP-Sr. reicht Kanüle an
Trokar	OP-Sr.	OP-Sr. reicht den zweiten Trokar an
Licht aus	Springer	Springer löscht das Raumlicht
Probier	OP-Sr.	OP-Sr. reicht Kanüle an
5 mm	OP-Sr.	OP-Sr. reicht den dritten 5 mm-Trokar an
Reinigen	Assistent I	Der Assistent reinigt die Optik

B. Surgery Schedule for a Laparoscopic Cholecystectomy

Umsetzen	Assistent I	Der Assistent setzt die Optik in den zweiten Trokar, der eingebracht wurde
Probier	OP-Sr.	OP-Sr. reicht Kanüle an
Trokar	OP-Sr.	OP-Sr. reicht den nun vorerst letzten 5 mm-Trokar an
Taststab	OP-Sr.	OP-Sr. reicht den Taststab an
Das ist jetzt Ihr Job	Assistent II	Der zweite Assistent bekommt den Taststab in die Hand zur besseren Exposition des Operationsfeldes

IV Präparation des Calot-Dreiecks		
Fasszange	OP-Sr.	OP-Sr. reicht die Fasszange an
PE	OP-Sr.	OP-Sr. reicht die PE-Zange an
Mittig	Assistent I	Assistent wird angewiesen, das gewünschte Operationsgebiet immer im Fokus der Kamera zu behalten
Ko	OP-Sr.	OP-Sr. appliziert Strom an der PE-Zange
Ko	OP-Sr.	OP-Sr. appliziert Strom an der PE-Zange (erfolgt während der gesamten Präparation als Anweisung)

V Clippen der Arteria Cystica		
Clip	OP-Sr.	OP-Sr. reicht einen resorbierbaren Clip an
Noch einen	OP-Sr.	OP-Sr. reicht einen resorbierbaren Clip an
Clip	OP-Sr.	OP-Sr. reicht den dritten Clip an
Schere	OP-Sr.	OP-Sr. reicht die Schere an
PE	OP-Sr.	OP-Sr. reicht die PE-Zange an
Ko	OP-Sr.	OP-Sr. appliziert Strom

VI Clippen des Ductus Cysticus		
Clip	OP-Sr.	OP-Sr. reicht die Clipzange an
Noch einen	OP-Sr.	OP-Sr. reicht erneut die Clipzange an
Und noch einen	OP-Sr.	OP-Sr. reicht die Clipzange ein drittes Mal an
Schere	OP-Sr.	OP-Sr. reicht die Schere an

VII Auslösen der Gallenblase aus dem Leberbett		
PE	OP-Sr.	OP-Sr. reicht die PE-Zange an
Abstand	Assistent I	Der Assistent geht mit der Optik auf Abstand, während der Operateur mit der Fass- bzw. PE-Zange nachgreift

Ko	OP-Sr.	OP-Sr. appliziert Strom an der PE-Zange (erfolgt während der gesamten Präparation als Anweisung)
----	--------	--

VIII Bergen der Gallenblase		
Bergebeutel	OP-Sr.	OP-Sr. reicht einen Bergebeutel an
PE	OP-Sr.	OP-Sr. reicht die PE-Zange an
Licht an	Springer	Springer schaltet Raumlicht an
Gas umsetzen	Assistent I	Der Assistent bringt den Gasschlauch am Trokar Nr. 2 an
Schere	OP-Sr.	OP-Sr. reicht die Schere an
Pean	OP-Sr.	OP-Sr. reicht eine Pean-Klemme an
Pean	OP-Sr.	OP-Sr. reicht eine weitere Pean-Klemme an
Schere	OP-Sr.	OP-Sr. reicht eine Lexer-Schere an
Umlage	OP-Sr.	OP-Sr. reicht eine Kompresse an
Elise	OP-Sr.	OP-Sr. reicht eine Alice-Klemme an
Zweite	OP-Sr.	OP-Sr. reicht eine zweite Alice-Klemme an
Sauger für Sie	OP-Sr.	OP-Sr. reicht dem Assistenten einen Metallsauger an
Schere	OP-Sr.	OP-Sr. reicht dem Operateur erneut die Lexer-Schere
Saug	Assistent I	Der Assistent saugt die Galle ab, nachdem die Gallenblase vom Operateur indiziert wurde
Alice	OP-Sr.	OP-Sr. reicht eine Alice-Klemme an
Sauber machen	OP-Sr.	OP-Sr. reinigt mit einer Kompresse die Alice-Klemme von Galle und Steinmaterial
Sauber machen	OP-Sr.	OP-Sr. reinigt erneut die Alice-Klemme
Pean	OP-Sr.	OP-Sr. reicht evtl. noch eine Pean-Klemme an
Finger drauf	Assistent I	Der Assistent dichtet mit einem Finger die Trokarstelle am Bauchnabel nach endgültiger Bergung der Gallenblase ab
Trokar	OP-Sr.	OP-Sr. reicht erneut den 10 mm-Trokar an
Gas ran	Assistent I	Der Assistent bringt den Gasschlauch wieder an dem 10 mm-Trokar an
Taststab	OP-Sr.	OP-Sr. reicht den Taststab an

IX Blutstillung im Gallenblasenbett		
Saug-Spül	OP-Sr.	OP-Sr. reicht die Saug-/Spül-Einheit an
Elefantenfuß	OP-Sr.	OP-Sr. reicht den Handsauger elektrisch an

B. Surgery Schedule for a Laparoscopic Cholecystectomy

Ko	OP-Sr.	OP-Sr. appliziert Strom
Ko	OP-Sr.	OP-Sr. appliziert erneut Strom (Vorgang wiederholt sich während der Blutstillung)
Jetzt noch am Hilus	Assistent I	Der Assistent richtet die Kamera auf den Hilus aus, während der Operateur nach Blut-trockenheit schaut
Hinter der Leber	Assistent I	Der Assistent schaut mit der Kamera noch einmal hinter die Leber, während der Operateur die restliche Spülflüssigkeit absaugt

X Ende der Operation		
Trokare unter Sicht raus	Assistent I	Der Operateur entfernt die Trokare nacheinander unter jeweiliger Inspek-tion der Einstichstellen von Innen durch den Assistenten
Finger drauf	Assistent II	Der zweite Assistent wird angewiesen, die Trokarstellen mit den Fingern abzudichten
Drainage	OP-Sr.	OP-Sr. reicht die 20er-Robinson-Drainage an, welche anschließend durch den 10 mm-Trokar am Bauchnabel eingebracht wird
Trokare unter Sicht raus	Assistent I	Der Assistent inspiziert noch einmal beim Entfernen der Trokare die Einstichstellen
Vielen Dank	Alle	Ende der Operation

Bibliography

- [1] <http://www.intenv.org/>. Accessed: 2017-05-13.
- [2] Sheetal Agarwal, Anupam Joshi, Tim Finin, Yelena Yesha, and Tim Ganous. A Pervasive Computing System for the Operating Room of the Future. *Mobile Networks and Applications*, 12(2-3):215–228, 2007.
- [3] Rasiah Bharathan, Rajesh Aggarwal, and Ara Darzi. Operating room of the future. *Best practice & research. Clinical obstetrics & gynaecology*, 27(3):311–322, 2013.
- [4] M. K. Chmarra, C. A. Grimbergen, and J. Dankelman. Systems for tracking minimally invasive surgical instruments. *Minimally invasive therapy & allied technologies*, 16(6):328–340, 2007.
- [5] Laila Dybkjær, Niels Ole Bernsen, and Wolfgang Minker. Evaluation and Usability of Multimodal Spoken Language Dialogue Systems. *Speech Communication*, (43):33–54, 2003.
- [6] H. Feussner, S. B. Reiser, M. Bauer, M. Kranzfelder, R. Schirren, J. Kleeff, and D. Wilhelm. Technische und digitale Weiterentwicklung in der laparoskopischen/offenen Chirurgie. *Der Chirurg; Zeitschrift für alle Gebiete der operativen Medizin*, 85(3):178, 180–185, 2014.
- [7] Hubertus Feussner. The Operating Room of the Future: A View from Europe. *Surgical Innovation*, 10(3):149–156, 2003.
- [8] Hubertus Feussner and Dirk Wilhelm. Minimalinvasive Chirurgie und ”robotic surgery”: Chirurgie 4.0? *Der Chirurg; Zeitschrift für alle Gebiete der operativen Medizin*, 87(3):189–194, 2016.
- [9] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An Ontology-based Context Model in Intelligent Environments. In Montgomery T. Shaw and William J. MacKnight, editors, *Introduction to Polymer Viscoelasticity*, pages 1–6. John Wiley & Sons, Inc, Hoboken, NJ, USA, 2005.
- [10] Pablo A. Haya, Germán Montoro, and Xavier Alamán. A Prototype of a Context-Based Architecture for Intelligent Home Environments. In David Hutchison et al., editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290 of *Lecture Notes in Computer Science*, pages 477–491. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

Bibliography

- [11] Tobias Heinroth, Dan Denich, and Alexander Schmitt. OwlSpeak - Adaptive Spoken Dialogue within Intelligent Environments. In Christian Becker, editor, *8th IEEE International Conference on Pervasive Computing and Communications workshops (PerCom workshops)*, 2010, pages 666–671, Piscataway, NJ, 2010. IEEE.
- [12] Tobias Heinroth and Wolfgang Minker. *Introducing Spoken Dialogue Systems into Intelligent Environments*. Springer, New York, NY, 2013.
- [13] Hansjörg Hofmann, Ute Ehrlich, and André Berton. Proactivity in Spoken Dialog Systems. In Dirk Schnelle-Walka, Stefan Radomski, Jochen Huber, Oliver Brdiczka, Kris Luyten, and Max Mühlhäuser, editors, *3rd IUI Workshop on Interacting with Smart Objects*, pages 7–10, 2014.
- [14] Darko Katić, Anna-Laura Wekerle, Fabian Gärtner, Hannes Kenngott, Beat Peter Müller-Stich, Rüdiger Dillmann, and Stefanie Speidel. Knowledge-Driven Formalization of Laparoscopic Surgeries for Rule-Based Intraoperative Context-Aware Assistance. In David Hutchison et al., editors, *Information Processing in Computer-Assisted Interventions*, volume 8498 of *Lecture Notes in Computer Science*, pages 158–167, Cham, 2014. Springer International Publishing.
- [15] Michael Kranzfelder, Armin Schneider, Adam Fiolka, Sebastian Koller, Silvano Reiser, Thomas Vogel, Dirk Wilhelm, and Hubertus Feussner. Reliability of sensor-based real-time workflow recognition in laparoscopic cholecystectomy. *International journal of computer assisted radiology and surgery*, 9(6):941–948, 2014.
- [16] Michael Kranzfelder, Armin Schneider, Adam Fiolka, Elena Schwan, Sonja Gillen, Dirk Wilhelm, Rebecca Schirren, Silvano Reiser, Brian Jensen, and Hubertus Feussner. Real-time instrument detection in minimally invasive surgery using radiofrequency identification technology. *The Journal of surgical research*, 185(2):704–710, 2013.
- [17] Michael Kranzfelder, Armin Schneider, Sonja Gillen, and Hubertus Feussner. New technologies for information retrieval to achieve situational awareness and higher patient safety in the surgical operating room: the MRI institutional approach and review of the literature. *Surgical endoscopy*, 25(3):696–705, 2011.
- [18] Michael Kranzfelder, Christoph Staub, Adam Fiolka, Armin Schneider, Sonja Gillen, Dirk Wilhelm, Helmut Friess, Alois Knoll, and Hubertus Feussner. Toward increased autonomy in the surgical OR: needs, requests, and expectations. *Surgical endoscopy*, 27(5):1681–1688, 2013.
- [19] Wolfgang Minker. Slides for the lecture Dialogue Systems, Winter term 2016–2017. Part 1: Spoken Language Dialogue Systems Overview.
- [20] Wolfgang Minker, Ramón López-Cózar, and Michael McTear. The role of spoken language dialogue interaction in intelligent environments. *Journal of Ambient Intelligence and Smart Environments*, 1(1):31–36, 2009.

- [21] OWL Working Group. Web Ontology Language (OWL). <https://www.w3.org/OWL/>, 2012. Accessed: 2017-05-14.
- [22] Nicolas Padoy. *Workflow and Activity Modeling for Monitoring Surgical Procedures*. Dissertation, Université Henri Poincaré, Nancy, 2010.
- [23] Joanne Pransky. An intelligent operating room of the future – an interview with the University of California Los Angeles Medical Center. *Industrial Robot: An International Journal*, 28(5):376–380, 2001.
- [24] D. B. Roe and Y. Wang. A voice-controlled network for universal control of devices in the OR. *Minimally invasive therapy & allied technologies : MITAT : official journal of the Society for Minimally Invasive Therapy*, 9(3-4):185–191, 2000.
- [25] Alexander Schmitt and Wolfgang Minker. *Towards Adaptive Spoken Dialog Systems*. Springer, New York, 2013.
- [26] Steven A. N. Shafer, Barry Brumitt, and J. J. Cadiz. Interaction Issues in Context-Aware Intelligent Environments. *Human-Computer Interaction*, 16(2):363–378, 2001.
- [27] Vera Siegler. Der intelligente OP-Saal: Neuer Forschungsoperationssaal erleichtert Arbeit durch moderne Technik. <https://www.tum.de/die-tum/aktuelles/pressemitteilungen/detail/article/31781/>, 2014. Accessed: 2017-05-20.
- [28] Willem-Jan Sutherland, Jeff; Van den Heuvel. Towards an Intelligent Hospital Environment: Adaptive Workflow in the OR of the Future. In Ralph H. Sprague, editor, *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, Los Alamitos, Calif., 2006.
- [29] Bill Swartout, Ramesh Patil, Kevin Knight, and Tom Russ. Toward Distributed Use of Large-Scale Ontologies. In *Proceedings of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*, pages 138–148, 1996.
- [30] Stefan Ultes and Wolfgang Minker. Managing adaptive spoken dialogue for Intelligent Environments. *Journal of Ambient Intelligence and Smart Environments*, 6(5):523–539, 2014.