Design of Multimodal Dialogue-based Systems

THÈSE N^{O} 4081 (2008)

PRÉSENTÉE LE 20 JUIN 2008

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE D'INTELLIGENCE ARTIFICIELLE

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Miroslav MELICHAR

Master en informatique de l'Université de Brno, République tchèque
et de nationalité tchèque

acceptée sur proposition du jury:

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2008

Contents

List of Figures

List of Tables

Acknowledgements

Above of all, I would like to express special thanks to my supervisor Martin Rajman for not only giving me the opportunity to pursue my thesis at EPFL and for his help and support throughout the whole studies, but also for providing me with a lot of inspiration, for making me working hard before deadlines and also for his understanding when the deadlines turned out to be impossible to keep. Under Martin's supervision, I have learned a lot about research, about interpreting my findings and about reporting them in a structured and convincing way. I also appreciate that Martin always found some time in his incredibly busy schedule to discuss the thesis with me.

Next, I would like to thank the members of the jury, Sebastian Möller, Andrei Popescu-Belis, and Pierre Dillenbourg. They have carefully analyzed my work, densely annotated the manuscript and gave me invaluable advices and comments. All this made my private defense an interesting, fruitful and inspiring discussion.

My work on the thesis was also strongly influenced by two colleagues: Agnes Lisowska and Marita Ailomaa. Without them, the Archivus system would look completely differently and I would never have been able to collect so much experimental data during Wizard of Oz studies. Agnes and Marita, thank you for your great help!

Regardless how well you do, it is useless if the others can only hardly understand it. This is even more important for an interdisciplinary work, as the one I did. In this perspective, many thanks go to my girlfriend Markéta Vlčková for reading the first version of my manuscript and for trying to improve its understandability even for those who are not computer scientists.

For the financial support, I am grateful to the Interactive Multimodal Information Management project (IM2, funded by the Swiss National Science Foundation). This project gave me an opportunity to work in a larger scientific team and to have a close collaboration with many senior researchers. To name at least some of them: Susan Armstrong, Mike Flynn, Alex Jaimes, Denis Lalanne, and Pierre Wellner (and all the others).

Last, but not least, I wish to thank number of colleagues from the EPFL-LIA lab and all my friends in Lausanne and Basel for the very friendly atmosphere, great evenings, and for everything they did to make my stay in Switzerland an enjoyable experience. Thank you and take care!

Abstract

Multimodal dialogue systems integrate advanced (often spoken) language technologies within human-computer interaction methods. Such complex systems cannot be designed without extensive human expertise and systematic design guidelines taking into account the limitations of the underlying technologies. Therefore, this thesis aims at *reducing the time and effort* needed to build such systems by creating application-independent techniques, tools and algorithms that automate the design process and make it accessible for non-expert application developers.

The thesis proposes an *interactive system prototyping methodology*, which (together with its software implementation) allows for rapid building of multimodal dialogue-based information seeking systems. When designed with our methodology, even partially implemented system prototypes can immediately be tested with users through *Wizard of Oz simulations* (which are integrated into the methodology) that reveal user behavior and modality use models. Involving users in early development phases increases the chances for the targeted system to be well accepted by end-users.

With respect to dialogue system design, we propose a *two-layered dialogue model* as a variant of the standard frame-based approach. The two layers of the proposed dialogue model correspond to local and global dialogue strategies. One of the important findings of our research is that the two-layered dialogue model is *easily extendable to multimodal systems.*

The methodology is illustrated in full detail through the design and implementation of the Archivus system – a multimodal (mouse, pen, touchscreen, keyboard and voice) interface that allows users to access and search a database of recorded and annotated meetings (the *Smart Meeting Room* application).

The final part of the thesis is dedicated to an overall qualitative evaluation of the Archivus system (user's performance, satisfaction, analysis of encountered problems) and to a quantitative evaluation of all the implemented dialogue strategies.

Our methodology is intended (1) for designers of multimodal systems who want to quickly develop a multimodal system in their application domain, (2) for researchers who want to better understand human-machine multimodal interaction through experimenting with working prototypes, (3) for researchers who want to test new modalities within the context of a complete application, and (4) for researchers interested in new approaches to specific issues related to multimodal systems (e.g. the multimodal fusion problem).

Keywords: multimodal systems, dialogue systems, dialogue management, rapid dialogue prototyping, Wizard of Oz experiments, human computer interaction (HCI), graphical user interface (GUI), system evaluation

Résumé

Les systèmes multimodaux à base de dialogue intègrent des techniques avancées de traitement de la langue (ou de la parole) au sein de méthodes plus générales d'interaction homme-machine. La conception de tels systèmes, souvent très complexes, peut difficilement être envisagée sans une large expertise humaine et une méthodologie de conception qui prenne en compte les limitations intrinsèques aux techniques d'interaction utilisées. Dans cette perspective, l'objectif de cette thèse est de proposer une méthodologie, des outils et des algorithmes permettant, d'une façon aussi indépendante que possible des applications visées, une automatisation accrue du processus de conception de systèmes multimodaux à base de dialogue, avec comme conséquence une réduction substantielle des durées et des coûts de développement, ainsi que du degré d'expertise spécifique requis de la part des développeurs impliqués.

La thèse présente une méthodologie de prototypage de systèmes interactifs qui, couplée à des techniques d'implémentation spécifiques, permet le développement rapide de systèmes multimodaux de recherche d'information intégrant des techniques de dialogue. En particulier, la méthodologie proposée permet la conception de prototypes partiels (i.e. des prototypes dont certaines fonctionalités ne sont pas encore, ou pas complètement, implémentées) qui peuvent être utilisés pour des simulations de type "Magicien d'Oz" permettant d'éliciter des modèles de comportement et des modèles d'utilisation de modalités caractéristiques des utilisateurs confrontés au prototype. Une telle prise en compte des utilisateurs dans les toutes premières phases de développement permet d'augmenter les chances que le système finalement produit soit bien accepté par les utilisateurs cibles.

Pour ce qui est de la conception de systèmes de dialogue, nous proposons une approche à deux niveaux inspirée de l'approche standard à base de "frames". Les deux niveaux identifiés correspondent respectivement à des stratégies de gestion de dialogue locales et des stratégies de gestion de dialogue globales et l'une des contributions importantes de ce travail est de montrer qu'une telle approche peut être étendue de façon efficace à des systèmes multimodaux.

La méthodologie proposée est illustrée par la description détaillée du développement d'un système multimodal spécifique, le système Archivus, dont l'objectif est de proposer une interface multimodale (voix, clavier, souris, écran tactile) permettant la recherche d'information au sein d'une base multimedia d'enregistrements annotés de réunions (application "Smart meeting room").

Une part importante du travail de recherche a également été consacrée à l'évaluation qualitative du système Archivus et l'évaluation quantitative des diverses stratégies de dialogue implémentées au sein de ce prototype.

Notre méthodologie concerne (1) les concepteurs de systèmes intéressés par le développement rapide de systèmes multimodaux dans un domaine applicatif particulier, (2) les chercheurs intéressés par une meilleure compréhension de l'interaction homme-machine multimodale et qui ont besoin de prototypes exploitables pour des expériences, (3) les chercheurs intéressés par l'intégration de nouvelles modalités au sein d'applications, et (4) les chercheurs intéressés par le développement de nouvelles techniques spécifiques à l'interaction multimodale (par exemple la fusion de modalités).

Mots-clés: systèmes multimodaux, systèmes de dialogue, gestion de dialogue, prototypage rapide, expériences Magicien d'Oz, interaction homme-machine, interfaces graphiques, évaluation.

Chapter 1

Introduction

Human-computer interfaces have come a long way from the early days of punched cards and manual switches. At that time, people working with computers were skilled machine operators (experts) rather than ordinary end-users. This has gradually changed with the introduction of text-terminals controlled with keyboards, which were later transformed into graphical terminals operated with pointing devices (mouse, pen). Such technological advances enabled the design of new interaction techniques, such as the direct manipulation of graphical objects and windows-like interaction metaphors. The proliferation of personal computers further led to a small revolution in the technology society: the few machine operators became countless computer users, communicating everyday with personal electronic devices in all possible situations. A new research field was born: the human-computer interaction.

Nevertheless, the human-computer interaction techniques are still too constrained and inflexible when compared to existing human-to-human communication mechanisms. Humans use speech, gestures, writing, drawing, facial expressions, gaze and other methods, allowing them to easily and naturally express their communication needs. Clearly, humans spontaneously use many interaction channels (hereafter called *modalities*), and many of these modalities are better suited for communication than the modalities provided by traditional computer interfaces. For instance, people can easily and naturally specify information needs using voice (e.g. *"Find me all financial reports concerning the lab budget, which I have received from John last year"*), while performing the same task might be quite complicated with mouse and keyword searches. In fact, simply finding and invoking the necessary system functionalities is often obscure in many existing interfaces.

With the growing importance of computer systems, more and more people need computers for everyday tasks of still increasing complexity. Such non-expert computer users need the interface to be natural, easy to use, ergonomic and supportive. Unfortunately, the traditional computer interaction paradigms cannot satisfy such user needs because of their inherently limited and inflexible ways of *understanding* the human user. As a consequence, an experience with such interfaces often becomes an important source of user frustration.

One of the promising research directions towards more natural interfaces are dialogue systems, which allow users to perform various tasks through (often spoken) natural language. Nowadays, hundreds of such applications are running all over the world, typically accessible as phone vocal services. Such dialogue systems combine various advanced technologies (automatic speech recognition, speech synthesis, natural language processing and dialogue management) to better communicate with users. However, the required technologies are not yet robust enough for understanding the user requests in a fully satisfactory way and therefore are still often not well accepted by users.

Another promising research direction towards natural and easy-to-use interfaces are multimodal systems, offering to the users various interaction channels in addition to the traditional keyboard and mouse. Examples of such additional modalities are: handwriting, speech, typed natural language, haptics, pen-based input, or inputs from webcameras enabling for instance detection of user presence in front of the computer, recognition of facial expressions, of gestures, etc. The main advantage of the availability of multiple modalities is increased system usability: the weakness of one modality (e.g. non-robust speech recognition) might be compensated by the strength of another. In addition, multimodal interfaces also improve system accessibility: they can be easily used by physically impaired people, who can control the system through the modalities that they can rely on. Similarly, people with no impairment can sometimes be considered as "situationally impaired", for instance in hands- or eyes-busy environments (driving, cooking).

In general, the fact that *multimodal human-computer interaction results in greater flexibility and naturalness of expression* in contrast to unimodal input methods has been reported by many researchers (e.g. [37, 12, 78, 40, 97, 115]). However, designing a multimodal interactive system that smoothly blends coherent natural language communication together with direct object manipulation (graphical user interface) is recognized as a *very complex task* in practice. Indeed, multimodal interfaces require the integration of a number of non-traditional technologies (speech recognition, speech understanding, and other modality sensors) into one single system. Often, these technologies are interfering with each other. For instance a graphical user interface influences the natural language used by users, the precision of pointing on a tactile screen can impose the necessary minimum size of objects manipulated by users, etc. In other words, many technical and design-related problems have to be solved in order to create a usable multimodal interface. As a consequence, the design becomes a time consuming process where a lot of effort has to be invested in implementing and fine-tuning of numerous system features, many of which might in addition turn out to be unused by the end-users. Therefore, efficient development guidelines and tools are of great help.

Within this perspective, we propose *a methodology for prototyping interactive systems*, which (together with its software implementation) allows for rapid building of multimodal dialogue-based information seeking systems. When designed with our methodology, even partially implemented system prototypes can immediately be tested with users through *Wizard of Oz simulations* (which are integrated into the methodology) and can thereby reveal behavior and modality use models specific for the targeted end-users. An important consequence of the early user involvement in the development phases is the increase of chances that the designed system is well accepted by end-users.

The thesis is organized as follows: Chapter 2 provides an overview of the state of the art in dialogue systems and multimodal systems. In Chapter 3, we introduce our methodology for prototyping multimodal systems, explain how concrete systems can be designed with it, and what issues system designers need to take into account in order to make the targeted multimodal systems operational. Since the systems are designed iteratively using Wizard of Oz simulations, Chapter 4 provides a description of our simulation environment. In Chapter 5, we present the Archivus system – an illustrative example of a system designed by our methodology, together with the individual design steps and the necessary modifications and improvements that were required after its pilot evaluation by 40 users. Chapter 6 presents the overall qualitative evaluation of the final version of Archivus as resulting from experiments involving 90 users, as well as the quantitative evaluation of the dialogue strategies used in Archivus. Finally, Chapters 7 and 8 respectively present the main conclusions of our research and perspectives for future work. In order to facilitate the reading, most of the chapters starts with an introduction and are concluded with a summary that respectively synthesize the main issues considered and the main results obtained.

Chapter 2

State of the art

2.1 Background

The term *dialogue system* refers to a computer program that communicates with a human user in a more or less natural manner. The interaction protocol used for the communication between the program and the human is called *the dialogue.* Dialogue is also used by pairs (or groups) of humans for the communication among them. Dialogue can be seen as a sequence of information exchange that stays coherent over the time. Humans usually participate in dialogue because they want to achieve a certain goal. We say that the interaction is goal driven. Examples of goals are: finding the most suitable restaurant in a foreign city, booking the cheapest flight to a given city, controlling the state of the devices in a home, but the goal might also be the interaction itself (chatting).

A cooperative dialogue system is designed in a way that it helps the user to achieve a goal using dialogue. Such a dialogue system must therefore have an extensive knowledge about the domain of communication and have some basic communication skills. However, the communication skills and intelligence of the current computer-based programs are still far from a state that would make it possible to say that the behavior of the computer program is comparable with the one of humans.

2.1.1 Dialogue is a complex protocol

Even if dialogue is the most natural means for human-to-human communication, it is necessary to realize that dialogue is a very complex protocol. There are several reasons for it [81]: the dialogue utterances are often imperfect – ungrammatical or elliptical. Humans usually use their extensive knowledge and reasoning capabilities to understand the conversational partner. Dialogues also follow certain conventions or protocols that are adopted by participants. Despite those problems, from the human perspective, natural dialogue is perceived as a very expressive, efficient and robust manner of communicating.

Many researchers were investigating human-to-human dialogue as such, in order to better understand different aspects of the communication. An example of their achievement is the *speech act theory*, early proposed by Austin [7] and further developed by Searle [103]. The term *dialogue act* (i.e. “speech act used in the dialogue and updating the dialogue context”) has been first used by Bunt [14]. Dialogue acts are classified by their *semantic* *content* (task-oriented acts) and *communicative function* (dialogue-control acts). Details about the dialogue act functions can be found in [15].

Even if theories of dialogue exist, robust computational mechanisms for dialogue processing are still missing. This is largely due to the following features of natural dialogue:

- *Ellipsis and fragments*: people often utter partial phrases to avoid repetition. The problems arise when analyzing such segments. Since the missing information is usually important, it must be reconstructed from the context of the dialogue [2].
- *References*: another complication for proper analysis of the utterances is that the meaning of some words (e.g. this/that, me/you/he/it) can only be interpreted as a function of the context.
- *Indirect meaning* (also called conversational implicature): the meaning of the discourse might be far from literal meaning. Humans usually use this feature for effect and efficiency, e.g. “Here is a phone call from your boss” – “I am not here” – “Okay”.
- *Turn taking*: people seem to know well when they can take their turn in the dialogue, the overlap is very small (5%), the gaps are very limited (less than $1/10^{th}$s of a second), and a natural dialogue seems to be fluid. It is not obvious how humans decide when to talk [81].
- *Conversational fillers*: phrases like “a-ha”, “yes”, “hmm” or “eh” are often prompted in order to fill the pauses of the conversation, to indicate the attention or reflection. The challenge here is to recognize when they should be understood as a request for turn taking and when they should be ignored.
- *Adjacency pairs*: they are intuitive structures of the form question-answer, greeting-greeting, offer-acceptance. They are often embedded in each other, but they may also overlap and interfere.

2.1.2 Dialogue system

A computer based dialogue system can be defined as *an artificial participant in the dialogue* [81]. It contains algorithms and procedures that support a computer’s participation in a cooperative dialogue. It needs to have an extensive knowledge about the domain. It may have a model of the other participants and it may have to cope with unexpected or unusual input.

As mentioned earlier, human dialogue is a really complex protocol; therefore the artificial dialogue systems are still far from being close to real humans. The situation is further complicated by the fact that users do not interact with a machine exactly in the same way they would interact with a real human. As pointed out in [21], human-to-human dialogues exhibit much more varied behaviors, including clarifications, confirmations, other communicative actions, etc. Some researchers have argued that because humans interact differently with computers than they do with people [26, 32], the goal of developing a system that emulates a *real* human dialogue behavior is neither appropriate, nor attainable [26, 108]. On the contrary, others have argued that the usability of current natural language systems, especially voice-interactive systems in a telecommunications setting, could greatly benefit from techniques that allow the humans to engage in situations similar to ones found in their typical spoken conversations [43]. The whole situation is quite different in multimodal systems equipped e.g. with a screen. The communication model is then not comparable to a human-to-human interaction. Therefore, multimodal systems are typically designed in a way that explains to a user, in terms and with concepts they are already familiar with, how to interact with an unfamiliar application and what its functionalities might be. The set of known terms and concepts (typically taken from natural human life) forms *the communication metaphor* of the application [30].

According to [1], a task-oriented dialogue system that naturally interacts with humans must deal with at least the following tasks:

- *Disambiguation of user inputs*: since the natural language expressions are often ambiguous, the system has to implement mechanisms to disambiguate user's input. Disambiguation can be implicit (deduction of the meaning from the context of the dialogue) or explicit (asking the user).
- *Relaxation*: the system should be able to drop some of the user's constraints in the case when they do not generate any solution (i.e. user request is over-constrained). Intelligent systems should drop the constraints that are not of primary importance for the user.
- *Confirmation*: the system should ask the user to confirm an information that has been obtained with a low confidence score or is likely to be wrong within the given context of the dialogue.
- *Completion of missing required information*: the system must take the initiative to elicit the information that is needed to successfully complete the task.
- *Description of otherwise invisible semantic actions*: in the case when the system has dropped some of the user constraints or when it supposes something implicitly (and it is not clear that user would agree), the system should notify the user about it.
- *Detection of user confusion/error recovery*: since dialogue systems are still imperfect, it is very important to implement several strategies that solve the problems occurring in the communication. If the user does not know how to answer the question, the system must be able to give him/her a help or ask another question in the hope of executing the user's request. The detection of user confusion can be indicated by an "I do not know" answer or detectable from auditory or audiovisual cues [48, 38]. Another not so obvious indication is the lack of progress in the dialogue.
- *Topic switching*: the system should be able to recognize when the user request is out of the current system context. When possible, the system should then switch to the new context (topic) of the conversation.

A user model in a dialogue system is a knowledge source that contains explicit assumptions on all user related aspects that may be relevant for the dialogue behavior of the system [118]. Knowing the model of the user can smooth the dialogue flow and increase the satisfaction of the user when interacting with the system.

Current commercially available dialogue systems typically have a small vocabulary (often around 100 words), are applied in closed domains, and are characterized by strong system initiative. On the other hand, current research systems use larger (but still small) vocabulary (around 1000 words), they have closed domain and we can observe (limited) mixed initiative between the computer and the human.

Some of the most innovative applied research projects are represented by:

- SUNDIAL [EU, 88–93]: flight/train timetables
- MAIS/RAILTEL [EU, 94–95]: train timetable
- VerbMobil [DE, 93–00]: assisted translation, meeting scheduling
- TRINDI [UK, SE, 98–00]: instructional dialogue, SmartHome
- Clarissa [NASA, 02–05]: a procedure navigator for astronauts at the ISS
- VoiceXML [W3C, 99–07]: W3C's standard XML format for specifying interactive voice dialogues

Most common application domains include: flight and train timetable information and reservation, switchboard services, automated directory enquiries, weather information, yellow pages enquiries, appointment scheduling, multilingual spoken dialogue real-time translation systems (e.g. VerbMobil).

2.2 Dialogue system architectures

This section presents the *key components of multimodal dialogue systems* and shows how these components are integrated together in various *software architectures*. Having a flexible software architecture for component integration has been identified as a key to successful dialogue systems, i.e. systems that are more advanced than the sum of their sub-components [64].

A spoken dialogue system involves the integration of a number of components that typically provide the following functionalities:

- *Speech recognition*: transforms speech input into a list of transcriptions, possibly with confidence scores.
- *Language understanding*: extracts the meaning of the textual utterances. The meaning is expressed in a formal representation understandable by the dialogue manager. Depending on used techniques, this process may involve several sub-processes, e.g. syntactic analysis, semantic analysis, or discourse analysis.
- *Dialogue manager*: central component that controls the interaction flow with the user, based on a certain *dialogue model*. The input to this process is a formal representation of the user utterance and the output is a formal message to be generated as response. The dialogue manager also communicates with the application task manager.
- *Application task manager* (or *domain knowledge manager*, since it works with *the domain task model*) encapsulates and handles the application specific operations, e.g. communicates with the application database. In the ideal case, the dialogue manager operates on a general, task independent level, while the application task manager handles application and domain specific knowledge. The denomination

term used for this component is not very standardized, and is usually derived from the function this component plays in a particular system. Depending on the focus of the system, the provided functionality can be data-access, reasoning, planning or problem solving management.

- *Response generation*: produces a (textual) natural language message to be communicated to the user on the basis of the formal message generated by the dialogue manager.
- *Speech synthesis*: transforms a textual message to the corresponding speech signal by text-to-speech synthesis or concatenation of pre-recorded speech.

A *multimodal dialogue system* is a system that supports several channels of communication with the user (clicking, gestures, visual output, etc). Multimodal dialogue systems comprise the additional components for processing the information from other modalities than speech (i.e. components like face detection, emotion recognition, gaze and gesture recognition) and the output of such a system is usually richer as the screen is used to present multimedia in a synchronized manner. The multimodal input is combined in *the fusion component* and the multimodal output is controlled by *the fission component*. In the multimodal dialogue systems, the dialogue manager is often referred to as *the interaction manager*.

The classification above is the most traditional classification, but it cannot be considered as the ultimate one. Due to practical reasons, dialogue systems very often combine two or more functionalities within one module (e.g. the response generation module is part of the dialogue manager). On the other hand, dialogue systems often incorporate other additional modules (e.g. the modules for maintaining the user model and dialogue history, or modules for operating the telephony interface).

As far as the organization of the dialogue system components is concerned, the *pipeline architecture* is the simplest one. Modules are simply connected linearly into a sequence in the order they process the input, as shown in Figure 2.1. The main advantage of the pipeline architecture is its relative implementation simplicity. A commonly mentioned disadvantage is its inflexibility: modules can interact only with the modules they are connected with and there is no shared memory that can be used as a blackboard.

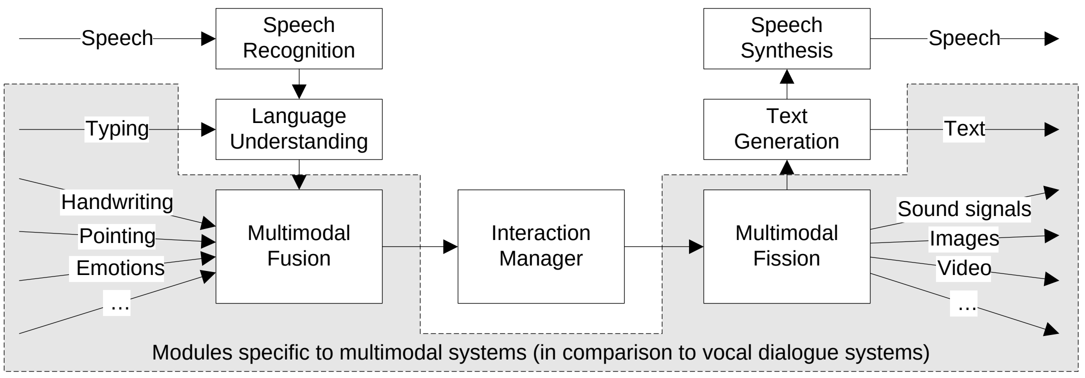


Figure 2.1: Pipeline architecture of multimodal system

A dialogue system is usually a complex system that needs to integrate a large number of modules. It is therefore convenient if these modules are specified as reusable modules that can be designed by different people and shared among different dialogue system applications. The integration and building of the dialogue system can indeed be simplified, if the modules are designed to be compatible with some common *software framework* (or *platform*). Such a framework should provide rules for the module design and the necessary infrastructure for communication between modules. In the ideal case, it should represent the whole development environment that supports development and debugging of the complete system. It should come with pre-defined components for immediate use (e.g. modules for speech recognition, speech synthesis, dialogue management, etc). This would allow researchers to deploy the functionality of the module they are interested in, without the need of solving the problem of module communication and data exchange.

Frameworks that can be considered as satisfying the above criterions are *Galaxy-II* [104] and *Open Agent Architecture* [63]. Because the Galaxy-II is specifically targeted to the speech application domain and because both architectures share similar principles, we will now focus on Galaxy-II only.

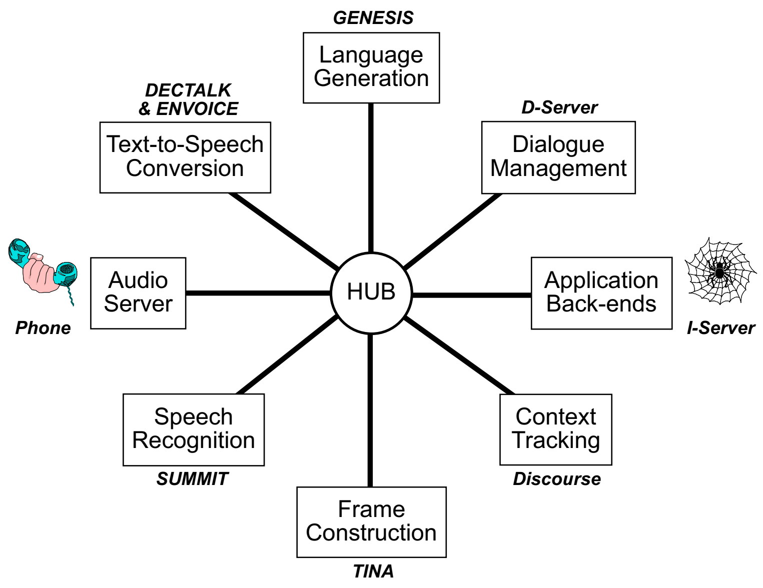


Figure 2.2: Architecture of GALAXY-II [104]

Galaxy-II is a distributed, message-based, hub-and-spoke infrastructure that was especially optimized for spoken dialog systems. The central component is the hub, which is used to coordinate the communication with the other components that are implemented as servers. Servers are organized within a star topology (central is the hub). The Figure 2.2 shows a skeleton of a Galaxy-II based spoken dialog system. Components send messages (frames) to the hub and the hub forwards them to the appropriate component. The hub logs all the routed messages. The configuration of the hub is done by a script that defines the routing of the messages. Such architecture allows any desired communication between components. The implementation of the Galaxy-II is released as open source software and is maintained by MITRE Corp. The distribution contains the system infrastructure (hub and libraries for construction of components). Some implemented modules are available as Open Source Toolkit[1]. The implementation of all necessary modules is called CMU Communicator[2] and has been released by Carnegie Mellon University.

[1] `http://communicator.sourceforge.net/`

[2] `http://www.speech.cs.cmu.edu/Communicator/`

The Galaxy-II is a very general architecture that does not impose any predefined communication paradigm on the dialogue system. On the other hand, the distributed nature of the framework can make it more difficult to debug and test the system, as well as to start to use the system at the beginning. Since each module must be implemented as a server, problems with module synchronization might become difficult to track.

2.3 Computational models for dialogue management

A dialogue manager controls the overall interaction between a system and a user. The essential role of the dialogue manager in the framework of human-computer interaction can be reduced to two basic actions [54]:

- Interpret observations (usually user inputs) in context and update the internal representation of the dialogue state.
- Determine the next action of the dialogue system, depending on some dialogue management policy (with the effect of affecting the mental state of the user).

Although these two functionalities coexist in all dialogue managers, each of them is non-trivial, leading to a proliferation of different computational approaches.

The dialogue manager may draw on a number of *knowledge sources*, which are sometimes referred to collectively as the dialogue model. A dialogue model might include the following types of knowledge relevant to the dialogue management [64]:

- *A dialogue history*: A trace of the dialogue realized so far, in terms of the propositions that have been discussed and the entities that have been mentioned. This representation provides a basis for conceptual coherence and for the resolution of anaphora and ellipsis.
- *A task record*: A representation of the information to be gathered in the dialogue. This record, often referred to as a form, a template, or a status graph, is used to determine what information has not yet been acquired. This record can also be used as a task memory [4] for cases where a user wishes to change the values of some parameters, such as an earlier departure time, but does not need to repeat the whole dialogue to provide the other values that remain unchanged.
- *A world knowledge model*: This model contains general background information that supports any commonsense reasoning required by the system, for example, that Christmas day is December 25.
- *A domain model*: A model with specific information about the applicative domain in consideration, for example, flight information.
- *A generic model of conversational competence*: This includes knowledge about the principles of conversational turn-taking and discourse obligations, e.g. that an appropriate response to a request for information is to supply the information or provide a reason for not supplying it.
- *A user model*: This model may contain relatively stable information about the user that may be relevant to the dialogue (such as the user's age, gender, and preferences),

as well as information that changes over the course of the dialogue, such as the user's goals, beliefs, and intentions.

2.3.1 Finite state models

Some of the first dialogue managers used the concept of *finite state automaton* to model their behavior. The state of the dialogue manager is determined by the current state of the finite state automaton, which specifies the dialogue action (prompt) and the grammar that is used to interpret the input of the user. The semantics of the user utterance (or the utterance itself) determines the transition to another node, resulting in changing the state of the dialogue manager. Even though this is an extremely simple approach, it can already describe an infinite number of dialogues. Finite state systems are well suited to applications in which the interaction is well-defined and can be structured as a sequential form-filling task or a tree, preferably with yes/no or short answer questions.

The advantage of finite state models is that they are formally well-understood [46] and computationally attractive [52]. The design of such systems is relatively straightforward and intuitive and their behavior is predictable. The ease of development is further improved by existing visualization toolkits [23]. One of the most popular is the Rapid Application Developer of CSLU Toolkit [111], which allows the designer to specify the dialogue as a finite state model using a drag-and-drop interface. In addition, probabilistic finite-state models can be used for automatic machine learning of optimal dialogue strategies [53, 87, 33] using reinforcement learning or Q-learning.

The main disadvantage is that the finite state approach typically leads to an *unnatural dialogue*, where the information is elicited from the user in the form of a sequence of questions. The dialogue strategy is very inflexible: the user must follow the structure of the dialogue and answer the system questions. Any additionally provided information is ignored by the system. Attempting to extend the system to enable repair mechanisms (reaction to misunderstanding, clarification, etc) leads to combinatorial explosion of states and transitions [66], thus making the system very hard to manage. Another limitation of the finite state methodology is its complete *domain and application dependency*: porting a finite state dialogue model to a new domain or application typically results in the development of a completely new finite state model. The reason is that the finite state system lacks of a systematic distinction between the task (what the dialogue manager wants to achieve) and the dialogue strategy (how the dialogue manager should proceed towards its goal).

These limitations are well-known and alternative approaches have been investigated. One of them is to associate each state with executable code that updates some shared variable. The state of the automaton is then determined by both the state and the value of the variable [66, 65]. Another approach is to embed another finite-state network into one state, making the whole finite state network easier to understand and manage. Some other systems (e.g. EasyFlight [123]) use the finite state approach in combination with plan based control [124].

2.3.2 Frame-based (slot-filling) systems

Another computational approach to dialogue flow control is based on the *frame structure* [70] (other authors use the term *form* or *template*). The frame consists of *slots*. Each slot is related to a specific category of information "understood" by the system. The frame cumulates the information provided by the user during the dialogue and is used to determine the next dialogue action.

The dialogue system has questions associated with slots. The user is asked questions that enable the system to fill-in slots in order to perform a task. The task might be finding the restaurant conforming the user's request (BeRP [42], InfoVox[3][114]), finding some train timetable information or changing the state of a domestic devices (Inspire[4][71, 49, 10]). A frame-based system typically implements some mixed initiative: the user takes the dialogue initiative predominantly at the beginning of the dialogue where he/she freely specifies his/her requirements. The system analyzes the user's response and fills-in the appropriate slots. The missing information (unfilled slots) is then elicited by the system.

The term dialogue strategy in frame-based systems refers to a decision-making mechanism that is used to determine the next information to be obtained from the user and describe how to solve potential problems during the dialogue.

i. Variations of the standard frame-based approach

Many successful systems were implemented based on the slot-filling paradigm (see e.g. [36, 16, 75, 82, 28, 50, 106, 125, 6, 44, 110, 105, 69], with variations on the structure of the frame and on the way of describing the dialogue strategies.

One of the frame variations is *the E-form* [35]: slots are augmented with priorities and marked as mandatory or optional. E-forms have been used in the WHEELS dialogue system [69] to capture different user preferences about car parameters (e.g. parameters like price or color do not usually have the same importance).

Another modification of the frame structure are *hierarchical frames* (also referred to as *frame type hierarchy*), in which one slot might be represented by a sub-frame. Such a representation better reflects the hierarchical structure of real-world objects. For example, the slot `person` might be decomposed into the slots `first name`, `family name` and `address`, and the slot `address` might be in turn decomposed into slots `city`, `institute` and `office number`. The chronological ordering of the subtasks in the dialogue is still available (e.g. left-to-right depth-first traversal) and the dialogue strategies can exploit the information about the object under discussion. Examples of systems using such a representation are [3, 39, 83, 90, 116].

An even more generic approach is to describe various relations between slots by a *task structure graph* [122]. Such a description can be exploited both by the dialogue management strategies and the natural language understanding algorithms. In [3], a frame description method using the Web Ontology Language (OWL) is proposed with the expectation of increasing portability of the dialogue manager and with the assumption that the OWL annotated pages can be used as a knowledge base for the dialogue manager.

[3]`http://liawww.epfl.ch/Research/infovox.html`

[4]`http://www.ist-world.org/ProjectDetails.aspx?ProjectId=0c83b96d2f764e3fae85a9b5e8566949`

The frame (or its modification) describes the structure of the information required by the user and the dialogue strategy is used to select the reaction of the system to the last user request. Some systems use action lists with associated preconditions (*event driven approach*) to describe the behavior of the dialogue. Only actions with fulfilled pre-conditions might be selected by the system. In the case that several actions can be selected, other factors (such as priorities) are used. An example of such a system is the Philips SpeechMania [6] or the Communicator system [120].

Another possible approach is to directly encode the logic (dialogue manager decisions) as a script in the dialogue manager. In this case (hard-wiring of the logic in the code of the dialogue manager), designers usually try to formulate the logic in the form of *a general slot-filling dialogue model* that can be reused when the system is ported to another domain (i.e. the frame description has changed). Such algorithms have been described by several researchers, e.g. [109, 51, 86, 5].

The combination of the above mentioned techniques is used in [13], where a general slot-filling dialogue model selects the main category of the next system prompt and the conditional approach is used to select among several compatible prompts (this is typically used to alternate the formulation of the prompt or/and to make it more contextualized).

ii. Advantages

The frame-based approach is appropriate for well-defined tasks, where the dialogue manager can respond to the user's initial query with a sequence of clarifications in order to obtain enough information to complete a specific action. The slot-filling approach is the most frequently used dialogue management technique in practical systems [47]. This is partly due to the large number of available toolkits: languages like VoiceXML[5] or Philips' HDDL [85] are based on frames and environments for their interpretation exist (e.g. IBM WebSphere Voice Server[6], OptimTalk[7], Philips SpeechMania [6]). Techniques for rapid dialogue prototyping have also been proposed, e.g. the rapid dialogue prototyping framework developed at LIA-EPFL [94, 95, 93, 13, 91]. This latter work is further extended in this thesis, making it applicable to design of multimodal systems.

A frame-based system fulfils a number of dialogue design requirements, as identified by the dialogue team at Philips [6], including the following:

- there should not be a rigid question-answer scheme to obtain the required values;
- there should not be more questions asked than necessary;
- there should not be more required confirmation than necessary;
- information given by the caller, prior to the system asking for it, should be used.

Since the development of a dialogue system requires considerable efforts, an important aspect of dialogue system development is *portability* (i.e. the ability to easily customize the dialogue system for a new language or domain). In frame-based systems, the task and the dialogue strategy can be clearly separated. The task is defined by a (domain

[5]`http://www.w3.org/TR/voicexml21/`
[6]`http://www.ibm.com/software/pervasive/voice_server/`
[7]`http://www.optimtalk.cz/`

specific) frame, while the strategy for filling-in the frame is rather domain independent (the strategy might be designed to be independent of the contents of the slots).

iii. Disadvantages

Even though task and dialogue strategies are separated (which is beneficial for portability), it is an open issue to measure *how scalable the approach is* [47]. Extending the existing systems with another useful dialogue strategy usually requires a considerable amount of hand-coding (e.g. EPFL dialogue prototyping methodology [13]) or is almost impossible (e.g. the Form Interpretation Algorithm in VoiceXML 2.0). Developers need to handle large sets of rules or types of system reactions to particular situations. It is difficult to predict which rule fires under specific circumstances or all the consequences of a dialogue strategy modification. A considerable amount of experimentation may be required to ensure that the system does not produce an inappropriate question under some circumstances that had not been foreseen at design time [64].

Another problematic issue in frame-based systems is that they model the dialogue as elicitation of several parameters in order to perform a predefined task. However, a dialogue is a more complex process, typically with several possible topics for the conversation; both the system and user might ask questions or change the topic; and the system should be able to detect the user's dialogue goal (which is usually expressed implicitly in the user queries), as well as be capable of simple reasoning (e.g. "When do you want to watch the movie?" – "After the Simpsons finishes"). Generally, frame-based systems do not support mechanisms for *topic detection*, nor for *explicit representation of user goals*, as the goals are implicitly encoded in the structure of the frame. Because of practical reasons, the frame-based systems usually relay on quite simple (but relatively robust) natural language understanding techniques (such as concept spotting). In order to model more complex dialogues, advanced natural language capabilities are needed. The result of complex utterance analysis is then more complex than a simple attribute-value list (typically it might be a logical form [96]), but the flat frame structure is unable to store such complex results and therefore the available dialogue strategies cannot exploit it.

2.3.3 Plan-based systems

Plan-based systems view the communication as *a planning process* motivated by the achievement of *certain goals*. Achieving the goals is decomposed into a sequence of actions and sub-goals to be achieved (and sub-goals are again decomposed in the same manner) [61]. The actions are represented by dialogue acts [14], and are performed both by the user and the system. Knowing the goal, the system can optimize the plan of how to achieve it, taking into account the history of the dialogue (i.e. all dialogue acts executed so far by both participants). The next system action (system dialogue act) is simply determined by the next action in the optimized plan. However, the planning problem is not trivial, since the system executes just half of the actions; the second (possibly different) plan is executed by the user (and is unknown to the system). The system must therefore infer the plan of the user, based on already observed actions (i.e. user dialogue acts) and assuming cooperativeness of the user. Further complication can arise from the fact that there can be several possible goals and the system must infer the one desired by the user.

The plan-based approach is quite sophisticated and has the potential for handling complex dialogues. However, it suffers from today's technological limitation and the overall performance of integrated systems still remains quite low. The current technology is not able to reliably recognize the speech acts corresponding to the user's utterances. In addition, in more complex cases, the planning problem can become computationally intractable [64]. The theoretical limitation of plan-based systems is that they assume cooperative users and do not provide explanation of dialogue activity in the case of participants that do not share mutual goals.

2.4 Multimodality processing

Interaction modalities are all the communication channels between the user and the machine. For a quite long time, speech has been the favored modality for input and output in telephone-based dialogue systems. In the past two decades, however, so called multimodal dialogue systems start to emerge. In multimodal dialogue systems, the machine is sensitive to several input modalities and produces output in several output modalities. For example, besides the speech, the dialogue system can be sensitive to keyboard input, mouse clicks, facial expressions, gestures, gaze, etc. The system output can be performed by speech and visual information. In case that the system is represented by an animated virtual character (e.g. Baldi[8] in CSLU Toolkit), the outputs can also be augmented by virtual character facial expressions and emotions.

Several modalities are particularly useful in situations when one modality is not applicable (e.g. eyes or hands are busy, silent environment, etc) or difficult to use (e.g. small devices with limited keyboard and small screen). Having a richer set of communication modalities is also believed to make the dialogue more efficient, namely because of two reasons: (1) Some modalities are more efficient or effective than others for certain types of content, tasks, users or contexts (e.g. mouse clicks can compensate speech technology limitations) and (2) dialogue strategies can be adopted to use the most convenient way of presenting the system output (e.g. long vocal confirmation might be favorably replaced by short visual indications). Altogether, multimodality aims at improving speed, accuracy, and naturalness of the human computer interaction.

In a multimodal environment, the computer needs to understand inputs coming from different modalities (happening sequentially or in parallel) and it needs to be able to integrate these inputs in order to get a full understanding of the current situation. In addition, users frequently do not provide complete utterances when interacting with a computer. Often the utterances need to be disambiguated by looking at the context of occurrence or by looking at information coming from other modalities. Research in multimodal interaction is concerned with the interpretation of isolated, sequential and parallel interaction utterances with the computer. In particular, it is useful to develop models in which the knowledge obtained from different input modalities can be integrated. This also involves knowledge representation and reasoning formalisms. From a human-computer interaction perspective, it is interesting to look at the various multimodal ways people interact with the environment and with each other, and to design systems that are

[8]`http://cslu.cse.ogi.edu/FAQ/`

sensitive to what the user wants without providing any explicit command or information about it.

Notice that there exist several different ways of understanding the term multimodal [99]. Among the most often mentioned ones are the following:

- *Different communication interfaces* that allow the users to use the means of the communication they prefer, e.g. speech, mouse, or keyboard. The main characteristic of this approach to multimodality is that the user voluntarily selects one of the communication channels (or their combination).
- The term multimodal also refers to *multiple coordinated streams* generated by the user, not necessarily by conscious choice. An example can be speech and lip movements. Both of them can be used together to improve the accuracy of speech recognition, but they represent one information source (seen from the perspective of the dialogue manager).
- We can also think of *virtual streams* of information that are delivered from the same physical signal but processed differently. For example, the voice can include linguistic, prosodic or emotional information and each of them can be used by the application as a different information source.
- Yet another approach is to think about multimodality as a set of *independent streams* of information that are only loosely related to the ongoing communication act, but provide a useful context for interpretation of the user's message. An example is the user's location in the room (in front of a lamp) when saying "switch *it* on".

In the rest of the document, we will mostly refer to the first definition and therefore consider the term multimodality as a different means of communication.

In multimodal systems, the term *multimodal fusion* is used to describe the process of combining input coming from different channels into one abstract message that is then processed by the dialogue manager. The process of fusion may happen at several levels. Nigay and Coutaz have identified three such levels: lexical fusion, syntactic fusion and semantic fusion [76]. Their PAC-Amodeus model uses *melting pots* as a metaphor for hierarchical fusion of modalities, taking into account the temporal occurrence and the semantic meaning of the modality to the dialogue (see [76] for more details). The importance of interpretation of every input event within the context of its local dialogue turn is emphasized by Pfleger [84] in the COMIC system.

Similarly, the term *multimodal fission* refers to the process of concretely realizing an abstract message (produced by dialogue manager) through some combination of the available output channels. For example, in multimodal system the output can be realized by using synthesized speech, text, graphic, or their combinations. The techniques for fission are usually considered as practical issues. Nevertheless, some user preferences have been identified and practical guidelines have been proposed [9], for instance that speech and graphic output need to be coordinated and unnecessary redundancy between the two channels should be avoided (the speech should convey just the short version of the main message and the details can be displayed on the screen).

Some general-purpose languages for multimodal content abstraction exist, for example the two languages developed by the World Wide Web Consortium (W3C): Extensible MultiModal Annotation language and Synchronised Multimedia Integration Language.

Extensible MultiModal Annotation language (EMMA): this is a markup language for describing the interpretation of user input. Examples of user input interpretation are (1) the transcriptions into word sequences of various raw signals, for instance speech, pen or keystroke input, (2) a set of attribute/value pairs describing their meaning, or a set of attribute/value pairs describing a gesture. The interpretation of the user's input is expected to be generated by the signal interpretation processes, such as speech and writing recognition, semantic interpreters, and other types of processors used by components (such as interaction managers) that act on the user's inputs.

Synchronised Multimedia Integration Language (SMIL): is a meta-language for multimodal presentations. The goal of SMIL is to be an integration format for presentable single modality formats; that is, to allow the authors to specify what should be presented when. SMIL allows the integration of images, audio and video clips, animations, and formatted text. Several SMIL 2.0 compliant commercial players are available, for example, the GRiNS player, Internet Explorer 6.0, and RealOne all understand SMIL 2.0 markup. For the complete survey of multimodal fission activities and available languages, see [31].

Chapter 3

Multimodal interactive systems prototyping

This chapter presents the methodology we have developed for rapid prototyping of multimodal interactive systems, hereafter referred to as the *interactive systems prototyping methodology* – ISPM.

Designing a multimodal interactive system that smoothly combines coherent natural language communication together with direct object manipulation (typically realized by a pointing device and screen display) is generally recognized as a *very complex task*. Compared to the design of traditional GUI-based systems, the designer must not only foresee optimal system's functionalities and graphical layouts, but also consider how users might use the novel modalities to interact with the system. Because such systems are still not wide-spread, one cannot rely on some generally accepted models for multimodal interfaces. The challenge is therefore to create an interface that makes the multimodal interaction natural and smooth, while remaining manageable from the perspective of the system.

In particular, we believe that the following factors make the traditional development cycle (i.e. design, prototyping and evaluation) of multimodal systems more complex compared to traditional GUI-based interactive systems:

- The natural language technologies, such as speech recognition, natural language understanding and dialogue management, have to be efficiently integrated.
- The presence of non-traditional modalities (tactile screen, spoken language) influences the graphical design and system functionalities.
- The user language inputs are influenced by the linguistic features (e.g. prompts, labels, etc.) of the system prototype that the user is faced with.
- Due to strong interdependencies between modalities and their combined impact on user behavior [92], the modalities and the interaction techniques have to be evaluated within a context of a fully integrated system prototype.

Consequently, the system design becomes a time consuming process, where a lot of effort is invested in implementing and fine-tuning system features, which might in addition turn out to be unused (or to be used only very infrequently) by end-users.

However, the complexity of the system development cycle might be significantly reduced, if (1) it is possible to build the first system prototype on top of already available components, (2) if exploitable design guidelines are available, and (3) if it is possible to observe user-system interactions even before the whole system is completely implemented. The *interactive systems prototyping methodology* (ISPM) presented in this chapter, together with its Java implementation (hereafter called *the prototyping platform*) proposes convenient solutions for all the above mentioned issues.

Our methodology is intended (1) for designers of multimodal systems who want to quickly develop a multimodal system in their application domain, (2) for researchers who want to better understand human-machine multimodal interaction through experimenting with working prototypes, (3) for researchers who want to test new modalities within the context of a complete application, and (4) for researchers interested in new approaches to specific issues related to multimodal systems (e.g. the multimodal fusion problem).

In the ISPM, the design of a first system prototype starts with the definition of the required application knowledge (i.e. the domain model, section 3.2). The application is then decomposed into elementary building blocks (sections 3.3 and 3.4), which are used to define the task model of the application, i.e. to describe what is possible, when it is possible and how the overall screen layout is organized (section 3.5). The software architecture of the prototype must be easily modifiable in order to add or remove modules (section 3.6). Some of the modules can even be replaced by a hidden human operator during Wizard of Oz simulations (Chapter 4), allowing to speed up the development process. Finally, at the end of this chapter, we provide an overview of applications created using the ISPM (section 3.7) and compare vocal-only and screen-equipped dialogue systems (section 3.8).

3.1 Information seeking systems

Our methodology is intended for prototyping of information seeking systems. In such systems, the user is looking for information within a potentially huge *search space* by specifying various *search constrains* of the sought information. Such searches are often iterative – the user refines or modifies his/her request based on current state of the search space.

An example of an information seeking system is a public transport information service [25]. Such system can provide several semantically different types of information, for example “trip information” (what bus lines one has to use to travel from one place to another at a particular time of the day) or a complete timetable for a given bus line and station. In order to find such information, the user has to specify the departure and arrival locations, the time of departure or arrival, and perhaps additional constraints (maximum number of transfers, restriction to specific types of transport types, etc). Of course, not all of these constraints have to always be specified, and often the need for specifying them depends on the type of sought information and/or previously provided constraints.

A natural language dialogue is suitable for such systems, because it allows users to formulate their information needs in a naturally intuitive manner and can optimally advance the information search. Typically, the user freely specifies his/her initial information need at the beginning of the dialogue. Depending on the current search space and the predefined system initiative, the system can then ask for additional information or the user can provide additional information in a more structured way after he/she has examined the current search space.

However, verbal communication alone is not necessarily always the optimal mode of interaction. As for the output modality, the screen can quickly provide much more information than a spoken prompt. For example, the screen can at one time display the search constraints provided so far, an overview of the current search space or the signalization of conflicting search constraints. The user can voluntarily focus his/her attention on the information that is of his/her interest. The verbal prompts are more intrusive and they can therefore better focus user attention on a specific situation, with a risk that the user might be annoyed by that information in case if he/she was already aware of it or if the verbal prompt is too long. We believe that equipping the system with both the screen and verbal output allows for optimal system output.

Concerning the input modalities, natural speech allows the user to directly and quickly express several constraints in one dialogue turn, as opposed to pointing inputs that might be considered as too restrictive with low expressive power. However, the user's final choice of input modality is often influenced also by other factors, like the performance of natural language technologies, the willingness of the user to speak/type or the user's background knowledge. An example of background knowledge influencing the choice of modality is obvious in case of providing the departure bus stop information [25]: if the user does not know the name of the actual bus stop and only knows that it is in a certain area, asking the user to provide a name of bus stop is not of much help. In these cases pointing on a map is much more useful. On the other hand, the map interaction requires that the user knows the geographic location of the bus stop. This is not always the case, especially if the user is not familiar with the town. In such cases, it might be better to enter the name using speech or typed text input.

All above mentioned reasons lead us to believe that dialogue systems extended with multimodal inputs and outputs are well suited for information seeking tasks.

3.2 Domain model

The domain model represents a specific knowledge about the applicative domain under consideration. The implementation of such knowledge can vary depending on the complexity and size of the domain. It can take a form of a simple relational table (e.g. commands for controlling smart home devices) or a large relational database schema (e.g. flight information). The domain model can also include a complex problem solver (e.g. public transport routing information). In order to encompass all such varying representations, the ISPM needs to take a unified view on the domain model.

In ISPM, the domain model is seen as a set of *information targets* sought by the user. Each target is individually described by a set of *attributes* in form of `name:value` pair. The domain model is implemented as an object with the following methods (functionalities):

- `getAttributes` indicates the names of all attributes that are used to describe the targets. Such attributes are presented to the user as possible search constraint categories.

- `setConstraints` selects targets compatible with given search constraints expressed by the user. The search constraints are in the form of a set of `name:value` pairs. The targets constrained by the search constraints are called the *current search space*.
- `getCurrentSearchSpaceSize` indicates the overall number of targets compatible with current search constraints. The information is used by the dialogue manager to decide if the search targets are few enough to be directly presented to the user, or if the dialogue should advance with the search criteria elicitation or if the over-constrained situation (i.e. when no targets are compatible) should be solved.
- `getCompatibleValuesForAttribute` For a given name of an attribute, the domain model has to return all values of that attribute name, such that those values are used to describe the targets in the current search space. In other words, the domain model indicates a set of values (of given name) compatible with the current search space.

  The results of this function are (1) presented to the user in order to enable a direct selection of a value which reduces the current search space without generating an over-constrained situation, (2) shown to the user in order to indicate the characteristics of the current search space (e.g. after selecting a departing airport, the system shows only destinations directly reachable from that airport), and (3) used by the dialogue manger to optimize the constraint elicitation phase of the dialogue, as the dialogue manager does not elicit from the user preferences for attributes that have less than 2 possible values.
- `getCompatibleObjects` In certain cases, some individual attributes form a logical object. Hereafter, we call such a set of attributes the *composite object*. An example of a composite object is `Person` that is decomposed into individual attributes `FirstName` and `FamilyName`. From a user's perspective, it is natural to view and select such objects directly as tuples consisting of individual attributes, while the possibility of selecting only individual attributes should still be preserved (for instance, a user may know and select only a first name of a person). The domain model supports that option by implementing this method, which returns tuples of values describing the current search space based on the given set of individual attribute names.

  In other words, this method is similar to the `getCompatibleValuesForAttribute` method, but it operates on a set of attribute names and returns tuples of values. The set of attribute names *is not* formed by random attribute name combinations, but only by the attribute combinations that form logical objects (such as person, date, address, etc) which are expected to be presented as tuples of values to the user during the interaction.
- `getExtraInfo` The domain model might be required to provide additional specific information about attribute values or targets, for example bus stop coordinates or to indicate the number of targets associated with each available constraint value. Such information is not crucial for a multimodal dialogue system, but can improve graphical output of the system and thus facilitate the information search process for the user.

The domain model can actually be seen as a virtual table shown in Table 3.1, where the domain model functions are selections or projections on columns. The presented virtual table is the domain model of an application for providing information about restaurants.

Each restaurant is described by its location, type of food and opening hours (the opening hours attribute is decomposed into day and time of day). Those attributes can be used by the user (and the system) to search for a relevant restaurant. The unique identifier of the restaurant (information target) and the detailed restaurant information (which are to be provided to user) are on the right side of the table.

		Opening hours		*Restaurant*		
Location	*Foodtype*	*Day*	*Time*	*ID*	*Name*	*Site*
...				...		
church	chinese	Sat	10 - 22	6	Wok	`http:// ...`
station	null	Sun	15 - 20	7	Le Café	`http:// ...`
station	null	Mon	8 - 18	7	Le Café	`http:// ...`
...				...		

Table 3.1: The domain model of an application in form of virtual table.

As it can be seen in the virtual table, the restaurant "Le Café" is described by several values of `Opening hours` attribute. This fact has to be represented using several rows in the table, although the single target (restaurant "Le Café") is still uniquely identified by a single unique ID. Unfortunately, this approach leads to a combinatory explosion of the number of rows in cases when an information target is described by several multi-value attributes.

Another aspect is that the information targets do not need to be identified by all attributes. For example, the "Le Café" does not provide any food and therefore the `Foodtype` attribute has `null` value. This example also illustrates a possibility of modeling several semantic types of information targets – restaurants and bars (with no food). If this feature is important for the application or user, the target description can be extended by another attribute `TypeOfService`, which takes the type of information target into account.

In summary, the concept of virtual table makes it easier to understand the domain model and the rest of the ISPM. Nevertheless, implementing the domain model directly in form of a table has several drawbacks and we recommend it only for simple and rather small domains. In such situations, the ISPM facilitates the implementation of the domain model by providing a Java object that only needs to be initialized with data in file with a simple textual tabular format. The advantage is a quick design of small domains, and no need to implement anything in Java.

In case of more complex domains, it is more reasonable that the system designer implements the domain model functions using any convenient domain knowledge representation (SQL, problem solver, composition of webservices, etc). The only requirement imposed by our framework is that the domain model is finally wrapped as a Java object that implements all necessary domain model functions (mentioned above) through a specific interface within our Java implementation of the ISPM.

3.3 Multimodal generic dialogue node (mGDN)

In our approach, an elementary building block of a multimodal dialogue application is called multimodal generic dialogue node (mGDN). The mGDN serves as a *context-specific* *communication mediator* between the core of dialogue management and the user. The mGDN represents the current focus of the dialogue (the question under discussion) and controls all multimodal inputs and outputs within its scope. In particular, the mGDN communicates with the user using natural language and provides interactive graphical component.

The concept of the mGDN comes from its predecessor GDN, the generic dialogue node used to define the communication context (i.e. the question under discussion) in vocal dialogue systems. The term GDN was first used by Bilange [8], and adapted by Rajman et al. [94, 95] and Melichar [66].

The system designer has to decompose the foreseen application into a set of communication contexts and associate each context with one mGDN. Because this can be done properly only when knowing the expressive power of mGDN, this section first introduces the mGDN design principles and then provides examples of already implemented reusable mGDNs. The generic, internal operational model of an mGDN is presented in the following section 3.4. Section 3.5 is dedicated to building an application from individual mGDNs.

Typically, an mGDN is associated with an attribute of a domain model and its role is to perform a simple multimodal interaction with the user in order to obtain a value representing the user's preference for the associated attribute (a search constraint). An mGDN can also be associated with multiple attributes – this is required in cases when the user should be able to make a selection of compound attribute values (person or address). Besides mGDNs associated with some attribute in the task model, the system also contains additional mGDNs used either to present search results (target) to the user, or mGDNs invoked by the dialogue manager in specific situations, such as at the beginning of the dialogue or when the user over-specifies his/her request.

3.3.1 Design principles

Depending on their role in the system, various mGDNs can have different graphical layouts and take different models of interaction with the user. However, in order to ensure that the complete system composed from individual mGDNs is coherent as a whole, every mGDN has to respect the following design principles:

- The mGDN is *an elementary building block* of the multimodal interface. There are several types of mGDNs, each encapsulating a particular kind of interaction and providing various graphical layouts (an overview is in section 3.3.2). This allows rapid building of multimodal dialogue systems by exploiting existing building blocks.
- The mGDNs represent *the only interaction channel* of the user with the system. In other words, all inputs/outputs going to/coming from the system must be managed by some mGDN. The underlying design implication is that every active piece of the graphical user interface must be related to some mGDN.
- Every mGDN is *fully multimodal*, enabling the user to interact with the mGDN using any combination of the modalities available in the application. This ensures that the user can interact with the system in the most convenient and comfortable way, and allows the user to switch modalities if a particular modality does not have

the desired effect on the system. Thus, the use of fully multimodal mGDNs improves the robustness of the system itself.

- Only *one single mGDN is in focus* at any given time. This principle ensures the overall feasibility of the system by constraining, in a logical fashion, the possible interactions with the system and the interpretation contexts for multimodal fusion at any given point in time. The mGDN in focus should be clearly marked.

  Even though only one mGDN is in focus, several other mGDNs can be *active*, i.e. ready to process the input provided to the mGDN in focus. This functionality is needed to support the mixed initiative dialogues.

- The mGDNs must be *independent entities* clearly separated from the rest of the system so that everybody can design their own mGDNs and plug them into the system. This is important since building practical multimodal systems for a particular domain might reveal the need for mGDNs supporting a special kind of interaction or a special treatment of modalities. At the same time, it is useful to design mGDNs generic enough so they can be reused in situations with similar interaction needs. Then, the application can be created from a small number of reasonably generic, customizable mGDNs. A convenient solution for mGDN design (used in the ISPM implementation) is their customization with a simple declarative configuration file, leading to fast design of systems through reusability of already implemented mGDNs.

3.3.2 Examples of mGDNs with various interaction models

Any methodology for prototyping of software systems needs to find its *balance between flexibility and speed* of design.

Designing a system using predefined elementary blocks is fast, although the available blocks might not be completely appropriate for the targeted system. Having a huge library of miscellaneous building blocks is not an optimal solution either, as the cost of building such a library would be too high and it is still not guaranteed that all the necessary blocks are available in the library. A better approach is to allow each building block to be customized with some parameters. The cost of initial implementation of such a block is slightly higher than in case of a non-customizable block, but the reusability of building blocks is substantially enhanced.

This solution was adopted for the design of mGDNs. An mGDN is implemented in form of a *Java class* with a well-specified interface and an *associated graphical component*. The mGDN implementation can be *customized* with declarative parameters, mainly concerning linguistic information (prompts, grammar), tailoring it to its specific applicative use.

In our experience, an application is composed of a larger number of mGDNs, most of them having the same internal functionality and mechanisms for interaction with users. It is therefore straightforward to use one (or a low number of) mGDN implementation and customize it with context-specific parameters. The parameters usually represent the data that the mGDN manipulates with (prompts, grammar, and items to be presented to the user) and can therefore influence the interaction behavior of the mGDN only to a certain extent.

If the internal and external functioning of an mGDN (i.e. the interaction model) needs to be substantially altered, it is better to use another mGDN implementation with its own set of customization parameters. We have therefore implemented several mGDNs, which do differ in the way of interaction with the user or in their main purpose. We give an overview of the customizable mGDNs in the following subsections together with examples of situations where they should be used.

If no suitable mGDN is available for a given application, we give the system designer a possibility to implement (in Java) a specific mGDN, which can fully satisfy the application needs. In order to preserve the coherence and scalability of the overall solution, we strongly advice to keep the design principles formulated in section 3.3.1 during the design and implementation of new mGDNs.

We believe that giving the designer a possibility of building an application from a few existing types of main blocks (mGDNs) that can be easily customized to a specific context of use, as well as a possibility of implementing own specific building blocks, brings the right balance between speed and flexibility of the system design. The very first prototype can be quickly build only from predefined blocks, and later it can be iteratively improved by replacing the generic building blocks (mGDNs) by newly implemented blocks (which is, of course, a more time-consuming process).

In the rest of this section, we provide an overview of existing examples of mGDNs with various interaction models and layouts.

i. Simple value selection

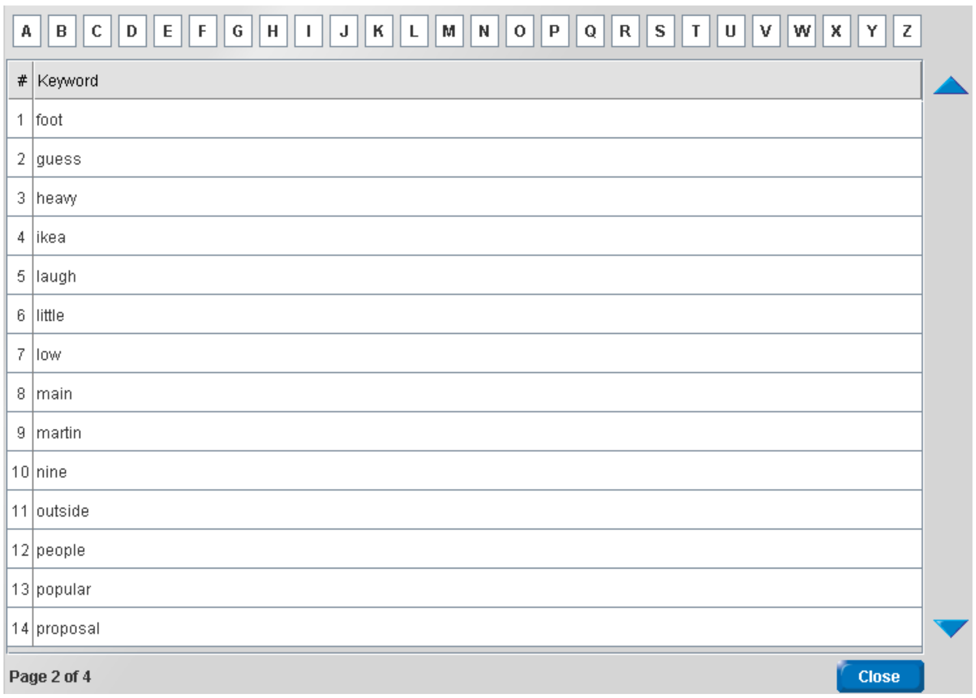
This mGDN is intended for situations when the user is supposed to select a value for an attribute from a predefined list of values. The values usually represent different search criteria of the same category that are to be specified by the user.

An associated textual and vocal prompt first asks the user about his/her preference for a value of the given attribute and displays a list of possible values. The user may directly provide the value, browse the list for available options, ask for clarification or repetition of the prompt. The user may also decide not to select any value by simply closing the mGDN (see Figure 3.1).

The user may provide the value in several ways:

- The user can *directly say or type* the value. This option is available even when the value is not visible on screen.
- The user can select a value by indicating the associated *numerical identifier* displayed next to the desired value. This option is convenient in cases when the performance of a speech recognition module is low or when the value is linguistically too complex or too long to be effortlessly pronounced.
- The user can use the *pointing device* (mouse, pen, touchscreen) to click on the desired value. In those cases, scrolling the list in order to display the desired value is essential.

The values displayed in the list can be from a predictable set of values (e.g. days of weeks, months of year) or from unpredictable set of values (e.g. movie titles, search keywords).



#	Keyword
1	foot
2	guess
3	heavy
4	ikea
5	laugh
6	little
7	low
8	main
9	martin
10	nine
11	outside
12	people
13	popular
14	proposal

Page 2 of 4

Figure 3.1: mGDN layout: selecting a search constraint (keyword) from a list

The possibility of browsing through the list is especially useful when values are from an unpredictable set. The user has an option of scrolling the list up and down or jumping directly to values starting with a specific letter – all such actions may be performed using any modality combination. The "letter indicators" disappear from the top of the screen when all values fit in one screen and hence no scrolling is needed.

This mGDN is intended for situations when values are from both a predictable and an unpredictable set of values. The number of values in the list can be small or large. However, when the graphical component is displaying a large number of unpredictable values, the performance is not optimal as the user has to scroll over many pages. In those situations, another mGDN offering, for instance, an approximate search within values would be recommended [80].

ii. Geographical data selection

This mGDN is suitable for situations when the user should indicate a geographical location. The geographical locations are displayed on a map, see Figure 3.2.

Displaying geographical locations on a map is particularly useful when the user is looking for locations in a specific area or near some place, or when the user is only partially familiar with the local geographical situation – the user may roughly know the map of the area, but does not remember the local names.

The geographical locations are displayed on the map in two colors: in red are the locations compatible with the current search constraints and in black with grey names are all other locations. This is convenient for the user because of (at least) two reasons:

- When values compatible with the current search constraints are indicated, the user can avoid selecting a search constraint that causes the search to be over-constrained,

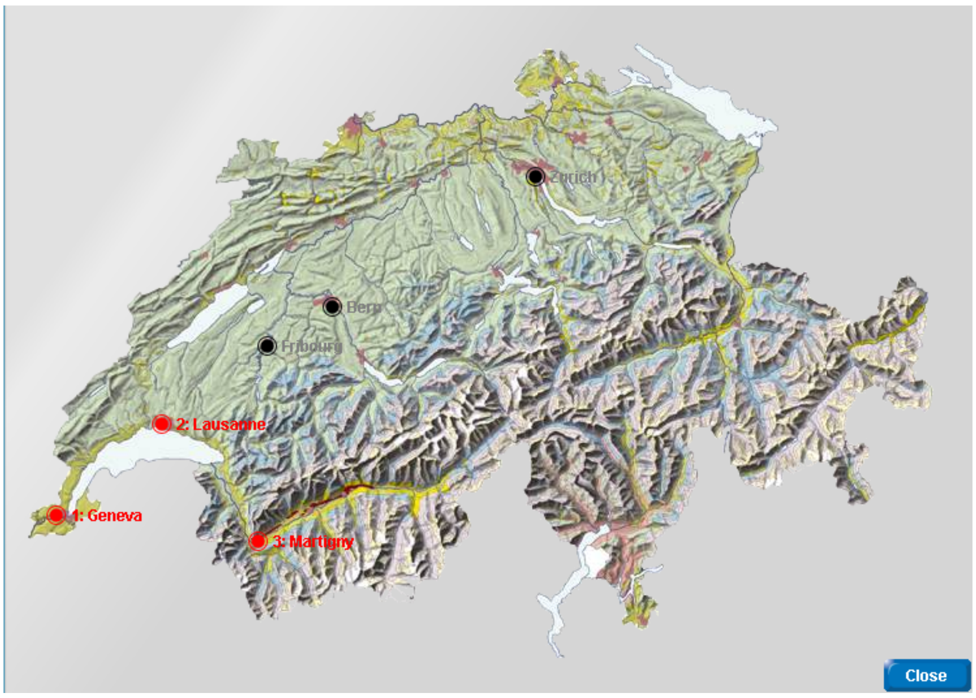


Figure 3.2: mGDN layout: selecting a city on a map

i.e. that no solution satisfy all the search criteria. This feature may therefore accelerate the search process when multiple values are candidates for search constraints.

- The mGDN gives an overview about the current search space. For instance, a restaurant information system can indicate that the user-selected Chinese restaurants with low prices are available only in certain cities.

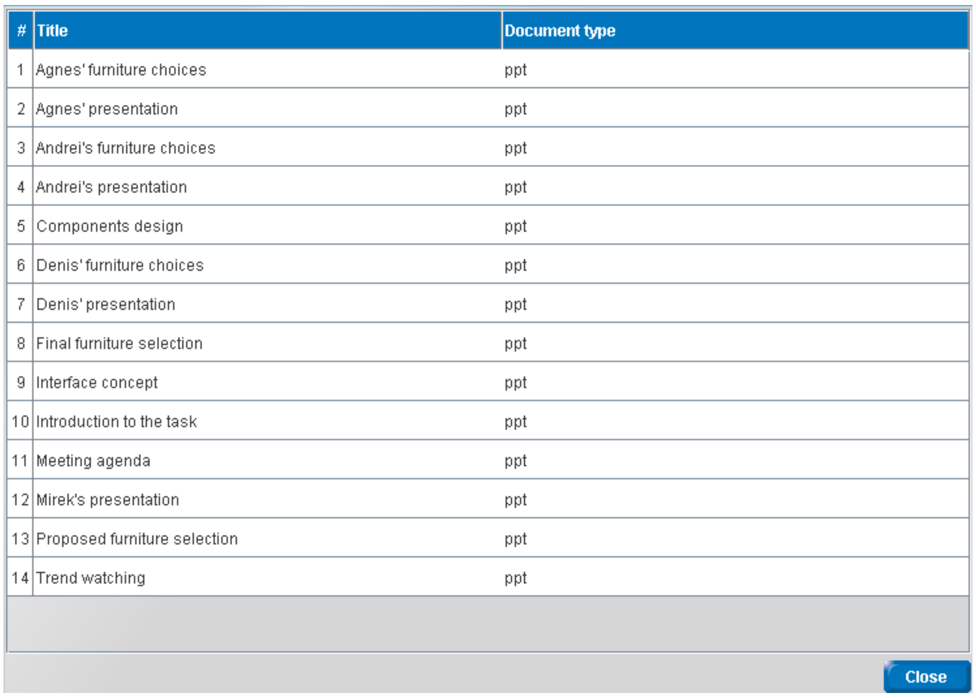
The mGDN gets the information about values compatible with the current search constraints by calling the functions of the domain model (see section 3.2).

In this particular mGDN implementation, we have not implemented zooming and scrolling features of the map, because the static map of Switzerland fully satisfied our needs. Adding those features should be relatively straightforward, analogical to scrolling features of the mGDN for simple value selection.

iii. Composite value selection

Often, users have to indicate a composite object, i.e. an object consisting of several elementary attributes. Examples of such objects are: person (first name, family name, address), address (street, city, state), document (name, type, publisher), date (day, month, year), etc.

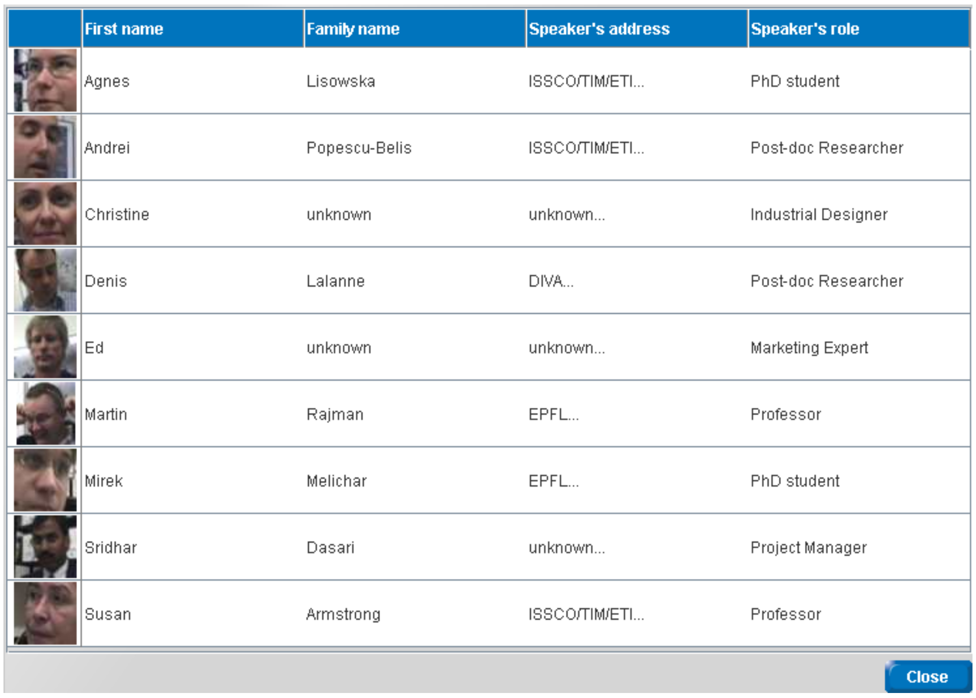
Although it is possible to decompose such objects into elementary attributes and ask the user to specify individual values for those attributes, it is much more convenient for the user to select directly the whole object, especially in cases when the number of these objects is reasonable low. In such a situation, the mGDN for composite value selection can be used. The layout of that mGDN is shown in Figure 3.3. Every line represents an individual object and the columns indicate the individual attributes. The option of



#	Title	Document type
1	Agnes' furniture choices	ppt
2	Agnes' presentation	ppt
3	Andrei's furniture choices	ppt
4	Andrei's presentation	ppt
5	Components design	ppt
6	Denis' furniture choices	ppt
7	Denis' presentation	ppt
8	Final furniture selection	ppt
9	Interface concept	ppt
10	Introduction to the task	ppt
11	Meeting agenda	ppt
12	Mirek's presentation	ppt
13	Proposed furniture selection	ppt
14	Trend watching	ppt

Figure 3.3: mGDN layout: selecting a document (title, type) from a list

selecting only the individual attributes is still available and the individual attributes can be accessed by selecting their name in a column header.

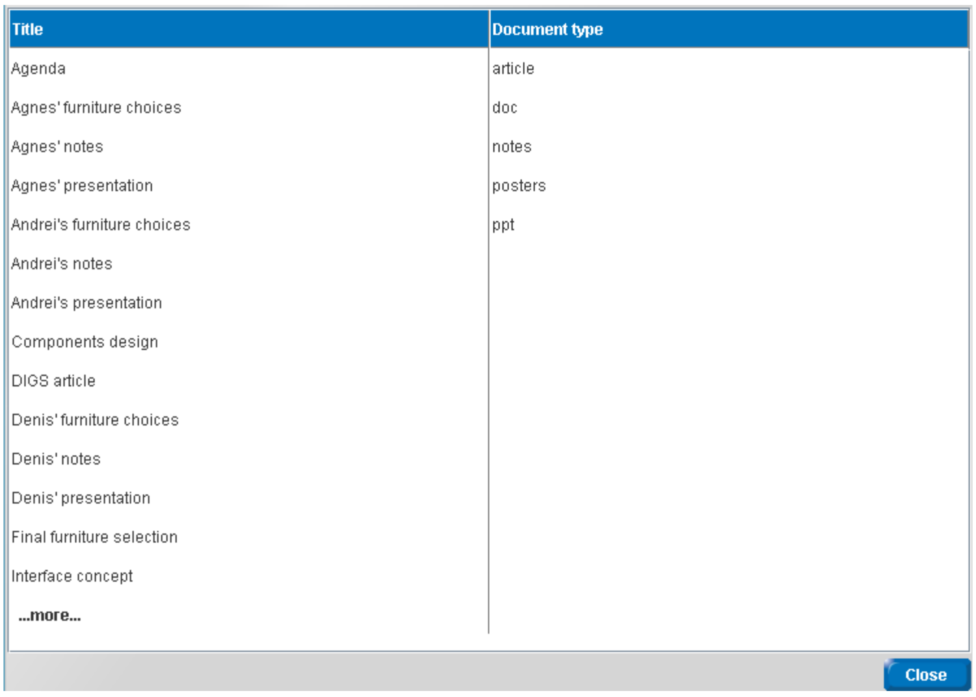


	First name	Family name	Speaker's address	Speaker's role
	Agnes	Lisowska	ISSCO/TIM/ETI...	PhD student
	Andrei	Popescu-Belis	ISSCO/TIM/ETI...	Post-doc Researcher
	Christine	unknown	unknown...	Industrial Designer
	Denis	Lalanne	DIVA...	Post-doc Researcher
	Ed	unknown	unknown...	Marketing Expert
	Martin	Rajman	EPFL...	Professor
	Mirek	Melichar	EPFL...	PhD student
	Sridhar	Dasari	unknown...	Project Manager
	Susan	Armstrong	ISSCO/TIM/ETI...	Professor

Figure 3.4: mGDN layout: selecting a meeting participant (first and family name, address, role) from a list

Since composite objects often represent physical objects in the real world, an image of such an object may help the user to select the sought object (for instance, a person's face). We have therefore also implemented a possibility of displaying object icons together with the objects, as shown in Figure 3.4.

Although the mGDN gives the user an impression of selecting whole objects, the objects are internally represented as a set of individual attributes related to underlying mGDNs. As a consequence, by selecting a value in a composite mGDN the user actually selects a set of individual values associated with the object.



Title	Document type
Agenda	article
Agnes' furniture choices	doc
Agnes' notes	notes
Agnes' presentation	posters
Andrei's furniture choices	ppt
Andrei's notes	
Andrei's presentation	
Components design	
DIGS article	
Denis' furniture choices	
Denis' notes	
Denis' presentation	
Final furniture selection	
Interface concept	
...more...	

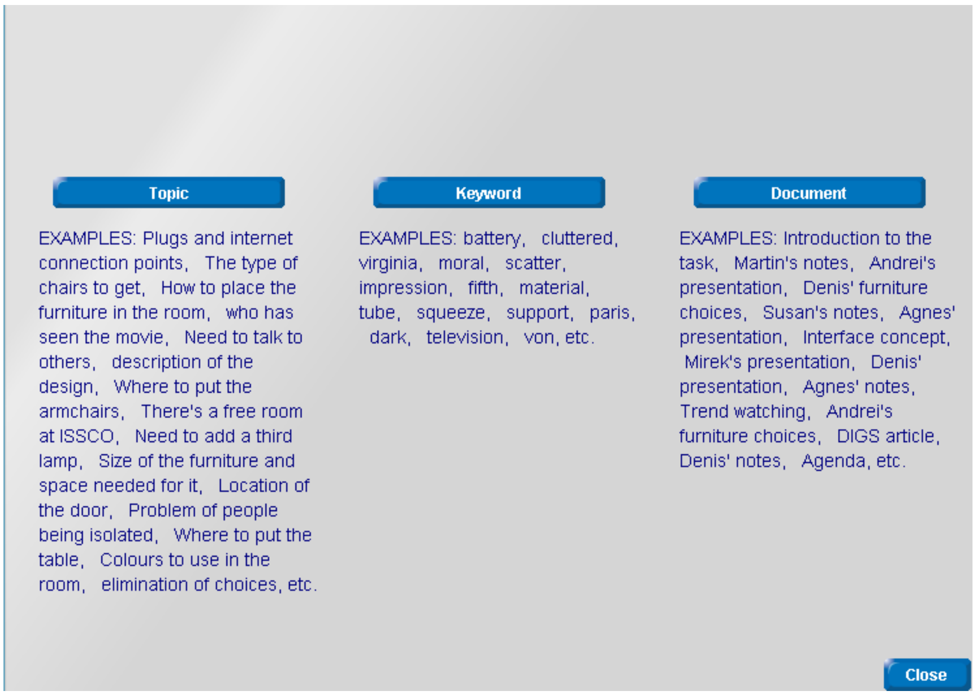
Figure 3.5: mGDN layout: selecting individual attribute values of a document (title, type) from columns, because the number of documents is too high to fit in one screen

In cases when the number of complex values is too high and cannot fit in one screen, the mGDN automatically changes its layout to a column layout (see Figure 3.5). This layout displays values of individual attributes, rather than whole composite objects. In many cases, the number of values for individual attributes is low enough to comfortably fit in one screen, in contrary to the number of objects composed from those attributes. The user may directly select individual attributes (e.g. types of sought documents or persons with a given function). As soon as the number of composite objects becomes reasonably low, the mGDN changes its layout to display again the whole objects.

iv. Switch

Often, the attributes in the system are only loosely coupled. They do not form composite objects, but they still have some common properties. It is a good practice to group such attributes into one visual space, helping the user to access individual or composite attributes deeper in the hierarchy.

A switch mGDN is perfectly suited for this type of hierarchical grouping (see Figure 3.6). Concerning the internal functionality, it behaves similarly as the composite value selection mGDN with column layout. However, from the user perspective, it is more like a menu with sub-menu items. In order to better indicate the meaning of each sub-menu item, examples of values available in the sub-menu items are displayed on the screen.



EXAMPLES: Plugs and internet connection points, The type of chairs to get, How to place the furniture in the room, who has seen the movie, Need to talk to others, description of the design, Where to put the armchairs, There's a free room at ISSCO, Need to add a third lamp, Size of the furniture and space needed for it, Location of the door, Problem of people being isolated, Where to put the table, Colours to use in the room, elimination of choices, etc.

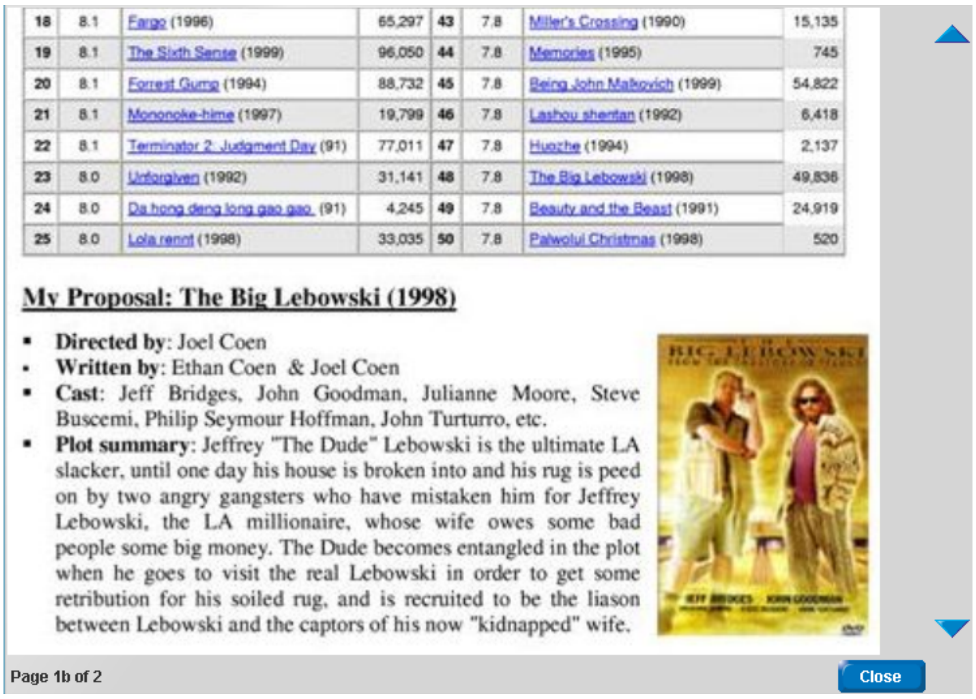
EXAMPLES: battery, cluttered, virginia, moral, scatter, impression, fifth, material, tube, squeeze, support, paris, dark, television, von, etc.

EXAMPLES: Introduction to the task, Martin's notes, Andrei's presentation, Denis' furniture choices, Susan's notes, Agnes' presentation, Interface concept, Mirek's presentation, Denis' presentation, Agnes' notes, Trend watching, Andrei's furniture choices, DIGS article, Denis' notes, Agenda, etc.

Figure 3.6: mGDN layout: a menu with three sub-menus, examples of values in sub-menus are shown

v. Document browser

Not every mGDN gathers the search constraints or preferences of a user (i.e. is not necessarily associated with a constraint attribute of the domain model). A document browser mGDN is intended for presenting a document (or any other page-based graphics) to the user and is typically associated with an attribute of the domain model that describes



18	8.1	Fargo (1996)	65,297	43	7.8	Miller's Crossing (1990)	15,135
19	8.1	The Sixth Sense (1999)	96,050	44	7.8	Memories (1995)	745
20	8.1	Forrest Gump (1994)	88,732	45	7.8	Being John Malkovich (1999)	54,822
21	8.1	Mononoke-hime (1997)	19,799	46	7.8	Lashou shentan (1992)	6,418
22	8.1	Terminator 2: Judgment Day (91)	77,011	47	7.8	Huozhe (1994)	2,137
23	8.0	Unforgiven (1992)	31,141	48	7.8	The Big Lebowski (1998)	49,836
24	8.0	Da hong deng long gao gao (91)	4,245	49	7.8	Beauty and the Beast (1991)	24,919
25	8.0	Lola rennt (1998)	33,035	50	7.8	Palwolui Christmas (1998)	520

My Proposal: The Big Lebowski (1998)

- **Directed by**: Joel Coen
- **Written by**: Ethan Coen & Joel Coen
- **Cast**: Jeff Bridges, John Goodman, Julianne Moore, Steve Buscemi, Philip Seymour Hoffman, John Turturro, etc.
- **Plot summary**: Jeffrey "The Dude" Lebowski is the ultimate LA slacker, until one day his house is broken into and his rug is peed on by two angry gangsters who have mistaken him for Jeffrey Lebowski, the LA millionaire, whose wife owes some bad people some big money. The Dude becomes entangled in the plot when he goes to visit the real Lebowski in order to get some retribution for his soiled rug, and is recruited to be the liason between Lebowski and the captors of his now "kidnapped" wife.

Figure 3.7: mGDN layout: document browser

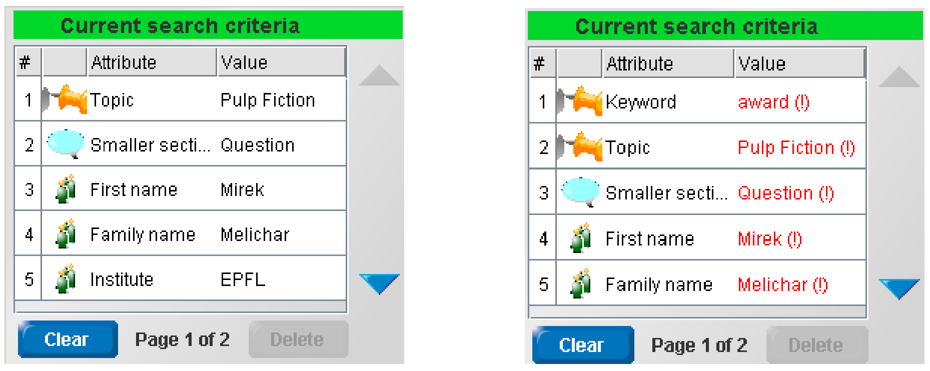
search targets. The user can scroll the document up and down and even partial views on document pages are possible, see Figure 3.7.

The document selection (specification) is not a part of this mGDN. The document is selected as a result of interaction with another mGDN(s). The document pages are obtained in form of links to document pages (graphical images) from the domain model.

vi. Current search criteria

This mGDN allows to manage a set of current search constraints, previously specified by user. Following functionality is available in the mGDN:

- The mGDN provides an *overview of all constraints* understood by the system, and thus the structure of the current query (see Figure 3.8(a)). It is especially important if natural language recognition is involved – the mGDN provides feedback and confirmation about what the system has understood. It is also useful when the interaction is longer and the user is unsure what are all the constraints currently used by the system.
- The mGDN *indicates situations when conflicting search constraints were specified* and no solution is therefore available (see Figure 3.8(b)). The conflicting search constraints are in red color, flashing and marked with exclamation mark in order to attract user's attention to the problem.
- The user can *erase* some of the search constraints, either as a reaction to over-constrained situation or voluntarily when he/she wants to modify the query. Search constraints can be erased either one-by-one using the erase command or all together using the clear command.


Current search criteria

#	Attribute	Value
1	Topic	Pulp Fiction
2	Smaller secti...	Question
3	First name	Mirek
4	Family name	Melichar
5	Institute	EPFL

(a) Search constraints previously provided by user

Current search criteria

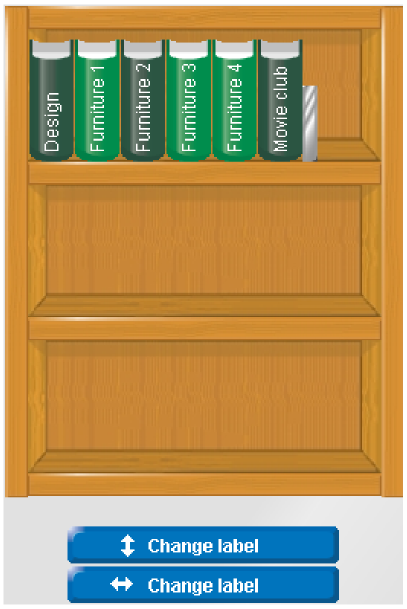
#	Attribute	Value
1	Keyword	award (!)
2	Topic	Pulp Fiction (!)
3	Smaller secti...	Question (!)
4	First name	Mirek (!)
5	Family name	Melichar (!)

(b) Indication of over-constrained situation

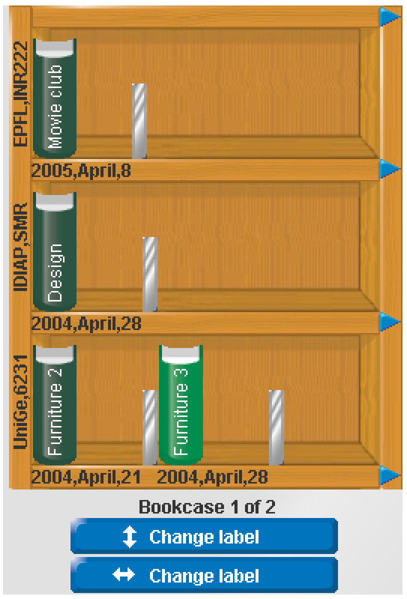
Figure 3.8: mGDN layout: visualization of current search constraints

vii. Search space overview

During the information search dialogue, the user is progressively expressing search constraints in order to find the sought information. An overview of the search space helps the user to better imagine the nature of the current search space and therefore more efficiently steer the constraint specification process.



(a) Targets satisfying the search query are displayed as light-green books



(b) Books can be organized in two dimensions, using shelf and leg labels

Figure 3.9: mGDN layout: a search space overview – the library metaphor

In our implementation, we present the search space using the library metaphor (see Figure 3.9). The main units of sought information are displayed as books in a bookcase. The books containing information relevant to the current query are displayed as light-green books and the others are displayed in dark-green. The user can open the book in order to find details of the given book (the book is displayed in context of another mGDN).

Sometimes, the information sought by the user is actually spread over the whole search space. For instance, imagine a database of meetings and a user wanting to find out where most of the meetings took place. In such case, specifying individually each possible location of meetings and counting how many meetings happened there becomes a tedious process.

However, our search space overview mGDN allows arranging the books by miscellaneous criteria in two dimensions: labels can be associated with the legs and the shelves of bookcase, see Figure 3.9(b). The labels can be selected using corresponding buttons at the bottom of the bookcase – the association is done in context of another mGDN, typically a simple value selection mGDN. When the bookcase becomes too small to display all books, blue scroll arrows appear on left and/or right of the bookcase and the user can move to other bookcases as he/she would do in a normal library.

The question of "where most of the meetings took place?" can be now easily solved by assigning the place of meetings to the legs of bookcase and the most occupied shelf indicates the place where most of the meetings took place.

3.4 Operational model of an mGDN

A role of an mGDN is to intermediate communication between the user and the core of dialogue management in a specific context. A context is typically a system request to select a value for an attribute from a predefined list of possible options, presenting search results to the user or resolving some dialogue problem. After the core of the dialogue management selects the context (an mGDN), the control is passed to the selected mGDN, which becomes *responsible for the communication with user.* The mGDN returns the control back to the dialogue management core after a satisfactory response has been obtained from the user. All *local* interaction (such as scrolling the list of values, providing the help, etc) is resolved at the mGDN level.

The mGDN is initialized with information about the state of the global dialogue manager, has permission to read information contained in the domain model, is associated with a corresponding graphical component and communicates with input and output managers. The input and output managers are system modules responsible for gathering the multimodal response from the user, for issuing system prompts in natural language and for all necessary synchronization (timings) of the multimodal interaction. The input and output managers are in practice connected to a chain of other modules that actually gathers and processes the user's response and realizes the system output.

Once set under focus, an mGDN operates in a loop decomposed in three phases as described in the following sections.

3.4.1 Phase 1: Presenting a multimodal request to the user (multimodal fission)

Based on the system state (number of values to be presented, previous response of the user, etc), the system request is prepared in the form of a textual and vocal natural language prompt. The mGDN also updates the layout of the associated graphical component and the displayed values.

Depending on the system state, the mGDN may decide to present the request as an open-ended question (*"What is your preferred X?"*) or as a command (*"Select your preference for X"*). The system may give help to the user (*"Possible values for X are: ..."*) and decide to present a short or long (or more or less polite) version of the textual/vocal prompt. In cases when the system asks the user to select an item among several options, the vocal prompt can either enumerate all these options or the options can be displayed only on the graphical component and the prompt can only refer to displayed options (*"Select one of X displayed on the screen"*). The choice of graphical presentation is influenced by a combination of factors like the *size of the screen*, *precision of the input device* (mouse, pen, touchscreen), the *size of information to be displayed* (number of options the user is asked to select from) and the *nature of the displayed information* (map, list of persons). We showed some examples of developed mGDNs and their information presentation strategies in section 3.3.2.

In our Java implementation of the prototyping methodology, mGDNs modify the state of the associated graphical component directly by invoking component-specific methods. The vocal and textual versions of the natural language output are prepared and sent to the input-output manager (IOmanager). The system contains one single IOmanager shared by all mGDNs. Having only a single IOmanager contributes to a coherent presentation of natural language requests to the user: the textual version of the prompt is always displayed in the same area and the vocal version of the prompt keeps the same audio properties as it is always synthesized by the same text-to-speech synthesizer. The IOmanager also consistently manages all synchronizations related to barge-in and input-output timings. This approach also makes it straightforward to add new output modalities: the new output modalities will be centrally realized by the IOmanager and the mGDN's definition of multimodal output messages will be extended. However, if the new output modality is highly context-sensitive, the control and realization of the multimodal output can be partially performed by the mGDN itself (similarly to how it is done with the control of graphical components).

Although we use our own format for describing the multimodal output messages, it is realistic to assume that mGDNs can prepare the whole multimodal presentation in one of the standardized languages, such as SMIL (Synchronized Multimedia Integration Language, see section 2.4 on page 22). Then, the IOmanager would simply be replaced by a SMIL-compatible player.

3.4.2 Phase 2: Gathering user's response (multimodal fusion)

When a multimodal output starts to be presented to the user, the system must be ready to receive the user's response. The user can respond using any interaction channel or their combination.

In our approach, we foresee a *specialized recognizer for each interaction channel.* The recognizers can be configured dynamically based on the current interaction context (mGDN) and are required to *produce a set of semantic pairs* (set of `name:value` pairs) describing the user's input in a given modality. Each semantic pair is augmented by a timestamp to facilitate the fusion of results of different recognizers. For instance, a speech recognizer coupled with natural language understanding modules is considered as one recognizer, with the possibility of adjusting the language model and interpretation process depending on the current mGDN. The output of such a recognizer is a set of semantic pairs expressing the semantics of the user's last utterance within the context of the current mGDN. Another example of a recognizer is a graphical component that produces semantic pairs as a response to a user pointing on it.

The 'semantic pairs set' formalism is sufficient for most of user inputs provided within a restricted interaction context in a specific domain. For instance, the user utterance *"I would like a big pizza with ham and cheese and two beers"* can be expressed as the following set of semantic pairs: {`pizza_size:large`, `pizza_topping:ham`, `pizza_topping:cheese`, `drink_quantity:2`, `drink_type:beer`} (assuming the semantic pair names and values were predefined in the domain). However, this formalism is unable to efficiently express relations between provided attributes – it cannot correctly describe user inputs like *"I want one small pizza with pepperoni and one big pizza with mushrooms"*. Since such a limitation is inherent to all typical frame-based dialogue models (see section 2.3.2), we consider the set of semantic pairs formalism to be sufficient for our (essentially frame-based) approach to dialogue modelling.

Sets of semantic pairs produced by individual recognizers are *fused* in the modality fusion module. The fusion is usually done by simply merging sets of semantic pairs with the possibility of resolving any cross-modality ambiguities using semantic pair timestamps (notice however that real semantic ambiguities or contradictions in the user's input are not resolved by the fusion module, but can be resolved later by the dialogue management). For instance, the classical example "Put that there" [11] can be resolved in the fusion module by the temporal alignment of the semantic pairs coming from the speech recognizer {`action:move`, `object:?`, `location:?`} and semantic pairs from the pointing modality {`pointingTo:objectX`, `pointingTo:placeY`}, resulting in a final set of semantic pairs completely describing the user's multimodal behavior in the last dialogue turn {`action:move`, `object:objectX`, `location:placeY`}. Yet another responsibility of the fusion module is the time synchronization of recognizers employed in the system and handling their timeout related operations.

The resulting set of semantic pairs describing the user's last multimodal input is then received by the mGDN under focus.

The advantage of this approach is that mGDNs can be created to a certain extent independently of modalities used in the system, thus making experimenting with different modalities simple (no need to change existing mGDNs when adding or removing modalities). Having the user's whole multimodal input expressed in the simple form of semantic pairs enables replacing real recognizers by a human expert during Wizard of Oz simulations (as discussed in Chapter 4). Furthermore, the whole framework for multimodal fusion outlined above can be replaced by another scheme (the research on possible frameworks for multimodal fusion is still ongoing, see [76, 117, 113, 55, 84]), as long as the result is expressed in a form of semantic pairs. All the external components for multimodality processing are actually independent of mGDNs, but these components have information about the currently focused and active mGDNs. Based on this information, the external recognition components can adapt their behavior – the currently focused and active mGDNs naturally define *the context* for interpretation of user's input.

Similarly to the multimodal output of the system, the formalism for expressing user's multimodal input is a subject of W3C standardization activities. The standardized language is called EMMA (Extensible MultiModal Annotation markup language), and it is realistic to assume that EMMA can replace our formalism of semantic pairs. Although EMMA was still a working draft at the time of writing this thesis, it seems that EMMA is an extended version of the "set of semantic pairs" formalism and for that reason, it should be easy to use any EMMA-compatible multimodal fusion framework in our system.

3.4.3 Phase 3: Treating the user's response (local dialogue strategies)

After the user's multimodal response has been expressed in form of semantic pairs, the system has to decide on the *next step in the dialogue.* The decision making mechanism of the dialogue manager is referred to as dialogue strategy. In our approach, the dialogue-management handles dialogue strategies at two levels: local and global.

Local dialogue flow management strategies handle situations that are fully in the scope of a given mGDN. The goal of these strategies is to guide the user towards providing some information related to the mGDN's role in the system. Each mGDN implements itself the local strategies and when the local strategies take place, the control remains at the mGDN level.

A typical responsibility of local dialogue strategies implemented within every mGDN is (1) to provide a help to the user, (2) to resolve situation when recognition of users input fails (namely speech recognition), (3) to react when user is not providing any input, and (4) to handle a user's request for last system prompt repetition or reformulation. In these situations, the mGDN has to classify the situation and to select a relevant system prompt. The processing then continues by Phase 1 (see above in section 3.4.1).

Additional local dialogue strategies depend on the particular type of interactions implemented by a given mGDN. The mGDN that allows selecting a value for an associated attribute from a list (see Figure 3.1) has additionally to strategies (1-4) also to implement reactions to (5) scrolling of the list up and down, (6) selecting a value based on its associated numeral index and (7) direct scrolling to values starting with a specific letter. The mGDN that presents geographical information (see Figure 3.2) has to implement reactions to (8) a user request to zoom in/out and (9) scrolling the map left/right/up/down. The mGDN that presents multimedia recordings (video) has to have (10) reactions to requests of stopping/starting/rewinding the multimedia.

All the above mentioned reactions (local dialogue strategies) are determined by the mGDN by a run-time analysis of semantic pairs produced by individual modality recognizers in Phase 2 of the mGDN operational loop. The information about which modality has produced a particular semantic pair is typically irrelevant. The user can, for instance, require the list to be scrolled down by clicking on a scroll arrow, giving a vocal command, typing a command, or the request for scrolling may be detected from eye gaze of the user. The mGDN reaction is always the same, regardless the modality that the user used to perform the action.

As soon as no local strategy can be applied or the user has provided information required by an mGDN, the mGDN loop terminates and the control is given back to the global dialogue manager.

Global dialogue strategies control the dialogue flow at the level above mGDNs. This includes determining and controlling which mGDNs need to be treated to accomplish the goals, activating those mGDNs, processing newly acquired search constraints from the user and selecting the next most relevant mGDN. Additionally, the global management strategies treat issues such as resolving incoherencies in input, restarting the dialogue on user request, handling dead-ends in the dialogue, or ending the dialogue altogether. The decision about which global strategy should be applied is based on an analysis of the existing set of constraints provided so far by the user during the dialogue, on the user's explicit preference about the next mGDN under focus, and on the model of the application task. These strategies are implemented in the global dialogue manager and will be described in section 3.5.2.

From the perspective of an mGDN, it is important to recognize semantic pairs related to global dialogue strategies and forward them together with power of control to the global dialogue manager. The Table 3.2 provides an overview of all semantic pairs defined in the system.

type		*description*	*naming convention*
global		a value provided by user as a result of mGDN interaction	`<mGDNname>:<value>`
		user's request for new focus (new mGDN to be selected)	`global.newfocus:<mGDNname>`
		user's decision of terminating the interaction with a given mGDN	`global.close:<mGDNname>`
		user's request for system restart	`global.control:restart`
local	shared	request for help	`local:help`
		request for prompt repetition	`local:repeat`
		recognizer failing to understand user input (no match)	specific state of a recognition component, not sem. pair
		user does not provide any input (no input)	
	list	scrolling up/down	`local.scroll:<up|down>`
		scrolling to items starting with selected letter	`local.jumpto:<A..Z>`
		selecting a value from list based on its associated numeral index	`local.list:<1..X>`
	map	zooming map	`local.map.zoom:<in|out>`
		scrolling map	`local.map.scroll:<up|down|left|right>`
	playback	control of media playback	`local.playback:<start|stop|pause|forward|rewind>`
	...	local semantic pairs can be added with new mGDNs	...

Table 3.2: An overview of semantic pairs used by the system.

However, the mGDNs need to resolve situations where *multiple* local strategies could apply. For instance, the user may ask to scroll the list down, ask for help and provide a new value, all that at the same time. Although such a situation seem to be unrealistic, the system must be able to handle it as this may happen not only in cases when a user is testing "what the system can do", but also as a consequence of imperfect recognition of the user's last input. This is usually solved by a careful analysis of priorities of different strategies and their mutual exclusivity.

Yet an additional mechanism is needed for situations where the user provides an input concerning local strategies but for another mGDN. In this case, the current mGDN can recognize the semantic pairs for another mGDN (they are augmented by information about which mGDN they apply to) and treats them as they were semantic pairs for global strategies, i.e. forwards them, together with the control of the system to the global dialogue manger. It is then the responsibility of the global dialogue manager to offer such semantic pairs for further treatment to the concerned mGDNs.

3.5 Task model

A *task* in a dialogue system is an activity that should be performed in order to reach the goal of the communication. A *task model* indicates the logical user activities supported by an application [73, 74]. In an information seeking system, the user performs two main logical activities: provides search constraints to the system and browses the search results. Ideally, the system should support alternation of those two activities, allowing the user to be more efficient during both phases of the dialogue.

In the previous sections, we have defined the domain model that describes the domain knowledge of an application. We have also defined an mGDN that serves as an elementary communication unit and as a building block of an application. Based on those two constructs, we can define the application task model.

The application task model consists of three main parts: (1) the application-dependent declarative specification of all mGDNs in the system and relations between them, (2) the layout of the graphical user interface and (3) the application-independent global dialogue flow management strategies. While the mGDN structure and the layout of the graphical user interface must be defined by the designer of a particular application, the dialogue strategies have been implemented as a part of the dialogue platform and they are generally not supposed to be modified by application designers.

3.5.1 A structure of mGDNs

The system designer has to associate one mGDN with every attribute of the domain model, thus allowing the user to specify any search criteria through an interaction with the associated mGDN.

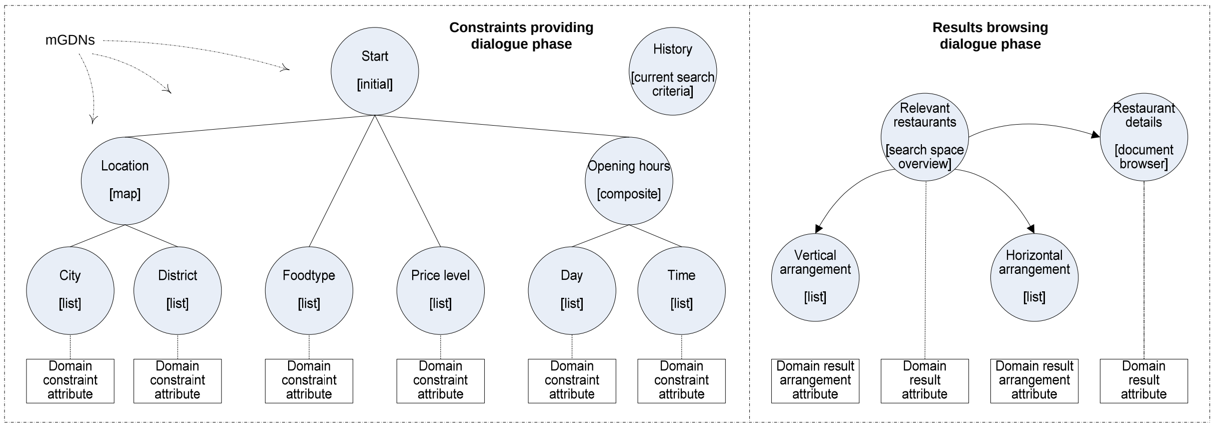

mGDNs
Constraints providing dialogue phase
Start [initial]
History [current search criteria]
Location [map]
Opening hours [composite]
City [list]
District [list]
Foodtype [list]
Price level [list]
Day [list]
Time [list]
Domain constraint attribute
Results browsing dialogue phase
Relevant restaurants [search space overview]
Restaurant details [document browser]
Vertical arrangement [list]
Horizontal arrangement [list]
Domain result arrangement attribute
Domain result attribute

Figure 3.10: Task model: decomposition of an application into a set of mGDNs. The example shows the structure of restaurant information application. The structure supports the two phases of the dialogue communication. The name and type of every mGDN is indicated.

In case of a too large number or variety of such mGDNs in the system, the mGDNs can be organized in a hierarchical structure as shown in Figure 3.10. The mGDNs for composite value selection or with a switch layout are used as inner nodes of the hierarchical structure. The leave nodes of the structure (typically a simple value selection mGDNs) are accessed through inner nodes, making a sort of a hierarchical menu. Because the graphical interface of the application (section 3.5.3) and the interaction flow reflects the hierarchical structure, grouping attributes with similar meaning increases the understandability of the interface and can resolve problems with limited space on the screen. The mGDN hierarchy also defines a natural distance between different concepts in the system and thus can be exploited during disambiguation of natural language input.

In addition to mGDNs associated with the constraint attributes of the domain model, the system also contains specific-purpose mGDNs, such as the Start mGDN in the root of the mGDN hierarchy. The Start mGDN is used at the very beginning of the dialogue and its purpose is to welcome the user, ask a general open-ended question (e.g. *"Which restaurant are you interested in?"*) and provide more detailed information about the system upon user request. Another specific mGDN used during the constraint specification phase of the dialogue is an mGDN for visualization and manipulation with the current set of constraints. This mGDN can be accessed by the user at any time, or the system decides to access it in case when conflicting constraints were provided, as defined by global dialogue strategies (section 3.5.2).

Yet another set of mGDNs has to be defined for the result-browsing phase of the dialogue. The relations between mGDNs are in this case more sequential, rather then hierarchical – once user provided a value for one mGDN, the system has to transit to the following mGDN. For instance, after the user has selected a restaurant from a list, the system should display the details of that restaurant. The transitions between mGDNs in the browsing phase are rather static, in contrast to the constraint specification phase of the dialogue, where the transitions between mGDNs are dynamic, based on already provided search constraints and on the shape of the current search space.

The purpose of explicit indication of relations between mGDNs is to enable the dialogue strategies to properly behave in following situations:

- Select a next mGDN to be set in focus (i.e. the next dialogue context) after a *value for the current mGDN has been provided*.
- Select a next mGDN to be set in focus after the mGDN under focus was *closed* by the user.
- Define a set of mGDNs *where the system can transit to* from a given mGDN on explicit user request.
- Define which mGDNs can be *directly accessed* from a given mGDN, i.e. which transition buttons the given mGDN has to display on the user screen.
- Define which mGDNs are *displayed simultaneously* together on user screen (this is in part defined by an application graphical layout, see section 3.5.3).
- Define which other mGDNs can process the current natural language input (i.e. degree of *mixed initiative*).

3.5.2 Global dialogue strategies

The global dialogue strategies are the dialogue management decision mechanisms about the next step in the dialogue operating *above* the mGDNs (as opposed to the local dialogue strategies that operate *within* each mGDN, see section 3.4.3). The global dialogue strategies were designed as *application independent*, however operating on per-application defined structure of mGDNs. As such, the strategies are not supposed to be modified by an application designer and we describe them in the following sections in order to fully explain the model of the dialogue system behavior. When the default strategies do not satisfy the needs of the application, they can also be modified. However, such step requires programming in Java and our experience shows that it is often difficult to foresee all possible dialogue situations and therefore to correctly predefine system behavior.

Because the model of the user behavior substantially depends on the presence or absence of the screen output, we have two different sets of global dialogue strategies for system behavior: one set for truly multimodal case when screen output is available and another set for vocal-only settings. We present here both sets of strategies. A detailed comparison of user behavior and corresponding dialogue strategies is given in section 3.8. A quantitative evaluation of the strategies is provided later in sections 6.3 and 6.4.

i. Dialogue strategies for multimodal systems equipped with screen output

Branching logic Branching logic is the elementary strategy for selecting a next mGDN after a value for present mGDN has been acquired, after the user closed the mGDN under focus or after the user explicitly changes the focus to another mGDN.

The branching logic is statically defined for mGDNs participating in the result browsing phase of the dialogue and follows the relations between mGDNs described by the task model. For mGDNs related to the constraint providing phase of the dialogue, the transitions between mGDNs follow the hierarchical structure. However, in order to speed-up the interaction, the strategy defines that the 'return link' does not necessarily go back to the direct parent of the current mGDN, but rather to the first (grand) parent mGDN that can offer more than one value (or complex value) compatible with the current search space (compatible values are provided by the domain model, see section 3.2). The return links are defined as a transition after a value for mGDN under focus was provided (and it is not a multi-value mGDN) or when the user decided to close it. The return link is also used in situations when user provided a value for another mGDN, but the current mGDN then has at most one possible value compatible with the current search space.

Concerning the transitions requested by the user (using explicit request for focus change), they are normally not denied, unless another strategy applies.

Presenting possible solutions (search results) The system automatically selects the mGDN for browsing results, as soon as the number of solutions compatible with the current search constraints is sufficiently low to be efficiently displayed. This action is executed immediately after the last needed search constraint has been provided by the user or the user closed a constraint mGDN. The action is not taken when the user makes an explicit focus change request.

The implementation of the condition "number of solutions is sufficiently low" depends on the way the solution targets are displayed (and is not necessarily related to the actual *number* of solutions). For instance, when the solution targets are paragraphs in a document collection, then the system can automatically display the document as soon as all compatible solutions (even a higher number of them) are only within one document. The document should then highlight all paragraphs compatible with the user's request.

Dialogue dead-end management The dead-end management deals with situations when a user request is over-constrained, i.e. when no solution targets are compatible with the current search constraints. In such cases, the dialogue management changes focus to the mGDN designed for manipulation with the current search criteria and asks the user to erase some search constraints. No transition to mGDNs for adding search constraints is allowed at this point and any such request results in changing the dialogue focus on the mGDN for manipulating with current search criteria.

Nevertheless, the user is still allowed to enter to the result-browsing phase of the dialogue on his/her explicit request and browse the search space. No results are then highlighted (like when no constraints are specified) and a vocal prompt announces the over-constrained situation. We allow the search space browsing in dead-end situations, as we assume the user might be frustrated in case of restricted system interaction.

Dialogue restart The dialogue restart in a system equipped with an interactive screen is never automated, but is always available on user request. A button with this choice is a part of common controls still visible on the screen. The restart of the system clears all the search constraints of current query and also resets any user-defined views on the search space into an initiatory state.

Indications of important interaction states This dialogue strategy is not intended for changes of focus, but rather for an indication or confirmation of miscellaneous interaction states.

- A set of *new search constraints* acquired by the system in the last dialogue turn is indicated visually by blinking of the constraints in the mGDN for search constraint manipulation (this mGDN is visible on the screen all the time). Visual indication is preferred over vocal indication as it is perceived less intrusive by users. The confirmation of newly added search constraints is particularly useful when the user provided them vocally and may have doubts about what the system understood.
- Normally, the system allows only one value per one search constraint type (e.g. `John` as `Name`). When the user later during the interaction specifies another value, the system *automatically replaces* the old value by the new one. This is indicated by a vocal prompt (*"Search constraint John was replaced by Jim"*) issued by the global dialogue manager. The vocal prompt indication was chosen because value replacement is difficult to indicate visually.
- When the search constraints become incompatible, the dead-end strategy is applied and the system changes the focus to the mGDN for manipulation with current search constraints. In addition to it, the mGDN indicates this situation by *red flashing of the constraints* and by a vocal prompt indicating this situation. The search space browsing mGDN also uses a warning prompt when user starts to browse the search space that contains no valid solutions.

ii. Dialogue strategies for systems with vocal-only output

The dialogue strategies in systems with voice-only output have to be more proactive, as the user cannot discover all the possibilities of the vocal interface as easily as in systems with an interactive screen. Additionally, the hierarchical structure of mGDNs cannot be presented as such to the user and therefore we suggest to use only a flat sequence of mGDNs for vocal-only systems.

The hierarchical nature of some attributes (e.g. a person being decomposed into first name, family name and age) can be transformed into a sequence (person, first name, family name, age). The branching dialogue strategy assures optimal progression in the dialogue, without asking the user for a value of attributes that are already uniquely determined by the current shape of the search space.

Branching logic The branching logic defines a next mGDN to be activated in cases when no other strategy applies. It selects the first constraint mGDN (in the order defined by task model) for which the user has not stated any preference yet and whose associated attribute in the domain model has more than one value compatible with the current solution set.

This approach ensures that the user is asked for stating preferences for each attribute in the sequence defined by the task model, but the dialogue is optimized by skipping attributes that do not offer multiple values compatible with the shape of the current solution set.

Dialogue dead-end management This strategy deals with cases where the goal of the dialogue cannot be reached (zero solutions). To cope with dead-end situations, we the following relaxation strategy:

1. Determine how many solutions are compatible with all the search constraints *but one*. If the obtained number of solutions is smaller or equal to a predefined threshold (called the dead-end management threshold), then provide all the relaxed solutions to user and ask him/her to select the desired one. Otherwise, determine a set of all "relaxation attributes", individually corresponding to a non-zero number of solutions when relaxed.
2. For each "relaxation attribute", ask user whether he/she is wiling to reconsider a value he/she previously provided. When he/she agrees, remove this value from set of constraints and continue the dialogue according to the standard activation rules (branching strategy).
3. If user rejects all relaxation possibilities, reset the dialogue.

Confirmation The confirmation strategy is a procedure used during the dialogue to obtain the user confirmation of the values that have been acquired by the system. There are two possible approaches:

- Explicit confirmation: the confirmation is simply obtained by explicitly asking the user.

- Implicit confirmation: the confirmation is induced from the reaction of the user to some confirmation information automatically associated with the next question.

Implicit confirmation usually leads to shorter dialogues and is often considered as more natural by the users. Explicit confirmation is useful in special cases, such as the invocation of irreversible actions or when the recognition confidence of the last user's utterance is low.

Dialogue termination The idea behind the dialogue termination strategy is that it might be more efficient, once a limited number of solutions has been reached during a dialogue, to simply display/utter the solution list and let the user choose the correct one, instead of trying to continue the dialogue to refine user's request in order to reduce the solution set to a unique one.

Incoherencies This strategy is necessary to deal with the situations where a user provides two different values for one search constraint type (attribute). In such case, the system has to ask the user for the preferred value once again.

If several simultaneous incoherencies occur, only one is processed and all other new values that lead to incoherencies are automatically removed (not considered by the system). The rule for choosing the particular incoherence pair to be processed is following:

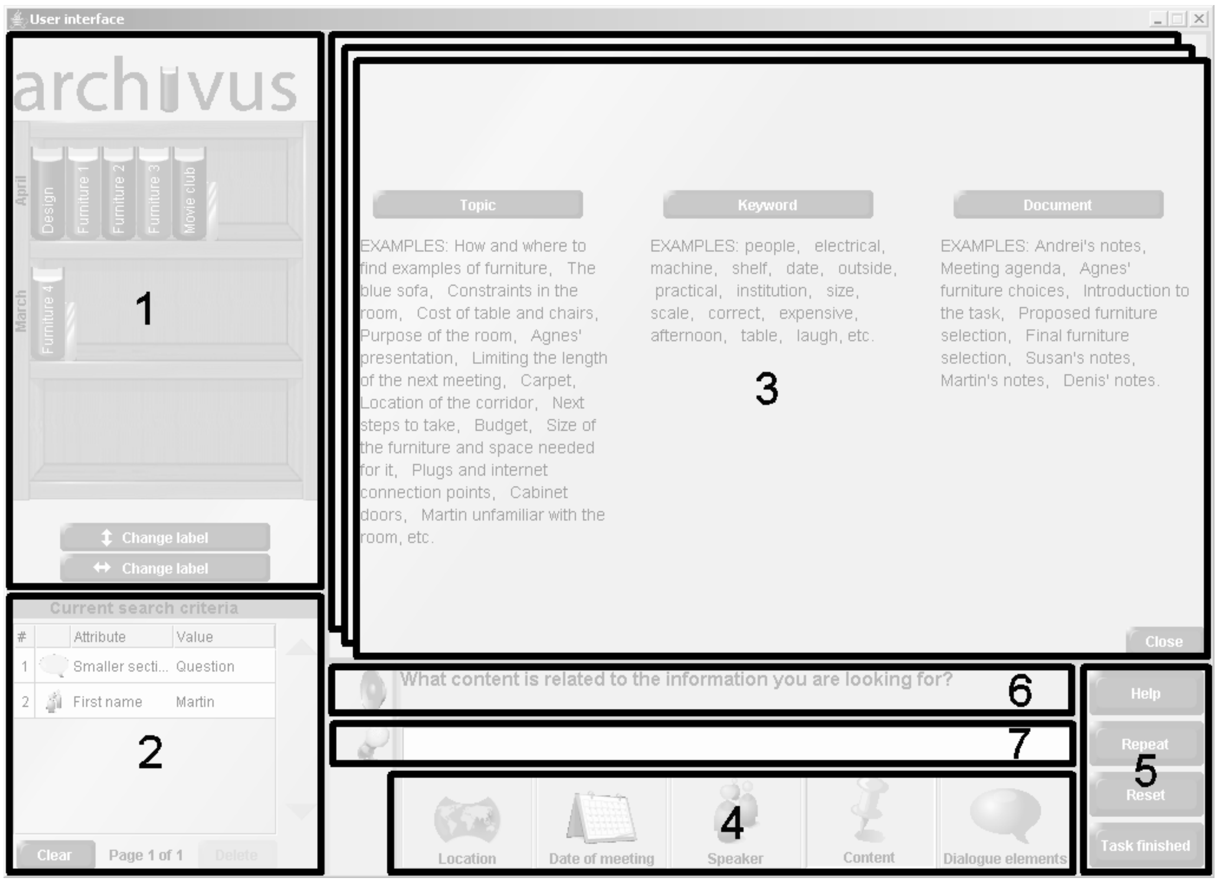
1. If the current mGDN is a source of the incoherence, then this incoherence should be processed.
2. Otherwise, the incoherence corresponding to the attribute associated with the mGDN coming first in the order defined in the domain model should be processed.

3.5.3 Graphical user interface (GUI) decomposition

The information seeking dialogue consists of two phases: (1) eliciting from the user the constraints that are needed to identify a small set of possible solutions and (2) giving the user the possibility to browse the current solution set in order to select the relevant solution. With respect to this dialogue structure, we propose the general GUI structure that is depicted in Figure 3.11.

The graphical interface of information seeking system, regardless the application domain, should include seven basic elements:

1. An area that *visualizes the solution space* and highlights the solutions that meet the current constraints defined by the user. The user should have the possibility to switch between various visualization modes, rearrange the solution space or browse the solution space. The search space overview mGDN controls the interaction within this area.
2. An area visualizing the *set of current search constrains*. Ideally, this area should be interactive, allowing users to easily browse and delete previously entered values of the current query. The area is controlled by the associated mGDN for current search constraints visualization and manipulation.



Current search criteria

#	Attribute	Value
1	Smaller secti...	Question
2	First name	Martin

EXAMPLES: How and where to find examples of furniture, The blue sofa, Constraints in the room, Cost of table and chairs, Purpose of the room, Agnes' presentation, Limiting the length of the next meeting, Carpet, Location of the corridor, Next steps to take, Budget, Size of the furniture and space needed for it, Plugs and internet connection points, Cabinet doors, Martin unfamiliar with the room, etc.

EXAMPLES: people, electrical, machine, shelf, date, outside, practical, institution, size, scale, correct, expensive, afternoon, table, laugh, etc.

EXAMPLES: Andrei's notes, Meeting agenda, Agnes' furniture choices, Introduction to the task, Proposed furniture selection, Final furniture selection, Susan's notes, Martin's notes, Denis' notes.

What content is related to the information you are looking for?

Figure 3.11: A general schema of graphical user interface for information seeking systems.

3. The *central zone* is an area used by all mGDNs, except the two described just above. However, in accordance with the mGDNs design principles (section 3.3.1), only the mGDN in focus is visualized here at one time. The functionality provided by this area is determined by the specific mGDN in focus.

4. A *selector* for explicitly switching between various constraint selection mGDNs. The content of the selector needs to be defined on a per-application basis and contains the uppermost mGDNs from the mGDN hierarchy (just below the root). The graphical component of the selected mGDN then appears in the central zone (3).

5. System *control options*, e.g. access to help, submission of a new query, or exiting the system altogether.

6. A display area for *system prompts.* Since the prompts are presented also in audio modality, the user can mute the system using the loudspeaker icon.

7. An area for *natural language input.* The text editing component visualizes what the user has typed and the microphone icon is used to control the speech input processing (i.e. muting the microphone).

While the graphical elements 1-3 are controlled directly by the associated mGDN, the elements 4-7 are shared by all mGDNs (actually, controlled via the IOmanager).

3.6 Software architecture

The previous sections were focused on the design of algorithms responsible for the interaction with the user. Only little has been said about *how such algorithms are integrated* within a software framework. This section describes the software architecture that we have designed in order to integrate all functionalities required by a multimodal dialogue system.

Multimodal dialogue systems are in general complex pieces of software composed of a large number of modules. Often, modules are created by various authors and were originally targeted for different kinds of applications. Each module requires different data sources for initialization, as well as information from different sources to produce its outputs. Moreover, the number and types of modules in multimodal dialogue systems are not fixed and there is no consensus on what exactly is the responsibility of every single module.

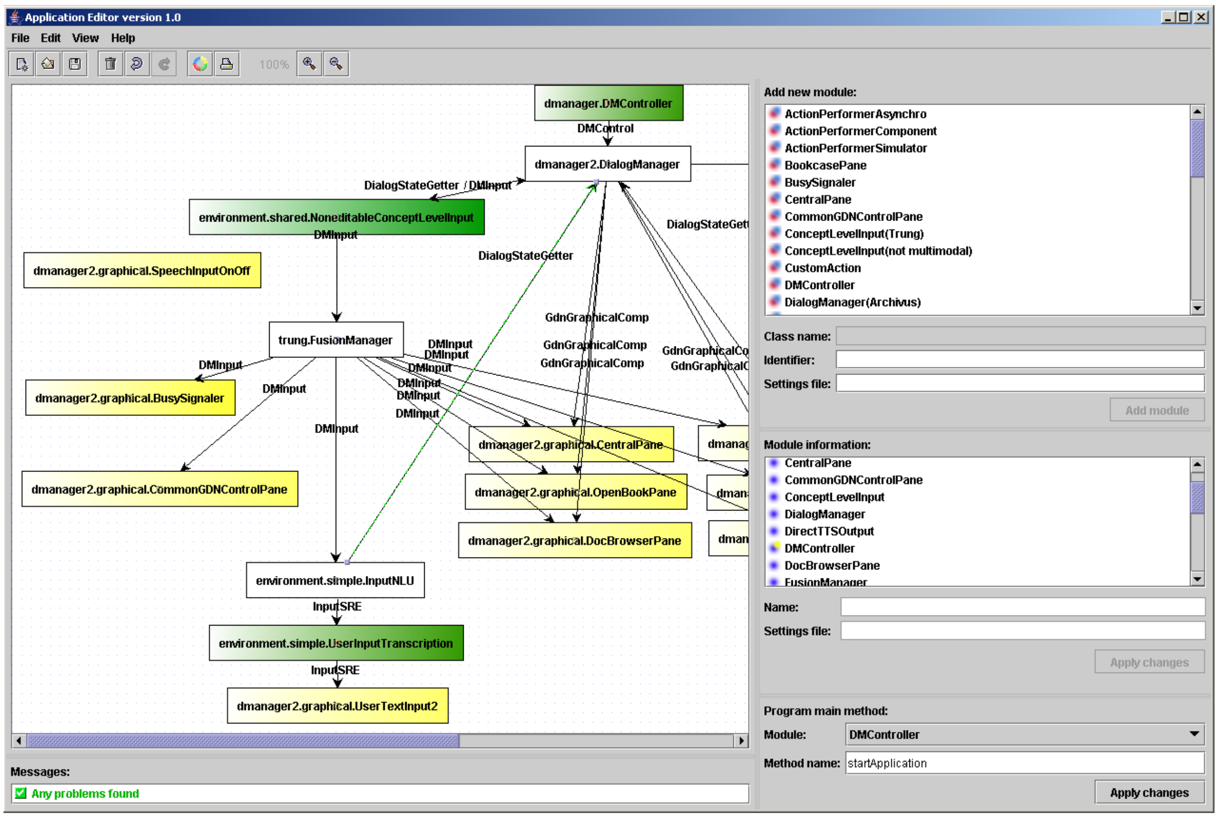
These factors show the need for a flexible architecture that allows for simple modification or addition of new modules. Ideally, the architecture is packaged with modules ready for immediate use (such as modules for speech recognition and synthesis or dialogue management) and with an environment supporting all phases of the development cycle. This speeds up the development process and allows researchers to focus on the functionalities of the system rather than the implementational details of the various components. Existing frameworks that might satisfy these criteria, such as Galaxy-II [104] and the Open Agent Architecture [63] are, in our view, too general and do not impose a predefined communication paradigm on the dialogue system. Moreover, the distributed nature of these systems makes debugging difficult.

For all these reasons, we have chosen another approach to module composition. Each module is implemented as a Java class and the communication with other modules is performed through calling their methods in well-defined interfaces. The interconnection of the system modules is defined in the *application configuration file*. When the system is launched, a special module called the *application loader* creates instances of all the modules, interconnects them as specified in the configuration file and initializes them. Since the initialization of each module can be unique (as far as the resources and start-up parameters are concerned), there is an independent configuration file associated with each module which is used only by that module for its own initialization.

The system contains two types of modules: *graphical* and *functional-only*. Graphical modules are extended version of functional-only modules and theirs graphical component is laid-down during the initialization process by the application loader according to the *application layout* configuration file. The application layout file defines all graphical windows of the application and each graphical module is located at one window at a specified position and with a specified size.

In order to facilitate creating new systems, we have implemented a tool (see Figure 3.12, [100]) that provides a GUI for the visual composition of modules. The tool allows for a *functional composition* of modules and stores it to (or loads it from) the application configuration file. We plan to extend the tool by implementing the option of visual definition of *layout* of graphical components.

Our approach provides a highly flexible and easy-to-configure system, allowing for simple module development and debugging. At the same time, the possibility of some distributed



Messages: Any problems found

Module: DMController

Method name: startApplication

Figure 3.12: Application Editor: enables to build a new application from existing modules and define module interconnections

processing is still fully open. On the local system, any functional module can be replaced by a proxy that forwards the method calls to another computer (where the real operation take place) and then receives results. From the point of view of the rest of the system, this process remains fully transparent.

The most complex situation is when a graphical module has to be displayed on a remote machine(s). Our solution to this problem relies on the standard VNC protocol[1]. We have implemented a *VNC virtual window container* for graphical modules. The virtual window does not appear on the local machine, but runs a VNC server and the content of the window can be displayed using a VNC client on any remote machine. This solution makes the distribution of an application extremely simple – an application is developed and tested locally, and the local windows are replaced by VNC windows only in the final phase of the development. This approach also allows having multiple VNC virtual window containers – each of them just has to run one VNC server with a unique port number. The disadvantage could be higher network bandwidth consumption compared to the truly distributed case.

Since the Wizard of Oz simulations are an important part of our methodology (Chapter 4), we have to be able to *supervise the functionalities* of certain modules or even *substitute them* by a Wizard (human operator) using a special Wizard's control GUIs during the dialogue. This can be easily achieved by inserting a graphical WOz module as a proxy of the module that we want to supervise. The WOz module can be located on a VNC virtual window, making it possible to supervise the system from a remote machine. An

[1]VNC systems use the RFB protocol, `http://en.wikipedia.org/wiki/RFB_protocol`

example: if the goal is to supervise the quality of the speech recognition, the graphical module is plugged in between the SRE and the NLU. Then, the recognized utterance is displayed to the Wizard who is able to modify it, if necessary. Similarly, the Wizard can check the semantic pairs resulting from the NLU module.

In the IM2 project (that our work was part of), yet another software framework for module composition was developed. The framework is called JFerret[2] and is based on plugins that can be dynamically added to or removed from an application. The main difference compared to our framework is that JFerret uses message broadcasting to communicate between modules (in contrast to direct method invocation used in our systems). Such an approach is useful in cases when many modules produce or consume only few types of messages. This is however not our case – a multimodal dialogue system has usually a one-to-one connection between modules and the communication is performed only between those modules. This implies that each connection of two modules in JFerret would have to use a unique communication channel, making the system configuration more cumbersome than in our approach.

The next section describes a generic schema of multimodal dialogue systems. Mostly, it can be used directly without any modifications, for instance when the aim is to quickly study user behavior with the new type of system. Nevertheless, our flexible software architecture enables the option of replacing some of the *modules* with newly created ones, for instance when the impact of performance of different speech recognizers on the user communication model is studied within the context of a whole application. Alternatively, some *parts* of the whole framework can be replaced. For instance all modules for multimodal fusion or fission can be replaced by more efficient and powerful schemes.

3.6.1 A generic schema of a multimodal dialogue system

Although our software architecture provides means for flexible interconnection of modules, we have observed that the main schema of multimodal dialogue system architecture tends to have the same skeleton. The schema that we propose is depicted in Figure 3.13.

The *Dialogue manager* controls two groups of modules – the input and output modules. The role of the *Fusion manager* is to combine the semantic pairs from the different input sources (our existing fusion algorithm is described in section 3.4.2). The *Text input field* module (area 7 in Figure 3.11) allows the user to type-in some text, that is consequently translated by the *Natural language understanding* (*NLU*) into semantic pairs. Similarly, the text produced by the *Automated speech recognition* (*ASR*) is translated by the *NLU* into semantic pairs. Note that the *ASR* might be disabled using the *Mute microphone* GUI module and the *ASR* result might be corrected by Wizard's *Recognition supervision* module. The user GUI is decomposed into *mGDN graphical component(s)* (areas 1, 2 and 3 in Figure 3.11) and into other static control elements (help and repeat buttons, buttons for selecting a new mGDN in focus, indication of the system being busy by processing of the users input, i.e. areas 4 and 5 in Figure 3.11). Pointing on the graphical components results produces semantic pairs, which are in turn sent to the *Fusion manager*.

All the semantic pairs resulting from the fusion process are supervised by the Wizard in the *Semantic pairs supervision* module and then are sent to the *Dialogue manager*.

[2]http://www.idiap.ch/mmm/tools/jferret/jferret_home

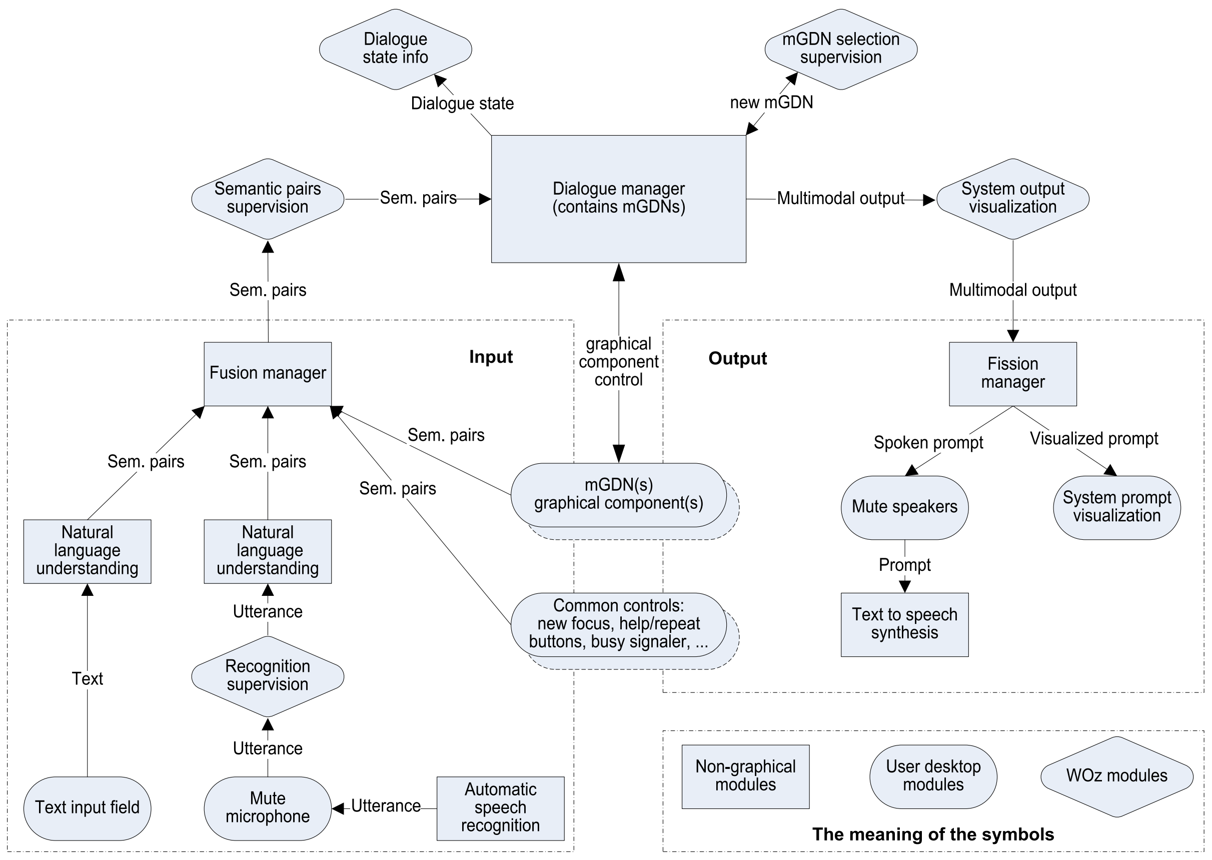


Figure 3.13: Proposed module composition for the multimodal dialogue system and the main data flows. Every module has also an access to information about a state of dialogue manager, but this is not indicated in the figure due to readability reasons.

The *Dialogue manager* processes the semantic pairs and selects the next mGDN under focus (the decision can be modified by the Wizard in the *mGDN selection supervision*). The dialogue state information is then updated, the mGDN updates the state of its graphical component and the multimodal output is issued. The output is sent by the *Fission manager* to the *System prompt visualization* module (area 6 in Figure 3.11) that displays the prompt on the screen and sends it to the *Text to speech synthesis* module which gives a vocal feedback to the user.

Each of the modules in the system can be sensitive to the global state of the dialogue (e.g. the mGDN in focus, the list of active mGDNs) and can adapt its behavior accordingly (e.g. using an appropriate mGDN-specific grammar for speech recognition). The information about the dialogue state can be obtained by accessing the information published by the dialogue manager.

3.7 Application examples

3.7.1 RestInfo

A system for providing information about restaurants in the city of Martigny, Switzerland. The user can search for restaurant using any of the following search criteria: location, type of food, price level and opening hours. A large part this dialogue model was designed during the InfoVox[3] [114, 94, 95] project.

3.7.2 SmartHome

The SmartHome system is an interactive dialogue system for wireless command and control of home appliances. The system enables the user to control the following activities: to turn lamps on or off and dimming them, putting the blinds up or down, turning the fan on and off, manipulating messages on the answering machine, switching on and off the TV and selecting different channels, consulting an electronic programming guide (EPG) as presented on the television screen and recording a movie, or setting a reminder for the start of a movie.

Obviously, the SmartHome is not directly an information seeking system, but rather a command performing system. Therefore, we had to express the commands in the system as information targets described by elementary attributes of the commands (device, location, action, day, time, movie title). The goal of the dialogue communication is actually to identify the target (command), which is functionally equivalent to searching for it. Only a partial modification of dialogue manager was required – as soon as the information target (i.e. a system command) was identified, it was sent to the smart home environment for execution rather than being presented to the user.

The system exists in German and English language versions. The German version was intensively tested by test subjects in a real home environment, in a smart home lab of Philips campus in Eindhoven, Netherlands. The work was carried out in the framework of the Inspire[4] project [71, 49, 10].

3.7.3 Archivus

The Archivus system, described in detail in Chapter 5, is a multimodal interface (mouse, pen, touchscreen, keyboard and voice) that allows users to access a multimedia database of recorded and annotated multimodal meetings. Specifically, the database contains the original video and audio from the meetings (recorded in special meeting SmartRooms), electronic copies of all documents used or referred to in the meetings as well as handwritten notes made by participants during the meeting, and a text transcript of the meeting itself. In order to facilitate retrieval of information, selected annotations have also been made on the data, specifying elements such as dialogue acts, argumentative structure and references to documents, as well as the date and location of the meetings and information about the meeting participants [56, 60]. The Archivus system and the ISPM were developed within the framework of IM2 project[5].

[3]InfoVox: Interactive Voice Servers for Advanced Computer Telephony Applications, funded by Swiss Commission for Technology and Innovation, CTI grant 4247.1.

[4]Inspire: INfotainment management with SPeech Interaction via REmote microphones and telephone interfaces, European grant IST-2001-32746.

[5]IM2: Interactive Multimodal Information Management, `http://www.im2.ch`, funded by the Swiss National Science Foundation.

3.7.4 Other applications

Using our methodology, several other applications have been implemented. For example, an electronic catalog of cars allows searching a database of second-hand cars for sale. Another application allows searching for apartments available for rent. We have also studied a possibility of designing a system for searching books in a library and a flight reservation system.

3.8 From vocal-only to screen-equipped multimodal dialogue systems

Thinking of a multimodal dialogue as an extended voice-only dialogue is a simplified view. Since we have an experience with both vocal [66, 13, 10] and screen-equipped multimodal dialogue systems [18, 68], we decided to highlight the observed differences in *user behavior* when interacting with such systems [67]. The differences in user behavior are the main motivations for using two different sets of global dialogue strategies in the ISPM (see section 3.5.2).

3.8.1 Towards user-driven dialogue strategies

In systems that only permit vocal input, the user typically expresses some initial wishes at the very beginning of the interaction and the system then progressively asks for missing information, guiding the user towards the goal of the interaction (i.e. to find objects in the database satisfying the user's needs). It is important to guide the users in vocal-only systems, as they may not know what piece of information they need to supply to the system in order to limit reasonably the number of objects satisfying their needs. Each time the user provides some new information, the vocal dialogue manager analyses the current solution set and selects the next most appropriate piece of information the user should be asked for. This is done by changing the focus of the interaction to the GDN associated with a slot that is associated with this piece of information.

However, in multimodal systems equipped with a screen, users have visual feedback on the current context of the interaction, for example, they can quickly *see* what type of information the system requires, and they may have an instant global view of the current solution space as well as of the constraints that have been provided to the system so far. In such situations, users better understand the current context of the interaction and it is therefore reasonable to assume that they will prefer to participate more actively in the interaction.

The system GUI gives the user a natural possibility to explicitly change the focus (i.e. to select the next mGDN) by clicking, and therefore the multimodal dialogue manager must be able to handle this change. This is an extension to the original vocal dialogue management strategies (see page 47), where such a feature seems to be very unnatural as it would correspond to user utterances like *"Now I want to specify the speakers of the meeting that I am looking for"* or *"Ask me for the names of speakers in the meeting"*. The explicit focus change feature is not needed in vocal dialogue systems with mixed initiative since speech can express directly value(s) different from those expected in the current dialogue step. For example, if the system is asking *"What was the location of the meeting?"*, the user may vocally answer *"It was in Lausanne and John was speaking"*, while this is not possible (without explicit focus change) by mouse pointing when only a list of locations is displayed on the screen.

Our other finding is that the dialogue strategies for screen-based multimodal systems need to be less active. Specifically, in case of the Archivus system (see Chapter 5), the initial vocal strategy for selecting the next GDN in focus (strategy called branching logic) was negatively perceived by users. This might have two possible reasons: (1) the graphical interface allows users to clearly see the hierarchical structure of the task, and the user is confused when the system automatically selects an mGDN in another part of this hierarchy (the behavior of the system does not seem to be consistent and stable), and (2) the task that is solved using the Archivus system is more complex (larger number of slots) than tasks that can be solved by voice only systems. This makes the automated selection of the next mGDN (based on the current solution set) partly inappropriate from user's perspective, as the system may ask for information that the user does not know or has no preferences for.

Therefore, we decided to make the branching dialogue management strategy more passive: after a value for the current mGDN is acquired (or the user indicates that he/she does not want to provide a value by selecting *Close*), the focus is automatically changed, but only to the closest ancestor of the current mGDN in the mGDN hierarchy such that the ancestor covers at least one slot whose value needs to be provided by the user. The hierarchical structure of the task is actually presented to the user as a sort of hierarchical menu. From the perspective of the user, our updated dialogue strategy therefore corresponds to going back to one of the upper menu items in the menu hierarchy.

Another observed difference between voice-only and screen equipped systems concerns solving of over-constrained situations (the dead-end strategy). Since the systems equipped with screen allow the user to directly see and modify the conflicting constraints, the user does not need any additional mechanism to resolve dead-end situations. This is in contrast to vocal-only systems, where a relatively complex procedure is needed to inform the user about the existing problem and propose constraints that can lead to problem resolution.

3.8.2 New role of system prompts

When extending the unimodal vocal-only dialogue prototyping methodology to the multimodal systems, the design of system prompts has to be addressed differently. In a voice-only system, prompts are the only means of conveying information to the user. However, in the case of a screen-equipped multimodal system, users are no longer interacting with solely one mGDN in focus. Instead, they can see several other active mGDNs displayed on the screen at the same time and each of these mGDNs can contribute with some useful information. In addition to the GUI for entering a value for the slot associated with the mGDN in focus, the user can also easily see the information in other mGDNs; for instance the already submitted search constraints, solutions satisfying these constraints, and the signalization of an over-constrained situation. This gives the user a better idea of the current context of the conversation compared to voice-only systems.

Since the original role of the prompts, i.e. request for user's input, is often taken over by the graphical representation, the vocal prompts can be either omitted entirely in some situations or used to provide the user with context-dependent suggestions and advice. For those reasons, we make the distinction between "information requesting" prompts and "advising" prompts. For the system developer, it is not easy to identify the context-dependent advising prompts and integrate them into the multimodal dialogue model, compared to the requesting prompts in unimodal models. Typically, an advising prompt should be used if the user ignores some important signal given by the graphical interface. For example, in Archivus, meetings relevant to the user's current request are displayed as light green books in the bookshelf, while the irrelevant books are dark green. When the user tries to open a dark green book, an advising prompt should be played informing the user that this meeting does not contain any relevant information.

An advising prompt can be also useful in situations when the default prompt chosen by the system is redundant. For example, if the user has interacted with the system for a while, he/she does not need to hear the information provided by the default prompt again. Instead, an advising prompt can point out an interesting aspect of the system, teaching the user to use the system more effectively. If there is no suitable advising prompt available, the vocal part of the prompt can be entirely omitted.

Since advising prompts should be played only when certain circumstances occur, they are ideally implemented with conditional prompts (i.e. prompts associated with a logical expression and used by system only when the associated expression is evaluated as true in the context of current interaction). In order to discover the necessary advising prompts relevant for a given multimodal dialogue system, a system evaluation using Wizard of Oz studies is crucial (see Chapter 4).

3.9 Summary

Building an initial system prototype is an important part of any software development process. The complexity of this task is increased when novel types of interactions between the user and the system are foreseen, such as dialogue interactions using multiple input and output modalities.

In our work, we propose a methodology that allows for the design and implementation of multimodal dialogue systems for the information seeking domain. The main characteristics of this methodology are the following:

- the methodology is *generic*, i.e. it is applicable to a wide range of information seeking systems.
- the methodology allows for the *rapid* implementation of an initial prototype by providing guidelines for the design process and ready-to-use software components. In addition, the methodology does not require the whole system to be fully implemented, as it provides integrated support for Wizard of Oz simulations (see Chapter 4) helping the designer to avoid the time consuming implementation of system features and functions, which may finally turn out to be unused by the end-users.
- the developed prototype is *fully multimodal and dialogue driven.* This is guaranteed by allowing the designer to compose the targeted interface from predefined building

blocks (mGDNs) that individually provide appropriate graphical outputs together with consistent access to all chosen modalities. Modality fusion is performed in a specific module external to the mGDNs, but taking into account the mGDN in focus.

A system prototype designed with our methodology can be immediately tested with users and thus reveal end-user behavior and modality use models. As argued in [93, 119, 22], involving users in an early development phase increases the chances that the final system is well accepted by them.

As far as dialogue design is concerned, we propose a *two-layered dialogue model* that corresponds to an extension of the standard frame-based approach. In our approach, the first dialogue layer (modeled with generic dialogue nodes – GDNs) defines the interaction within specific dialogue contexts and based on local dialogue strategies associated with the individual slots of a global dialogue frame. The second layer controls the overall dialogue progress and relies on global dialogue strategies that provide a higher-level planning of the dialogue. One of the important findings of our research is that such a two-layered dialogue model is *easily extendable to multimodal systems.*

Concerning the dialogue system software architecture, we propose a variant of a plugin approach. The resulting plugin system is, in general, intended to run as a single process on one machine, thus making debugging and development simple. However, if needed, the system's graphical interfaces can be easily distributed over several machines using the VNC protocol and a standard VNC client.

Chapter 4

Experimenting with Wizard of Oz simulations

Wizard of Oz (WOz) simulation is a technique commonly used to design and evaluate dialogue systems. The main principle of WOz experiments is the *simulation of the targeted system* (or some of its parts) by a hidden human operator called a *wizard*. The purpose is to observe users interacting with the system before the system is actually fully implemented. Since, from the user perspective, the system seems to be fully operational (the users are not aware of wizard's interventions), WOz simulations allow to gather realistic interaction data and to take advantage of it at any stage of the system design.

In particular, the availability of user interaction data allows for more precise identification of user's needs and actions (hereafter called the *user model*) while performing specific tasks. The obtained user model then serves for the optimal implementation of the various *system functions* that are necessary to satisfy users' needs and to respond to users' actions. The targeted system functions can be categorized into two main groups: *domain related functions* (i.e. functions related to the content of the domain database and to the domain attributes used to search for information) and *dialogue related functions* (i.e. related to the various dialogue situations that need to be resolved – clarification, reformulation, etc).

Wizard of Oz simulations are traditionally used for the design of natural language systems [29, 27, 34, 20, 62] and are generally considered as a suitable approach for the identification of sound design solutions [102], as they allow to get valuable information about how users naturally describe their needs and how they proceed in the dialogue.

We have extended the standard WOz approach for the design of multimodal systems created with the ISPM (Chapter 3). Although several attempts of extending WOz simulations to multimodal systems have already been published [101, 45, 112], we feel that a detailed description of this method (especially in the multimodal case) is still missing. We therefore first review the WOz methodology and development cycle used for natural language dialogue systems (section 4.1). Then, we explain the impact of a multimodal environment on the initial and advanced phases of multimodal system design (sections 4.2 and 4.3 respectively). The section 4.4 provides details about the hardware configuration of the WOz environment and section 4.5 describes the graphical interfaces that our wizards used to control the system. We evaluate the wizard's performance when using the provided interfaces in section 4.6, and practical guidelines for carrying out WOz studies are given in section 4.7.

4.1 Free and Constrained WOz simulations for voice-only systems

Having voice as the only interaction channel allows for complete simulation of the whole system by wizard – we call such experiments *Free WOz simulations.* The advantage of Free WOz simulations is a quick setup (no implementation of any specific system needed, the user and wizard are usually connected only over telephone line, sometimes wizard uses a text-to-speech synthesizer for his outputs). In addition, any interaction problems are easily identified and naturally solved as in human-to-human interaction, because the wizard's decisions are not limited by existing automated system components. On the other hand, the wizard's behavior should still be consistent with the capabilities of targeted system and with the performance of existing NL technologies, because otherwise the user may suspect the wizard's presence. This would invalidate the experimental results, because human-to-human interactions are known to be different from man-machine interactions [41, 32, 26, 107]. Moreover, the observed interactions become useless for system design when the complexity of observed interactions is incompatible with existing technologies. The Free WOz simulations are typically used at the very *first stage* of vocal system development for *user requirement elicitation.*

Once the essential system functions have been identified, most of them can be directly implemented. Those typically include the domain knowledge database, the set of interaction contexts (i.e. mGDNs in systems designed using ISPM) and the corresponding set of predefined system prompts or prompt patterns. At this stage of the development, the design decisions can be *validated* by means of *Constrained WOz simulations.* The main difference (compared to Free WOz simulations) is that the wizard simulates specific, well-defined functionalities of the system as opposed to the system per se. In this way, it is possible to observe users' reactions to the system before its full and expensive implementation is made. However, the overall appearance and performance of such a system is *from the user perspective close to the final product* and the gathered interaction data are far more realistic compared to the Free WOz simulation. On the other hand, not all interaction problems can be resolved by the wizard, since the wizard cannot add instantly new functionalities to the system when need for them arises. The aim of Constrained WOz simulations is to prove that there are *no major interaction problems* unhandled by the existing or envisioned system functions and that the *already implemented functions work adequately.* The Constrained WOz simulations also allow gathering *detailed specifications* for the implementation of functions currently simulated by the wizard.

4.1.1 Development cycle

The development cycle for natural language interfaces starts with gathering user requirements using the Free WOz simulations. It allows identifying all essential system functions and building the very first system prototype from those functions. The functions that do not require too much development time are directly implemented, while functions difficult to implement or those requiring training data for correct implementation are simulated by the wizard during the Constrained WOz simulation.

Outcomes of Constrained WOz simulations are an evaluation of the quality of implemented functions and an identification of potential problems unhandled by the system.

The designer then *iteratively* (with the help of Constrained WOz simulations) refines the problematical functions and progressively *automates* the prototype, finally reaching a fully automated system.

While the early WOz studies can be carried out with any user (co-workers, colleagues, friends), at a certain point it is important to start using naïve test subjects. The term 'naïve test subject' refers to persons who have no previous experience with the system and are not aware of the wizard simulating the system. Only such users can give a realistic view on the complexity of the interaction and naturally discover problems in the existing prototype. In contrast, users who know the prototype can be more useful during the initial stages, when the system needs to be debugged and the whole experimental procedure tested.

Validating all system design decisions by WOz simulations with naïve users enables to get an early feedback on the decisions and thereby increases chances that the final automated system is better accepted by users [93, 119, 22].

4.2 User requirements elicitation in a multimodal environment

The previous section described the development cycle of vocal dialogue systems using WOz simulations. Our goal is to adapt the WOz technique for the design and development of multimodal dialogue systems.

However, the Free WOz simulations of multimodal systems equipped with screen output are normally not that easy as simulations of voice-only systems. The main problem is that in the situation when the model of user requests is unknown, it is impossible to predict and predefine all necessary *graphical* system outputs. Online synthesis of graphical outputs involving synchronized text, graphics and video is too much time consuming and therefore beyond the wizard's capabilities. This task is apparently much harder than online generation of system responses in natural language, where the wizard typically writes the response on keyboard and the response is then vocalized using a text-to-speech synthesizer.

The initial phase of multimodal dialogue system development (user requirement elicitation) cannot usually be done simply by means of Free WOz simulations. Instead of it, the user requirements and necessary system functions are identified by other techniques. One of the employed techniques is *scenario-based design* [98], based on informal narrative description of user activity and experience when performing a task. The scenario, created together with the potential user, describes what a user would have to 'do and see' step-by-step in performing a task using a given system. Another technique relies on creating *low-fidelity prototypes* of the system [19] in the form of paper mockups of the user interface (rough sketches of the main interfaces and dialogs). These two techniques can be combined in the form of electronic mockupsor storyboards – a set of system screens interconnected to represent an interaction scenario, prepared usually in presentation programs like MS PowerPoint, Macromedia Flash, etc.

The goal of all abovementioned techniques is to brainstorm and get first user feedback. In contrast to Free WOz simulations, the system is never presented to the user as a working prototype, but always as a design example. The low-fidelity prototypes are advantageous because it is easy and cheap to change them and because they maximize the number of interactions before committing to certain functions and coding.

4.3 Constrained WOz simulations in a multimodal environment

Once the user requirements were collected and the necessary system functionalities were identified, the system can be at least partially implemented. We have adapted the Constrained WOz technique for multimodal systems developed using the Interactive System Prototyping Methodology (ISPM, see Chapter 3). Our methodology directly supports simulations of missing system components by providing a Wizard's Control Interface (WCI). The resulting system can be considered as a high-fidelity prototype, because it looks and reacts as the targeted system, allowing for a detailed examination of all usability and interactions issues. This is important especially in later stages of system development, because (as we believe) the final quality of the multimodal system depends on a lot of fine-tuning.

In order to highlight the differences between the design and evaluation of voice-only systems versus multimodal systems, we will first describe the particularities of WOz simulations for voice-only interaction in section 4.3.1, since vocal dialogue systems can be seen as a special case of multimodal dialogue systems. In section 4.3.2, we explain the specificities of WOz simulations in multimodal environments.

4.3.1 Particularities of WOz simulations with vocal dialogue systems

In earlier work we developed a methodology for rapid design of vocal dialogue systems with mixed initiative [13]. Using this methodology, the designer of the interactive vocal system only needs to provide a domain model of the application and the application-specific language resources (grammars, prompts). Our Java implementation accepts those resources and makes the dialogue system operational; it also automatically provides a graphical interface for the wizard which can control or simulate some system functionalities.

In our experience, the modules usually simulated or supervised are the speech recognition engine, the natural language understanding module, and sometimes the decisions of the dialogue manager. WOz experiments allow for the simulation of full grammar coverage (with an excellent speech recognition rate) and can also be used to find the minimal speech recognition performance necessary for smooth interaction flow [49].

We have found that WOz simulations in voice-only settings are relatively uncomplicated from the technical point of view. This is due to the fact that the dialogue manager and the wizard's interface are running as one application on one single computer (fully under the wizard's control) and only audio signals need to be transmitted to the user (test subject), making the hardware and software setup relatively simple.

The cognitive load of the wizard in vocal dialogue systems is lower in comparison to WOz simulations with a multimodal dialogue system, because the wizard chooses his actions based mainly on listening to the ongoing dialogue, i.e. auditory input. In multimodal systems equipped with a graphical user interface, the wizard must also control and be aware of what is happening on the screen, which increases his load.

We also found that users tolerate the wizard's reaction time (which forms a large part of the overall system response time) if it is within a few (approx. 5) seconds, since they are in general not yet familiar with speech interfaces and seem to understand that processing of speech takes time. However, users will not accept slow response times with GUI-equipped systems, since they are accustomed to fast reaction times with such systems.

4.3.2 Extending the WOz methodology for multimodal systems

When designing a dialogue based multimodal system, we must look at four elements simultaneously – the dialogue management, the language model, the use of all interaction modalities and the graphical interface – each directly influenced by user needs. These elements also influence one another, making their decoupling during the design and evaluation process impossible. For example, the sizes of the graphical elements might depend on whether touch is used as an input modality, the choice of graphical elements depends on the dialogue management elements which in turn constrain the types of linguistic interactions possible. These in turn play a more general role in influencing which modalities a user will choose for a specific task (see Figure 4.1 for the types of influences and the scope of the traditional Wizard of Oz vs. the proposed extended methodology).

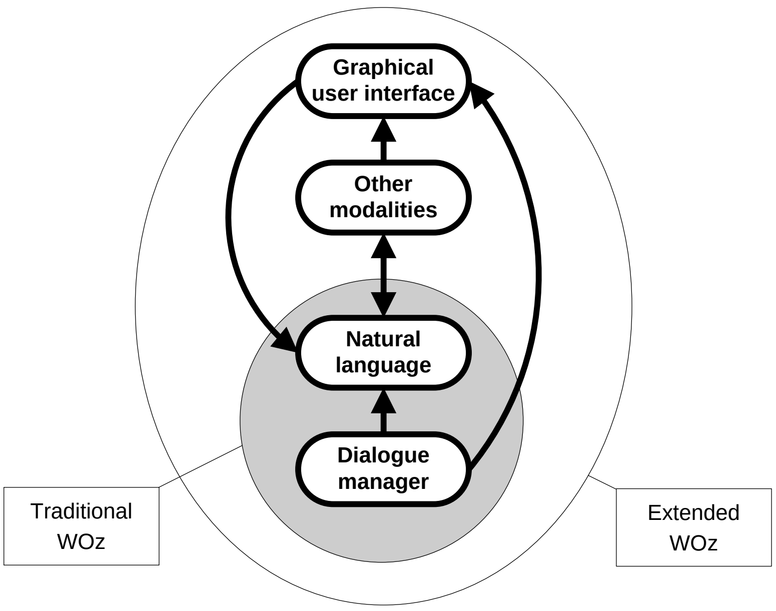


Figure 4.1: Influences of elements and scope of the WOz methodologies.

In terms of the wizard's simulation tasks there are several differences between controlling a unimodal language interface and a multimodal system. First, the wizard has to manually interpret or confirm generated interpretations from inputs in all modalities (for instance pointing, speaking and typing). To do it in a uniform way, the wizard needs to have the same interpretation formalism for all modalities. Moreover, the degree of automation of modules for the different modalities changes during system development, since modules for some modalities are easier to automate than others. For instance, mouse pointing is easy to automate because the interpretation of a click is unambiguous. On the other hand, interpretation of speech requires a sufficient-quality speech recognizer and appropriate NLU algorithms. This means that the wizard will produce the interpretations for this modality manually in the early phases of system development. The same is true for keyboard input, a modality not being used at all in voice-only systems. Consequently, multimodal systems allowing for both voice and keyboard input automatically require extra development steps.

In addition to higher cognitive load of the wizard when simulating the missing parts of a multimodal system, his reactions must be fast, precise and consistent in recurring situations. In order to decrease cognitive load, Salber and Coutaz [102] suggested to use multiple wizards, each specialized on specific tasks.

Finally, the wizard's reaction time should be balanced for all modalities, as a long reaction time of one modality may possibly influence a user natural modality selection during the experiments. The problem arises when both pointing and language are represented in the system. Pointing can be processed automatically and is therefore very fast, whereas language input requires manual interpretation by the wizard, which naturally takes more time. To balance the time difference, either the speed of the pointing modality can be degraded (although this is not advisable since most users expect pointing to be fast and may react negatively to the degradation), or the wizard has to be very fast when interpreting the language input. The latter requires that the interface for controlling the system by wizard has to allow for very efficient interpretation techniques.

4.4 Physical settings and hardware configuration

In this section, we describe the physical settings and hardware configuration of Constrained Wizard of Oz simulations that we have used to design and evaluate a multimodal system. Since the settings are generic enough to be used in any experiments with a multimodal system and because the literature on the topic is missing, the description is sufficiently detailed to allow the reader to reconstruct the experimental environment.

In our Wizard of Oz environment the evaluator of the interface, *the user*, sits at a standard desktop PC with a 15 inch touchscreen and a wireless mouse and keyboard (Figure 4.2). Users are also given a small lapel microphone. The user can therefore provide an input in *spoken* or *written* natural language, using mouse *clicks* or directly *point* on elements on the screen. The system gives *graphical* and *textual* feedback using the screen and *audio* feedback (natural language prompts) using loudspeakers.

The user's actions are recorded by two cameras situated on tripods. One camera faces the user and records his/her facial expressions as he/she interact with the system. The second camera is positioned to the side and slightly behind the user in order to record the modalities he/she is using. The computer desktop screen (multimodal graphical interface) and audio from lapel microphone are recorded as well, giving the experimenters a total of *three views* to work with during the analysis of multimodal behavior of the user.

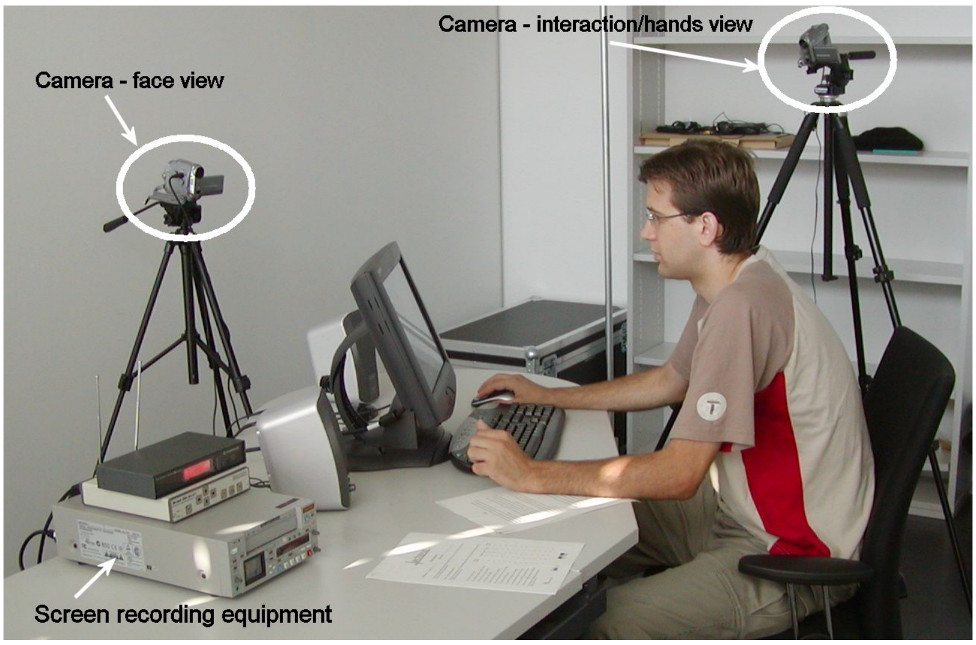


Figure 4.2: View of the user's work environment.

The wizard sits in a separate room since the user must be given the impression of interacting with a fully automated system. The wizard's room (Figure 4.3) is equipped with two observing monitors providing a view on the user's desktop and the user's face. The audio from the user room is also transferred to the wizard's room. Such a setup gives a sufficient overview of the situation in the user room: the wizard sees what the user is doing with the interface and the user's reactions to the interface, including facial expressions allowing detecting when the user is confused or does not know what to do. Since the user is normally left by the experimenter alone in the room (in order to eliminate a possible influence from the experimenter's presence or feeling of pressure), the facial camera is additionally useful in unexpected situations that could influence an ongoing experiment, for

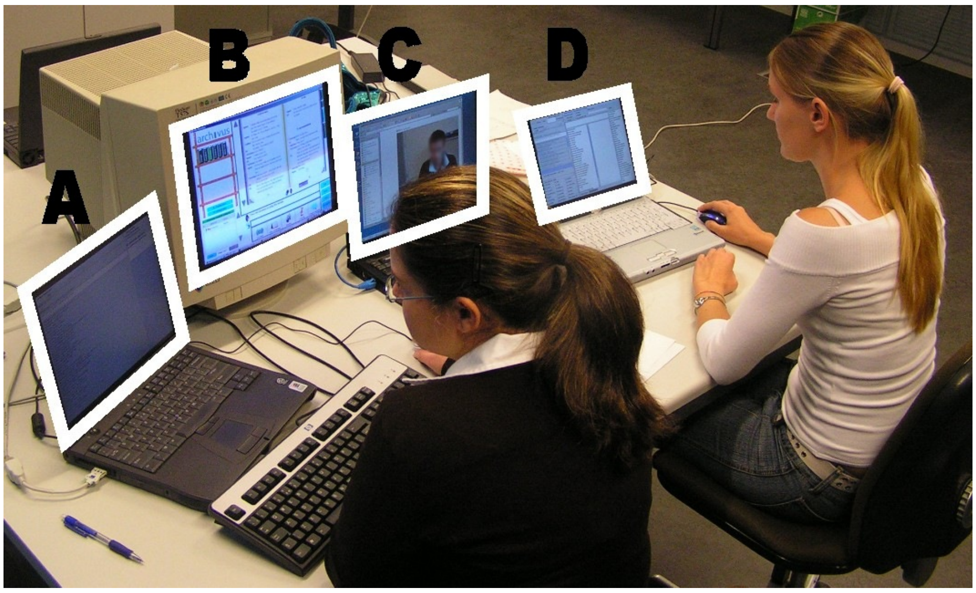


Figure 4.3: View of the wizard's environment. (A) Output Wizard's Control Interface, (B) Mirror of the user's desktop, (C) User's face, (D) Input Wizard's Control Interface.

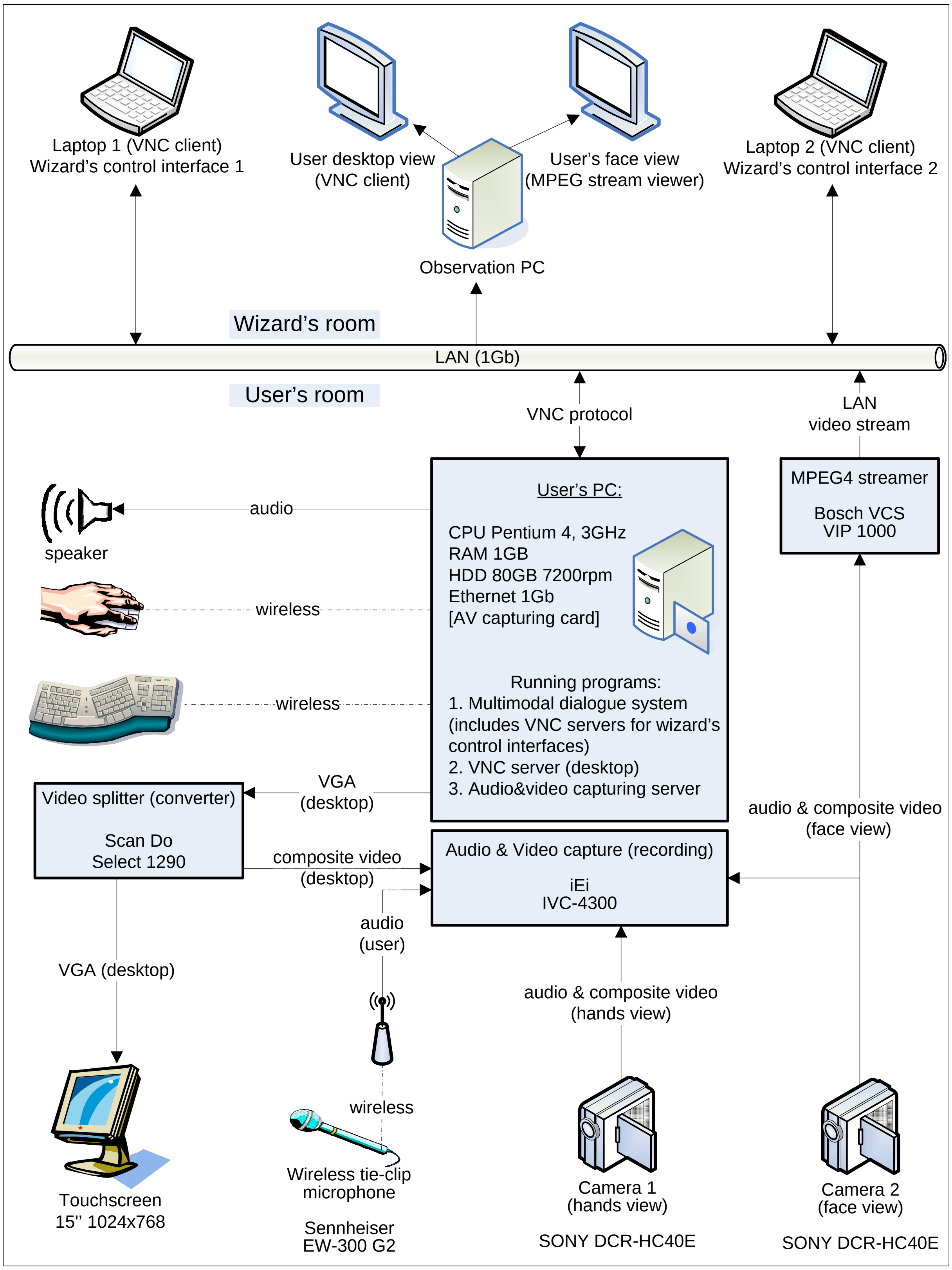


Figure 4.4: Wiring diagram of Wizard of Oz environment with specification of main hardware components.

instance a user ignoring instructions given to him/her by the experimenter, the wireless mouse not working due to discharged batteries, the user writing an SMS message during interaction, technical or cleaning personnel entering the room during the experiment (these are examples of real situations we have experienced during our WOz experiments).

Besides a general view on the interaction through observation screens, the wizard also has his own laptop running the Wizard's Control Interface (WCI, see section 4.5) with information about the internal dialogue state, and which allows the wizard to simulate or supervise specific system functions. In case of multiple wizards, each of them has his own laptop.

Technically, the user's and wizard's rooms are interconnected with an Ethernet network with IP protocol (Figure 4.4). The audio and video from face view camera is streamed by an MPEG4 streamer (VCS VIP 1000) over the Ethernet. We found the streamer convenient, as it has good quality and only low latency and is running independently of the main User's PC. On wizard's side, the streaming browser is launched on the Observation PC at the beginning of the series of the experiments and is continuously running, similarly as the VNC client showing the user's desktop. The VNC server for the user's desktop view is installed as a service on the User's PC and it is automatically launched on the User's PC start (further decreasing experimenter load, as he does not need to control it all).

4.4.1 Audio and video recording solution

An important issue for Wizard of Oz experiments is the *capturing and recording of the experimental sessions*. The requirements were to record and digitally store audio and video from all three channels (hands, face and screen views) in a quality sufficient for later analysis of the experiments. The three videos had to be mutually synchronized during playback together with the logfiles of the multimodal system, thus giving full overview of the interaction state. The required precision of the synchronization was defined as lower than one second – sufficient enough for reviewing of the recorded interaction by a human expert. Considering all those requirements, we have finally decided for the IVC-4300 MPEG AV capturing board[1]. The board is able to simultaneously capture up to four audio and video channels and store them into separate AVI files on hard drive. The video encoding is done on the board without using the power of the main CPU. Therefore, the board can be plugged into the User's PC without any loss of computing power, while an option of using an external PC for recoding is still available – for instance when User's PC is only a small tablet PC or laptop.

We have implemented a server application allowing control of the capturing board (start/stop recording, storing to videofiles) over TCP/IP sockets. In this way, recording is directly controlled by the tested multimodal interface, the timestamps of start/stop of recording events are marked in the main logfile of the application, and the captured videofiles are automatically stored on disk. All this guarantees trouble-free synchronized documentation of the experiments, without any further action required from the experimenter. Reducing the load of the experimenter is an important factor, as he already needs to perform a number of procedures, for instance giving consistently the same information

[1]We acknowledge the help of our colleagues from Fribourg University and IDIAP research institute with the installation of the IVC-4300 capturing board and technical support.

to all users, starting the correct version of the experimental interface, and keeping track of all documents given to user (consent form, manual, tutorial, tasks given to the user, questionnaire).

Prior using the IVC-4300 capturing board, we were recording the experiments manually. Each camera recorded an experiment on mini DV tape and the User's PC desktop was recorded by DVCAM recorder. This setup turned out to be quite problematical, as the experimenter had always to insert blank MiniDV tape to the recorders, start each recorder separately before the experiment and launch a synchronization signal on the User's PC (three audio beeps and video flashes) in order to allow post experiment synchronization. After finishing the experiments, all MiniDV tapes had to be encoded into MPEG files, cut according to the synchronization signal and stored on disk for further use. Every experiment post-processing took about three hours for 40 minutes of recorded interaction. Using IVC-4300 capturing board does not require any specific action from experimenter during the experiments, minimizes experiment post-processing and makes the experimental setup cheaper (no need of expensive MiniDV tape recorders).

4.5 The wizard's control interfaces

The main role of the wizard is to simulate the system's functionalities that have not yet been implemented, or supervise functionalities that are not yet working correctly. To do this, the wizard uses a Wizard's Control Interface (WCI).

The WCI consists of a set of graphical windows, each dedicated to the simulation or supervision of one specific system functionality. When the multimodal system is running in WOz mode, the data flows are redirected to the graphical modules of WCI, allowing the

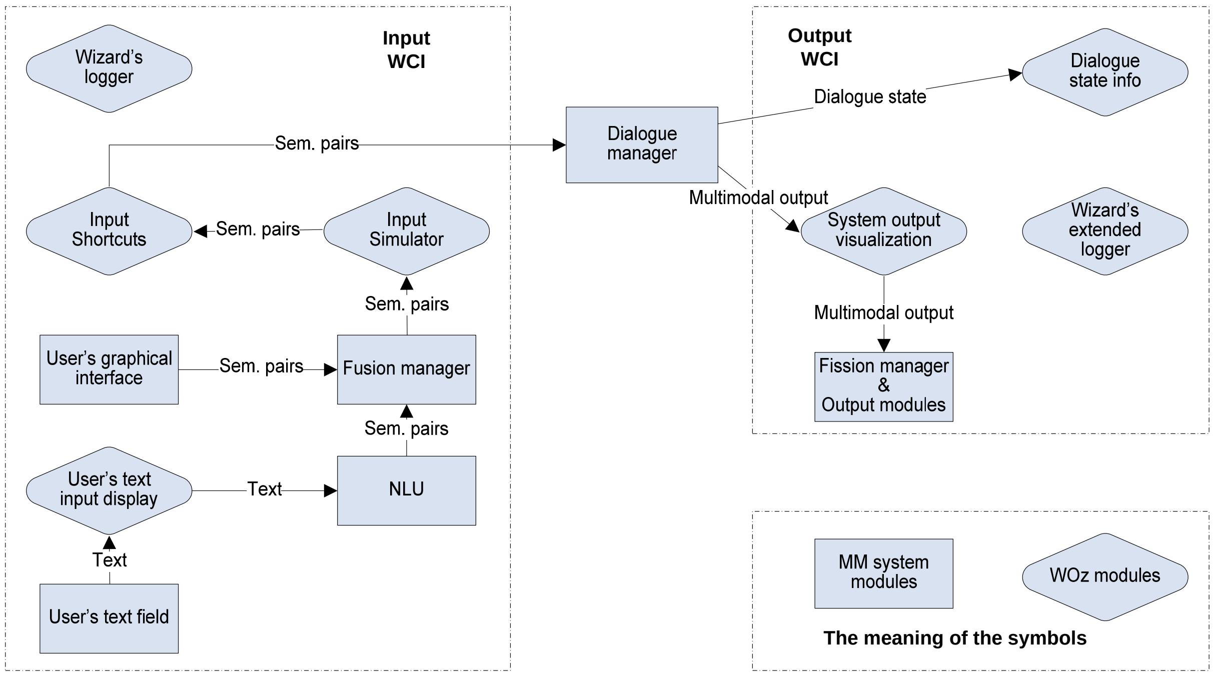


Figure 4.5: A simplified architecture of multimodal system with integrated wizard's modules. The modules correspond to individual graphical windows of WCIs shown in Figures 4.6 and 4.7.

wizard to check, modify, or even create new pieces of data if necessary. Once the wizard is satisfied with the data, he confirms them and the dialogue processing can continue. The wizard's graphical modules are an integrated part of the multimodal system, and are *plugged into the system as every other module* (see section 3.6 on page 50 for a description of the software architecture of our multimodal systems). Although the WCI is running as a part of the multimodal system on the User's PC, the graphical windows of the WCI are not displayed on the user's screen. The wizard uses a standard VNC client to connect to the multimodal system and to see and control the WCI (as described in detail on page 51).

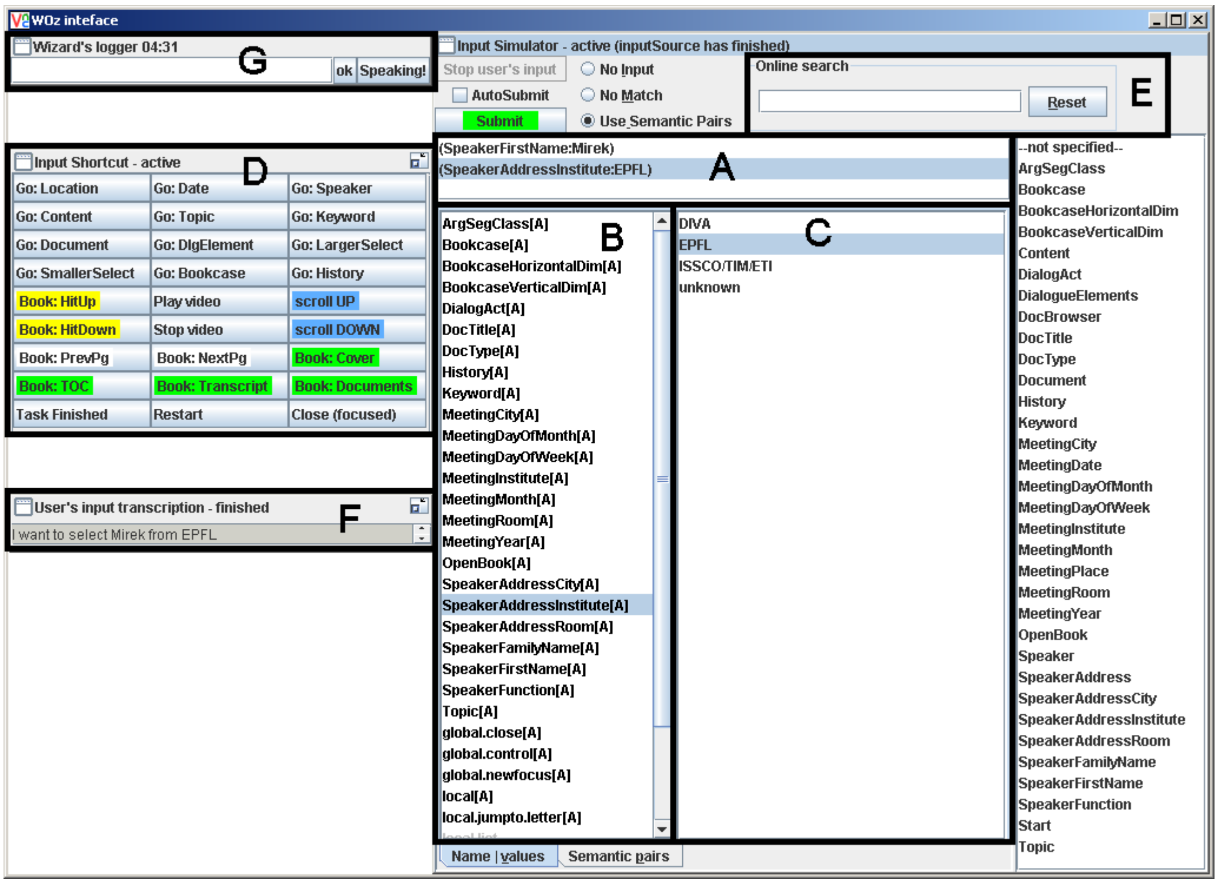
The wizard can control any functionality of the system that can be 'externalized' into modules with well-defined interface. In the rest of this chapter, we focus on those used during our simulations. In our settings (Figure 4.5), we wanted the wizard to *simulate* the speech recognition engine (SRE), as the speech recognizer was not integrated to our system. The performance of the natural language understanding module (NLU – transforms textual representation into a formal representation) was very limited, so we wanted the wizard to *supervise* it. On the system output side, we wanted the wizard to *supervise* the textual (and vocalized) response of a system, as the quality was not always sufficient in our system. We also had a setup allowing the wizard to supervise decisions made by dialogue strategies, but this is described elsewhere [79].

In order to increase efficiency and minimize the wizard's cognitive load, we propose sharing the wizarding tasks by two wizards, each managing a different part of the interaction: the interpretation of the user's input (Input Wizard) and the control of the system natural language output (Output Wizard).

4.5.1 Input wizard's interface

The user of the multimodal system can provide his/her input using several modalities, making the overall interpretation of his/her input more challenging. Therefore, in our approach each modality is processed separately, yielding (in some cases empty) sets of semantic pairs (see page 39 for details). The obtained semantic pairs may have associated confidence scores and can also be ambiguous (if the input provided through a given modality is ambiguous). The semantic pairs generated by the input from each modality are then combined (and disambiguated in the context of current mGDN) by the fusion algorithm into a single set. The role of the Input Wizard is *to check that the set correctly represents the user's overall input* and to make corrections or add new semantic pairs if necessary. In this way, the Input Wizard simulates or supervises the functionality of the SRE, NLU, and Fusion management modules in the system (or other modules processing other modalities, if they are present in the given system).

The Input Wizard's interface is shown in Figure 4.6. The wizard can see the semantic pairs resulting from automated system processing in (A). These can be removed from the set using the *delete* key. To add a semantic pair, the wizard has to select its category (name) from the list (B) first, then select the appropriate value for it (C). The list of available semantic pairs is generated automatically from the domain model (see section 3.2) and information provided by the mGDNs at application start-up. Once the wizard is satisfied with the interpretation of user's last input, he confirms it with *Submit* button.



[✓ Use Semantic Pairs]

(SpeakerFirstName:Mirek)
(SpeakerAddressInstitute:EPFL)

I want to select Mirek from EPFL

Figure 4.6: Input Wizard's Control Interface: (A) semantic pairs generated automatically by the system or added by the wizard; lists that allow the wizard to create a new semantic pair: (B) the list on the left shows names of semantic pairs, (C) the list on the right displays all possible values associated with the selected name of the semantic pair; (D) shortcut buttons; (E) semantic pairs filter; (F) a request typed-in by user using keyboard; (G) a textbox for wizard to write down a time-stamped free-form notes about the experiment.

Since the wizard's response time has to be as short as possible, we have optimized the interface during several runs of pilot experiments. One of the improvements is a panel with (D) *shortcut buttons* that replicates common actions on the user's interface. Those are typically actions that the users can access in 'one click' on their multimodal interface (we call them *navigation commands*). Slow reactions of the system on these actions when using other modalities (especially voice) could influence the user's natural selection of modality for the next use of navigational commands, negatively affecting the validity of the experimental data. By clicking on the shortcut button, the corresponding semantic pair is generated and immediately submitted. For instance, as a reaction to user's input *"Show me the available locations"*, the wizard had to click on *Go:Location* button, generating a `newFocus:Location` semantic pair.

Another improvement is a *quick-search mechanism* for finding items in the semantic pair (SP) list. Although the two-level organization (first select SP name, then SP value) is convenient in certain cases, the online search field (E) restricts the displayed SPs to only those containing the typed expression. This is important particularly when the whole list of possible SPs contains too many entries (in our case over 2100 entries in 35 categories). When the number of SPs is lower than a predefined threshold (less then 20), the SPs are displayed in a simple list (instead of two-level lists) in order to facilitate SP selection. The quick-search mechanism also helped the wizard to *disambiguate* uncertain situations by highlighting the items possible in the given context.

The Input WCI contains also the *Stop user's input* button. This button is used by the wizard to signal to the user that the system (wizard) is busy processing his/her last input. Signalization is done by showing an inactive, grayed-out user's screen and mouse pointer displayed as an hourglass.

Another control element on the WCI is the *autosubmit* checkbox. When ticked off, the system does not wait for the wizard to modify the semantic pair set resulting from automated processing. The button is useful in situations when input recognition modules are fully automated or when the user is using only modalities that can be reliably transformed into semantic pairs (e.g. clicking on graphical elements).

Altogether, the structure of the wizard's interface permitted fewer 'human-like' interpretations of the user's input, since the number and types of possible wizard's actions is limited. This was particularly useful given the amount of information to be processed by the wizard in a short time. Part of the overhead of thinking about what the system would or would not be capable of processing was thus eliminated. On the other hand, we allowed the wizard to write down any notes about the ongoing experiment in an unstructured form. The note is typed into a text box (G) and automatically annotated with a timestamp. However, it turned out that the wizard is unable to use this text box during interpretation of user's input due to already high cognitive load.

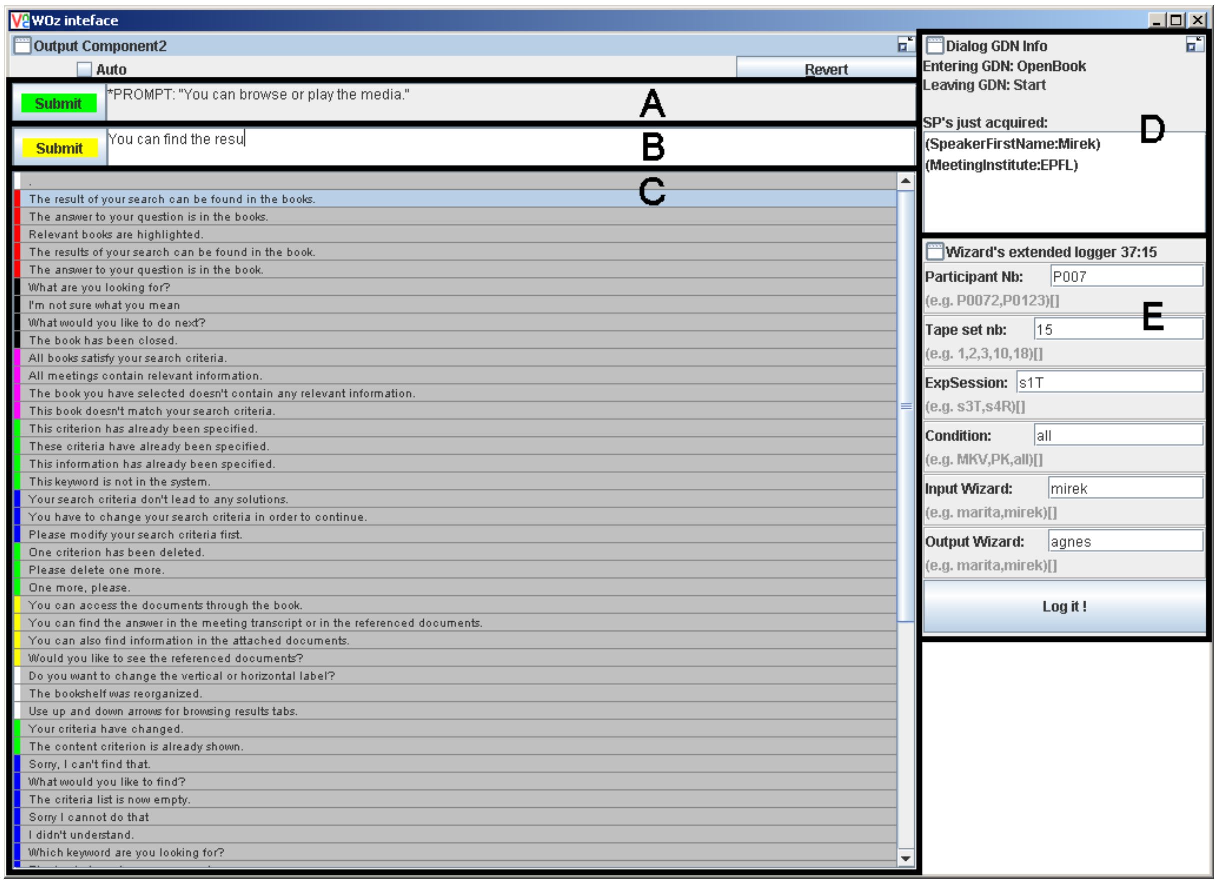
4.5.2 Output wizard's interface

The second wizard controls the vocal NL output given by the system using another interface on a different machine (Figure 4.7). The modification of the output was necessary in cases when the default prompt (A) suggested by the existing dialogue management module was insufficient or inappropriate given the current dialogue situation.

Depending on the actual internal state of the dialogue system (D), the wizard would select a new prompt from a predefined list of prompts (C), or create a new one on the fly (B), which was then automatically added to the list. This ensured that the *same set of prompts* was available during the experiments and thus introduced *less unwanted variety* in what the system could answer. All modified and new prompts together with the corresponding dialogue system states are logged for further analysis of dialogue strategies quality.

The system-suggested prompt is determined by the dialogue manager only after the Input Wizard has finished the interpretation of the user's last input. In order to minimize the delay before the Output Wizard select the prompt, our interface allows the Output Wizard to pre-select the prompt even before the Input Wizard finishes his work. When the system-suggested prompt appears to be appropriate, the wizard can confirm it by the first *Submit* button (A), otherwise the output wizard confirms the modified prompt by the other *Submit* button (B). This ergonomic feature turned out to be very useful for reducing an overall system response time perceived by user.

Because the task of Output Wizard is cognitively slightly less demanding compared to the task of Input Wizard, we have decided to make him responsible for logging of the



*PROMPT: "You can browse or play the media."

You can find the resu

.
The result of your search can be found in the books.
The answer to your question is in the books.
Relevant books are highlighted.
The results of your search can be found in the book.
The answer to your question is in the book.
What are you looking for?
I'm not sure what you mean
What would you like to do next?
The book has been closed.
All books satisfy your search criteria.
All meetings contain relevant information.
The book you have selected doesn't contain any relevant information.
This book doesn't match your search criteria.
This criterion has already been specified.
These criteria have already been specified.
This information has already been specified.
This keyword is not in the system.
Your search criteria don't lead to any solutions.
You have to change your search criteria in order to continue.
Please modify your search criteria first.
One criterion has been deleted.
Please delete one more.
One more, please.
You can access the documents through the book.
You can find the answer in the meeting transcript or in the referenced documents.
You can also find information in the attached documents.
Would you like to see the referenced documents?
Do you want to change the vertical or horizontal label?
The bookshelf was reorganized.
Use up and down arrows for browsing results tabs.
Your criteria have changed.
The content criterion is already shown.
Sorry, I can't find that.
What would you like to find?
The criteria list is now empty.
Sorry I cannot do that
I didn't understand.
Which keyword are you looking for?

Entering GDN: OpenBook
Leaving GDN: Start

SP's just acquired:
(SpeakerFirstName:Mirek)
(MeetingInstitute:EPFL)

Participant Nb:	P007
(e.g. P0072,P0123)[]	
Tape set nb:	15
(e.g. 1,2,3,10,18)[]	
ExpSession:	s1T
(e.g. s3T,s4R)[]	
Condition:	all
(e.g. MKV,PK,all)[]	
Input Wizard:	mirek
(e.g. marita,mirek)[]	
Output Wizard:	agnes
(e.g. marita,mirek)[]	

Figure 4.7: Output Wizard's Control Interface: (A) the prompt automatically determined by dialogue strategies; (B) the prompt suggested by wizard, which can be freely modified; (C) a list of predefined system prompts used by wizard; (D) an internal system state information used by wizard to determine most appropriate prompt; (E) information about interaction session entered by wizard.

interaction session information (E). This information is essential for keeping track of all performed experiments.

4.6 Performance of the Wizards

In this section, we provide a quantitative evaluation of performance of Input and Output Wizards, which serves for an estimation of the quality of the interface. The data used for the analysis come from Wizard of Oz experiments that we have carried out in order to evaluate the Archivus system. Details about the Archivus system and the experimental setup are given in Chapters 5 and 6 respectively.

4.6.1 Input wizard performance

Although only one human operator (wizard) controls the Input WCI at one time, two distinct persons acted as the Input Wizards due to their limited availability for the WOz experiments. Both wizards underwent an intensive training during pilot experiments with the interface. We wanted to know *how quickly and consistently the wizard responds* to

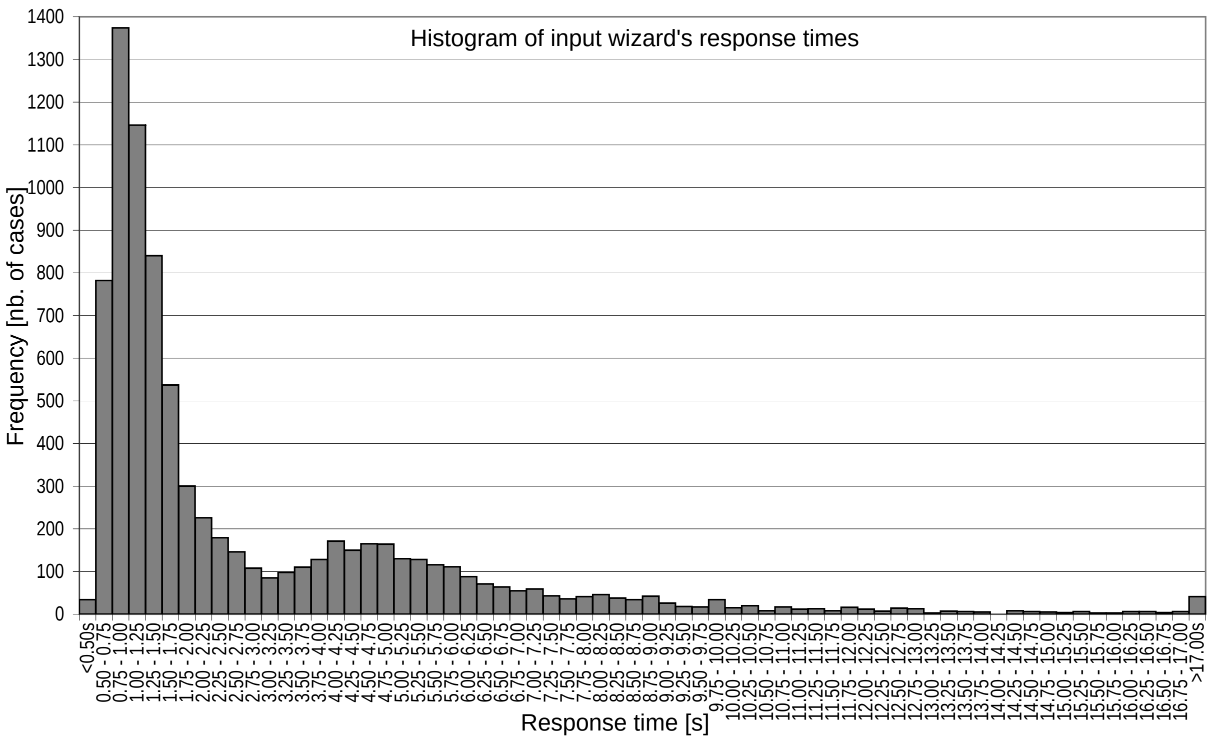


Figure 4.8: Histogram of the Input Wizard's response times

various user requests. The users did not have any previous experience with the tested system.

The collected data consists of 8215 cases when the wizard had to confirm, modify or create an interpretation of user's multimodal input (set of semantic pairs). The system contained about 2100 possible semantic pairs to choose from, organized in 35 available categories. The user produced his/her input using mouse, pen, voice or keyboard. For each case, we have logged not only the wizard's response time, but also his identification (W1, W2), the method that the wizard used to submit the data (either using the shortcut buttons or by the submit button), and the modality used by user.

An overall average Input Wizard's response time was 2.8s (with standard deviation 3.0s). As depicted in Figure 4.8, the response times are not evenly distributed – in fact, the wizard's response was in 68% of cases shorter than 2.8s. The histogram clearly shows two main peaks of the response time – the first around 1.0s and the second around 4.5s. We have further investigated the reasons for such a distribution. For such analysis, we removed 0.5% of data from each extreme side of the histogram (i.e. response times shorter than 0.5s and longer than 17s). The very long response times were caused by technical problems or wizard's inattention, rather than by actual need of so much time to prepare the response.

The results of the further analysis are summarized in Table 4.1. Firstly, we found that the average reaction speed of the two wizards slightly differs (0.6s) and that the slower wizard is slightly less reaction-time consistent (higher standard deviation). However, this does not explain the two peaks in the histogram of wizard's responses.

Secondly, the average wizard's reaction time depends on the modality that the user used to produce his/her last input. This follows from the absence of a speech recognizer, thus

Category	*Response time [s]*		*Remark*
	Average	*Std. dev.*	
By wizard:			
W1	2.5	±2.4	
W2	3.1	±3.2	
By modality used by user:			
mouse or pen (MP)	1.2	±0.7	confirms SP
voice (V)	4.0	±3.0	creates SPs
keyboard (K)	4.2	±3.2	modifies SPs
By method used by wizard:			
used shortcut button (VK)	1.6	±1.1	
used submit button (V)	5.4	±2.9	creates SPs
used submit button (K)	4.3	±3.2	modifies SPs

Table 4.1: Input wizard response times by various categories.

the wizard had to generate the whole semantic interpretation each time a voice input was used. For the typed natural language, the system contained a simple NL understanding module. However, the performance of the module was quite low and the wizard often had to modify the interpretation proposed automatically by the system. For pen and mouse input, the system always produced a correct semantic interpretation and the wizard had only to confirm it (or extend it with other semantic pairs in case when the user used a combination of modalities – this however happened so rarely that we do not consider it in this analysis). As a result, the wizard's reaction time primarily depends on the type of action that the wizard had to take – confirmation, modification or generation of semantic interpretation.

Finally, we found that the generation of semantic pairs is largely facilitated by the shortcut buttons created for user's navigational commands in natural language. Using these buttons, the input in natural language (voice or keyboard) can be processed nearly as fast as pen or mouse input (1.6s vs. 1.2s).

We can conclude that the wizard's interface allows for a quick response (within 1–2s) to any pointing modality (mouse, pen) and to navigational commands in natural language. As for the other requests in NL, the wizard is able to serve them in about 5s on average. This explains the two peaks in the histogram of the wizard's response times. The higher response time of NL non-navigational requests is caused by two facts: (1) appropriate semantic pair has to be found among 2100 semantic pairs used in the system; (2) about 17% of non-navigational requests in NL require more than one semantic pair for interpretation.

4.6.2 Output wizard performance

The Output Wizard needs to *quickly* propose another NL prompt in situations where the default prompt selected by the system is insufficient or inappropriate given the current dialogue situation. During our experiments, only one wizard controlled the output WCI.

The wizard usually selects the prompt from a *predefined list*, but can also *create a new one*, which is then automatically appended to the list. The initial list was collected during several runs of pilot experiments and contained 38 prompts. During further experiments

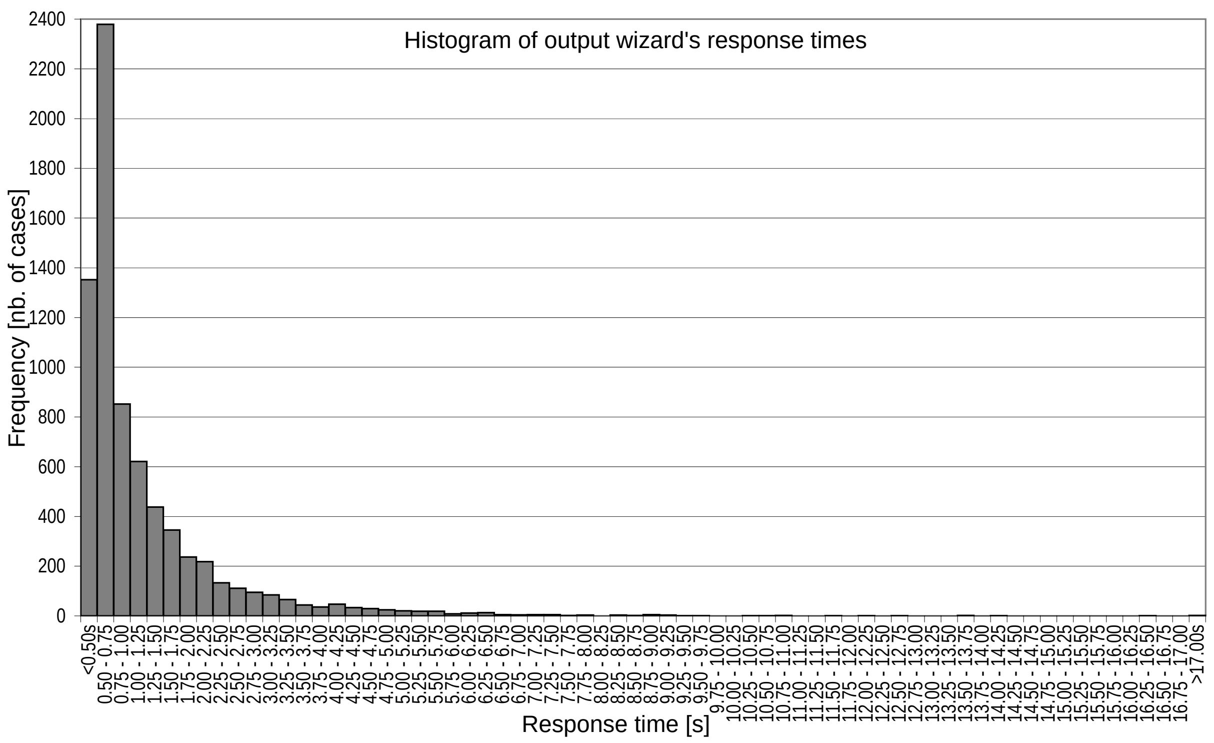


Figure 4.9: Histogram of the Output Wizard's response times

(91 users, each interacting with the system for 40 minutes), the list was extended to 60 prompts. The prompts in the list were slightly typographically modified between the experimental sessions and reorganized in the list according to their context of use.

Action type	*Response time [s]*		
	Average	*Std. dev.*	*Median*
all actions	1.2	±1.2	0.7
confirm prompt	1.2	±1.1	0.8
select prompt from list	1.2	±1.3	0.7
define new prompt	5.2	±5.0	3.0

Table 4.2: Output wizard response times by various types of taken actions.

Concerning the wizard's response times, the histogram in Figure 4.9 clearly shows that in most of the cases, the wizard was able to respond within very short time (less than two seconds) and that longer response times (more than four seconds) are very rare. The comparison of average response times for different wizard's actions in Table 4.2 confirms, that the most frequent actions (i.e. the confirmation of system-suggested prompt and the selection of prompt from a list) are almost equally fast. The only action that takes several seconds is a definition of a new prompt, which is however very infrequently used (0.3% of all wizard's actions).

Interestingly, the median of time necessary for selecting a prompt from list is smaller than the median of prompt confirmation time. We speculate that this is caused by fact that the Output Wizard can pre-select the prompt from the list during the time the Input Wizard needs to process the user's last input.

The submission of the pre-selected prompt is very quick, while the system-suggested prompt has to be reviewed before submitting it (the system prompt is determined and appears on wizard's interface only after the Input Wizard has finished his work). This validates our hypothesis that the two wizards (Input and Output) can, in a number of situations, work simultaneously and thus the WCI should support the concurrent operation of both wizards.

4.7 A general design of the WOz study and the collected data

For WOz simulations, it is important to define a certain evaluation protocol and consistently give the same amount of information to all participants in the study, in order to minimize the risk of introducing a new, unwanted, and uncontrolled variability to the experiments.

Since a detailed specification of the evaluation protocol depends on a number of criteria (such as the maturity of system, its purpose, or an overall objective of the study), we do not provide any specific guidelines about the evaluation protocol. The evaluation protocol for our WOz experiments was carefully designed by our colleagues from the University of Geneva within framework of another doctoral thesis. The interested reader is referred to [57] for further details. In this section, we discuss some general issues to be taken into account and give more detailed information about the data (logfiles) that are collected during the experiments.

Our experience shows that several runs of initial *pilot studies* with the system are indeed necessary. They typically involve only few users (4 to 8) and can very quickly reveal problematical parts of the interface even without any formal evaluation. During the pilot studies, the interface can undergo quite substantial changes, the code is debugged, the wizards get necessary training, and the whole experimental protocol is tested. The pilot studies are particularly useful for systems that were not designed using Free WOz simulations and where several design decisions still need to be validated.

Concerning the typical procedure of the experimental protocol, the experimenter should welcome the user, provide him/her with a short description of the project and inform him/her about the interaction being recorded during the whole session. The presence of wizards should not be revealed at this point. Although some authors [32] argue that it is ethically incorrect to deceive users about wizard's presence, another author [27] claims that it is acceptable to reveal the truth at the end of experiment. If the user agrees with the experimental conditions, the experimenter asks him/her to sign a consent form.

The user then receives instructions on how to use the system. This can be done by means of a manual or tutorial. While the manual is useful for systems that are intuitive to use, the tutorial is more suitable for systems involving new types of complex interactions. However, defining an unbiased tutorial procedure (i.e. a tutorial teaching every user in the same way how to use the system, without favoring using of some specific modality for specific action or implying any predefined interaction pattern) is quite a challenging task [58].

Before starting the actual experimental session, the user is given a precise interaction context with scenario defining the situation for using the system (*"You have arrived into a city that you do not know and your hotel is equipped with a multimodal information kiosk"*). The user is then connected to the system and is asked to perform a set of tasks (e.g. *"Find a Chinese restaurant in the city and reserve there a table for five persons tonight"* or fact checking *"How many Chinese restaurants are there in the old town?"*). The experimenter should not help the user during this phase in order to minimize the risk of biasing the interaction. Ideally, the experimenter leaves the room, but is available (via ring, or in the next room) in case of unexpected technical problems.

After the user has finished all necessary interaction sessions and the interaction data have been collected (objective indicators), the user is asked to complete the final questionnaire. The questionnaire typically contains the demographical and background information about user, questions concerning user's satisfaction with the system, usability related questions, the interaction related questions, and other subjective measures. Even though the questions requiring structured answers (scale or yes/no) are much easier to evaluate, the questionnaire should also give the user a possibility of freely expressing his/her opinions of the system and of the experienced interaction. In addition to it, we have also acquired a lot of valuable information during an informal interview with users at the end of the experiment. Finally, the users are often given a compensation for their participation (a gift or financial reward).

4.7.1 Collected data

During the experiments, the entire interaction is videorecorded from three views (screen view, user's face view and hands view) and the system produces logfiles – a structured information about the interaction. While the videofiles are suitable for manual review and annotation of the interaction by a human expert, the logfiles are a great source for automated analysis of experiments.

However, it might take a substantial time and effort to analyze the logfiles without an optimal log format. Since the author of this thesis experienced certain problems when analyzing the logfiles, the experience is reported and recommendations are made in this section.

Our logfile has a textual format. Every logical action of the modules in the multimodal system is annotated (in the module source code) with a specific message and with a call of logging routine. The logging routine appends the message to the logfile, together with the module identification and the timestamp. The resulting logfile is a human-readable: it is a time-stamp organized sequence of actions that the multimodal system performed during the interaction with user.

Our first encountered problem was that the logfiles did not always contain the necessary information, although we have used the policy 'log as much as you can'. We have learned that it is crucial to *define beforehand the necessary parameters* that need to be logged and verify that all necessary information is then indeed consistently logged by the system. For instance, although we have always logged the semantic pair set produced by wizard, we did not log the semantic pair set displayed to the wizard for correction. This would have helped to easily identify the modality used by the user in his/her input (we finally managed to gather this information by inferring it from other sources, but the identification routine was complex).

For convenient data processing, we have transformed the logfiles into a SQL database. The relational table contains one record for each dialogue turn, clearly separating dialogue state changes. While using the SQL database instead of sequential textual logfiles significantly facilitated the evaluation, the transformation procedure turned out to be very complex. The identification of the log messages belonging to the same dialogue turn was problematical, because no message was clearly separating two dialogue turns. Although we finally managed to identify them, the procedure was again quite complex and had to deal with many of specificities (due to multiple threads in our system, the messages had no fixed order, often having the same timestamp, etc).

Based on this experience, we propose that each logged message should contain (besides the timestamp) a logical *identification of the session* (a sequential number, increasing at every start of the multimodal application) and a *turn number* (a sequential number, increasing at every new dialogue turn). The session and turn identifiers can be automatically appended to the logged message within the logger routine (similarly as the timestamps). Such logfile organization enables a simple identification of starts and ends of the experimental session and assigning them to particular users, and is also convenient for further annotation (e.g. manual transcription of user's utterances), since annotations can be uniquely associated with dialogue turns. Transformation of the logfiles into a sequence of dialogue states in SQL table is then trivial. The turn-based organization makes it also easy to detect information missing in some of the turns (i.e. information logged inconsistently).

4.7.2 Data exploitation tools

In order to review experimental sessions, we have implemented a tool that enables *simultaneous playback* of several recorded videofiles aligned with a view on the logfiles. Such tool gives to a human expert a full and comfortable view on the experiments.

Another tool that we have used were simple *queries in SQL language*. Once the data were consistently organized in a database, it was easy to retrieve various statistical indicators about the interactions. However, as mentioned earlier, the transformation of data from textual logfiles into a database was sometimes quite cumbersome.

In order to retrieve information that was initially forgotten to log, we have implemented a tool named *dialogue simulator*. The dialogue simulator repeats the entire dialogues recorded in logfiles by supplying the dialogue manager with the same semantic input as during the experiments. Because the dialogue manager works in a deterministic manner, the simulated run of the dialogues involves exactly the same dialogue strategies, but the information can be logged in greater detail during this simulated run. Of course, the newly retrieved data can only expand the information about the system steps – no further information about the user actions can be retrieved automatically. The user actions can be further annotated only by manual review of the videofiles.

4.8 Summary

One of the most productive exploitation of rapid prototyping to date has been to use it as a tool for iterative user requirements elicitation and human-computer interface design [77]. Indeed, observing the prototype in interactions with users reveals real user needs and their preferred ways of interaction. However, such an observation cannot be achieved without a nearly fully functional high-fidelity prototype, as the degree of maturity of the prototype strongly influences the user's reactions.

We propose *Wizard of Oz techniques* for the testing of *multimodal systems*. Although WOz simulations have been traditionally used for vocal dialogue systems only, we show that they can be extended to multimodal systems.

Even though the idea of using WOz simulations for the design of multimodal systems is not new, to the best of our knowledge, there is no available literature describing the experimental environment in detail. Therefore, our contribution consists of three main parts: (1) a detailed description and analysis of the requirements for the experimental environment, (2) an experimentally validated integration of WOz techniques within a generic design methodology and (3) practical recommendations for carrying out WOz simulations. A major part of our environment is generic enough to be used for testing of any multimodal system, not necessarily only of those designed with our methodology.

The *Wizard's Control Interface* (i.e. the software used by wizards to simulate not yet fully implemented functionalities) can be tailored to the targeted application. In addition, the interface can be *easily distributed* over several computers to be simultaneously used by several wizards. We describe two particular examples of wizard's interfaces that are typically used at initial stages of any multimodal system development. In these examples, the interfaces allow the wizards to interpret inputs from different modalities and to produce/adapt system responses accurately and efficiently. They are implemented in the form of modules integrated within the multimodal system, and can therefore be *reused* for a broad range of applications. We describe the interfaces and their functionality and discuss usability and ergonomic issues. In this perspective, our main conclusions are:

- The wizard's interfaces must be *carefully incorporated* into the system in order to allow the wizards to have a sufficient degree of control over the system and its interaction with the user.
- The interfaces have to ensure *speed and consistency* for wizard responses and ergonomic issues have to be very carefully taken into consideration (e.g. shortcuts for frequently used actions, fast access to a database with interpretation elements, support for several wizards working simultaneously). Not surprisingly, the overall wizard's (in)efficiency has a strong impact on the perception that the users have of the system's efficiency.
- *Several runs of pilot experiments* are necessary to correct initial potentially wrong system design decisions and to train the wizards.

Chapter 5

Case study: the Archivus system

The Archivus system was the main case study we used for testing, evaluating and validating the Interactive System Prototyping Methodology described in Chapter 3 and the Wizard of Oz simulations presented in Chapter 4. In addition, the Archivus system also served as an experimental framework for the evaluation of the efficiency of our local and global dialogue strategies (Chapter 6).

The Archivus system was designed in the framework of the IM2 project[1] in tight collaboration between the University of Geneva and EPFL. As such, it also served as an experimental framework for two other doctoral theses: the first one [57] focused on determining which input modalities are the most useful and appropriate for meeting browsing and retrieval; the second (currently ongoing) investigates the role of natural language in multimodal interfaces.

In this chapter, we present the Archivus system as an *illustrative example* of a multimodal dialogue-based system developed with our prototyping methodology. In section 5.1, we first start with a description of the system. The section 5.2 presents the individual steps for designing such a system using our ISPM. In section 5.3, the improvements made during the iterative development of the Archivus system are summarized, in order to illustrate the very valuable experience for designing of any multimodal dialogue-based system that was acquired during such a process.

5.1 System description

Archivus is a multimodal dialogue system for accessing a database of recorded and annotated meeting data [60, 17] (the *Smart Meeting Room* application). In short, the system helps the user to answer questions like *"What were John's questions related to the budget in the meeting in April?"*. The user can retrieve specific pieces of information about what happened in different meetings, for example topics that were discussed, decisions that were made, documents that were presented, or people who were active in proposing ideas.

[1]IM2: Interactive Multimodal Information Management, `http://www.im2.ch`, funded by the Swiss National Science Foundation.

5.1.1 Input and output modalities

The Archivus interface was designed and developed to be *flexibly multimodal*, meaning that each system action can be invoked with any available modality. Flexible multimodality gives the users freedom in controlling the system: the users can interact with the system either using only one fixed modality all the time, or they can flexibly switch between modalities in a way that is most comfortable to them. It is also possible to control the system truly multimodally and use several modalities simultaneously.

The input modalities available to the user are of two main kinds: pointing (mouse, pen, touchscreen) and natural language (voice, keyboard). We believe that the voice and complex natural language expressions can be particularly useful in the meeting search domain and we also believe that this domain encourages users to try out and consistently use such novel input modalities.

The output modalities used by the system are: audio responses in natural language, screen based textual feedback, images and video.

5.1.2 Accessed data

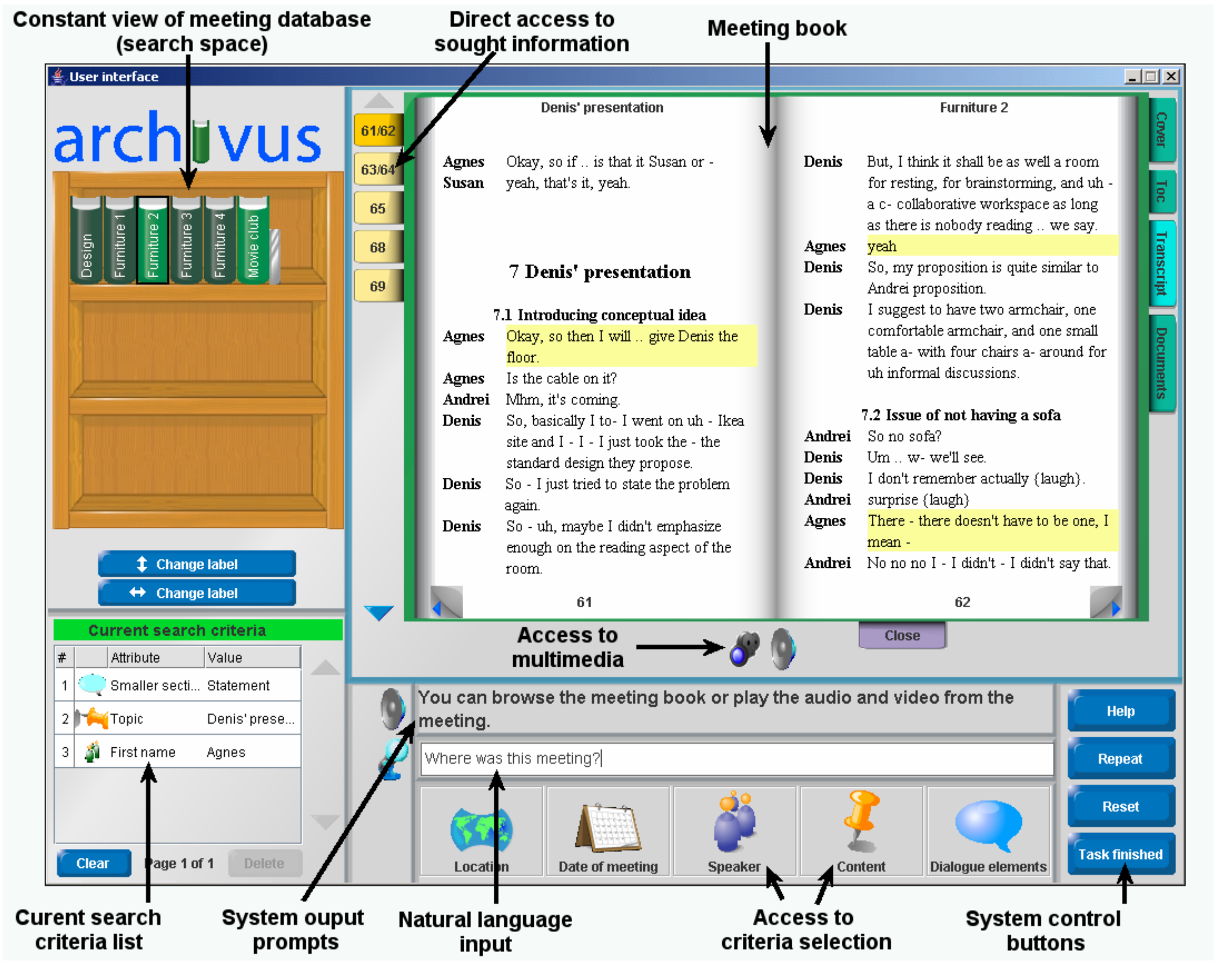
The Archivus meeting database contains captured *audio and video* from the meetings (recorded in special meeting SmartRooms [72]) and electronic copies of all *documents used* (paper artifacts, slides, etc). In order to facilitate retrieval of information stored in the database, manual annotations have been made on the data. The participant discussions were *transcribed* and annotated with *dialogue acts*, *topic segmentation*, *argumentative annotation*, and explicit *references to stored documents.* Each of the meeting is also stored with metadata such as the *date* and *location* of the meeting and information about the *meeting participants.*

Overall, our database includes 6 meetings (192 minutes of video data) held in English by a total of 8 different participants with typically 4 participants in one meeting. The rooting scenario of four meetings is a room furnishing, while the other scenario is a movie club meeting and a meeting to determine a design of a remote control.

5.1.3 Functionalities of the system

The major tasks that the user can perform using Archivus are:

- find meetings, parts of a meeting, or specific information in a meeting based on following criteria or any of their combinations:
  – the date, location or participants in a meeting
  – the topics covered, keywords spoken, or documents used in a meeting
  – the dialogue acts (i.e. questions, statements, etc.) or argumentative sections (discussions, arguments, etc.)
- get an overview of all meetings that are relevant to a user's goals
- browse quickly and easily through only the meetings or meeting sections that are relevant to the user's goals



#	Attribute	Value
1	Smaller secti...	Statement
2	Topic	Denis' prese...
3	First name	Agnes

Denis' presentation

Agnes Okay, so if .. is that it Susan or -
Susan yeah, that's it, yeah.

7 Denis' presentation

7.1 Introducing conceptual idea

Agnes Okay, so then I will .. give Denis the floor.
Agnes Is the cable on it?
Andrei Mhm, it's coming.
Denis So, basically I to- I went on uh - Ikea site and I - I - I just took the - the standard design they propose.
Denis So - I just tried to state the problem again.
Denis So - uh, maybe I didn't emphasize enough on the reading aspect of the room.

61

Furniture 2

Denis But, I think it shall be as well a room for resting, for brainstorming, and uh - a c- collaborative workspace as long as there is nobody reading .. we say.
Agnes yeah
Denis So, my proposition is quite similar to Andrei proposition.
Denis I suggest to have two armchair, one comfortable armchair, and one small table a- with four chairs a- around for uh informal discussions.

7.2 Issue of not having a sofa

Andrei So no sofa?
Denis Um .. w- we'll see.
Denis I don't remember actually (laugh).
Andrei surprise (laugh)
Agnes There - there doesn't have to be one, I mean -
Andrei No no no I - I didn't - I didn't say that.

62

You can browse the meeting book or play the audio and video from the meeting.

Where was this meeting?

Figure 5.1: The Archivus system graphical user interface

- view documents from a meeting and browse through them
- watch video, listen to audio or read the text transcript of a meeting
- browse through any meeting without specifying search criteria
- customize the organization of the entire database of meetings based on one or two criteria, for example by date and speaker

5.1.4 The Archivus metaphor

An interaction metaphor uses the terms and concepts already familiar to users, in order to explain them how to interact with an unfamiliar application and what its functionalities might be [30]. Since the Archivus system is a novel type of application, it was vital to find a suitable metaphor to which to situate the system.

For Archivus, we have chosen a *library metaphor* (Figure 5.1) as we have found that meetings can be mapped quite easily and naturally to the content of a standard book and that a database of meetings can be mapped to the structure of a library.

Thus, in Archivus, the database of stored meetings is represented by a series of bookcases containing meetings data. Each individual meeting is represented by a *book* (detailed views on possible book layouts are in Figure 5.4 on page 89), and a set of related meetings as a volume of a series. The title of the book becomes the main topic of a meeting, while the authors are the meeting participants. The publisher's information page contains the time, date and location of the meeting. The table of contents corresponds to the agenda of the meeting, listing the topics of the meeting. Chapters in the book always represent individual topics in a meeting, chapter sections reflect the dialogue structure of the topic, and individual paragraphs are the specific utterances that participants make. The documents that were used or referred to during the meeting are shown as the appendix of the book.

5.1.5 Platform

The platform and environment for which the Archivus was designed is a laptop (tablet PC) or a desktop PC with touchscreen used in an office-like environment.

This decision was based on two facts. The first is that a PC environment reduces learning curves introduced by new hardware. The second is that while the use of PDAs and mobile phones is rapidly gaining popularity, we are not convinced that the general population is sufficiently familiar with their use for information browsing and retrieval, nor that the devices are powerful enough to handle the information types in question. However, modification of the Archivus interface for use on handheld devices may be investigated in future work.

5.2 Building the system using ISPM

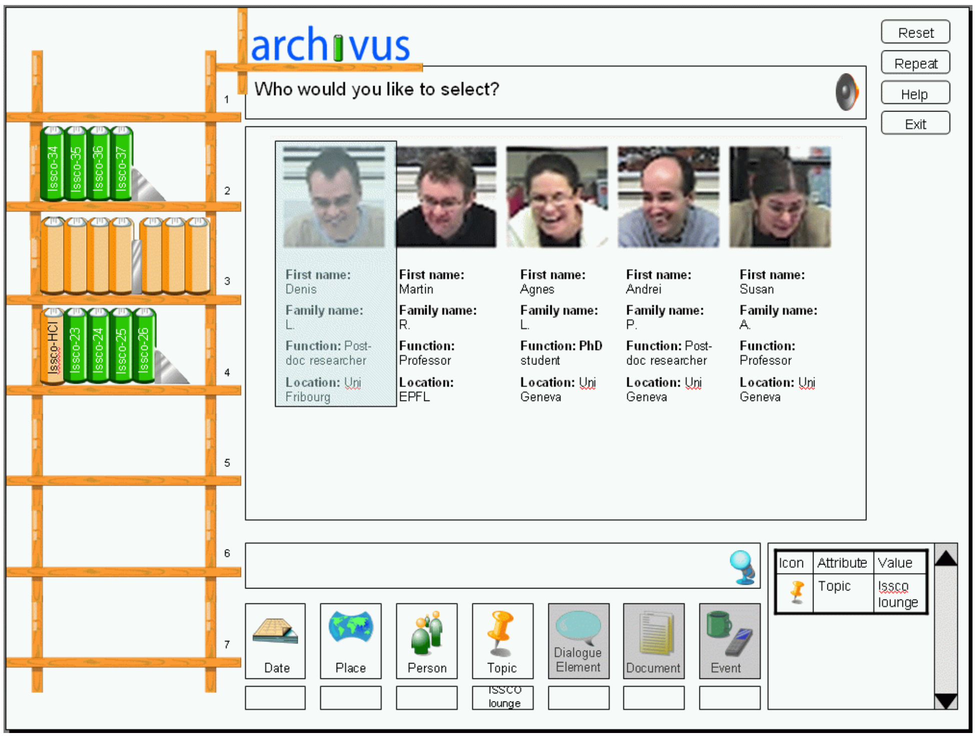
In this section, we present the individual steps of our Interactive Systems Prototyping Methodology that we took in order to design the Archivus system.

5.2.1 User requirements elicitation

The initial step is to identify the functionalities that the system must provide. This was in our case largely complicated by several factors. First, the *domain of application is entirely new* to the users. Specifically, no similar application exists and therefore users do not have any pre-existing set of functionalities that they might require from the system. Second, even the *set of potential users was unknown*, as this system was designed as a research prototype and not as an application for a specific client (user). Finally, the *use of multiple modalities is quite uncommon* in the office-like environment that we wanted to build.

The user requirement study under such complicated circumstances was carried out by our colleagues from University of Geneva and it is described in [56, 59]. Firstly, a set of *possible scenarios* and possible users was created based on the initial objectives of the application. Those were the following:

- A manager tracking employee performance
- A manager tracking project progress



Who would you like to select?

First name:	Family name:	Function:	Location:
Denis	L.	Post-doc researcher	Uni Fribourg
Martin	R.	Professor	EPFL
Agnes	L.	PhD student	Uni Geneva
Andrei	P.	Post-doc researcher	Uni Geneva
Susan	A.	Professor	Uni Geneva

Icon	Attribute	Value
📌	Topic	Issco lounge

Figure 5.2: An initial electronic mockup of the Archivus interface, created as a series of MS PowerPoint slides representing an example of multimodal interaction with the system. Compare with the final version of the system in Figure 5.1.

- A current employee who has missed a meeting
- A new employee who needs to learn about a project
- Fact checking: a person was in the meeting, but needs to recall or verify certain fact

In order to identify the tasks for which the Archivus system would be used, a user requirement study was carried out using the five above mentioned scenarios. The study involved 20 participants from different backgrounds, who were asked to list the *types of questions* that they would pose to the system, or the *type of information* that they would like to find.

As a result of the user requirement study and a series of in-house brainstorming sessions, we have obtained an electronic mockup of Archivus (Figure 5.2), which demonstrates a possible interaction with the system. The mockup and the set of user requirements was our initial entry point for the development of the Archivus system.

5.2.2 Domain model

Based on the natural language queries from the user requirement study, we have identified the *types of constraints* that users prefer to use for searches in the meeting domain. A full list of those constraints is in Table 5.1, nevertheless, the sought meeting information was essentially described in the following principal terms:

- Location of the meeting (`MeetingPlace`)
- Date of the meeting (`MeetingDate`)

- People participating in the meeting (`Speaker`)
- Topics of the conversation (`Topic`)
- Keywords uttered by meeting participants (`Keyword`)
- Names or types of documents used during the meeting (`Document`)
- Dialogue act types used by participants (`DialogAct`), such as questions, statements, answers, etc.
- Larger sections of the meetings (`ArgSegClass`), such as decisions, suggestions, presentations, etc.

The user requirement study also suggested the *targets* of the queries. These were mostly meeting utterances and information associated with utterances (speaker, date, referenced document, etc).

The constraints and the targets of user queries allowed us to implement the domain knowledge model of the Archivus application (for general approach to domain modeling see section 3.2 on page 25). Due to a quite high complexity of the underlying meeting data, we have decided to implement the Archivus domain model in a form of PostgreSQL relational database. The database schema is in Figure 5.3. The corresponding domain model functions are implemented as a dynamic construction of SQL queries. The complexity of the dynamically created SQL query depends on the actual set of constraints that were submitted via the domain model function `setConstraints`. Solely the constraints actually provided by the user are a part of the SQL query used for selecting the central information in the database – a compatible set of utterances (table `Utter`). The set of utterances compatible with the search constraints is then in turn used to generate compatible values for any given attribute (domain model function `getCompatibleValuesForAttribute`). For instance, when a year of meeting is a constraint, the compatible utterances are only those that happened during meetings held in the given year. In case when the user needs to know which persons were discussing in the given year, such persons can easily be identified as speakers of utterances selected in the previous step.

Although the outlined principle of SQL query construction is valid for most of the attributes of the domain model (see Table 5.1), there are some exceptions. For instance, the compatible values for bookcase dimension selectors are constantly the same (i.e. the set of possible selectors), or values of `History` attribute are basically the actual search criteria. In this respect, the `Bookcase` attribute is also specific: the values of `Bookcase` are always all meetings (both active and inactive), organized according to currently selected values of dimension selectors (vertical and horizontal). The corresponding SQL query returns all meetings (augmented with information about number of utterances compatible with search constraints) associated with information specified by the dimension selector type.

The SQL query execution time depends on the complexity of given query, varying from 40ms to 500ms. We consider such response times as sufficient for our application and database size. In case of bigger databases (hundreds of meetings), the database response times can be optimized by running the database on a separate machine or using database optimization methods (adding memory, storing the database on multiple disk volumes, etc). Another room for optimization lies in caching of frequently used parts of SQL queries (e.g. compatible utterances, results for bookcase, etc).

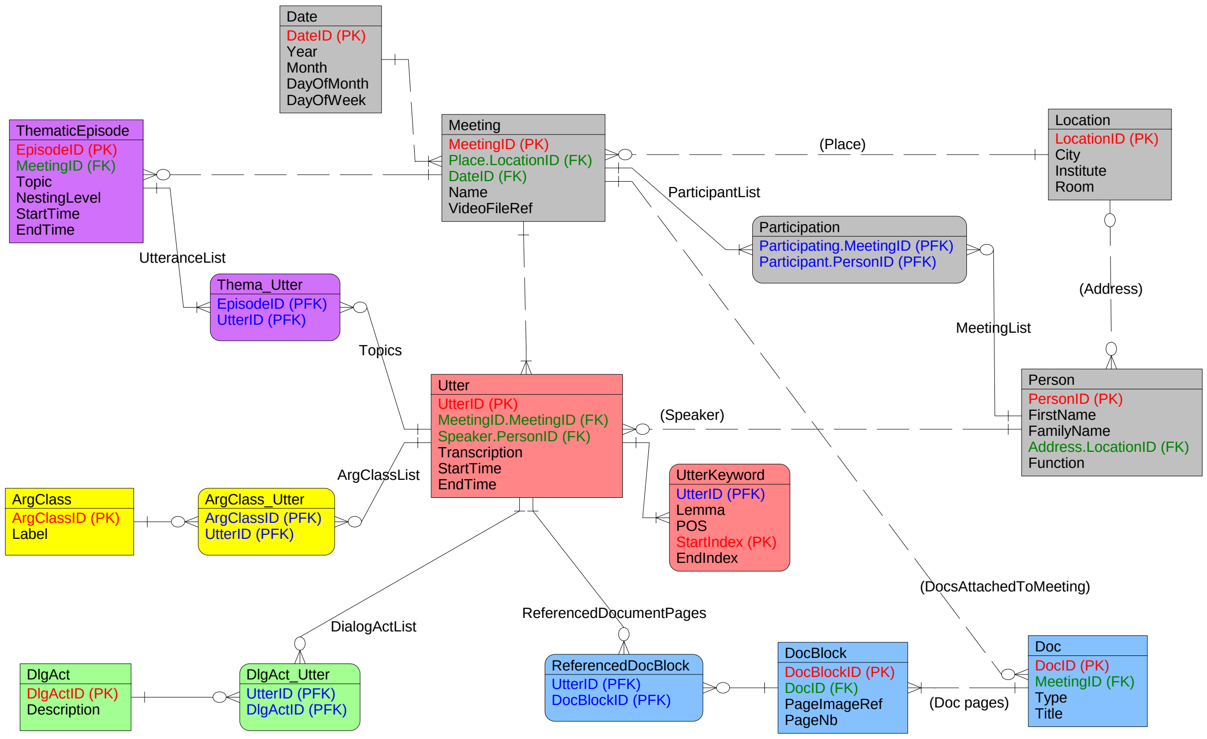


Figure 5.3: Domain model: an underlying schema of a relational database used to store meeting data. The color code is used to denote different annotations on the data.

5.2.3 Structure of mGDNs

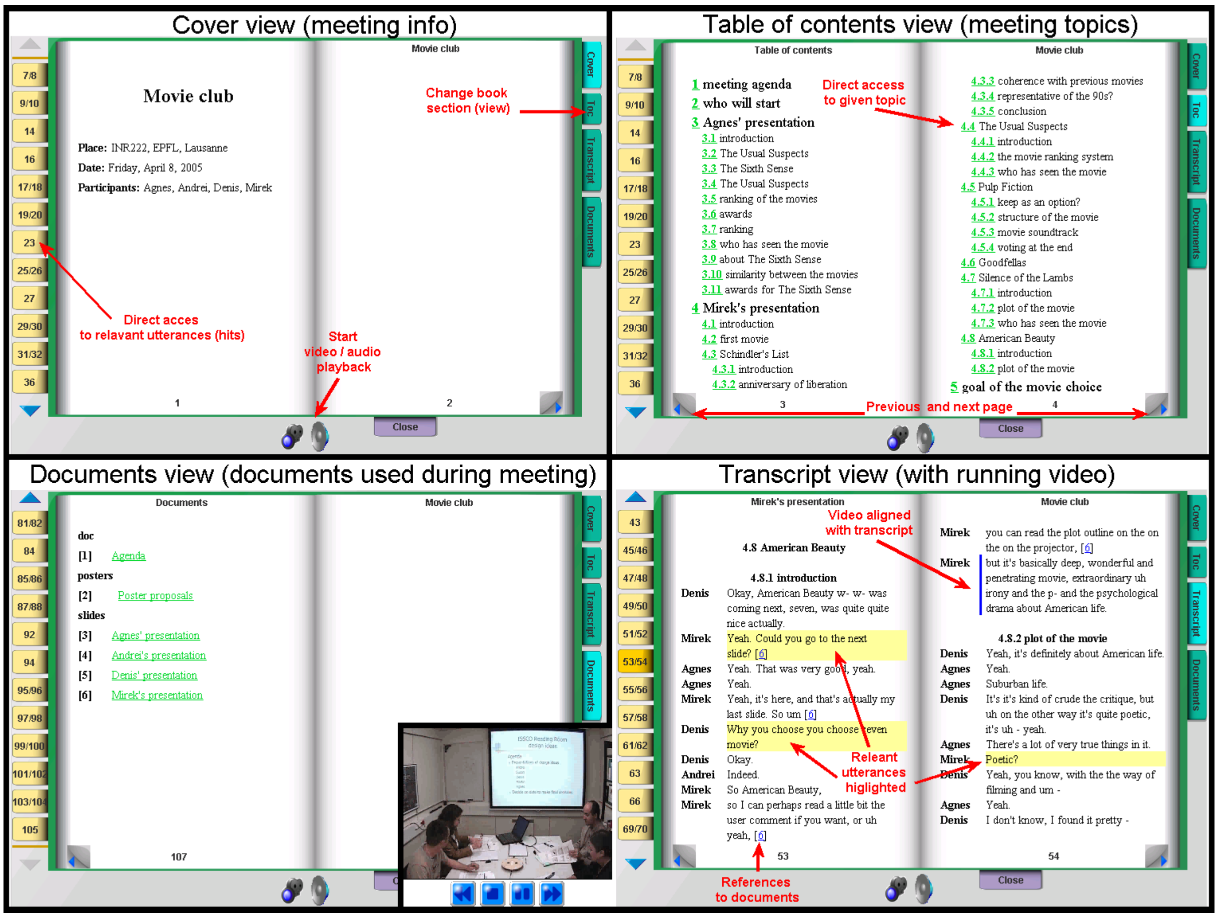
In order to define *how Archivus interacts* with the user, the attributes of the domain model were associated with mGDNs and transitional relations between the mGDNs were established. The mGDNs (section 3.3 on page 27) define the interaction in a specific context, while the structure of mGDNs is exploited by dialogue strategies to control the interaction with the user at a global level. The relation between the mGDN structure and the dialogue strategies is explained in section 3.5 on page 43.

The exact association of mGDNs with domain attributes is shown in Table 5.1. Nearly all mGDN types were already introduced in section 3.3.2 on page 29, except of the book browser mGDN. This is because the formerly mentioned mGDNs are regarded as general-purpose mGDNs, while the book browser is specific to Archivus, though it could be generalized to information seeking systems that use the book metaphor to present the sought information.

The book browser is quite a complex mGDN with multiple layouts (Figure 5.4), allowing the user to access various information about meetings. The entire meeting is always displayed, but the user can quickly access pages containing information relevant to the current search criteria using yellow tabs on the left side of the book. Additionally, the relevant information (utterances) on the page is highlighted in the transcript using a yellow background. Opening a meeting book is unfortunately quite a lengthy process (on average 6 seconds, ranging from 2.5s to 13s depending on meeting size), because all the book pages have to be generated before the book is displayed to the user (otherwise it would not be possible to display correctly yellow hit tabs and page references in table of contents). A progress bar appears when the book is being generated. In order to speedup

Attribute name	*Attribute type*	*mGDN type/layout*	*GUI area*
`Start`	initial	empty layout	3
·`MeetingPlace`	composite constr.	map layout	3
··`MeetingCity`	constraint	list	3
··`MeetingInstitute`	constraint	list	3
··`MeetingRoom`	constraint	list	3
·`MeetingDate`	composite constr.	table layout	3
··`MeetingYear`	constraint	list	3
··`MeetingMonth`	constraint	list	3
··`MeetingDayOfWeek`	constraint	list	3
··`MeetingDayOfMonth`	constraint	list	3
·`Speaker`	composite constr.	table with icons	3
··`SpeakerFirstName`	constraint	list	3
··`SpeakerFamilyName`	constraint	list	3
··`SpeakerAddress`	composite constr.	table layout	3
···`SpeakerAddrCity`	constraint	list	3
···`SpeakerAddrInstit`	constraint	list	3
···`SpeakerAddrRoom`	constraint	list	3
··`SpeakerFunction`	constraint	list	3
·`Content`	composite constr.	switch layout	3
··`Topic`	constraint	list	3
··`Keyword`	constraint	list	3
··`Document`	composite constr.	table	3
···`DocTitle`	constraint	list	3
···`DocType`	constraint	list	3
·`DialogueElements`	composite constr.	switch layout	3
··`ArgSegClass`	constraint	list	3
··`DialogAct`	constraint	list	3
`History`	constraints overview	special list	2
`Bookcase`	search space overview	bookcase	1
`BookcaseVerticalDim`	view arrangement	list	3
`BookcaseHorizontDim`	view arrangement	list	3
`OpenBook`	result browsing	book browser	3
`DocBrowser`	result browsing	document browser	3

Table 5.1: Archivus: a summary of domain model attributes with associated mGDN type and the placement on user interface (GUI area, referring to Figure 3.11 on page 49). A hierarchical organization of constraint attributes is denoted by indentation (using dots).


Cover view (meeting info)
Change book section (view)
Direct acces to relavant utterances (hits)
Start video / audio playback
Table of contents view (meeting topics)
Direct access to given topic
Previous and next page
Documents view (documents used during meeting)
Transcript view (with running video)
Video aligned with transcript
Releant utterances higlighted
References to documents
Cover
Toc
Transcript
Documents
Close

Movie club

Movie club

Place: INR222, EPFL, Lausanne
Date: Friday, April 8, 2005
Participants: Agnes, Andrei, Denis, Mirek

Table of contents

Movie club

1 meeting agenda
2 who will start
3 Agnes' presentation
3.1 introduction
3.2 The Usual Suspects
3.3 The Sixth Sense
3.4 The Usual Suspects
3.5 ranking of the movies
3.6 awards
3.7 ranking
3.8 who has seen the movie
3.9 about The Sixth Sense
3.10 similarity between the movies
3.11 awards for The Sixth Sense
4 Mirek's presentation
4.1 introduction
4.2 first movie
4.3 Schindler's List
4.3.1 introduction
4.3.2 anniversary of liberation
4.3.3 coherence with previous movies
4.3.4 representative of the 90s?
4.3.5 conclusion
4.4 The Usual Suspects
4.4.1 introduction
4.4.2 the movie ranking system
4.4.3 who has seen the movie
4.5 Pulp Fiction
4.5.1 keep as an option?
4.5.2 structure of the movie
4.5.3 movie soundtrack
4.5.4 voting at the end
4.6 Goodfellas
4.7 Silence of the Lambs
4.7.1 introduction
4.7.2 plot of the movie
4.7.3 who has seen the movie
4.8 American Beauty
4.8.1 introduction
4.8.2 plot of the movie
5 goal of the movie choice

Documents

Movie club

doc
[1] Agenda
posters
[2] Poster proposals
slides
[3] Agnes' presentation
[4] Andrei's presentation
[5] Denis' presentation
[6] Mirek's presentation

Mirek's presentation

Movie club

4.8 American Beauty

4.8.1 introduction

Denis Okay, American Beauty w- w- was coming next, seven, was quite quite nice actually.
Mirek Yeah. Could you go to the next slide? [6]
Agnes Yeah. That was very good, yeah.
Agnes Yeah.
Mirek Yeah, it's here, and that's actually my last slide. So um [6]
Denis Why you choose you choose seven movie?
Denis Okay.
Andrei Indeed.
Mirek So American Beauty,
Mirek so I can perhaps read a little bit the user comment if you want, or uh yeah, [6]
Mirek you can read the plot outline on the on the on the projector, [6]
Mirek but it's basically deep, wonderful and penetrating movie, extraordinary uh irony and the p- and the psychological drama about American life.

4.8.2 plot of the movie

Denis Yeah, it's definitely about American life.
Agnes Yeah.
Agnes Suburban life.
Denis It's it's kind of crude the critique, but uh on the other way it's quite poetic, it's uh - yeah.
Agnes There's a lot of very true things in it.
Mirek Poetic?
Denis Yeah, you know, with the the way of filming and um -
Agnes Yeah.
Denis I don't know, I found it pretty -

Figure 5.4: Book browsing mGDN: user can browse the meeting as a book. Several views on meeting data are available.

the book opening, we have implemented a cache for displayed data. In case when the user reopens the same book and no hits were changed, the response is nearly immediate (about 350ms).

The book browsing mGDN provides a multimedia player that is used to play the video and audio recorded during the meeting. Video is visually synchronized with the meeting transcription – a small blue vertical line is displayed next to the utterance that is being played. The player is tuned to start the recordings at the page from which it was started, giving users easy and quick access to very specific points in the meeting. The meeting book contains also references and list of all documents used during the meeting. The document browser is actually another mGDN, which is accessed immediately after a document was selected within a book. The document identifiers are the values of global semantic pairs resulting from the interaction with the book.

Concerning the overall Archivus interaction model, the *constraint mGDNs* are organized hierarchically as indicated in Table 5.1. The explicit hierarchical organization is encoded in a declarative config file and makes it possible for the branching logic (global dialogue strategy) to determine the next mGDN under focus after a value for the current constraint mGDN has been provided.

On the other hand, the relations between mGDNs participating in the *result browsing phase* of the dialogue (book and document browsers, search space overview mGDN and its horizontal and vertical arrangement mGDNs) are not encoded declaratively in the config file. Those are only implicitly encoded in the Java source code of the dialogue manager (within the main loop of the `interactWithUser()` method) together with commands for other dialogue strategies. Although the transition rules for 'result browsing mGDNs' represent in total less than 50 lines of Java source code (split in three blocks), we are conscious that an explicit formalism for relations between mGDNs is desirable. Such a formalism is currently missing in the implementation of our methodology and we regard it as future work. The reason for which we have not yet defined any explicit formalism for all relations between mGDNs is that we had not enough time (and examples of systems) to define it properly. Furthermore, we believe that a formalism with a weak expressing power could have caused the system designer more complications when reaching its limits. The adaptation of an insufficient formalism is more complex compared to the current need of adaptation of relatively few lines of existing Java source code.

5.2.4 Graphical layout of the application

We have organized the graphical components of mGDNs according to the principles postulated in section 3.5.3 on page 48. The visibility of various components is directly controlled by the mGDNs according to theirs status. Most of the graphical components become visible when their controlling mGDN is under focus. The only exceptions are the bookcase and history components – those are visible all the time.

Yet another aspect that influenced the graphical layout are the input and output modalities used in the Archivus system. In particular, we planned to use a 15 inch touchscreen. This imposed that all the control elements had to be big enough to be comfortably selected with finger-touches (the precision of touchscreens is lower than the precision of mouse pointing).

5.3 Improvements and modifications during pilot studies

The design of Archivus was an interactive process with many system modifications during the initial series of WOz experiments. The initial experiments were split into three main series, each lasting about five days and involving altogether 40 participants. An experiment with each participant took about two hours, hence allowing having a maximum of four participants a day. The number of participants could have been lower in case if the aim of the pilot studies was only to improve Archivus. However, the studies served the purposes of two other dissertations and thus had multiple aims, such as finding an optimal modality-unbiased experimental protocol, testing the hardware setup of the experimental environment, improving the wizard's control interface and the entire prototyping methodology. The overall aim of the pilot experiments was the preparation of the system and WOz environment for the data-collection experiments that are described in Chapter 6.

This section presents the major improvements and modifications of the Archivus system during the pilot studies in the form of the experience report, which is valuable for any system designer. It illustrates the possible outcomes of WOz experiments related to modifications of the designed system and clearly shows their importance. Because some of the necessary improvements formed our ISPM, this section can also be regarded as a justification for some of the ISPM design principles.

5.3.1 System outputs

Presentation of the system outputs underwent a number of modifications based on the informal post-experimental interviews with users. The initially used *prompts* were considered long, not very informative and too much repetitive. We have therefore carefully reconsidered wording of the prompts defined in the system, eliminated certain ambiguities in their meanings and implemented a possibility of defining multiple *prompt reformulations* that are randomly selected by the system (mGDN) in recurring situations. In specific situations (e.g. scrolling through the list of items) the consequent prompts are muted (no vocal output, only textual output). We have also extended our methodology with *conditional prompts*. Each conditional prompt is associated with a logical expression (condition) describing the situation in which this prompt should be used. If the condition is satisfied, the associated prompt replaces the default one.

The user-perceived system output is also largely influenced by the quality of *vocal* output. Initially, we have used the FreeTTS[2] solution – a text-to-speech synthesizer entirely written in Java and distributed free of charge. Nevertheless, in order to reach more naturalistic vocal output, we have integrated a commercial Nuance TTS[3] system, which is easy and pleasant to listen even repeatedly. The prototyping platform also offers a possibility of using fully prerecorded prompts, but this feature was not used in the Archivus system, as Nuance TTS is more practical and with a sufficient quality.

As a consequence of improved prompt quality (both content and audio form), we have observed that users hardly ever turned the audio output off during the experiments, which was not the case during the initial runs of experiments. Ensuring that the users naturally leave the prompts on was important for an evaluation of natural language use – we speculate that muted system prompts could discourage users from using their own voice for controlling the system.

Concerning other system outputs, we had to tune the *system confirmations* of newly acquired constraints. Our previous experience with vocal-only systems shows that some kind of confirmation is necessary, but vocal confirmation in systems equipped with screen is negatively perceived by users (long prompt, redundant with information displayed on screen). As a result, we have removed the confirmation prompt, but we attract the user's attention to the newly added constraints visually: they are flashing on screen for a short time (2.2 seconds) after they were recognized by the system. Similarly, we attract the user's attention to changes in the search space: each time there is a change in the result set, the relevant books (i.e. meetings containing sought information) in the bookcase are moving for a while. Those changes minimized the number of users repeatedly specifying the same constraints as they did not notice the effect of previously provided constraints.

[2]FreeTTS version 1.2.1: `http://freetts.sourceforge.net/`

[3]Nuance Vocalizer version 3.0.8: `http://www.nuance.com/realspeak/vocalizer/`

5.3.2 Functionality and usability improvements

During the pilot experiments, a need for extended types of interaction emerged. An example of extended interaction is a *map layout* for geographical location selection that replaced the initially used table layout. The list layout (Figure 3.1 on page 31) was found impractical for long lists and therefore we have introduced a possibility of jumping to a predefined part of the list using *alphabetical buttons*.

Other functional modifications concerned the refinement of the *hierarchy* of constraint mGDNs, with the aim of grouping together conceptually similar elements and enabling an option of using *multiple constraints of the same type* for searches (typically required for keywords). On the database side, the *labels* of some annotation types (dialogue acts and argumentative annotation) were reformulated in order to become more intuitive for non-expert users, and the labels were unified (such as lemmatization of keywords or turning the topic labels into singular forms without articles) in order to present the constraints in a coherent way.

Because the initial pilot experiments were carried out without tutorial, we could observe how intuitive the interface seems to be to users. From this perspective, the original label *"History"* of the mGDN for manipulation with search constraints was misleading, because users provided us with at least three different wrong interpretations of the purpose of that particular mGDN during the final interview. The wrong understandings were the following: (1) the component just enumerates all previous searches, (2) it provides an 'undo' functionality – when clicked somewhere in the middle, it rolls back to this point of interaction, (3) it gives a direct access to the displayed piece of information. Such misunderstandings were eventually resolved by the unambiguous label '*Current search criteria*'.

Another misleading functionality was triggered in the column layout (Figure 3.5 on page 34), where many users wanted to scroll down the columns. Therefore we have highlighted the hyperlink to the underlying mGDN (label '*...more...*'), where scrolling was possible. The overall intuitivity was also enhanced by a slight reorganization of the interface layout (compare Figures 5.1 and 5.2) and by using a new system graphics (the previous one was regarded as boring and old-fashioned by users during the final interview).

5.3.3 Book view

Although the book view follows a real-life metaphor, its mapping to meetings is a fairly novel approach. As such, it required several modifications and fine-tuning during the pilot experiments.

We have modified the labels on the tabs (see final version in Figure 5.4) in order to make them more intuitive: *'Contents'* become *'Toc'*, *'Appendix'* become *'Documents'*, and we have added a tab for the actual content of the book (*'Transcript'*). On the other hand, we have completely removed the list of keywords in the book (*'Index'*) as this was typically too long and not very usable. Besides highlighting the relevant utterances in yellow, we have also decided to highlight particular keywords in orange color as it was sometimes unclear to users why a given utterance was marked as relevant.

A functional enhancement was required in the part of the book that presents the meeting content in the form of a hierarchical list of topics. The initial prototype listed the topics, but did not allow directly accessing the corresponding part of the meeting. This was changed by adding an appropriate local dialogue strategy (responding to utterances like *"Go to topic 1.4"* and displaying hyperlinks next to each topic).

The system behavior was also improved in less frequently occurring situations, which were actually revealed only during WOz experiments. For instance, we had to decide how to behave in situations when the user (during book browsing) adds vocally a new search constraint, causing the currently selected book to become inactive, while another one remains active. We have considered the possibility of directly switching into that active book, but finally we have decided to vocally inform the user that the currently selected book does not contain any relevant information anymore, and it was up to the user to decide whether he/she wants to continue browsing the current inactive book, modify search constraints or to switch to the active book.

5.4 Summary

Designing a dialogue system with multiple input and output modalities is always a challenging task, even if methodological guidelines and predefined software building blocks are provided. In this perspective, concrete experience with building similar systems is of invaluable help.

Therefore, we present the full design process of such a system – the Archivus system. As the Archivus applicative goal (retrieving information from a multimedia meeting database) corresponds to a quite novel application, it is thus difficult to predict the behavior of the users when faced with the associated multimodal interface. We present and justify our design decisions and discuss the changes that were made during the iterative system refinement. Some of those changes might be regarded as fairly generic and might inspire the designers of future similar systems.

Chapter 6

Evaluation of dialogue strategies and the Archivus system

This chapter focuses on several evaluations based on the data collected during large-scale WOz experiments carried out with the Archivus system. These experiments involved 91 naïve users and were made with the most mature version of Archivus after several pilot experiment runs (the pilot experiments involved additional 40 users and the improvements made to the Archivus system during these experiments were summarized in section 5.3).

The main goals of this chapter are (1) to analyze how the various strategies and features implemented in the interactive system designed with our methodology are used and perceived by users in order to derive conclusions about their relative importance, and (2) to evaluate the quality of the Archivus system, from the perspective of user performance, satisfaction, and encountered problems.

The large-scale Wizard of Oz experiments were carried out in a collaborative framework with colleagues from the University of Geneva and the EPFL. The experimental protocol builds upon several groups of users with different access to the Archivus modalities, allowing an inter-group, modality-based comparison. However, for our specific evaluation purposes, we only took into account the PVK (pen, voice, keyboard) and MVK (mouse, voice, keyboard) conditions and we balanced our data in order to have the same number of users within each modality condition (see section 6.2 for details). No inter-group comparison was performed in our case, because the variable 'modality condition' which defines the groups is not in the main focus of our analysis.

The chapter is organized as follows: first, the experimental setup and the gathered data (users' demographic information and global interaction measures) are described (sections 6.1 and 6.2). Then, the use and significance of the global and local dialogue strategies is evaluated (sections 6.3 and 6.4). The use of various modalities and their role in the interaction is described in section 6.5. Next, we evaluate the Archivus system as such. In particular, we look at objective measures such as the speed and accuracy of the users when solving the assigned tasks, and we analyze the causes of problems encountered by the users during their interaction with the system (section 6.6). Finally, we provide an overview of user's subjective opinions about the Archivus system (section 6.7).

6.1 Experimental setup

The user (the experiment volunteer) was first welcomed and briefly introduced to the experimental evaluation by an experimenter. During the introduction, the users were warned that the experiment is recorded, but they were *not* informed about the presence of wizards (who sat in a separate room). After having signed a consent form, the users had to fill out a demographic questionnaire and then received detailed information about the experiment together with a description of the applicative scenario (*"Imagine you are a new employee of a company. Your manager asked you to find and check some information for him. He has told you to use the Archivus system that ...*").

The actual experiment was split into two twenty-minute sessions, during which the users solved tasks provided to them on paper cards. The tasks included true-false questions (statements) like *"The budget for the room furnishing was 1000CHF*" and short-answer questions like *"Who attended all meetings?*". As the original main purpose of the experiments was to identify the most useful modalities in Archivus and determine whether there is a modality-learning effect (see [57]), only a subset of all input modalities was available to the users during the first session (for example voice only, or pen and typed natural language only). In the second session, which immediately followed the first one, the users were allowed to freely use any of the modalities available in the system (speech, typed natural language, mouse or pen), with exception that pen and mouse were never allowed simultaneously. Table 6.1 shows all the combinations of modalities that were made available to the users. In order to familiarize users with the system and to introduce the modalities available to them, each twenty-minute session was preceded by an approximately fifteen-minute tutorial. The tutorial took the user through three sample tasks (similar to those that were given on paper cards) and explained how to control each part of the system with the available modalities using a guided interaction.

Group	*Modalities available during*		*Number of users*		
	first session	*second session*	*total*	*discarded*	*analyzed*
G1	M	MVK	9	1	8
G2	P	PVK	10	2	8
G3	MK	MVK	9	1	8
G4	PK	PVK	8		8
G5	MV	MVK	9	1	8
G6	PV	PVK	8		8
G7	PVK	PVK	11	3	8
G8	MVK	MVK	10	2	8
G9	V	MVK	5		8
G10	V	PVK	3		
G11	VK	MVK	6	1	8
G12	VK	PVK	3		

Table 6.1: Experimental setup: modalities (the modality condition) available to users during the first and second experimental sessions. In order to guarantee a uniform distribution of experimental conditions per users, some of the users were not considered for further analysis. Acronyms used: M – mouse, P – pen, V – voice, K – keyboard

At the end of the experiment, the users were asked to fill out a post-experimental questionnaire, the goal of which was to elicit their overall opinion of the system. The experimenter was only present in the room when instructing the users about the next evaluation steps and each user was left alone in the room when doing the two experimental sessions and tutorials. This helped to keep the experimental conditions constant and the experimenter could also act as an output wizard, thus reducing the overall number of operators needed during the experiments.

The experiment participants (naïve users) were recruited through flyer advertisements distributed on three Swiss universities (EPFL, University of Lausanne and University of Geneva). Volunteers were asked to contact the experimenters by email and were awarded 20 CHF (about 13 EUR) and a free drink for their participation in a two-hour experiment at EPFL or at the University of Geneva.

6.2 The gathered and analyzed data

Several types of data were gathered during the experiments: first (1) user's *personal information* collected from the demographic questionnaire and (2) user's *subjective opinions* of experience with the system from the post-experiment questionnaire. Next is (3) the *audio and video* recordings (in three channels: user's desktop, hands and face view) of each of the twenty-minute sessions, and (4) the *system logfiles* that keep track of all system actions and allow to determine user actions, such as where the user pointed, what he/she typed, what modality he/she used, etc. After the experiments, the logfiles were manually annotated and partially corrected using the video files (e.g. added transcription of user's utterances, corrected starts and ends of user tasks and aligned with the task IDs). The interactions during the tutorial were not recorded, as the users were supposed only to strictly follow the tutorial instructions. In addition, during the experiments, the wizards took (5) informal notes concerning any specificities that could possibly bias the collected data (for instance technical problems, user not willing to finish the tutorial, user performing tasks out of the given order, user saying he/she was ill, etc). The last type of data gathered were (6) the user's answers to the given tasks (true-false statements and short-answer questions).

Although we gathered the six types of data for all 91 users, we only use a part of them for the analysis performed in this chapter. Indeed, as already mentioned, the experimental setup was originally designed for the study of modality-learning effects and for intermodality comparison [57]. For our work, we only use the data from the second twenty-minute experimental sessions, during which all modalities were available to the users (with the exception of pen and mouse, which were never available simultaneously). The used data was selected in order to be balanced with respect to modalities available in the first twenty-minute sessions and we preferentially excluded users who experienced non-standard experimental conditions (e.g. users announcing that they were ill, users who solved tasks out of given order, etc). An overview of included and excluded data is given in Table 6.1.

In consequence, if not stated otherwise, the analyses presented in this chapter build upon the experimental data of 80 participants interacting with the system during the second experimental session (altogether 80 × 20 = 1600 minutes of interaction). The next two sub-sections (6.2.1 and 6.2.2) provide a more detailed presentation of the analyzed data.

6.2.1 The users – a demographic information

Among our 80 experiment participants, we had a fairly even distribution of males (58.75%) and females (41.25%), who ranged in ages from 18 to 55. Of these, just over half (53.75%) were aged between 18 and 24, 37.5% were between the ages of 25 and 35, 6.25% were between 36 and 45, and the other 2.5% were between the ages of 46 and 55. Of these, 85% were non-native speakers of English with a self-assessed good level of reading and speaking skills in English. Most of the volunteers (70%) were university level students. The others came from a variety of different professions ranging from researchers to engineers or translators. Nevertheless, 38.75% of them attended meeting a few times a month, with 12.5% attending a few times a week, and 5% attending meetings every day. Only 7.5% said that they never attended meetings and 21.25% said that they attended them only a few times a year.

Computer literacy was self-assessed by the volunteers. One of the questions that they were asked was how many hours they spent using a computer each day. 38.75% said that they spend 2 to 4 hours with a computer, 23.75% said it was 5 to 7 hours, and 21.25% said it was 7 to 10 hours. A surprisingly large number (13.75%) said that they spend over 10 hours each day in front of a computer. Only 2.5% said that they spent less than an hour. An overwhelming number of our users (95%) used Windows systems on a daily basis and were familiar with browsers (98.75%). Slightly fewer used word processors (90%) and audio players (82.5%) regularly. Only 63.75% of the users used video players regularly. A surprisingly high number of users (33.75%) said that they had used automatic speech recognition programs in the past and 23.75% said that they had used voice to control their computer. However, only 18.75% had experience with database tools.

We feel that in general, this user population is fairly representative of a population that could use multimodal dialogue systems of the given type.

6.2.2 Global interaction measures

The analyzed experimental sessions of 80 participants are on average 19 minutes and 58 seconds long, representing altogether 1598 minutes of interaction (see Table 6.2 for average, standard deviation, minimum, maximum and sum of most of the hereafter mentioned measures). During that period, users made 6641 dialogue turns (inputs to Archivus) in total. On average, this represents 83 dialogue turns per session, and in terms of speed, it is one user input every 14.4 seconds. Actually, the users were providing their inputs within 10 seconds on an average, and 4.5s is the average system response delay. The system response delay is caused partly by the wizards simulating or supervising some of the system functionalities (2.8s) and partly by the system itself (1.7s).

An average duration of system voice prompt[1] is 3.3s. The prompts are played vocally only in some of the dialogue turns (exactly in 3579 instances, which represents 54% of all

[1]The user can provide his/her input in the time when the system plays the voice prompt (barge-in). Therefore, such time period is included within the durations of user turns (and is not considered as a

Interaction measure	*# of data*	*Avg*	*Dev*	*Min*	*Max*	*Sum*
Duration of sessions	80	19m:58s	±25s	18m:29s	21m:11s	1598m
# of turns in sessions	80	83	±24	41	181	6641
User turn duration	6641	10.2s	±10.5s	0.1s	2m:41s	1125m
System turn duration	6641	4.5s	±5.3s	0.1s	37s	495m
Pure system duration	6641	1.7s	±2.9s	0.1s	14s	191m
Wizard processing dur.	6641	2.8s	±3.4s	0.1s	32s	304m
Prompt playback dur.	3579	3.3s	±1.3s	0.3s	18s	194m
# of tasks per sessions	80	10.5	±3.2	5	20	837

Table 6.2: Global interaction measures

dialogue turns). Altogether, 119 distinct vocal prompts were played by the system and the vast majority of them (117) were played less than 80 times (i.e. less than one times in a twenty-minute session), indicating that prompts are not very repetitive within one experimental session.

The experiments were carried out with two different modality conditions. In one of the condition, a tablet PC with tactile screen was used, enabling pen, voice and keyboard as possible input modality channels (hereafter called the PVK condition). The second hardware settings used a desktop PC with 15 inch monitor, mouse, voice and keyboard input (the MVK condition). We had 42 users following the MVK experimental condition, while 38 users followed the PVK condition. Unfortunately, the tablet PC was slower than desktop PC, making the overall average system responses longer by 1.8s. This resulted into a lower average number of dialogue turns performed during allocated 20 minutes (73 turns in PVK versus 92 turnd in MVK). In total, our data set contains 2759 (42%) dialogue turns from PVK condition and 3882 (58%) turns from MVK condition. We take this into account in analyses where such a disproportion might have an impact on the results.

Overall, we assume that the collected data is exhaustive enough to be representative of possible interactions that a great majority of users in a population will have with the Archivus system (within similar interaction scenarios). It allows us to draw some general conclusions on the use of dialogue strategies and modalities in Archivus and its overall usability.

6.3 Evaluation of global dialogue strategies

6.3.1 Automated presentation of possible search results

The Archivus system *automatically opens a book* with a meeting transcript in cases when *only one* book (meeting) contains information satisfying the search constraints provided by the user. Information matching the user's query is highlighted in the transcript and the user has a possibility of directly browsing the relevant pages using book tabs (the book tabs are shown in Figure 5.4 on page 89). The book opens automatically on the

system processing time – the system processing time is solely the time period when the user cannot interact with the system).

first page with relevant information. Further details about this global strategy are given on page 45. We were interested in *how often the strategy was used* and *whether it was accepted by user.*

The analysis of experiment logfiles revealed that a book was opened automatically by the system in 949 cases (43% of all book openings), while the user opened the book manually in 1268 cases (57% of all book openings). In total, the strategy of automatically presenting a possible search solution/result (i.e. utterances within meeting book) represents 7% of all interactions (dialogue turns[2]) with the system, making it a fairly frequent dialogue strategy.

As the strategy is frequently used, it is crucial to know whether the corresponding automated decision of the system is *accepted by users* or not. Because users were not explicitly asked about that issue in the post-experiment questionnaire, we had to establish measures (based on objective data in logfiles) in order to estimate the acceptance of the system strategy.

The first measure is a heuristic defining that the user is unsatisfied with the automatically opened book in situation if he/she closes the book (or accesses some other element on the interface) within a 'short time' after the book was automatically opened. Being aware of high subjectivity of the term 'short time', we rather define it as three potential durations[3]: 3s, 5s and 8s. The results are summarized in Table 6.3. Although the number of cases of 'quickly closing the book' changes with the definition of 'quickly' (short time), it can be seen that *a great majority of users examined the information automatically proposed by the system* (in 90% of cases, the users spent more than 8 seconds with a review of the proposed information).

An interaction time with automatically opened book	*# of cases*	*% of cases*
<3s	18	1.9%
<5s	50	5.3%
<8s	98	10.3%

Table 6.3: Acceptability of dialogue strategy for automated presentation of possible solutions: the table shows the number of cases when users closed the automatically opened book within the given time interval. The percentage fractions are with respect to all situations when a book was opened automatically (i.e. 949 cases)

The second measure compares the overall user behavior in situations (1) when the book was opened automatically by the system and (2) when it was opened manually by the user. In particular, we compare the number of dialogue turns and time spent in the book before it was closed (or another element on the interface was accessed). The underlying hypothesis is that if users spent at least the same amounts of time and dialogue turns in the automatically opened books as in the manually opened books, we can consider the

[2]We have chosen the dialogue turns to be the base numerator of the frequency-of-use of dialogue strategies, because one dialogue strategy is always triggered during every dialogue turn.

[3]The duration measures the real time that user spent in the given book (or documents associated with that book) before leaving it. In particular, it includes the time when the system is actively waiting for user's input and excludes the durations of the system processing time, such as the durations caused by wizards, document and video opening times, etc. The duration of the system prompt playback (which is on average 3 seconds) is included, as the prompts can be barged-in.

automated decision as to be accepted by users. The results are summarized in Table 6.4. Apparently, users spent more time in the books automatically opened by the system and they also performed slightly more dialogue turns in such cases. That indicates that the automatically opened books contained information relevant to the search query and that the users invested some time to carefully review it.

How was the book opened?	*avg duration*	*avg # of turns*
automatically by system	50s	3.8
manually by user	36s	3.6

Table 6.4: A comparison of time spent in a book (meeting view) and number of performed dialogue turns, depending on the way of opening the book (automatically or manually).

Based on our experimental results, we can **conclude** that the proposed strategy for an automatic presentation of possible search results (books) is *frequently used* in the Archivus system and *well-accepted* by users.

In consequence, we **recommend** that a multimodal information seeking system should automatically propose possible search results as soon as their number is sufficiently low to be efficiently displayed, provided that the system has enough evidence about the expected relevance of these results. Nevertheless, automatically presenting information should neither prevent users from having the possibility to further browse the search space nor to enter new search constraints.

6.3.2 Dead-end management

The dead-end dialogue management strategy is invoked in situations when a user request is over-constrained, i.e. when no solution targets are compatible with current search constraints. The strategy is described on page 46.

The dead-end situation represents 3.1% of all dialogue turns and 3.6% of all interaction time was spent in dead-ends.

As the user has an option of erasing *only one* (possibly conflicting) search constraint or *all constraints together* (using clear or reset command), we wanted to know *which option is more frequently used* by users. We found that users resolved the over-constraint situation in 57% of all dead-end cases by careful removal of one search constraint (or more, but one-by-one), while in 43% they erased all constraints all together (in 33% by the system reset and in 10% by using the clear button in the history). This result shows that there is not one unique approach preferred by users and therefore *systems should offer both options* (erasing constraints one-by-one and all together) for resolving the over-constrained situation.

Secondly, we wanted to know *whether users immediately understood the problem of an over-constrained situation.* We have observed that in 14% (32) cases of over-constrained situations, users attempted to add new search constraints. As adding a new search constraint to already over-constrained situation is completely irrational action, we assume that the users were not aware of the problem in those cases. Although the absolute number of ignoring the dead-end is not very high, it is still quite surprising considering the indications that are given to users: it is indicated by a vocal prompt[4] and the incompatible constraints are displayed on the screen in red color with an exclamation mark. This finding suggests that indication of problematical situation should never be underestimated by system designers and that users are not necessarily always paying full attention to the system prompts.

In conclusion, we **first recommend** that user-friendly constraint-based search systems should provide at least two ways of resolving over-constrained situations: (1) by erasing only a specific criterion and (2) by erasing all search criteria simultaneously, as both of these approaches are frequently used by users. Moreover, it is important to remain aware of the fact that, for an incompressible proportion of over-constrained situations (14% in our case), the dead-end will be initially ignored by the users, regardless of the warning methods used (in our system, audio prompts and constraints in red with an associated exclamation mark were used). Therefore, our **second recommendation** is to very clearly indicate (possibly with redundant techniques) the over-constrained situations and possibly enable the system to generate suggestions for resolving them. Our hypothesis is that users sometimes have their own understanding of what to do with the system and they therefore tend to ignore nonspecific warnings(e.g. *"Your search criteria are incompatible"*). We expect that users will better react to more specific system suggestions concerning already provided user preferences (e.g. *"No such cars are available. But if you accept the price 15.000-20.000$ instead of 14.000$, then there is 7 of them. Is it ok to increase the price?"*). This type of suggestions was studied by our colleague [89].

6.3.3 Use of mixed initiative

Mixed initiative systems allow both the user and the system to control the interaction. In practice, mixed initiative is achieved by *permitting the user to specify more information than required by the system in a given dialogue context.* Archivus supports mixed initiative by processing the information (semantic pairs, SPs) intended for other mGDNs than currently in focus. In our approach, mixed initiative is not problematical at the dialogue management level (as the dialogue manager can process SPs independently of how they were generated), but the *complexity and robustness of the NL understanding algorithms* that generate SPs (based on user's NL input) strongly depends on the degree of mixed initiative allowed by a system.

During our experiments, the natural language understanding was simulated by wizards and hence we have permitted a full mixed initiative behavior of users. However, for fully automated systems, it is important to know *to which degree the mixed initiative input is naturally used by users* in order to correctly implement algorithms for natural language processing.

We use the following method to determine the frequency of mixed initiative use: the semantic pairs[5] resulting from the interpretation of the user's last NL input are categorized according to their relevance to the mGDN currently in focus (i.e. the mGDN controlling interaction at the time when user provided his/her input). Then, the number of SPs

[4]The warning system prompt in the `History` mGDN is *"Your search criteria are incompatible. Modify your selections."* and in the `OpenBook` mGDN the prompt is *"This meeting does not match your search criteria very well."*

[5]An overview of all semantic pair types defined in the system is given in Table 3.2 on page 42.

matching the context of the mGDN in focus is compared with the number of all other (mixed initiative) SPs. Because every binary decision about SP's relevance to a given mGDN can be questionable, we rather define the following four categories of SP relevance with respect to the mGDN in focus (the categories are ordered from lowest to highest required degree of mixed initiative):

- *(C1) Concerns the current mGDN*: represents a value (constraint) corresponding to the mGDN in focus, all local SPs concerning a local interaction within the mGDN in focus, a command for closing the focused mGDN and a request for system restart. This category also includes values corresponding to the individual components of a composite mGDN in focus (e.g. values of `SpeakerFirstName` when the `Speaker` mGDN is in focus).
- *(C2) Focus change request*: this category contains the SPs corresponding to the user-initiated focus change requests (that is SPs like '`global.newfocus:...`').
- *(C3) Concerns visible mGDNs*: A value (constraint) and local SPs concerning mGDNs that are visible on the screen together with another mGDN in focus. In case of Archivus, these SPs correspond to the values and the local semantic pairs of the `Bookcase` and the `History` mGDNs (but only when these are not themselves in focus).
- *(C4) Concerns other mGDNs*: all other SPs, in particular SPs representing the values and invoking the local strategies of mGDNs that are not in focus and are not visible on the screen.

A proportional distribution of SPs generated by user inputs according to the above defined mixed initiative categories is shown in Table 6.5. With the assumption that SPs related to C3 and C4 categories are considered as the mixed initiative SPs, one can observe that 44% of SPs generated by users required the system to support the mixed initiative. Such result suggests that *the mixed initiative is an important part of the dialogue management.* The percentage is even higher for inputs wrote by user on keyboard. This is because a majority (70%) of all typings happened in the `Start` and `OpenBook` mGDNs and was used to specify search criteria (which is considered as a mixed initiative in the context of these mGDNs).

Type of input	*Percentage of semantic pairs concerning*			
	current mGDN (C1)	*focus change request (C2)*	*visible mGDNs (C3)*	*other mGDNs (C4)*
spoken	50%	10%	9%	31%
typed	23%	1%	3%	73%
all natural	47%	9%	8%	36%
language	56%		44%	

Table 6.5: Use of mixed initiative: fractions of semantic pairs (generated by user inputs) with respect to their different relevance to the mGDN in focus. Percentages sums to 100% in each row.

Further analysis of *all* NL inputs revealed that the mixed initiative (of C4 category) was most frequently used in the `Start` (674 cases) and `OpenBook` (465 cases) mGDNs. Detailed results are given in Table 6.6. As it can be seen, the mixed initiative frequency varies depending on the mGDN in focus. For many mGDNs (denoted as 'others' in the

mGDN	*# of C4 SPs*	*SPs used (only when >10%)*
`Start`	674	46% – C4 `Keyword`; 17% – C3 `Bookcase`; 11% – C2
`OpenBook`	465	66% – C1; 17% – C4 `Keyword`
`Bookcase`(visual)	69	37% – C1; 34% – C4 `Keyword`; 12% – C3 `History`
`Topic`	35	49% – C1; 46% – C4 `Keyword`
`DialogueElements`	28	37% – C2; 28% – C4 `Keyword`
`History`(visual)	23	62% – C1; 32% – C4 `Keyword`
others	$\leq$6	71% – C1; 21% – C2

Table 6.6: A list of mGDNs where mixed initiative (of C4 category) was most frequently used. The last column shows distribution (in descending order) of SP categories used by users within the given mGDN. Categories C3 and C4 are augmented with the name of mGDN for which the SP is targeted. Percentages are calculated from all SPs generated with natural language when given mGDN was in focus.

figure), the mixed initiative was almost never used. The mixed initiative was frequently used in the mGDNs explicitly enumerated in the figure, but then the users are likely to address only some of the mGDNs out of the focus (if mixed initiative of category C3 or C4 is involved, the typically addressed out-of-focus mGDNs are: `Keyword` mGDN (most frequent), `Bookcase` and `History`).

From our experimental results, we can therefore **conclude** that *mixed initiative* is *frequently used* in Archivus, but its extent of use varies significantly over the different application contexts (mGDNs). A detailed analysis leads to the identification of the mGDNs where users are *not using mixed initiative*. Similarly, for the mGDNs where mixed initiative is used, the analysis can reveal *the nature of the mixed initiative used*.

In consequence, our **recommendation** is to always perform an analysis of mixed initiative use before integrating automated NL processing in the system. Indeed, the robustness of the considered NL processing usually strongly depends on the size and variability of the used vocabulary as well as on the complexity of the used syntactic and semantic structures. The mixed initiative use analysis then allows for the identification of the NL resources to be taken into consideration, thus increasing the robustness and reliability of the final automated system.

6.4 Evaluation of local dialogue strategies

This section provides a quantitative evaluation of the local dialogue strategies, i.e. the strategies that do not change the current context of the dialogue. The local dialogue strategies are implemented internally within mGDNs and they are defined as a part of an mGDN operational model in section 3.4.3 on page 40. In this section, we focus only on the local strategies commonly defined by all the mGDNs.

6.4.1 User requests for help

Users *virtually never asked the multimodal system for a help advice* although the corresponding dialogue strategy was constantly available and the availability was also visually indicated with a help button on the screen. In fact, users asked for help only three times (0.02% of all user inputs) during the entire experiments. In one case, the user asked for help because he did not know where to find a document, while the other two cases of help strategy were triggered by user's question *"Why?"* after the system announced that the action (previously requested by user) cannot be performed.

We assume that users did not use the help button because (1) a majority of them found the system easy to use (85% of all users), (2) users felt comfortable working with the system (78%) and (3) users felt in control of the system (56%), according to the questionnaire they filled after the experiment. In such settings, users naturally do not require help advices.

Another aspect that eliminates the help requests could be that the users have learned using the system with a *tutorial* (lasting 10–25 minutes, 15 minutes on average). The tutorial makes them understand all functional aspects of the interface. The only problematical situations occur when the system cannot understand the user. In such cases, users typically reformulate the request, as indicated by some of them in the final questionnaire:

- Participant P122: "I did not use the help button. When the system was not providing required information, I tried other criteria, which helped in most of the cases."
- Participant P120: "The system indicates when it could not understand what I was asking for. In such cases, I simply tried asking the system with another sentence."

We **conclude** that although the help dialogue strategy was very infrequently used in the Archivus system, earlier experiments with vocal-only system for controlling home appliances demonstrated more frequent use of the help dialogue strategy (about 1.66% of all system dialogue turns). We therefore speculate that the need for a help dialogue strategy is lower in systems equipped with a screen output (as users better understand the interaction context compared to voice-only systems) and in the situations where the learning of system functionalities through exploration is reduced (e.g. by providing a tutorial).

6.4.2 User requests for the last prompt repetition

The *repeat* local dialogue strategy is used to handle user's requests for the repetition or reformulation of the last system prompt. The strategy was hardly ever invoked by users in the Archivus system – it has occurred only in 4 cases (0.03% of all user inputs). In three cases, the user asked the system to repeat the prompt message because the message was unexpected in the given context of interaction (the system message was inappropriate because the output wizard made a mistake in selecting the message). In one case, the user misinterpreted the concept of the repeat button and intended using it in order to replicate his last action.

A marginal use of this strategy in Archivus is not surprising, as the user can also see the last system message on the screen in a textual version. The strategy was primary developed for vocal-only systems where good understanding of system prompts is crucial for interaction.

Our **conclusion** is that the repeat strategy is useless in multimodal screen-based systems. The three above mentioned cases of use rather indicates that the users were actually listening to the system prompts, and hence do not provide any evidence about the fact that the repeat strategy is relevant for the multimodal systems.

6.4.3 Problems with processing of user inputs

In situations when the user's last multimodal input cannot be interpreted with any of semantic pairs, the system uses `NoInput` or `NoMatch` strategy:

- The `NoInput` strategy indicates that the system was *unable to acquire any signal* from the user within a predefined timeout, or the signal was very noisy. In systems integrating speech recognition, it means that the user (1) stayed silent for a long time or (2) that the user said something what could not be understood by the recognizer (due to acoustic or pronunciations problems, or because the user pronounced only speech fragments). In Archivus, we have used the strategy only in the latter case (and we have ignored the former case). Such situation is indicated to user by system prompts like: *"Sorry, what did you say?"*, *"I could not hear you. Could you say it again?"*. The user is advised to *repeat* more clearly his/her last input.
- The `NoMatch` strategy indicates that the signal of user behavior was acquired correctly, but the system has *failed to interpret the signal* within the context of the current dialogue. In particular, it means that the speech recognizer has recognized a sequence of words, but the consequent parsing with natural language understanding component has failed. This is indicated to the user by system prompts like: *"I cannot do that"* or *"I cannot find that word in the system"*. The user is advised to *change* his/her request.

In Archivus, the specificity of `NoInput` and `NoMatch` strategies is that they apply only to inputs in natural language. Pen pointing or mouse clicking cannot trigger the `NoMatch` (out-of-context) strategy, as the semantic pairs are produced in response to clicking (pointing) on active graphical components within current context (and all out-of-context components are not visible on screen and therefore not available for clicking). Although users can perform 'invalid pointings' by trying to click on unclickable components or having difficulties to physically perform the click with a pointing device like a pen or tactile screen, such invalid pointings are traditionally ignored by systems. The specificity of these two strategies also shows the different *expression power* of modalities: the expression power of pointing is limited to the context provided by the system (but the context can be changed with a well-chosen sequence of pointings), while the natural language allows the user to directly establish the context of the communication (but is limited by the understanding capabilities of the technology, which is indicated by the `NoMatch` strategy).

During our WOz experiments, the understanding of user's vocal or typed natural language requests was simulated by wizards. This allowed for *optimal performance* of natural language processing. Only very unintelligible or out-of-context user requests (that is requests that cannot be directly satisfied with existing system functionalities) lead to triggering of the `NoInput` or `NoMatch` strategy. We have analyzed *how often the strategies were triggered* and obtained the following results:

- `NoInput`: 1.9% of all *voice inputs* were unintelligible even to human wizard. The typical reason is a user pronunciation problem or user speaking too loud or silently to the microphone.
- `NoInput`: 0.3% of all *keyboard inputs* were classified as `NoInput`. This happened in infrequent cases when the user submitted only partial request or accidentally hit the Enter key.
- `NoMatch`: 3.1% of all *voice inputs* were not fully responded by wizard because the system lacked the required functionality or the user-specified search criterion was not available in the system.
- `NoMatch`: 12.1% of all *keyboard inputs* were classified as `NoMatch`. The proportional increase in the occurrence of this strategy compared to voice inputs can be partially explained with two facts. First, the users have repeated their previously `NoMatch`ing vocal input on keyboard (2.0% of all keyboard inputs). Still, such an attempt led to the same effect, with an increasing proportion of keyboard `NoMatch`es, as the keyboard is used ten times less frequently than voice. Secondly, the users have altered the previously (by keyboard) submitted queries in situations when they have previously made a typo or they have reformulated the queries containing numbers (e.g. the users have first submitted *"1000 CHF"*, and then tried again with a query like *"thousand francs"*). However, because neither the typo nor the wrong number format were the origin of a problem (the wizards were instructed to interpret modest typos correctly), the users has fallen into the same problem again (2.2% of all keyboard inputs).

  After eliminating all previously mentioned cases, 7.9% of all keyboards input were actually failing due to the unavailability of a given search criterion in the system. We do not have an explanation for this increase with respect to vocal inputs.

No correlation was found between the user-experienced number of fail events (`NoMatch` or `NoInput`) and subjective user's opinion expressed in the post-experimental questionnaire (in particular, we have investigated the user answers to the question *"Did you find that you could talk to the system easily?"* and the user agreements with the statement *"Being able to use language to interact with the system was helpful"*). We assume that the overall performance was good enough (90% of users experienced maximum 10 fail events during 40 minutes of interaction) to prevent the users to draw any negative conclusions about the ease-of-use or usefulness of the natural language for interaction.

We **conclude** that altogether 5.7% of *all inputs in natural language* (done by voice or through the keyboard, counting both `NoInput` and `NoMatch` situations) were not answered by the system. This happened either because the system lacked the required functionality or because the user's input was incomprehensible, even to human operators. We therefore consider the observed percentage as interesting information on the theoretical performance limit of any interactive system integrating natural language.

6.5 Evaluation of modality use

This section aims at determining the *role of available modalities from the perspective of the dialogue system.* In particular, we try to identify how complex the modality fusion algorithm needs to be and what types of operations are most frequently addressed by a user in natural language, thus suggesting a minimal necessary language model in the Archivus system.

The impact of available input modalities on users working with the Archivus system is thoroughly studied by Lisowska [57], who answers questions like how users perceive different modalities, what is the frequency of use of various modalities combinations, whether certain modality helps to learn the system easier and whether the available modalities have an impact on the user's speed and accuracy of task solving process. Therefore, those issues are not the main subject discussed here.

6.5.1 Simultaneous use of modalities

Archivus allows users to perform actions using any of the available modalities or their combination. In response to every user's (multimodal) input, the interface becomes grayed-out in order to signalize that the system is processing the user's last input. This way, the interaction is segmented into dialogue turns. From this perspective, users can behave 'sequentially multimodally' (i.e. switching between modalities used to perform dialogue turns, but using one modality within one dialogue turn), or users can perform actions 'simultaneously multimodally' (i.e. using several modalities in one dialogue turn). Since the latter type of multimodality requires more advanced multimodal fusion algorithm, we wanted to know *how frequently the simultaneous multimodality is used* by Archivus users.

From this perspective, we have found that the users often used the different modalities available, but used them simultaneously (within one dialogue turn) only in 1.85% of all dialogue turns(123 cases). This number includes also the situations when users tried to point with the mouse (or pen) on interface components that were not active and therefore such pointing was ignored by the system. The users then typically used another modality to provide the input. We feel that such cases do not represent a true example of simultaneous use of multiple modalities, so we can exclude them. This way, we get only 20 (0.3%) cases of truly simultaneous multimodality use. In most of those cases, users provided the same information in several modalities (e.g. saying *'close'* at the same time as clicking on close button), while only in 4 cases the users specified two different information, each of them with different modality. However, the inputs in modalities were always independent, in the sense that we had no case of interaction such as the users saying *"Show me questions from this person"* while indicating a person on the screen using a pointing device.

We can therefore **conclude** that simultaneous use of modalities is *extremely rare* in Archivus. When it happens, it mainly corresponds to the situations where speech is redundant with what is typed or pointed to. We thus believe that *no complex multimodal fusion algorithm is necessary* for the interpretation of user inputs in systems similar to Archivus. The existing fusion algorithm consisting in simple merging semantic pair sets from different modalities is fully sufficient.

6.5.2 Proportions of overall modality use

In this section, we compare how frequently are the individual modalities used by users. This could suggest inutile modalities which can be removed from the system, while other (frequently used) modalities should be optimized for robustness and speed of processing in a fully automated system.

Since our data collection contains two different modality conditions (MVK and PVK), we compare proportions of modality use in each condition separately. We define that a modality is used, when corresponding user's dialogue turn is performed with that modality. Moreover, since each user performed a different number of dialogue turns, we calculate an overall proportion of the modality use as an average of proportions of the modality use of every user (thus avoiding the results to be biased by users who performed above-average number of dialogue turns). The results are in Table 6.7.

Modality	*M/P*		*V*		*K*	
condition	*avg*	*dev*	*avg*	*dev*	*avg*	*dev*
MVK	54%	±26	41%	±27	5%	±7
PVK	39%	±24	57%	±27	4%	±6

Table 6.7: Overall proportions of modality use. Calculated as averages over proportions of modality use of every user. The deviations are standard deviations.

As it can be seen, the most dominant modalities are M/P and V, while *keyboard is used relatively seldom* (4-5%) in both conditions. On the other hand, the *presence of mouse or pen has quite strong impact on use of voice* – depending on whether mouse or pen is available, an overall proportion of voice use is 41% or 57%, respectively.

The results clearly show that *natural language (V+K) is very frequently used by users* (used in 61% or 46% cases compared to pen or mouse). As such, it gives an experimental justification of the effort directed from a GUI-based towards a natural language-enabled system. Nevertheless, one has to keep in mind that all NL inputs were processed by a wizard and not automatically by the system. Users therefore experienced almost perfect speech recognition and NL understanding rate, which is unlikely to be achieved with current NL technologies. It is likely that the use of speech would decrease in a system with ASR in favor of keyboard, or in favor of reliable pen pointing or mouse in a system with automated NL understanding.

It is also interesting to note that *functionally equivalent modalities (from the system perspective) are not perceived as such by users* [57]. Two modalities are considered to be functionally equivalent if they can provide input that has exactly the same semantic and functional content, and if the input is processed in exactly the same way by the system, resulting in exactly the same system output. In our system, functionally equivalent is pen and mouse, as well as voice and keyboard. However, we can see that voice is 8 to 14 times more frequently used than keyboard, and that mouse is 1.4 times more frequently used than pen in the Archivus system. Obviously, from the two functionally equivalent modalities, the users prefer the one which allows them to provide inputs faster, more reliably and with less effort. Such an experimentally verified fact should be taken into account by system designers when introducing new modalities into the system.

The calculated proportions of modality use (Table 6.7) exhibit quite high standard deviations indicting large spread of modality use proportions for individual users. We therefore inspect the underlying data in Figure 6.1.

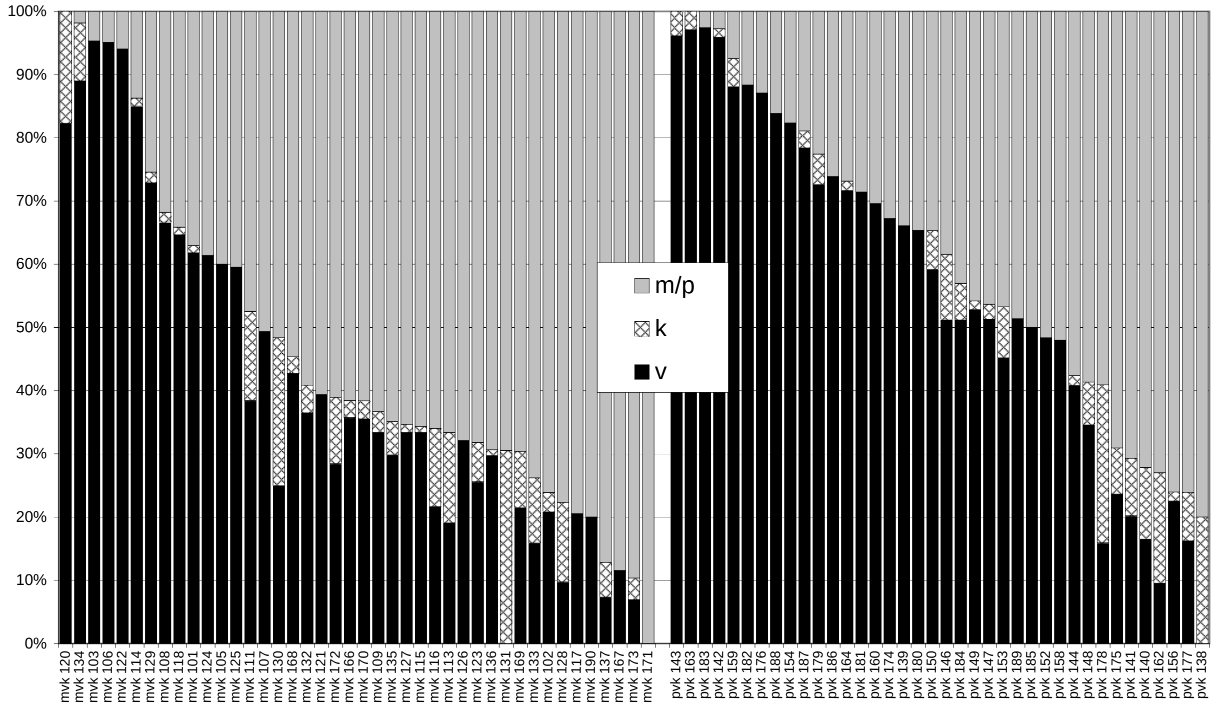


Figure 6.1: Proportions of modality use per individual user (users are labeled by their ID numbers), ordered by increasing proportion of mouse/pen use. On left are users with MVK modality, on right are users with PVK modality.

The display of individual proportions of modality use for each user clearly confirms that the use of modalities is indeed highly user-depended. Both extreme cases of the modalities use are in the figure, ranging from users controlling the system only through NL to users relying only on the mouse. The proportion of NL vs. pen use is fairly evenly distributed over all users. The PVK condition shows that although the users in general used keyboard only little, the proportion of using the keyboard rapidly increases when the overall use of language drops below about 35%. One could assume that these users found natural language useful for controlling of Archivus, but they felt uncomfortable with voice and substituted it with NL requests typed on keyboard.

From our experimental result, we can therefore **conclude** that natural language (voice and keyboard) is on average very frequently used by users, therefore providing an experimental justification of the efforts directed from GUI-only towards NL-enabled systems. However, the individual use (and thus user-perceived importance) of modalities largely varies among the users. We also observed large differences in the use of functionally equivalent modalities (pen/mouse and keyboard/voice). A lower pen use can be explained by physical difficulties of use experienced by users (when compared to mouse), while the rare keyboard use can be attributed to the fact that using the vocal input requires less effort and is therefore preferred (over typing) when the speech understanding performance level of the system is high enough.

In consequence, we can thus **recommend** the integration of the natural language modalities for multimodal interfaces. Nevertheless, the individual usefulness of various modalities available in the system has to be evaluated in an integrated prototype, because of the strong modality impact of one to others. Furthermore, only very little can be inferred about the expected modality use from an observed use of a functionally equivalent modality. We thus recommend that for any specific system, the modality usefulness is evaluated with a high number of users to cope with the expected large variance in the frequency of use related to personal preferences.

6.5.3 Proportions of modality use per action

Yet another aspect that we are interested in is *whether there are any correlations between a particular modality and the action performed by users.* In case if we find an action that is nearly always performed with a specific modality, the modality processing can be optimized (in terms of robustness and speed of processing) for that action. For instance, a language model does not need to include commands related to actions that are always performed by mouse, or conversely the language model must be adjusted with respect to the actions that are nearly always performed by natural language.

We distinguish four basic categories of actions that can be performed with the system:

- *Adding constraint (C)*: user specifies a new search constraint. This category contains actions like `C_Keyword` (user looks for meeting utterances with a given keyword), or `C_Topic` (user looks for parts of meetings with a given topic).
- *Navigating in interface (N)*: a semantic effect of such an action is the opening of a new interaction context (mGDN) or accessing additional information (i.e. accessing the next page in a list of values, in a book or in a document).

  Examples of actions in this category are `N_FocusOn` (user opens another mGDN), `N_OpenMtg` (user opens a specific meeting in the bookcase), `N_Close` (user closes the current mGDN), `N_ListUpDown` (user scrolls to the previous or next page of a list), `N_JumpToLetter` (user accesses a list of items starting with a specific letter, using the letter selection button feature of the mGDN with the list layout), etc.
- *Performing a command (Cmd)*: user resets the system or erases one or more search constraints. This category contains the following actions: `Cmd_Restart`, `Cmd_ClearHistory`, `Cmd_DelHistory`.
- *System failure (S)*: this category is related to situations when the system is unable to respond to a user's request (`S_NoInput` and `S_NoMatch`).

In order to determine a proportion of modalities per action, the following method is used: each of the actions is associated with semantic pairs that lead to that action. Then, for all the semantic pairs associated with a given action, we count how many times the semantic pairs were generated with each of the modalities. This method allows us to identify a proportion of each modality use for each of the actions. Nevertheless, a small modification of the method is necessary in situations where one single user's natural language input generates several semantic pairs (e.g. user's utterance *"Who suggested a blue sofa?"* generates `ArgSegClass:suggestion`, `Keyword:blue` and `Keyword:sofa`). In such cases, the natural language would gain three 'instances of use', while it was in fact used only once. Therefore, an 'instance of use' of an individual semantic pair is calculated

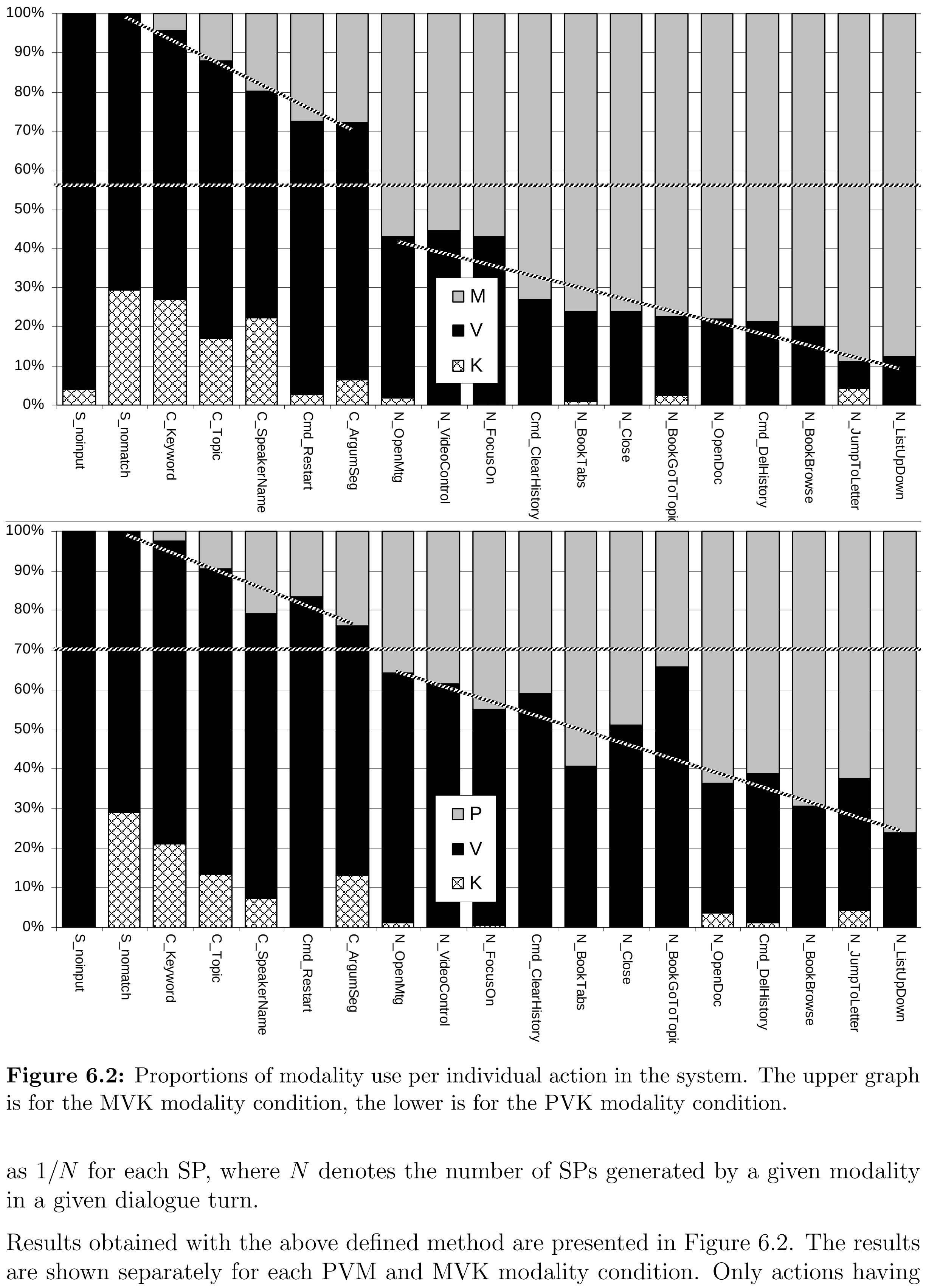


Figure 6.2: Proportions of modality use per individual action in the system. The upper graph is for the MVK modality condition, the lower is for the PVK modality condition.

as $1/N$ for each SP, where N denotes the number of SPs generated by a given modality in a given dialogue turn.

Results obtained with the above defined method are presented in Figure 6.2. The results are shown separately for each PVM and MVK modality condition. Only actions having at least 13 instances of use are shown. The shown actions have on average 170 instances of use (with median of 90 instances of use).

The figure shows that natural language (V+K) is dominantly used for S- and C-categories of actions. The MVK condition exhibits a clear drop of NL use for navigational actions (N), which is obvious even in the PVK condition. Interestingly, the two Cmd-actions (clear all constraints and delete one constraint) have low use of NL, while restarting the system is dominantly performed by voice. This becomes much clearer, when we realize that the users restarted (reset) the system at a moment when they have finished solving a task and when they were busy with inserting the task-card into a paper box. In those situations, users could not use the modalities operated by hands that easily, explaining why they used voice. This result experimentally confirms that voice is useful in hands-busy environments. Finally, no use of pointing (M/P) for S-actions is not surprising, since those actions are possible only when using natural language as discussed in section 6.4.3 on page 106.

Although we have not found any system action (except S-actions) always performed with the same modality, we draw other interesting **conclusions**:

- Search constraints are dominantly provided in natural language (voice or keyboard).
- Navigation within the interface is mainly performed with a pointing device (mouse or pen).
- Navigation is rarely performed by typing (keyboard).
- An increased use of voice is observed in hands-busy situations.

Our observations might hence suggest a *preferred role for language-based modalities* in multimodal interfaces: language is mainly used for providing search constraints (data) to the system and less frequently for controlling and navigating within a graphical interface. In consequence, it appears that existing vocal command-and-control desktop applications may not combine GUI and speech in a manner that is optimal and natural for users.

6.6 User performance in Archivus

During the experimental sessions, users are given tasks to be performed with the system. These tasks consist of questions about the content of the meetings and they have to be answered by using the Archivus system. The questions are given on paper cards (in a predefined order) and users solve them sequentially one-by-one, writing the answers directly on the corresponding paper card (users were allowed to leave the card blank if they were unable to find the answer). The questions themselves were chosen so as to give as much variety as possible in the type of content that users would have to access and to vary the amount of steps that users would need to take to find the required information. A list of the used tasks is shown in Figure 6.4 (together with additional information). The actual complexity of the tasks was tested during the pilot studies with the Archivus system. Furthermore, the influence of the question order was studied by cross-validation, and no influence was found. Further details about how the questions were selected can be found in [57].

By reviewing the answers provided by users on given questions (tasks), we can determine *how successfully the Archivus system allows users to perform their tasks.* A similar evaluation was performed by Lisowska in [57], but principally from the perspective of available modality impact on the user performance. It was observed that MK and MV modality conditions are slightly more effective than MVK and PVK, but a statistical significance of those results is not shown. In this section, we evaluate the Archivus system globally, without looking at particular modalities available to users. Therefore, we take into account all the tasks performed by users during both experimental sessions (i.e. the restricted modality and the full modality conditions).

One of the aims of this section is to give an impression about the accuracy and speed of users when solving the tasks given to them (subsections 6.6.1 and 6.6.2). This is a technique also used in the Browser Evaluation Test (BET [121, 88]) – a method designed especially for evaluation of meeting browsers. Unfortunately, our results cannot be directly compared with the results obtained in BET evaluations for other browsers (audio-based browsers [24] and TQB [88]), mainly because we have evaluated the browser on a different set of tasks (compared to BET) and we have six-times larger search database. The explanation for the different experimental settings is that the BET technique was not yet finalized at the moment when we prepared our experiments. We regard the BET-evaluation of Archivus as a future work.

In order to give a better impression of the actual user performance, we discuss the individual differences among users and we examine the individual complexity of the tasks (subsection 6.6.3). Finally, we identify problems preventing the users from answering certain tasks correctly (subsection 6.6.4).

6.6.1 User accuracy (correctness of task answers)

First, we wanted to know how often the users answered the given tasks (questions) correctly.

The user answers were marked as *correct*, *incorrect*, or as *not answered* in case if the card was left blank (users were instructed to leave the answer blank if they could not find an answer within a reasonable time). This allowed us to determine a proportion of correct, not answered and incorrect answers per user. Such proportions are then used to determine an overall accuracy of answers given by users. Due to the fact that the user tasks were questions of two types (true-false and short answer) with different baseline of accuracy (50% vs. 0% respectively), we also perform that analysis for each question type.

	Correct		*Not answered*		*Incorrect*	
Type of question	*avg*	*dev*	*avg*	*dev*	*avg*	*dev*
true-false	87%	±12	4%	±7	9%	±10
short-answer	65%	±22	15%	±15	21%	±14
all questions	76%	±14	9%	±9	15%	±9

Table 6.8: User correctness. Calculated as averages over proportions of correctness for every user. Deviations are the standard deviations.

Results are summarized in Table 6.8, showing that the users managed on average to correctly answer 76% of the tasks, while they provided incorrect answers in 15% of the cases (the various error types are analyzed later in section 6.6.4). The nearly four-time lower proportion of unanswered true-false questions when compared to short-answer questions indicates that the users are likely to select one of the two answers randomly in situations if they do not know the correct answer.

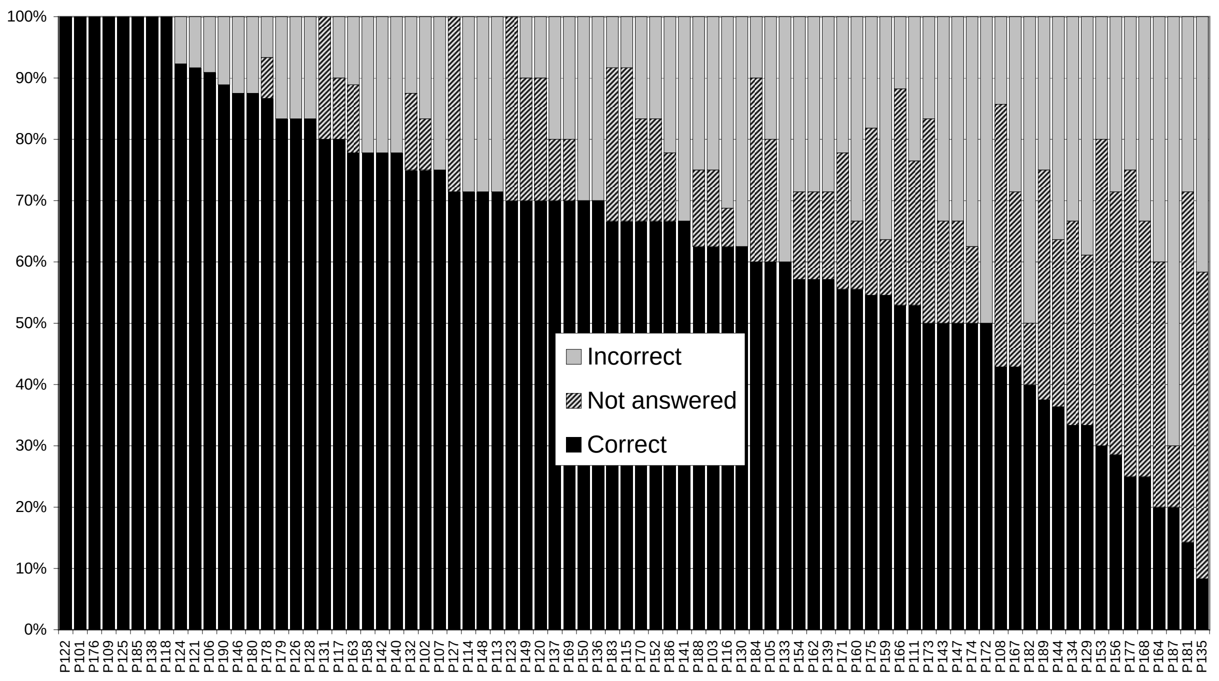


Figure 6.3: Proportion of correctly, incorrectly and not answered tasks per user.

As the data exhibit a non-negligible standard deviation, we also present the underlying data (Figure 6.3) that was used to compute the results already given in Table 6.8. On this figure, one can observe that, although 10% of the Archivus users (8) answered correctly all the given tasks, about 19% of the users (15) answered correctly less than half of the tasks. Interestingly, the latter users also tended to have a larger proportion of unanswered tasks (while their proportion of wrong answers is comparable with other users), indicating that they probably had difficulties to find the answers.

6.6.2 User speed (tasks per minute)

Our next interest was to know how long on average it took to users to solve a task and whether there are remarkable differences in speed among users.

A quick look at the data shows that within the allotted 40 minutes (actually split into 2×20 minutes), users worked on average on 20 tasks (answered or not), but this number varies from 11 to 37 tasks depending on the user.

However, such indications cannot be directly used to estimate the individual user speed, because of several reasons:

- The allotted time typically expired during the time when a user worked on a certain task – and this task does not count within the above given figures. In consequence, the actual total time necessary to solve all the completed tasks is lower than the allotted time.
- The allotted time was measured by an experimenter who sat in another room and acted as output wizard. In consequence, the allotted time is not always exactly 20 minutes, but it varies by 1-2 minutes.

- Because the experiments are carried out using WOz simulations, and on two machines (desktop PC and tablet PC) with different computing speed, the users have experienced different system response times. Consequently, the actual time available to user for the interaction with system is lowered by the overall system response time experienced by that user.

In order to accurately determine the user speed, we use the following method: for each user and interaction session, we measure the exact time it took to solve all of the completed tasks. Based on that, we compute the average time taken by a user to solve a task and the average speed of a user (measured in questions per minute). In the alternative approach, we reduce the task completion time by the duration of the experienced system responses, as this better estimates the user's speed in case of automated (i.e. no WOz simulation) and optimized system implementation. Results are in Table 6.9.

	Duration per task [s]		*Speed [tasks per min]*	
System response times	*avg*	*dev*	*avg*	*dev*
Include system response times	127s	±36s	0.51	±0.14
Exclude system response times	91s	±31s	0.73	±0.24

Table 6.9: Speed of task solving. Computed as average of individual user speeds and averaged user task durations.

It took about two minutes for the users to solve a task, but about almost one third of this time is due to the delay generated by the system (wizards) when processing the user's multimodal input. The expected user speed in a system providing immediate responses would therefore be about 0.73 tasks per minute. However, as already mentioned, the observed speed is quite user-dependent, as it can be seen when noticing the large standard deviation of that measure.

6.6.3 Complexity of tasks

We have shown in previous sections that there are fairly big differences among users in their average accuracy of the task solving process. The average user accuracy is computed over all tasks solved by a given user, regardless of his/her individual speed. In consequence, different users have solved a different number of tasks, thus the accuracy of certain users is possibly biased by the complexity of the individual set of tasks solved by that users.

For that reason, we decided to estimate the complexity of each task. For each task, we identify the proportion of users who answered the task correctly, incorrectly, or did not managed to find an answer (i.e. users inserted empty card into a box). Results are in Figure 6.4.

The figure shows 33 tasks, each of them solved by at least 10 users, ordered by proportion of users answering the task correctly (hereafter called the task correctness). Except the last four tasks, the task correctness is spread nearly linearly from 61% to 99%, indicating that the tasks are of variable complexity. Within the analyzed tasks, there is quite similar number of the true-false questions (TF, 17) and the short-answer questions (SA, 16). Not surprisingly, the TF tasks scored better – within the top 15 tasks (correctness $\geq$84%) are 11 of them (74%) of the TF type, while within the bottom 15 tasks (correctness $\leq$79%) are only 3 (20%) of the TF type.

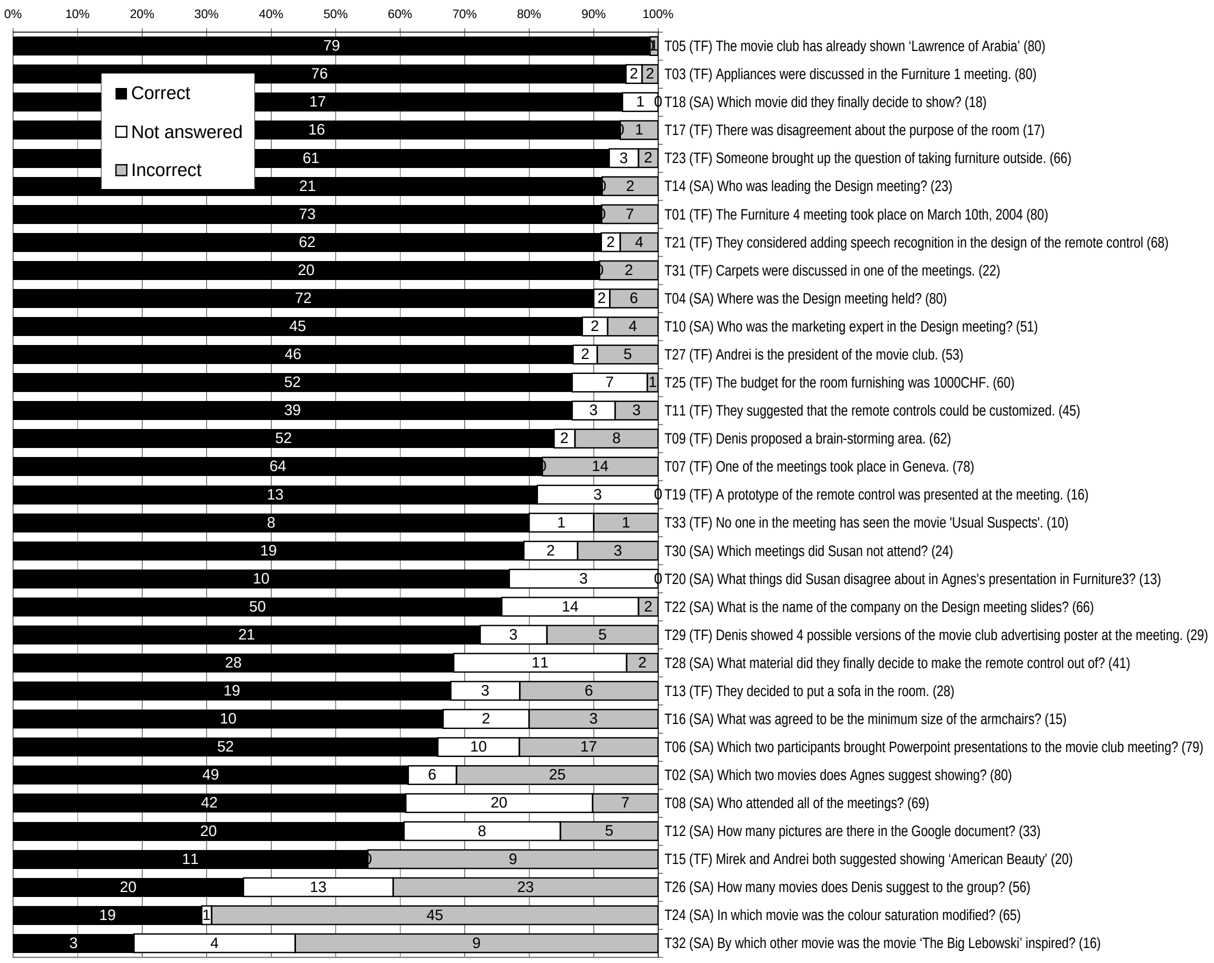


Figure 6.4: Tasks solved by users, sorted by proportion of correct answers. Each task is labeled with its task id (e.g. T5), task type – true-false (TF) or short answer task (SA), task formulation and total number of users (in last parentheses). The absolute number of users who answered the task correctly, incorrectly or did not find the answer is shown in the corresponding part of the bar. Only tasks solved by at least 10 users are shown (i.e. 33 tasks shown, 7 tasks not shown).

The last four tasks with lowest correctness are also having the biggest proportion of wrong answers, while the proportion of cases when these tasks were not answered is not exceptionally large (rather comparable with other tasks). The last three tasks are the short-answer tasks, where one could assume that the users will answer only if they are convinced about the correct answer. Surprisingly, many users provided the same wrong answer. Therefore, we will identify the reasons for the incorrect user answers in the next section.

6.6.4 Sources of task failures

Because we have found some of the tasks to be harder for the users than others (i.e. an important proportion of users failed to answer the task correctly), we wanted to know why the users have failed when solving such tasks (hereafter called the failing tasks). In particular, we focus on the experienced interaction problems and reasons that have caused them.

In order to do so, we decided to manually review the failing user interactions (i.e. the interactions where the users did not solve the task correctly) and to manually classify the reasons for a task failure. As reviewing all interactions would represent too much work, only several tasks were selected for the manual review. We have selected the tasks that have relatively low correctness score (indicating that the task is in principle problematical), the tasks that have reasonably low absolute number of failing users (so that we can review all cases for that task) and the tasks with either low or high proportion of users who did not find the answer. The tasks selected for manual review are the following:

T13: (TF) *They decided to put a sofa in the room.* (9 failing cases)

T16: (SA) *What was agreed to be the minimum size of the armchairs?* (5 failing cases)

T12: (SA) *How many pictures are there in the Google document?* (13 failing cases)

T15: (TF) *Mirek and Andrei both suggested showing 'American Beauty'* (9 failing cases)

T26: (SA) *How many movies does Denis suggest to the group?* (36 failing cases, 18 cases analyzed)

T24: (SA) *In which movie was the color saturation modified?* (46 failing cases, 10 cases analyzed)

Each of the analyzed failing cases was annotated (by one annotator) with the probable reason for the task failure. In situations where several reasons were assumed to contribute to the task failure, we have annotated the task with all of them (and we have also used a possibility of partial-only contribution). The results are summarized in the following list (including proportion of contribution of given reason to task failures with respect to contribution of all reasons):

A. (52.9%) The user finds correctly the part of meeting transcript containing the task answer, but is unable to infer (or infers incorrectly) the answer from reading the text.

B. (11.4%) The user uses improper search criteria, i.e. criteria that do not select the relevant parts of meeting transcript or documents.

C. (10.0%) The user finds the document containing the task answer, but is unable to infer the answer from inspecting the document or is not inspecting it entirely.

D. (4.3%) The user does not open the relevant document, because he/she looks for the answer only within the search hits, but the meeting documents are highlighted in our system with the search hits.

E. (3.6%) The answer cannot be found directly in the meeting transcript (but only in a related document) and the user is looking for the answer only in the transcript.

F. (3.6%) The user is unable to find an answer, because he/she does not use the semantic categories (dialogue elements) in the search query.

G. (2.9%) The user is having problems with using some of the system features (e.g. unable to browse search hits, unable to use document links or to manipulate the set of current search constraints).

H. (1.4%) The user is unable to find an answer, even if he/she is using the semantic categories (dialogue elements) in the search query.

I. (1.4%) Missing (or incomplete) annotation prevents the user from finding an answer.

J. (1.4%) The user has difficulties controlling a specific system feature when in the condition with limited access to modalities (e.g. when selecting a value from a list of topics using only mouse or pen).

K. (7.1%) Other reasons for failure

Clearly, the biggest source of a task failure is *user's misinterpretation of meeting transcripts or meeting documents* (A+C altogether accounts for about 63% of problems). The second most frequent source of a task failure is that *the users do not formulate their search constraints properly* (B+F, 15%). Some of these problems might be eliminated (D+E+G, 10.8%) if the implementation of the system was improved (e.g. displaying search hits also for meeting documents, hit tabs that allow for a direct access – not only for scrolling, etc).

Concerning the problem of the recurring wrong task answers provided by users (as outlined in the previous section), it is caused by a misinterpretation of meeting transcripts (and meeting documents) in situations where the meeting is not well structured or when documents (or tasks) are partially ambiguous. We illustrate this problem on the following two examples.

The first example of user's common misinterpretation of meeting transcript is the task T24: *In which movie was the color saturation modified?* The users usually found the part of meeting where participants spoke about the modification of the color saturation. Although the whole discussion is in the topical segment 'Saving Private Ryan', the color saturation is directly preceded a discussion (on opposite transcript page) about the awards received by the movie director (Steven Spielberg), and within this context, another movie 'E.T.' is mentioned. Most of the users then simply respond with the name of that movie, apparently without deeper reading of the movie context, and probably because that movie name is spatially closest to the search hit. This problem can possibly be eliminated with a permanent display of the full topical categorization of the presented transcript, or with a 'smoothed' transcript (e.g. with discourse fillers removed) presenting the meeting content in a more understandable manner.

Another example of partially ambiguous document is task T26: *How many movies does Denis suggest to the group?* In this case, the users also usually successfully found the relevant participant's PowerPoint presentation. However, the first part of the document presents the chart with 50 top-rated movies (justifying the choice of movie) and only second part of the document explicitly suggests one specific movie. In consequence, many users answered that the 50 movies were suggested to the group. We expect that the problem can be eliminated by annotating document contents similarly as the meeting transcript – then, the document would open directly at the page with movie suggestion, in case if the 'suggestion' was a search constraint.

6.7 Subjective user opinion about interactions in Archivus

This section evaluates the Archivus system from the perspective of subjective user opinion of the system and their experience with it. The data was gathered using a post-experiment questionnaire (the full version of the questionnaire can be found in [57]). We analyze only the first part of the questionnaire that evaluates the system in general and involves questions related to system usability, understandability, learnability, and user satisfaction with the system.

The subjective user opinion was collected using a series of statements. The users are asked to indicate how much they agree with the statement using one of the predefined responses: *strongly agree*, *agree*, *neutral*, *disagree*, *strongly disagree*. In order to measure an overall opinion of all users about a given statement, the numeric quantifiers (-2, -1, 0, 1, 2) were assigned to the answers and the overall opinion is defined as an average of the quantified answers to a given statement. The results are in Figure 6.5.

In general, the users had a positive reaction to the system and experienced interactions.

Most positive user reactions (averaged score 1.28) were to the statement Q6 *Being able to use language to interact with the system was helpful.* Only 4% of the users responded negatively. This result clearly shows a user-perceived importance of natural language for human-machine interaction. It justifies our efforts towards language-enabled systems.

Next, the users stated (Q3, 1.15) that they have indeed *understood the topic of the meetings.* This result enables us to assume that the chosen domain (meeting searching and browsing) does not caused major problems to users and thus the results reported in the previous sections are not biased by the choice of the application domain. This is further confirmed by fact that a majority of users (81%) declared that they *felt comfortable* when using the system (Q2, 0.99).

The users were also very positive (score 0.93–0.74) when answering the statements (Q1, Q9, Q4, Q10) related to the *ease of system use, learnability and appropriateness of the library metaphor.*

To some extent less positive were the Archivus users concerning their subjective feeling about being in control of the system and using it as they wanted (Q5 and Q11, score 0.51–0.48). About 9–13% of users disagreed with those statements and 34–38% of users stayed neutral about such statements (nevertheless, 53–54% of users had positive opinion). A further investigation will be necessary to discover whether some specific problems were experienced by the users who claim they couldn't use the system according to their needs.

Our examination of user's agreement with the statement Q8 *It was easy to go back and change a criterion* revealed that the users either agreed (48%) or disagreed (27%) with the statement and relatively low proportion of users stayed neutral, forming two evident groups of contradictory user answers. Such distribution of answers was found only by this statement. We speculate that this is actually caused by the formulation of the statement. In fact, it is not directly possible to 'go back' in our system as the system does not have any 'go back' button (hence it is difficult to go back), but it is possible to change search criteria and perform 'go back' in an indirect way (hence it is relatively easy to go back).

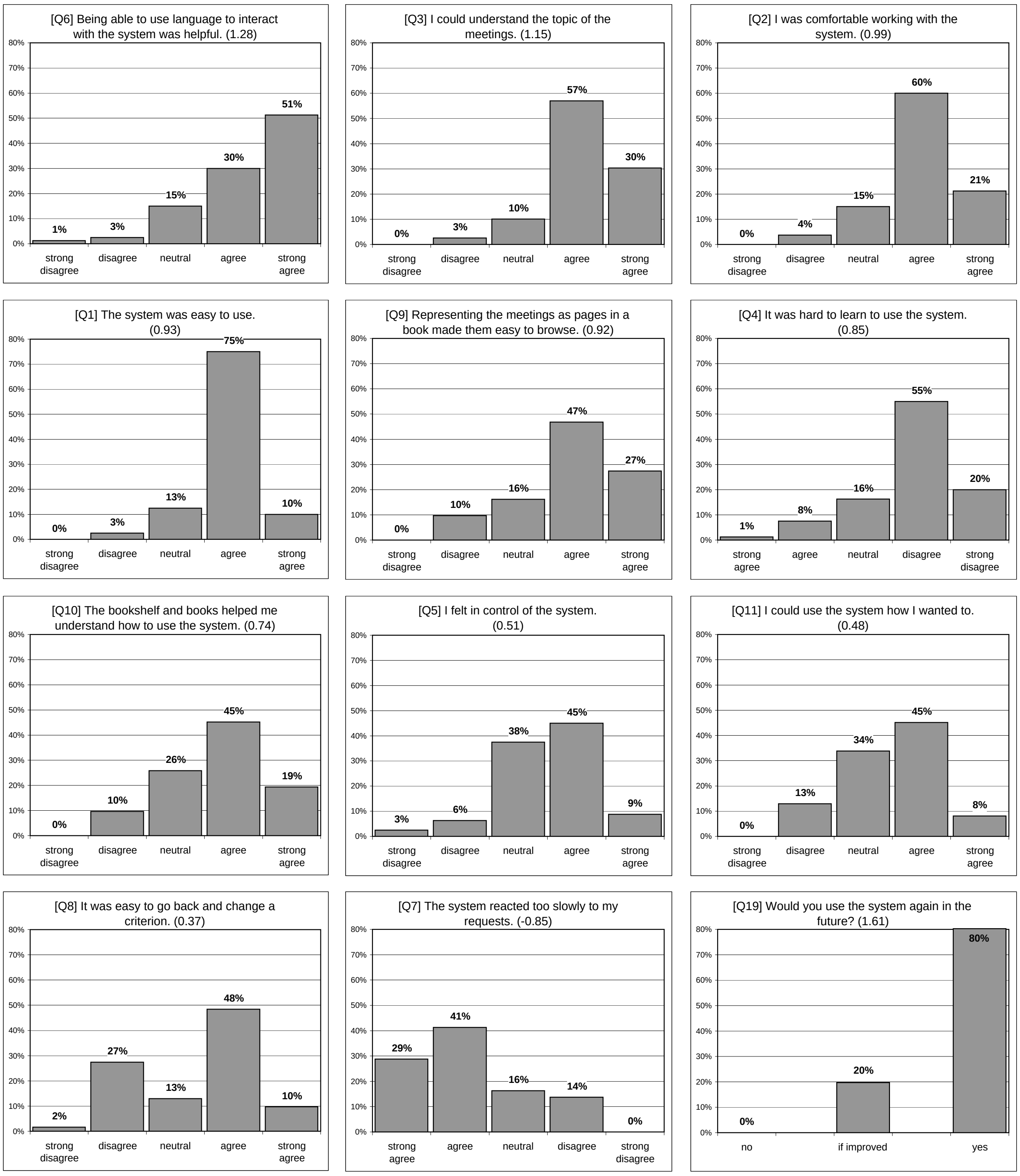


Figure 6.5: Results from the post-experiment questionnaire. Questions are ordered by an overall averaged *positiveness* of the answer (with respect to the system), scaled from 2 (positive) to −2 (negative), except Q19 which uses a different scale (displayed at the end). Notice that Q4 and Q7 are formulated in reversed sense, meaning that disagreement with the statement is actually a positive reaction to the system. The X-axis of Q4 and Q7 are therefore reversed.

We guess that the user answer depends on their first understanding of the statement. Consequently, we cannot consider this result to be completely valid.

The only negative result concerns the user opinion about the speed of the system. The 70% of users regarded the system reactions as slow (Q7, −0.85). We remind that the average system reaction time was 4.5 seconds, with standard deviation ±5.3s, minimum 0.1s and maximum 37s. The system reaction time is largely caused by the wizards and only partly by the real cost of the system operations, suggesting that this problem will be eliminated in the automated and optimally implemented system. A more general conclusion is that any GUI-enabled multimodal system with similar (or worse) response parameters will be regarded as slow by users. It is therefore indeed necessary to optimize the Wizard of Oz control interface and train the wizards.

Finally, 80% of users are willing to use the system in the future and 20% are willing to use if the system was improved. None of users said that he/she wouldn't use the system. We consider it as a very positive result, especially when bearing in mind that Archivus is still a system prototype. The result also stimulates our efforts in continuation of the research in multimodal dialogue-enabled systems.

6.8 Summary

This chapter provides objective and subjective indicators about the observed user interactions with the Archivus system, resulting from a large-scale Wizard of Oz experiment (involving 91 users, out of which 80 were taken into account in our work). We provide two main sets of results: the first set is related to the use and acceptance of various system features (dialogue strategies and available modalities) by users. This set of results is assumed to be valid for any system designed with our ISPM methodology. The second set of results is specifically related to the Archivus system and aims at evaluating user's speed and accuracy on tasks carried out with Archivus, followed by indications of task complexity and analysis of the most frequent sources of task failures. Finally, subjective user opinions about the Archivus system are presented.

The first set of results is assumed to be valid for any multimodal dialogue-based system with properties similar to systems designed with our ISPM. Our main findings are the following:

- The strategy for early automatic presentation of possible search results is frequently executed by the system and well-accepted by the users. Our recommendation is therefore to enable the automatic presentation of results as soon as their number is sufficiently low to be efficiently displayed, provided that the system has enough evidence about the expected relevance of these results. Nevertheless, automatically presenting information should neither prevent users from having the possibility to further browse the search space nor to enter new search constraints. In addition, the nature of the display used for presenting the results should take into account the specificities of the displayed data (for more details, see the analysis of the user failures given hereafter).
- In order to deal with over-constrained situations, the system should offer to the users at least two ways for resolving them: erasing a specific search criterion only

or erasing all search criteria simultaneously, because we found that both options are used by users to a similar extent. Furthermore, the indications of the existence of an over-constrained situation should be very explicit, as the users tend to ignore it and then become frustrated by (what they perceive as) system non-cooperativeness. We believe that more specific system-suggestions directly related to the search constraints provided by the users will be of great help for this issue.

- Mixed initiative is frequently used, but its extent of use varies significantly over the different application contexts (mGDNs). It is therefore recommended to perform for each particular system a detailed mixed initiative use analysis, in order to restrict the enabled degree of mixed initiative to the mGDN combinations where it is used only. This reduces the complexity and increases the robustness and reliability of the NL processing within the final fully automated system.
- The local strategies aiming at providing help to the users and at repeating the last system prompt if needed turned out to be useless for multimodal systems equipped with a screen.
- Although we benefited from optimal speech recognition and language understanding conditions (both performed by human operators through Wizard of Oz simulations), the system still turned out to be unable to process 5.7% of all user inputs provided via natural language. This happened either because the system lacked the required functionality or because the user's input was incomprehensible, even to human operators. We therefore consider the observed percentage as an interesting information on the theoretical performance limit of any interactive system integrating natural language.
- The simultaneous use of several modalities within one dialogue turn was extremely rare (0.3% of all dialogue turns), but all available modalities were quite frequently used, i.e. users switched modalities between dialogue turns. This suggests that our simple approach to modality fusion (relying on the assumption that SPs generated independently by individual modalities can be simply merged into one set) is adequate and sufficient for this type of systems.
- Natural language (voice and keyboard) was very frequently used: on average 61% and 46% of all interactions were done in natural language in the PVK and MVK conditions respectively). This provides an experimental justification of the efforts dedicated to the extension of GUI-only systems towards NL-enabled systems. However, the individual use (and thus user-perceived importance) of the NL modalities largely varied among the users, leading to a nearly uniform distribution, ranging from users controlling the system only through NL to users relying only on the mouse.
- Natural language was clearly preferred over pointing modalities (mouse, pen) *for providing search constraints* (i.e. entering data), while navigation within the graphical interface was most frequently done by pointing modalities. This suggests a general preferred role for language in multimodal interfaces, and indicates that existing vocal command-and-control desktop applications may not combine GUI and speech in a manner that is optimal and natural for users.

The second set of results is directly related to the Archivus system and thus constitutes useful contribution to the domain of meetings browsers:

- On average, the users were able to correctly answer 76% of the given tasks (consisting of a mixture of true-false and short-answer questions related to the meeting content), 15% of their answers were wrong, and in 9% of the cases they did not answer at all. More precisely, the average (correct, incorrect, non answered) percentages were (87%, 9%, 4%) for the true-false tasks and (65%, 21%, 15%) for the short-answer questions. Nevertheless, individual user accuracy varied in great range – 10% of the users correctly answered all the given tasks, while 19% of them answered correctly less than half of the tasks. Interestingly, the latter users also tended to have a larger proportion of unanswered tasks while their proportion of wrong answers is comparable with other users, indicating that they probably had difficulties to find the answers.
- It took users on average 127 seconds to answer a task, but nearly 30% of this time was spent in waiting for system response (including the time necessary for wizards to simulate some of system functions). With system response time excluded, the users answered a task in 91 seconds on average, and they should be able to solve on average 0.73 tasks per minute in a system with immediate responses.
- The analysis of the situations where users failed to correctly answer a given task (i.e. answered incorrectly or did not answer at all) revealed that the main reason (63%) for failure was *insufficient attention* or *misinterpretation* of the meeting transcripts provided by the system. Such a result suggests that the meeting transcript alone is not necessarily the optimal way for providing information about meetings. A possible improvement might be to either present the transcript in a 'smoothed form' (e.g. with discourse fillers removed) or to enrich it with additional information (e.g. make the topical classification of the displayed transcript part always visible).
- Users' subjective opinion of the Archivus system (and experienced interactions) in terms of system usability, understandability, learnability, and satisfaction with the system was in general *positive* or *very positive*. In our view, this is a very convincing indication of the adequacy of our approach. In addition, the only identified problematic aspect (the speed of the system) is inherent to the used WOz simulations. This indicates the importance of efficient wizard control interfaces and the need for proper wizard's training, but should disappear (or be significantly reduced) once the targeted system is fully automated and optimally implemented.

Chapter 7

Conclusions

This thesis provides three main contributions:

- A novel, experimentally validated approach for the integration of multimodal inputs and outputs within interactive dialogue-based systems.
- A methodology for the rapid design, prototyping and implementation of multimodal dialogue-based information seeking systems. The proposed methodology builds upon an iterative design process based on Wizard of Oz (WOz) simulations and is supported by an underlying software infrastructure (the prototyping platform), which can be easily customized for new application domains and includes ready-to-use wizard control interfaces for the simulation of not yet implemented parts of the targeted system.
- An operational prototype (the Archivus system) for a novel application – information seeking and browsing in multimedia meeting data (the Smart Meeting Room application). The uniqueness of the Archivus system lies in the application of a multimodal dialogue-driven interface to the already fairly novel domain of interaction.

Each of the three contributions is discussed in more detail in the following sections.

7.1 Multimodal dialogue-based system design

Our research first focused on the extension of dialogue-based systems with multimodal inputs and outputs (Chapter 3). The results presented in Chapter 6 illustrate how our approach allows for the smooth integration of natural language within a traditional GUI paradigm extended with more novel modalities (such as a tactile screen).

For the dialogue design, we propose a *two-layered dialogue model* that corresponds to an extension of the standard frame-based approach. In our work, the first dialogue layer (modeled with generic dialogue nodes – GDNs) defines the interaction within specific dialogue contexts and based on local dialogue strategies associated with the individual slots of a global dialogue frame. The second layer controls the overall dialogue progress and relies on global dialogue strategies that provide a higher-level planning of the dialogue.

Through experiments with the Archivus system we have shown that our two-layered approach to dialogue modelling is *easily extendable to multimodal systems*. The key idea was to extend existing GDNs to allow for a multimodal interaction with the user, thus leading to the notion of multimodal generic dialogue node (mGDN). The modifications required by the migration from vocal-only to multimodal dialogue management were explained in section 3.8.

A general evaluation of the proposed global and local dialogue strategies in a multimodal environment, presented in sections 6.3 and 6.4 for the Archivus system, has shown that (1) mixed initiative features are frequently used by users, though not necessarily to the same extent in all mGDNs, (2) the proposed strategy for early automatic presentation of possible search results is well-accepted by users, (3) users hardly made use of the help and repeat local dialogue strategies, and (4) despite optimal speech recognition and understanding conditions (both performed by human operators through Wizard of Oz simulations), the system is still unable to process 5.7% of all user inputs provided via natural language, which provides interesting information on the theoretical performance limit of any interactive system integrating natural language.

Furthermore, the analysis of the use of modalities (section 6.5) has shown (1) that the simultaneous use of several modalities (within one dialogue turn) is extremely rare (0.3% of all dialogue turns), while all available modalities were quite frequently used (i.e. users switched modalities between dialogue turns), thus suggesting that a quite simplistic approach to modality fusion, as the one used in Archivus, is in fact sufficient for most of the cases, and (2) natural language is clearly preferred over pointing modalities (mouse, pen) for providing search constraints (i.e. entering data), while navigation within the graphical interface is most frequently done by pointing modalities. This suggests a general preferred role for language in multimodal interfaces, and indicates that existing vocal command-and-control desktop applications may not combine GUI and speech in a manner that is optimal and natural for users.

7.2 Rapid prototyping with WOz simulations

The development of multimodal systems represents a significant amount of work. We addressed this issue by proposing a methodology that specifically focuses on *rapid design and implementation* of multimodal dialogue-based information seeking systems. The methodology is *generic* in the sense that new applications can easily be created thanks to the fact that the underlying application model is conceptually modular, with mGDNs as its main building blocks.

Furthermore, we complemented our methodology with an underlying software infrastructure and a number of reusable system modules. For the system infrastructure (*the prototyping platform*), we proposed a variant of the plugin approach (section 3.6), where the system modules can be either used directly, or customized to the needs of a given application. In general, our plugin system is intended to run as a single process on one machine, thus making debugging and development simple. However, if needed, the system's graphical interfaces can be easily distributed over several machines using the VNC protocol and a standard VNC client.

The VNC-based distribution of system modules is required for *Wizard of Oz simulations*, which we proposed for iterative testing and evaluation of multimodal systems under development.

Although the idea of using WOz simulations for the design of multimodal systems is not new, to the best of our knowledge, there is no available literature describing the experimental environment in detail. Therefore, our contribution consists of three main parts: (1) the detailed description and analysis of the requirements for the experimental environment, (2) an experimentally validated integration of WOz techniques within a generic design methodology and (3) practical recommendations for carrying out WOz simulations. A major part of our environment is generic enough to be used for testing of any multimodal system, not necessarily only of those designed with our methodology.

7.3 Archivus system

The main goal of the Archivus system was to serve as a central use-case for our prototyping methodology, allowing us to draw conclusions about its effective *applicability* and about the *acceptability* of the created systems, based on experiment with a large number of users.

Archivus also corresponds to a very novel application: multimodal dialogue-based information search within a multimedia database of annotated meeting recordings (the Smart Meeting Room application). The novelty of the application made it an interesting candidate for research on multimodal interfaces, as it was difficult to predict how users would use the provided interface for tasks they were not familiar with.

The methodology *applicability* is *validated* by the fact that the Archivus was created by the proposed methodology and is now a working system prototype. The iterative development process of Archivus is thoroughly described in Chapter 5.

The functionality and quality of the developed Archivus system were experimentally evaluated by means of WOz simulations. In this perspective, one of the main results is that on average the users were able to solve 91% of the given tasks, which we consider as a *very positive indication about the effectivity* of applications created by our ISPM (especially when taking into account the Archivus novelty). Furthermore, the fact that 15% of the tasks were on average answered incorrectly should not be considered as contradicting the former assumption, because our further analysis showed that the main reason (63%) for incorrect or no answer was the *insufficient attention* paid by the users to the meeting transcript (provided to them by the system), and could also be partially explained by the fact that the participants in our experiments were mostly non-native English speakers. This however suggests that meeting transcripts alone are not necessarily the optimal way of providing information about meetings.

Finally, concerning the acceptability of the Archivus system, the generally *positive or very positive subjective user's opinion* of the system (in terms of system usability, understandability, learnability, and satisfaction) is in our view a *very convincing indication of the adequacy of our approach.* The only identified problematical aspect, the speed of the system, should again not be considered as contradicting this assumption, as this problem is inherent to the used WOz simulations (a considerable delays in system responses are caused by Wizards and by the fact that the system is an imperfectly implemented prototype) and should disappear or be significantly reduced once the targeted system is fully automated and optimally implemented.

Chapter 8

Future work

The possible future directions resulting from our work can be split into two main categories: the ones concerning the developed methodology and the underlying dialogue model design (section 8.1), and the ones related to the possible extensions of the Archivus system (section 8.2).

8.1 Multimodal dialogue system design

8.1.1 Facilitate the dissemination of the methodology

The main purpose of the methodology described in this thesis is to facilitate the design and development of multimodal dialogue-based systems. As already mentioned in the conclusion chapter, we believe that the experimental results obtained during the development and evaluation of the Archivus system represent a convincing illustration of the very promising potential of our approach. However, the *implementation* of the methodology still needs to be improved and well documented in order to become an efficient and easy to use tool for people who did not participate to its development. Our intention is to make it available as an opensource software in order to further contribute to the dissemination of our work. However, the *ease of development* of new systems in other domains still remains to be tested, because so far, we have only some indications about the approach adequacy based on our experience with designing systems using earlier versions of the ISPM.

8.1.2 Improvements in the dialogue model

The development of systems for new application domains (possibly involving new types of modalities and interaction styles that were not yet tested in our framework), might reveal unforeseen *shortcomings of the underlying dialogue model* or uncover parts of the methodology implementation that should be made more flexible, for example:

– Interactive systems developed with our methodology might require the adjunction of some kind of *'undo'* (go back) feature that allows to discard the effect of the last performed action and to backtrack to the previous state of the system. Although a similar effect can often be achieved by a proper manipulation of the set of search constraints, using an 'undo' functionality is more straightforward and applicable in many situations.

We are therefore convinced that an 'undo' functionality should be added to our model, as several of the Archivus users have explicitly stated that they missed an undo button. Notice that providing such a functionality is far from being a trivial task: even if adding the possibility to backtrack to the previously visited mGDN is relatively easy (e.g. by memorizing its ID), substantial efforts are necessary to guarantee that that the full state of the previously visited mGDN is correctly reconstructed.

– The interactions enabled by our system model are strictly turn-based (i.e. the user's and the system's turns are strictly alternating). Although this is a quite desirable feature for WOz simulations (as there is a clear separation between the periods when the user or the system/wizard are in charge of the interaction control), it complicates the implementation of standard operations such as the *'smooth scrolling'* of long lists for which the fine-grained motion of the mouse dragging actions would require a system reaction after every mouse move (to ensure a really smooth scrolling), which is incompatible with the turn-based approach.

  For such operations requiring real-time response to very small changes in the system state, the underlying interaction model would need to be extended. A possible solution would be to define a new class of semantic pairs to be processed by the mGDNs in real-time without interrupting the turn-based higher level interpretation of the dialogue. Such semantic pairs should not be supervised by wizards directly, but wizards should only have the possibility to interrupt the 'low level' stream of semantic pairs, for instance in situations when 'freezing' the user interface would be necessary to ensure a coherent interpretation of user's inputs.

– The part of the task model defining the *relations between mGDNs* (see section 3.5.1) might be made more generic. Indeed, in its current state, it was sufficient to model the relationships between mGDNs during the constraints gathering phase of the dialogue, but this appeared to be insufficient to describe the possible transitions among mGDNs during the result-browsing phase. Currently, such transitions must be manually encoded in the Java source code of the dialogue manager (see the discussion in section 5.2.3 on page 89).

  Therefore, it would be useful to define a formalism that enables to express functional relationships between mGDNs that could be exploited within the dialogue strategies. A possible idea would be to use a formalism similar to the concurrent task trees (CTT [74, 73]).

– Another feature that might be further enhanced are the *conditional prompts* (i.e. prompts associated with a logical expression describing the situation in which the prompt should be used). Indeed, conditional prompts have been identified as very useful to (1) either add more context-sensitivity within mGDNs (the context being mainly defined by the mGDNs themselves, but refined by the associated conditions), or to (2) signalize some global interaction situations (e.g. the presence of a problem, of new solutions, confirmations) that are not necessarily related to the mGDN in focus. While the former (mGDN-related) conditional prompts can be declaratively defined in the mGDN configuration files, the latter (dialogue situation related) conditional prompts currently have to be directly encoded in the dialogue manager.

An interesting extension would therefore be to define a formalism allowing the efficient, declarative definitions of *all* conditional prompts and the associated conditions.

8.1.3 Techniques for automation of the targeted systems

The design of multimodal dialogue systems might be further improved and accelerated by providing techniques for the *automation of the wizarding tasks*. For example, an interesting research direction for the automation of *the output wizard decisions* would be to use machine learning techniques to learn rules for the modifications of the default system prompts, exploiting the logfiles resulting from WOz experiments that keep track of all the modified prompts and corresponding dialogue states.

8.1.4 Some additional more ambitious goals

Our prototyping methodology was created for designing information seeking systems that can rely on the type of domain model specified in section 3.2. Therefore, it would be interesting to study whether the methodology and the associated dialogue strategies can be adapted to systems that rely on other types of domain models (i.e. not information seeking systems). Such a study would generalize our methodology to a broader class of interactive systems, where the main limitation is that the interaction can be decomposed into the mGDNs – our elementary interaction units.

Another possible generalization concerns approaches beyond the frame-based dialogue models. In particular, frame-based systems rely on implicitly predefined relations between slots. However, certain user queries might require dynamically created relations between the provided information. For instance, a user query like: *"I would like to stay in a cheap hotel or I can accept even more expensive hotel, but then it has to be directly on the beach"* would need the system to interpret it as: `price:low OR (price:higher AND location:beach)`. A system capable of processing such queries would of course be more powerful, while special attention should be then given to the robustness of the interaction, as the reformulations and alternations of the attribute-relations can easily become cumbersome.

8.2 Archivus system

Although we have evaluated the performance of the Archivus system in terms of speed and accuracy, the obtained results are difficult to compare with other meeting browsers, mainly because of different experimental settings. Therefore, the Archivus browser should also be evaluated in a comparative framework, such as, for example, the Browser Evaluation Test (BET) using the experimental protocol described in [121, 88]. Such comparative evaluation might (1) reveal the types of information search tasks for which Archivus is best suited, (2) suggest the most useful meeting presentation techniques used in Archivus, and (3) confirm the added value brought by multimodal dialogue-based interfaces.

Another interesting research direction is the study of *the usefulness for meeting data searching and browsing* of various metadata used for annotating the Archivus database. For example, the usefulness of specific annotations for *searching* might be determined by analyzing user queries, while the usefulness of annotations for *browsing* might be partially evaluated by an analysis of access frequencies to meeting-book pages with different annotations, possibly with the help of some *eye-tracking devices*, which, in addition, might also reveal the most often viewed GUI elements within the Archivus screen (thus suggesting their importance and a possible need for reorganization or resize).

Finally, the ultimate step is to automate the Archivus system and thus make it *independent of wizard simulations and supervisions.* The automation of wizarding tasks such as prompt modifications (mentioned in the previous section) or NL understanding are crucial research issues to be considered in the future. The impact of the system automation on the user and the system performance has to also be analyzed.

Bibliography

[1] Alicia Abella, Michael K. Brown, and Bruce Buntschuh. Development principles for dialog-based interfaces. In *ECAI'96: Workshop on Dialogue Processing in Spoken Language Systems*, pages 141–155. Springer-Verlag, 1997.

[2] James F. Allen. *Natural Language Understanding*. Addison-Wesley, Reading, Massachusetts, USA, 2 edition, 1995.

[3] Masahiro Araki. OWL-based frame descriptions for spoken dialog systems. In *Proceedings of International Workshop on Semantic Web Foundations and Application Technologies*, Nara Prefecture Public Hall, Nara, Japan, March 2003.

[4] Maria Aretoulaki and Bernd Ludwig. Automaton-descriptions and theorem-proving: a marriage made in heaven? In *ETAI: News Journal on Intelligent User Interfaces*, volume 10, 1999. Special Issue on Intelligent Dialogue Systems edited by Jan Alexandersson, Lars Ahrenberg, Kristiina Jokinen and Arne Jönsson.

[5] Harald Aust, Martin Oerder, Frank Seide, and Volker Steinbiss. The Philips automatic train timetable information system. *Speech Communication*, 17(3-4):249–262, 1995.

[6] Harald Aust and Olaf Schröer. An overview of the Philips dialog system. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne Conference Resort, Lansdowne, Virginia, USA, February 1998.

[7] John L. Austin. *How to Do Things with Words*. Oxford University Press, 1962.

[8] Eric Bilange. *Dialogue personne-machine: modélisation et réalisation informatique*. Hermes, Paris, 1992.

[9] Dan Bohus and Alex Rudnicky. LARRI: a language-based maintenance and repair assistant. In Wolfgang Minker, Dirk Bühler, and Laila Dybkjær, editors, *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, volume 28 of *Text, Speech and Language Technology*, pages 203–218. Springer, 2005.

[10] Heleen Boland, Jettie Hoonhout, Claudia van Schijndel, Jan Krebber, Mirek Melichar, Dietmar Schuchardt, Hardy Baesekow, Rosa Pegam, Sebastian Möller, Martin Rajman, and Paula Smeele. Turn on the lights: investigating the Inspire voice controlled Smart Home system, October 2004.

[11] Richard A. Bolt. "Put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA, 1980. ACM.

[12] Ivan Bretan and Jussi Karlgren. Transparent natural language interaction through multimodality. In *ERCIM Workshop on Multimodal HCI*, 1993.

[13] Trung H. Bui, Martin Rajman, and Miroslav Melichar. Rapid Dialogue Prototyping Methodology. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *Proceedings of the 7th International Conference on Text, Speech and Dialogue (TSD 2004)*, Lecture Notes in Artificial Intelligence LNCS/LNAI 3206, pages 579–586, Brno, Czech Republic, September 2004. Springer-Verlag.

[14] Harry C. Bunt. Conversational principles in question-answer dialogues. In Dieter Krallmann and Gerhard Stickel, editors, *Zur Theorie der Frage*, pages 119–141. Narr Verlag, 1981.

[15] Harry C. Bunt. Information dialogues as communicative action in relation to partner modelling and information processing. In M.M. Taylor, F. Neel, and D.G. Bouwhuis, editors, *The Structure of Multimodal Dialogue*, pages 47–73. North-Holland, Amsterdam, 1989.

[16] Pedro Cardoso, Luis Flores, Thibault Langlois, and Joäo Neto. Meteo: A telephone-based portuguese conversation system in weather domain. In *Proceedings of the Third International Conference on Advances in Natural Language Processing*, pages 175–178, Faro, Portugal, June 2002. Springer-Verlag.

[17] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume lathoud, Mike Lincoln, Agnes Lisowska, Iain McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. The AMI meeting corpus: A pre-announcement. In *Machine Learning for Multimodal Interaction*, volume 3869/2006 of *Lecture Notes in Computer Science*, pages 28–39, Edinburgh, UK, February 2006. Springer Berlin / Heidelberg.

[18] Pavel Cenek, Miroslav Melichar, and Martin Rajman. A Framework for Rapid Multimodal Application Design. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *Proceedings of the 8th International Conference on Text, Speech and Dialogue (TSD 2005)*, volume 3658 of *Lecture Notes in Computer Science*, pages 393–403, Karlovy Vary, Czech Republic, September 12-15 2005. Springer.

[19] Corey D. Chandler, Gloria Lo, and Anoop K. Sinha. Multimodal theater: extending low fidelity paper prototyping to multimodal applications. In *CHI '02 extended abstracts on Human factors in computing systems*, pages 874–875, New York, NY, USA, 2002. ACM.

[20] Hua Cheng, Harry Bratt, Rohit Mishra, Elizabeth Shriberg, Sandra Upson, Joyce Chen, Fuliang Weng, Stanley Peters, Lawrence Cavedon, and John Niekrasz. A Wizard of Oz Framework for Collecting Spoken Human-Computer Dialogs. In *Proceedings of INTERSPEECH 2004 - ICSLP, The 8th International Conference on Spoken Language Processing*, pages 2269–2272, Jeju Island, Korea, 2004.

[21] Phil Cohen. Dialogue modeling. In Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, editors, *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, Cambridge, 1997.

[22] Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, editors. *Survey of the state of the art in human language technology*, chapter Chapter 13. Evaluation. Cambridge University Press, New York, NY, USA, 1997.

[23] Don Colton, Ron Cole, David G. Novick, and Stephen Sutton. A laboratory course for designing and testing spoken dialogue systems. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1129–1132, May 1996.

[24] Anita Cremers, Wilfried Post, Erwin Elling, Betsy van Dijk, Bram van der Wal, Jean Carletta, Mike Flynn, Pierre Wellner, and Simon Tucker. Meeting browser evaluation report. Technical report, AMI Project Deliverable D6.4, December 2006.

[25] Nils Dahlbäck, Annika Flycht-Eriksson, Arne Jönsson, and Pernilla Qvarfordt. An architecture for multi-modal natural dialogue systems. In *ESCA Tutorial and Research Workshop (ETRW) on Interactive Dialogue in Multi-Modal Systems*, 1999.

[26] Nils Dahlbäck and Arne Jönsson. An empirically based computationally tractable dialogue model. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society (COG SCI-92)*, Bloomington, Indiana, 1992.

[27] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. Wizard of Oz Studies – Why and How. In Wayne D. Gray, William Hefley, and Dianne Murray, editors, *International Workshop on Intelligent User Interfaces 1993*, pages 193–200. ACM Press, 1993.

[28] Jody J. Daniels and Susan McGrath. A spoken language interface for tasking agents. In *Proceedings of the Grace Hopper Celebration of Women in Computing Conference*, Hyannis, Massachusetts, USA, September 2000.

[29] Dan Diaper. The wizard's apprentice: a program to help analyse natural language dialogues. In *Proceedings of the fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and computers V*, pages 231–243, New York, NY, USA, 1989. Cambridge University Press.

[30] Alan Dix, Janet Finley, Gregory Abowd, and Russell Beale. *Human-computer interaction (2nd ed.)*. Prentice-Hall, Inc., 1998.

[31] Mary Ellen Foster. State of the art review: Multimodal fission. Deliverable 6.1 of the COMIC project, September 2002.

[32] Norman Fraser and Nigel Gilbert. Simulating speech systems. *Computer Speech and Language*, 5:81–99, 1991.

[33] Jeanne C. Fromer. Learning optimal discourse strategies in a spoken dialogue system. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, September 1998.

[34] Petra Geutner, Frank Steffens, and Dietrich Manstetten. Design of the VICO Spoken Dialogue System: Evaluation of User Expectations by Wizard-of-Oz Experiments. In *Proceeedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, 2002.

[35] David Goddeau, Helen Meng, Joe Polifroni, Stephanie Seneff, and Senis Busayapongchai. A form-based dialogue manager for spoken language applications. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'96)*, volume 2, pages 701–704, Philadelphia, Pennsylvania, USA, 1996.

[36] Hilda Hardy, Tomek Strzalkowski, Min Wu, Cristian Ursu, Nick Webb, Alan Biermann, Bryce Inouye, and Ashley McKenzie. Data-driven strategies for an automated dialogue system. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 71–78, Barcelona, Spain, July 2004.

[37] Alexander Hauptmann. Speech and gestures for graphic image manipulation. In *Proc. ACM CHI'89 Conference on Human Factors in Computing Systems*, pages 241–245, 1989.

[38] Julia Hirschberg, Diane Litman, and Marc Swerts. Prosodic cues to recognition errors. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU'99)*, 1999.

[39] Judith Hochberg, Nanda Kambhatla, and Salim Roukos. A flexible framework for developing mixed-initiative dialog systems. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 60–63, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[40] Kristiina Jokinen and Antti Raike. Multimodality - technology, visions and demands for the future. In *1st Nordic Symposium on Multimodal Interfaces*, 2003.

[41] Arne Jönsson and Nils Dahlbäck. Talking to a computer is not like talking to your best friend. In *Scandinivian Conference on Artificial Intelligence (SCAI)*, pages 53–68, March 1988.

[42] Daniel Jurafsky, Chuck Wooters, Gary Tajchman, Jonathan Segal, Andreas Stolcke, Eric Fosler, and Nelson Morgan. The Berkeley Restaurant Project. In *ICSLP'94*, pages 2139–2142, Yokohama, Japan, 1994.

[43] Demetrios Karis and Kathryn M. Dobroth. Automating services with speech recognition over the public switched telephone network: Human factors considerations. *IEEE Journal on Selected Areas in Communications*, 9(4):574–585, 1991.

[44] Andreas Kellner, Bernhard Rueber, Frank Seide, and Bach-Hiep Tran. Padis – an automatic telephone switchboard and directory information system. *Speech Communication*, 23(1-2):95–111, 1997.

[45] Alexandra Klein, Ingrid Schwank, Michel Généreux, and Harald Trost. Evaluating multi-modal input modes in a Wizard-of-Oz study for the domain of web search. In *Proceedings of the HCI01 Conference on People and Computers XV*, pages 475–484. Springer, 2001.

[46] Ivan Kopeček. Modeling of the information retrieval dialogue systems. In Václav Matoušek, Pavel Mautner, Jana Ocelíková, and Petr Sojka, editors, *Proceedings of Text, Speech and Dialogue: Second International Workshop*, Lecture Notes in Artificial Intelligence LNCS/LNAI 1692, pages 302–307, Plzeň, Czech Republic, September 1999. Springer-Verlag.

[47] Emiel Krahmer. The science and art of voice interfaces. Research report, Philips Research, Eindhoven, Netherlands, 2001.

[48] Emiel Krahmer, Marc Swerts, Mariët Theune, and Mieke Weegels. Error detection in spoken human-machine interaction. *International journal of speech technology*, 4(1):19–30, 2001.

[49] Jan Krebber, Sebastian Möller, Rosa Pegam, Ute Jekosch, Miroslav Melichar, and Martin Rajman. Wizard-of-Oz tests for a dialog system in Smart Homes. In *Proceedings of the joint congress CFA/DAGA*, Strasbourg, France, 2004.

[50] L. Lamel, S. Rosset, J.L. Gauvain, S. Bennacef, M. Garnier-Rizet, and B. Prouts. The LIMSI ARISE system. *Speech Communication*, 31(4):339–354, August 2000.

[51] Pat Langley, Cynthia A. Thompson, Renee Elio, and Afsaneh Haddadi. An adaptive conversational interface for destination advice. In *Cooperative Information Agents*, pages 347–364, 1999.

[52] James Alan Larson. *Interactive Software: Tools for Building Interactive User Interface.* Prentice-Hall, Inc., 1991.

[53] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of computer-human interaction for learning dialogue strategies. In *IEEE Automatic Speech Recognition and Understanding Workshop*, Santa Barbara, USA, 1997.

[54] Esther Levin, Roberto Pieraccini, Wieland Eckert, Pino Di Fabbrizio, and Shrikanth Narayanan. Spoken language dialogue: From theory to practice. In *IEEE Automatic Speech Recognition and Understanding Workshop*, Keystone, Colorado, USA, December 12 -15 1999.

[55] Hank Liao. Multimodal fusion. Master's thesis, University of Cambridge, 2002.

[56] Agnes Lisowska. Multimodal interface design for the multimodal meeting domain: Preliminary indications from a query analysis study. Project report IM2.MDM-11, University of Geneva, Geneva, Switzerland, November 2003.

[57] Agnes Lisowska. *Multimodal Interface Design for Multimedia Meeting Content Retrieval.* PhD thesis, University of Geneva, Switzerland, September 2007.

[58] Agnes Lisowska, Susan Armstrong, Mireille Betrancourt, and Martin Rajman. Minimizing modality bias when exploring input preferences for multimodal systems in new domains: the Archivus case study. In *CHI '07 extended abstracts on Human factors in computing systems*, pages 1805–1810, New York, NY, USA, 2007. ACM.

[59] Agnes Lisowska, Andrei Popescu-Belis, and Susan Armstrong. User query analysis for the specification and evaluation of a dialogue processing and retrieval system. In *Procedings of the LREC 2004 international conference*, pages 993–996, Lisbon, Portugal, May 26-28 2004.

[60] Agnes Lisowska, Martin Rajman, and Trung H. Bui. ARCHIVUS: A System for Accessing the Content of Recorded Multimodal Meetings. In *In Procedings of the JOINT AMI/PASCAL/IM2/M4 Workshop on Multimodal Interaction and Related Machine Learning Algorithms, Bourlard H. & Bengio S., eds. (2004), LNCS, Springer-Verlag, Berlin.*, Martigny, Switzerland, June 2004.

[61] Diane J. Litman and James F. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11(2):163–200, 1987.

[62] Kent Lyons, Christopher Skeels, and Thad Starner. Providing Support for Mobile Calendaring Conversations: A Wizard of Oz Evaluation of Dual–Purpose Speech. In *MobileHCI '05: Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices & Services*, pages 243–246, New York, NY, USA, 2005. ACM Press.

[63] David L. Martin, Adam Cheyer, and Douglas B. Moran. The Open Agent Architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2):91–128, 1999.

[64] Michael F. McTear. Spoken dialogue technology: Enabling the conversational user interface. *ACM Computing Surveys*, 34(1):90–169, 2002.

[65] Michael F. McTear, Susan Allen, Laura Clatworthy, Noelle Ellison, Colin Lavelle, and Helen McCaffery. Integrating flexibility into a structured dialogue model: Some design considerations. In *Proceedings of the 6th International Conference on Spoken Language Processing*, pages 110–113, Beijing, China, November 2000.

[66] Miroslav Melichar. A tool for rapid dialogue prototyping (in Czech). Master's thesis, Masaryk University Brno, Czech Republic, 2003.

[67] Miroslav Melichar, Pavel Cenek, Marita Ailomaa, Agnes Lisowska, and Martin Rajman. From vocal to multimodal dialogue management. In *Eighth International Conference on Multimodal Interfaces (ICMI'06)*, pages 59–67, Banff, Alberta, Canada, November 2-4 2006.

[68] Miroslav Melichar, Agnes Lisowska, Susan Armstrong, and Martin Rajman. Rapid multimodal dialogue design: Application in a multimodal meeting retrieval and browsing system. In *Poster presentation at MLMI'05*, Edinburgh, UK, July 11-13 2005.

[69] Helen Meng, Senis Busayapongchai, James Glass, Dave Goddeau, Lee Hetherington, Ed Hurley, Christine Pao, Joe Polifroni, Stephanie Seneff, and Victor Zue. WHEELS: A conversational system in the automobile classifieds domain. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'96)*, volume 1, pages 542–545, Philadelphia, Pennsylvania, USA, 1996.

[70] Marvin Minsky. Frame system theory. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 104–116. Association for Computational Linguistics, 1975.

[71] Sebastian Möller, Jan Krebber, Alexander Raake, Paula Smeele, Martin Rajman, Mirek Melichar, Vincenzo Pallotta, Gianna Tsakou, Basilis Kladis, Anestis Vovos, Jettie Hoonhout, Dietmar Schuchardt, Nikos Fakotakis, Todor Ganchev, and Ilyas Potamitis. INSPIRE: Evaluation of a Smart-Home System for Infotainment Management and Device Control. In *International Conference on Language Resources and Evaluation (LREC)*, volume 5, pages 1603–1606, Lisbon, Portugal, 2004.

[72] Darren Moore. The IDIAP Smart Meeting Room. In *IDIAP-Com 02-07*, page 13, 2002.

[73] Giulio Mori, Fabio Paterno, and Carmen Santoro. CTTE: support for developing and analyzing task models for interactive system design. *IEEE Trans. Softw. Eng.*, 28(8):797–813, 2002.

[74] Giulio Mori, Fabio Paterno, and Carmen Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, 30(8):507–520, 2004.

[75] Mikio Nakano, Yasuhiro Minami, Stephanie Seneff, Timothy J. Hazen, Scott Cyphers, James Glass, Joseph Polifroni, and Victor Zue. Mokusei: A telephone-based japanese conversational system in the weather domain. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 1331–1334, Aalborg, Denmark, September 2001.

[76] Laurence Nigay and Joëlle Coutaz. A generic platform for addressing the multimodal challenge. In *CHI'95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 98–105. ACM Press/Addison-Wesley Publishing Co., 1995.

[77] Scott P. Overmyer. Revolutionary vs. evolutionary rapid prototyping: Balancing software productivity and HCI design concerns. In *Proceedings of the Fourth International Conference on Human-Computer Interaction*, pages 303–307. Elsevier Science, 1991.

[78] Sharon Oviatt. Multimodal interactive maps: Designing for human performance. *Human-Computer Interaction*, 12:93–129, 1997.

[79] Sylvain Paillard. Extensions of rapid dialogue prototyping methodology. Semester project report, EPFL, February 2005.

[80] Sylvain Paillard. Venus – a dialogue-based virtual receptionist. Master's thesis, Swiss Federal Institute of Technology in Lausanne (EPFL), March 2006.

[81] Vincenzo Pallotta. Computational dialogue models. MDM research project deliverable, Faculty of Computer and Communication Sciences, Swiss Federal Institute of Technology, EPFL IC-ISIM LITH, IN-F Ecublens, 1015 Lausanne (CH), March 2003.

[82] Bryan Pellom, Wayne Ward, John Hansen, Kadri Hacioglu, Jianping Zhang, Xiuyang Yu, and Sameer Pradhan. University of Colorado dialog systems for travel and navigation. In *Proceedings of the Human Language Technology Conference (HLT-2001)*, San Diego, California, USA, March 2001.

[83] Bryan Pellom, Wayne Ward, and Sameer Pradhan. The CU communicator: An architecture for dialogue systems. In *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China, November 2000.

[84] Norbert Pfleger. Context based multimodal fusion. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 265–272, New York, NY, USA, 2004. ACM Press.

[85] Philips Dialogue Systems, Aachen, Germany. *SpeechMania 2.0: HDDL Reference Manual*, 1997.

[86] Roberto Pieraccini, Esther Levin, and Wieland Eckert. AMICA: The AT&T mixed initiative conversational architecture. In *Eurospeech'97*, pages 1875–1878, Rhodes, Greece, 1997.

[87] Olivier Pietquin and Thierry Dutoit. Aided design of finite-state dialogue management systems. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME 2003)*, Baltimore, Maryland, USA, July 2003.

[88] Andrei Popescu-Belis, Philippe Baudrion, Mike Flynn, and Pierre Wellner. Towards an objective test for meeting browsers: The BET4TQB pilot experiment. In *Machine Learning for Multimodal Interaction*, volume 4892/2008 of *Lecture Notes in Computer Science*, pages 108–119, Brno, Czech Republic, February 2008. Springer Berlin / Heidelberg.

[89] David Portabella. *Improving user confidence in decision support systems for electronic catalogs*. PhD thesis, EPFL, Switzerland, December 2007.

[90] Alexandros Potamianos, Egbert Ammicht, and Hong-Kwang Kuo. Dialogue management in the Bell Labs communicator system. In *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China, October 2000.

[91] Silvia Quarteroni, Martin Rajman, and Miroslav Melichar. Introducing reset patterns: An extension to a rapid dialogue prototyping methodology. In *The IEE International Workshop on Intelligent Environments*, University of Essex, Colchester, UK, June 28-29 2005.

[92] Martin Rajman, Marita Ailomaa, Agnes Lisowska, Miroslav Melichar, and Susan Armstrong. Extending the Wizard of Oz methodology for language-enabled multimodal systems. In *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy, May 2006.

[93] Martin Rajman, Trung H. Bui, Andréa Rajman, Florian Seydoux, Alex Trutnev, and Silvia Quarteroni. Assessing the usability of a dialogue management system designed in the framework of a rapid dialogue prototyping methodology. *ACTA ACUSTICA united with ACUSTICA, the Journal of the European Acoustics Association (EAA): International Journal on Acoustics. Vol. 90, no. 6 pp. 1096-1111 (Nov./Dec. 2004) - ISSN 1610-1928 S. Hirzel Verlag - Stuttgart*, 2004.

[94] Martin Rajman, Andréa Rajman, Florian Seydoux, and Alex Trutnev. Assessing the usability of a dialogue management system designed in the framework of a rapid dialogue prototyping methodology. In *First ISCA Tutorial & Research Workshop on Auditory Quality of Systems*, Akademie Mont-Cenis, April 23-25 2003.

[95] Martin Rajman, Andréa Rajman, Florian Seydoux, and Alex Trutnev. Prototypage rapide et évaluation de modèles de dialogue finalisés. In *Traitement Automatique des Langues Naturelles (TALN)*, Batz-sur-Mer, June 11-14 2003.

[96] Manny Rayner. *Abductive Equivalential Translation and its Application to Natural Language Database Interfacing.* PhD dissertation, University of Stockholm, September 1993.

[97] Leah M. Reeves, Jennifer Lai, James A. Larson, Sharon Oviatt, T.S. Balaji, Stéphanie Buisine, Penny Collings, Phil Cohen, Ben Kraal, Jean-Claude Martin, Michael McTear, TV Raman, Kay M. Stanney, Hui Su, and Qian Ying Wang. Guidelines for multimodal user interface design. *Communications of the ACM*, 47(1):57–59, 2004.

[98] Mary Beth Rosson and John M. Carroll. Scenario-based design. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 1032–1050. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 2003.

[99] Alexander I. Rudnicky. Multimodal dialogue systems. In Wolfgang Minker, Dirk Bühler, and Laila Dybkjær, editors, *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, volume 28 of *Text, Speech and Language Technology*, pages 3–11. Springer, 2005.

[100] Michael Ruflin. Contributions to the Wizard of Oz experiments (Application editor). Semester project report, EPFL, June 2005.

[101] Daniel Salber and Joëlle Coutaz. Applying the Wizard of Oz technique to the study of multimodal systems. In *EWHCI '93: Selected papers from the Third International Conference on Human-Computer Interaction*, pages 219–230, London, UK, 1993. Springer-Verlag.

[102] Daniel Salber and Joëlle Coutaz. A Wizard of Oz platform for the study of multimodal systems. In *CHI '93: INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, pages 95–96, New York, NY, USA, 1993. ACM.

[103] John R. Searle. *Speech Acts*. Cambridge University Press, 1969.

[104] Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. Galaxy-II: A reference architecture for conversational system development. In *Proc. ICSLP'98*, volume 3, pages 931–934, 1998.

[105] Stephanie Seneff and Joseph Polifroni. A new restaurant guide conversational system: Issues in rapid prototyping for specialized domains. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'96)*, volume 2, pages 665–668, Philadelphia, Pennsylvania, USA, 1996.

[106] Stephanie Seneff and Joseph Polifroni. Dialogue management in the Mercury flight reservation system. In *Proceedings of ANLP-NAACL Workshop on Conversational Systems*, pages 1–6, Seattle, Washington, USA, April 2000.

[107] Nicole Shechtman and Leonard M. Horowitz. Media inequality in conversation: How people behave differently when interacting with computers and people. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–288, New York, NY, USA, 2003. ACM.

[108] Ben Shneiderman. Natural vs. precise concise languages for human operation of computers: Research issues and experimental approaches. In *Proceedings of the 18th annual meeting on Association for Computational Linguistics*, pages 139–141, Morristown, NJ, USA, 1980. Association for Computational Linguistics.

[109] Bernd Souvignier, Andreas Kellner, Bernhard Rueber, Hauke Schramm, and Frank Seide. The thoughtful elephant: Strategies for spoken dialog systems. *IEEE Transactions on Speech and Audio Processing*, pages 51–62, 2000.

[110] Helmer Strik, Albert Russel, Henk van den Heuvel, Catia Cucchiarini, and Lou Boves. A spoken dialogue system for public transport information. In H. Strik, N. Oostdijk, C. Cucchiarini, and P.A. Coppen, editors, *Proceedings of the Department of Language and Speech*, volume 19, pages 129–142, Nijmegen, The Netherlands, June 1996.

[111] Stephen Sutton, Ronald Cole, Jacques de Villiers, Johan Schalkwyk, Pieter Vermeulen, Mike Macon, Yonghong Yan, Ed Kaiser, Brian Rundle, Khaldoun Shobaki, Paul Hosom, Alex Kain, Johan Wouters, Dominic Massaro, and Michael Cohen. Universal speech tools: the CSLU Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 3221–3224, Sydney, Australia, November 1998.

[112] Ronnie Taib and Natalie Ruiz. Wizard of Oz for multimodal interfaces design: Deployment considerations. In *Human-Computer Interaction. Interaction Design and Usability*, pages 232–241. Springer, 2007.

[113] Damien Touraine, Patrick Bourdot, Yacine Bellik, and Laurence Bolot. A framework to manage multimodal fusion of events for advanced interactions within virtual environments. In *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, pages 159–168, Aire-la-Ville, Switzerland, 2002. Eurographics Association.

[114] Robert van Kommer, Martin Rajman, and Herve Bourlard. Heading towards virtual-commerce portals. In *ComTec Journal*, pages 9–12, September 2000.

[115] Jan van Kuppevelt, Laila Dybkjær, and Niels Ole Bernsen. *Advances in Natural Multimodal Dialogue Systems*. Springer, 2005.

[116] Gert Veldhuijzen van Zanten. Adaptive mixed-initiative dialogue management. In *Proceedings of 4th IEEE International Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA-98)*, pages 65–70, Turin, Italy, September 1998.

[117] Ming Vo. *A Framework and Toolkit for the Construction of Multimodal Learning Interfaces*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 1998.

[118] Wolfgang Wahlster and Alfred Kobsa. Dialogue-based user models. In *Proceedings of the IEEE*, volume 74, pages 948–960, 1986.

[119] Marilyn Walker, Candace Kamm, and Julie Boland. Developing and testing general models of spoken dialogue system performance. In *Language Resources and Evaluation Conference*, 2000.

[120] Wayne Ward and Bryan Pellom. The CU Communicator system. In *Workshop on Automatic Speech Recognition and Understanding*, Keystone, Colorado, December 1999.

[121] Pierre Wellner, Mike Flynn, Simon Tucker, and Steve Whittaker. A meeting browser evaluation test. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 2021–2024, New York, NY, USA, 2005. ACM.

[122] Jerry Wright, Allen Gorin, and Alicia Abella. Spoken language understanding within dialogs using a graphical model of task structure. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, volume 5, Sydney, Australia, 1998.

[123] Xiao-Jun Wu, Fang Zheng, and Wen-Hu Wu. A hybrid dialogue management approach for a flight spoken dialogue system. In *Proceedings of the First International Conference on Machine Learning and Cybernetics*, volume 2, pages 824–829, Beijing, China, November 2002.

[124] Xiao-Jun Wu, Fang Zheng, and Mingxing Xu. Topic forest: a plan-based dialog management structure. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2001*, volume 1, pages 617–620, Salt Lake City, Utah, USA, May 2001.

[125] Victor Zue, Stephanie Seneff, James R. Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington. Jupiter: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–95, January 2000.

Curriculum Vitae

Miroslav Melichar

Born on April 5th, 1979 in Brno, Czech Republic

Languages spoken: English (fluent), French and German (basic), Czech (native)

Main interests: I am interested in human-machine communication. My main focus is on (multimodal) dialogue management, but I am also interested in related natural language processing fields, such as natural language understanding, speech recognition and speech generation.

Education and qualifications

2004 – 2008	**PhD Candidate** at the Artificial Intelligence Laboratory (LIA), School of Computer and Communication Sciences (IC), Swiss Federal Institute of Technology in Lausanne (EPFL)
2003 – 2004	**Doctoral School** at IC, EPFL
2003	**European Masters in Language and Speech certificate** (Euromasters) from the International Speech Communication Association (ISCA) and the European Chapter of the Association for Computational Linguistics (EACL)
2002	**Postgraduate specialization in Language and Speech Engineering certificate** from the EPFL
2002	**Six-month stay** related to Master's thesis at LIA, EPFL • InfoVox project – Interactive Voice Servers for Advanced Computer Telephony Applications, Swiss CTI grant 4247.1
1997 – 2003	**Master's degree** in Computer Science, Faculty of Informatics, Masaryk University, Brno • Specialization: Natural language processing • Master thesis: A tool for rapid dialogue prototyping • Member of Laboratory of Speech and Dialogue (LSD)

Work experience

2003 – 2008	**Research Assistant / Teaching Assistant** at LIA, EPFL • IM2 project – Interactive Multimodal Information Management, `http://www.im2.ch`, funded by the Swiss National Science Foundation • INSPIRE project – INfotainment management with SPeech Interaction via REmote microphones and telephone interfaces, European grant IST-2001-32746
2000 – 2003	**Software engineer** at Photon Systems Instruments (PSI) • Software and algorithms for image and signal processing data in biology and medicine • Embedded system programming (low-level software for scientific instruments) • GUI development
1997 – 2001	**Software developer** in several projects • Ticket reservation system (used by number of Czech hockey clubs) • Game engines (action game 'Katapult' and Online virtual casino)

Computer skills

- Expert knowledge of Java, Java Server Pages (environments: JBuilder, NetBeans)
- Expert knowledge of C/C++ (environments: Borland C++ Builder, Microsoft Visual C++)
- Expert knowledge of Delphi, Pascal (environment: Borland Delphi)
- Expert knowledge of dialog systems (VoiceXML)
- Databases (PostgreSQL, InterBase)
- PHP, JavaScript
- Matlab
- Digital image processing
- Embedded systems programming (ARM, C51, ISP1581-USB2.0)
- Platforms: Win32, Win CE

Publications

Book chapters

1. Pavel Cenek, Miroslav Melichar, and Martin Rajman. A Framework for Rapid Multimodal Application Design. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *Proceedings of the 8th International Conference on Text, Speech and Dialogue (TSD 2005)*, volume 3658 of *Lecture Notes in Computer Science*, pages 393–403, Karlovy Vary, Czech Republic, September 12-15 2005. Springer.

2. Trung H. Bui, Martin Rajman, and Miroslav Melichar. Rapid Dialogue Prototyping Methodology. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *Proceedings of the 7th International Conference on Text, Speech and Dialogue—TSD 2004*, Lecture Notes in Artificial Intelligence LNCS/LNAI 3206, pages 579–586, Brno, Czech Republic, September 2004. Springer-Verlag.

Peer-reviewed papers

1. Miroslav Melichar, Pavel Cenek, Marita Ailomaa, Agnes Lisowska, and Martin Rajman. From vocal to multimodal dialogue management. In *Eighth International Conference on Multimodal Interfaces (ICMI'06)*, pages 59–67, Banff, Alberta, Canada, November 2-4 2006.

2. Martin Rajman, Marita Ailomaa, Agnes Lisowska, Miroslav Melichar, and Susan Armstrong. Extending the Wizard of Oz methodology for language-enabled multimodal systems. In *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy, May 2006.

3. Marita Ailomaa, Agnes Lisowska, Miroslav Melichar, Susan Armstrong, and Martin Rajman. Archivus: A multimodal system for multimedia meeting browsing and retrieval. *Interactive presentation at ACL/Coling*, Sydney, Australia, July 17-21 2006.

4. Miroslav Melichar, Agnes Lisowska, Susan Armstrong, and Martin Rajman. Rapid multimodal dialogue design: Application in a multimodal meeting retrieval and browsing system. In *MLMI'05*, Edinburgh, UK, July 11-13 2005.

5. Silvia Quarteroni, Martin Rajman, and Miroslav Melichar. Introducing reset patterns: An extension to a rapid dialogue prototyping methodology. In *The IEE International Workshop on Intelligent Environments*, University of Essex, Colchester, UK, June 28-29 2005.

6. Sebastian Möller, Jan Krebber, Alexander Raake, Paula Smeele, Martin Rajman, Mirek Melichar, Vincenzo Pallotta, Gianna Tsakou, Basilis Kladis, Anestis Vovos, Jettie Hoonhout, Dietmar Schuchardt, Nikos Fakotakis, Todor Ganchev, and Ilyas Potamitis. INSPIRE: Evaluation of a Smart-Home System for Infotainment Management and Device Control. In *International Conference on Language Resources and Evaluation (LREC)*, volume 5, pages 1603–1606, Lisbon, Portugal, 2004.

7. Jan Krebber, Sebastian Möller, Rosa Pegam, Ute Jekosch, Miroslav Melichar, and Martin Rajman. Wizard-of-Oz tests for a dialog system in Smart Homes. In *Proceedings of the joint congress CFA/DAGA*, Strasbourg, France, 2004.

8. Heleen Boland, Jettie Hoonhout, Claudia van Schijndel, Jan Krebber, Mirek Melichar, Dietmar Schuchardt, Hardy Baesekow, Rosa Pegam, Sebastian Möller, Martin Rajman, and Paula Smeele. Turn on the lights: investigating the Inspire voice controlled smart home system. *Human Factors and Ergonomics Society Europe Chapter*, Delft, October 2004.