Ruhr-Universität Bochum
Fakultät für Angewandte Informatik

<u>Implementation and Evaluation of a Spoken Dialog System for Exam Result Enquiries</u>

Submitted in partial requirement of the Bachelor of Science in Applied Computer Science degree.

by Andreea Niculescu
Laerholzstr.80, C-409
44801 Bochum

30.09.2005

Supervisor:
Prof. Dr. Rainer Martin
Prof. Dr. Herbert Hudde

Acknowledgements

I would like to express my gratitude to Prof. Dr.-Ing. Rainer Martin for offering me this opportunity to do my Bachelors thesis in such a pleasant work environment as in the Institute of Communication Acoustics, Ruhr-Universität Bochum. I also thank M.Sc. Nilesh Madhu for his kind support.

Further on I am very grateful to Irina and Warren Treadhold for the generous support they've extended me during the course of my studies. Without their help I couldn't have achieved this step in my career.

Special thanks are reserved for my Singaporean friend Anantharaman Lakshminarayanan who believed in me and encouraged me to study Computer Science.

Thanks are also extended to Dipl. Ing. Christian Borß, Dipl. Inf. Sandra Krüger, the participants who cheerfully helped with the tests, and M.Sc. Sudhir Talupula. A special thank you to Fabian Bremmerich and Andre Gückel for their extraordinary support during the most trying times of my thesis.

Last, but not the least, I want to thank my family for all the moral support that I've received over the years.

Abstract

The goal of this work is to develop a speech-operated telephonic exam result enquiry system able to assist students in obtaining their exam scores. This study consists of analyzing and selecting an appropriate software tool (from various products currently available), implementing a dialog structure using the selected tool and testing it.
For the implementation of the dialog system three steps are required:

- developing of the relevant questions for each step of the process
- designing of the necessary vocabulary for the interaction
- programming the states according to the flowchart and interfacing the database containing the student ID and the scores with the system.

After the system was built it was tested upon 20 subjects (representing both German and foreign nationalities). The tests measure the quality of the system using objective criteria such as task success, dialog quality and dialog efficiency as well as subjective criteria such as system usability. For uniform testing, four scenarios were developed in which each participant had to interact with the system. After testing the system, all the participants completed an online questionnaire. The results based on both the objective and subjective criteria were analyzed to obtain a complete overview of the system performance. The work concludes with a presentation of the results and suggestions for further development.

List of Abbreviations

TTS	Text to Speech
RAD	Rapid Application Developer
ASR	Automatic Speech Recognition
Wrong I	Wrong Input
LI	Language Interference
SP	Spontaneous Speech Phenomena
UC	Utterance Complexity
LC	Lexical Coverage
BN	Background noise
SF	Speech feature
WSEtup	Wrong parameter setup
TF	Technical fails
TCR	Turn correction ratio
UTC	User Turns Correction Ratio
STC	System Turn Correction Ratio
S	Succeeded
SN	Succeeded in spotting that no answer exists
PS	Partial Succeeded
F	Fail
ITrain	Insufficient training

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Even though we live in a new era of communication technologies, where highly sophisticated systems are used to exchange information, natural, spoken language still remains the most popular media for interaction. The possibility of using spoken language when interacting with machines is extremely interesting, especially when speech as a natural and fundamental communication vehicle is the only interaction modality available, e.g. for telephone users. Moreover, the use of speech recognition and answering systems in telephone applications like travel booking and information, financial account information, and directory assistance is becoming more and more attractive due to the significant drop in the cost of implementing such voice-activated systems.

At the Institute of Communication Acoustics, Ruhr-Universität Bochum, we intend to use such a speech-operated dialog system for assisting in the dissemination of exam results. The scenario envisaged is as follows: the database containing the exam results will be connected to a speech dialog system, accessible to the user through a telephone line. The students desirous of knowing their results ring up the server containing the application and are guided through the task step-by-step. For such task-oriented applications a developer can design rules for understanding a user's requests and generating the system's responses for a limited vocabulary, so that recognition mistakes can be avoided.

According to their interaction capacity, there are four categories of spoken dialog systems[1], viz:

- simple question-answer systems,
- interactive dialog systems,
- voice-operated dialog systems, and
- conversational dialog systems.

The simple question-answer (Q&A) systems were developed in the 1960's and are only able to respond to elementary questions like team names, football player names or game scores. These systems are steered by the user and the system is rather passive. The dialog flow is more or less one-sided.

The systems in the second category interact with the user, presenting a two-sided dialog exchange. These systems are used, for example, by phone companies. They allow the user to communicate with the system using a reduced vocabulary (e.g. selection of options from a menu) and/or using the DTMF tones of the telephone (i.e. "for no, press one, for yes press two...etc").

The voice-operated dialog systems of the third category try to navigate the user to his/her goal. For this, a complex interaction is required between the system and the user. Such systems are capable of detecting and correcting recognition mistakes, preventing misunderstandings. Examples of such dialog systems already in commercial use include the Philips automatic train timetable information system for German intercity trains – available since February 1994 (under 0241/604020), the German railway information system (01805 - 996622) or the IBM traffic jam information system (06221 593129). The system that we have developed falls in this third category.

Systems belonging to the fourth category are the most complex of the lot. Such systems are able to converse with the users in a natural, task-independent manner, giving the impression of conversing with a human. The development of such systems requires sophisticated artificial intelligence features and is the focus of attention in artificial intelligence communities.

[1]http://www.ika.ruhr-uni-bochum.de/ika/lehre/praktika/v10/v10_intr.pdf

This works is organized as follows: chapter 2 contains a short overview about the existing tools for spoken dialog systems developing and offers a detailed description of the most important features of the tool we selected for our implementation.

Chapter 3 presents the manner in which we structured the dialog flowchart according to pre-determined principles rules established in the literature.

In the forth chapter we provide all the implementation details, the dialog states setup, data base connection and parameter settings.

Chapter 5 offers short overview of the evaluation methods in use for dialog systems and presents the results of the evaluation tests we performed.

Finally, the work is summarized in chapter 6.

Chapter 2
Tool selection process

2.1 Structure of a spoken dialog system

Spoken dialog systems have a complex software architecture since they need to analyze and produce speech units. In Fig.1 below, we present the circle of interaction between system and user where the communication takes place over a telephone line.

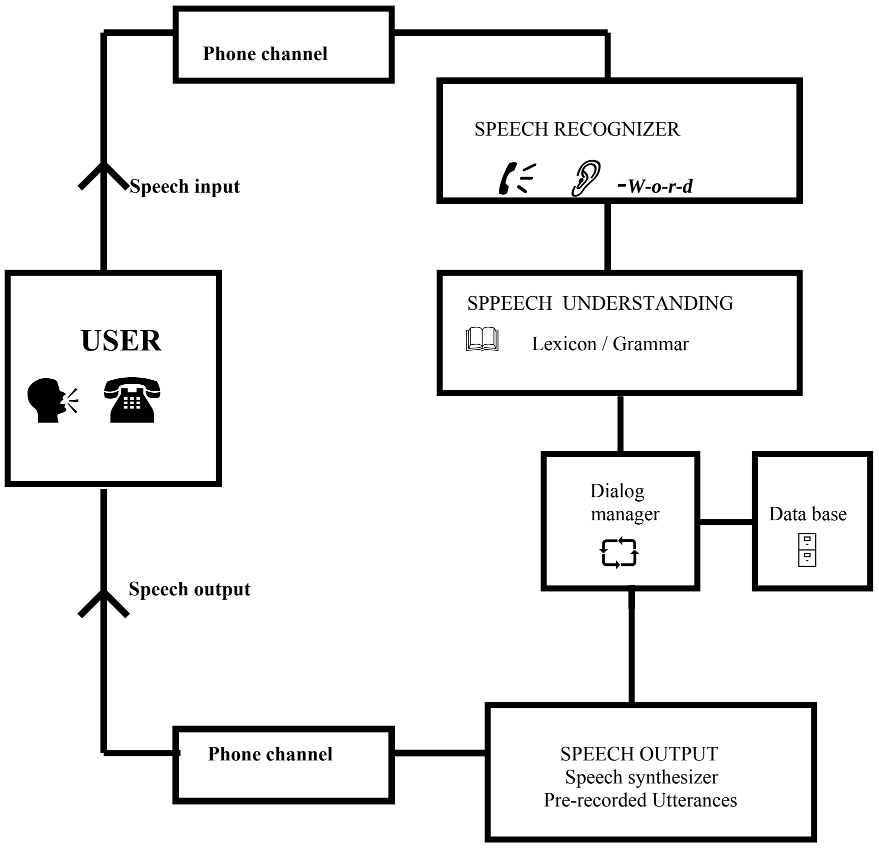


Fig. 1 Architecture of a spoken dialog system

The user's speech input consists of a string of acoustical signals that are converted into a sequence of words by the speech recognition module. To achieve this goal, most **speech recognition** modules use statistical methods – such as Hidden Markov Models (HMMs). First, they generate a word lattice of the *n-best* word solution sequences, with simple models to compute approximate likelihoods in real-time. Then, more accurate likelihoods are computed and compared with a limited number of hypotheses to determine the most likely word sequence.

The **speech understanding** module takes the word sequence delivered by the speech recognizer and analyzes it syntactically, pragmatically and semantically. The aim is to determine the intended meaning of the word sequence. This process is called **parsing**. Usually the parsing process uses a grammar which describes how words in an utterance are combined.

The decoded information is then sent to the **dialog manager**. This unit plays a central role in operating the dialog – a difficult task. Very frequently the recognized words delivered by prior modules are fragmented and incorrectly modeled. Background noise, human noises such as sneezes, coughs, unequal emphasis, word accents, inarticulate and incomplete pronunciations, word or syllable recurrences and different acoustical realizations of the phonemes can significantly affect the speech recognition performance.

If the user's speech input can at least be partially decoded by the parser, the dialog manager's task is to find to the right match for it. For this purpose the dialog manager uses dialog or word predictions. If the information from the user is not sufficient, the dialog manager will get back to the user to obtain the missing data.

If the complete information is available, the dialog manager will develop an answering strategy according to the implemented design and the information contained in the database.

The response will be converted into text and forwarded to the **speech output** module. The text can be either read out by a speech synthesis unit or, pre-recorded, natural language utterances can be played. Thus, the information processed by the system gets back to the user in an acoustical form.

2.2 Tool selection

For the development of our speech-operated dialog system dedicated to providing examination results, we required the following:

Software –components	Hardware -components
1. Data base for inserting student IDs and exam scores	1. PC on which this system can run
2. Speech recognizer	2. Telephone card / Telephone system
3. Speech output module: a synthesis or pre-recorded prompts to be played	
4. Dialog manager for dynamic system implementation	
5. Interface for data base access	
6. Interface for telephone cards installation	

Tab. 1 Software/Hardware requirements

The first step was the selection and installation of the required tools and appropriate development environment. For this purpose, several tools (see Appendix E) were analyzed with respect to their suitability for the task. The relevant criteria for the selection were:

1) the tool should be open source,
2) it should be suitable for building dialog systems and
3) it should have all the required components required for this project such as speech recognition, TTS and available API's for installing interfaces.

We analyzed **PublicVoiceXML**, **TellMe Studio**, the **CSLU Toolkit** and a free version of **IBM Voice Server**, available at the institute. We decided to try the last two tools, primarily due to the positive experience we had with them during past projects.

After testing both tools we decided to use the **CSLU toolkit**, because it has very good documentation, ease of implementation and last, but not the least, easy access to a database using a special interface, **TclODBC**.

2.3 Brief tool overview[2]

The CSLU toolkit, provided by the *Oregon Graduate Institute of Science and Technology* has a modular, open architecture supporting distributed, cross-platform, client/server-based networking and includes interfaces for standard telephony (**Dialogic driver software**) and audio devices, software interfaces for speech recognition, text-to-speech synthesis (**Festival**), dialog manager and animation components (3D talking heads).

For speech recognition the toolkit supports four languages – English, Spanish, Italian, Portuguese – and several approaches to speech recognition including Artificial Neural Network (**ANN**) classifiers, hidden Markov models (**HMM**) and segmental systems. It comes complete with a vocabulary-independent speech recognition engine, plus several vocabulary-specific recognizers (e.g. alpha-digits). In addition, it includes all the necessary tutorials and tools for training new ANN and HMM recognizers.

On the speech synthesis side, the toolkit integrates the Festival text-to-speech synthesis system, developed at the University of Edinburgh (Black & Taylor, 1997). Festival provides a complete environment for learning, researching and developing synthetic speech, including modules for normalizing text (e.g., dealing with abbreviations), transforming text into a sequence of phonetic segments with appropriate durations, assigning prosodic contours (e.g. pitch, amplitude) to utterances, and generating speech using either diphone or unit-selection concatenative synthesis.

The toolkit also includes the Rapid Application Developer (RAD), which makes it possible to quickly design a speech application using a simple drag-and-drop interface. RAD seamlessly integrates the core technologies with other useful features such as word-spotting, barge in, dialog repair, telephone and microphone interfaces, and open-microphone capability. This software makes it possible for people with little or no knowledge of speech technology to develop speech interfaces and application sin a matter of minutes. Software for a telephony interface is included in the bundle as well.

The Toolkit is written at two levels: the Tcl script level and the C level. The Tcl level allows for easy implementation and modification of high-level procedures such as speech recognition, file selection, and recognizer evaluation. Computationally intensive procedures, such as wave I/O, neural network training and classification, and Viterbi search, are implemented at the C level. An intermediate level allows the C-level routines to be called from the Tcl-level scripts.

[2] This paragraph was adapted from the CSLU tutorial website: http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/prefs/.

Chapter 3
Dialog Design

The capability to convert thoughts into speech is a human feature. Thereby people transform unconventional content (thoughts) into conventional form (speech) and communicate efficiently[3] and effectively about any subject using dialogs. A dialog can be described as a two-way interaction leading to a reciprocal information exchange between two agents in order to accomplish a task. Each task can be seen as a transaction string of several subtasks and each subtask has one or more complementary turns (question-answer pairs). For example, in our case, when we want to ask about an exam result (*task*), we need to exchange information about the course name (*subtask 1*) and the student ID (*subtask 2*). For each subtask we require to exchange one or more *turns* with our dialog partner.

Although there are systems in existence today that are able to give intelligent answers, they cannot achieve the same levels of communicativeness as their human counterparts. This is because of several reasons: firstly, the recognition of continuous speech is still a big challenge for such systems; secondly, the vocabulary used by such systems is kept small to enhance speech recognition; thirdly, due to the very small vocabulary, some complex or elliptical statements cannot be recognized and, fourthly, because dialog systems are task-oriented, i.e. they are able to carry out a dialog only about one particular task.

However, even if a dialog system cannot completely replace a human dialog partner, we can design it to be friendly and cooperative by respecting a set of rules. Listed below are a couple of design principles we took into account during system development. Most of these principles are described by [Bernsen et al., 1998][4], but we have also added other rules we found useful from our previous work experience.

3.1 Design Principles

1. Target user group
Before starting the design of a dialog system, the designer should identify the users who are going to utilize the services offered by the dialog system. Users can be classified using the following criteria:

a) **Gender**
Men and women talk differently. This difference is visible when humans talk to machines too. Women talking to computers will generally formulate whole sentences, indicating everything they would like to know. In contrast, men will use mostly keywords to communicate with the system.

b) **Background knowledge and expectations (novice/expert)***
System developers need to recognize relevant differences among users and user groups in terms of their background knowledge. These differences not only concern novice/expert distinction among users but also other types of background knowledge differences and may even include recognition of widespread erroneous background assumptions among users. The system must also take into account the expectations users may have with respect to the system's background knowledge. If the system does not take into account possible user inferences by analogy, this may invite users to ask clarifying questions or leave the system, probably discontentedly so, with unanswered questions.

[3] Although, this could be debated upon given the current global political situation.
[4] The principles designed by [Bernsen et al., 1998] are marked with an asterix.

2. Provide a mixed dialog initiative*

Designing the appropriate dialog initiative for a task-orientated dialog system is important for successful task completion. Speech is a dynamic process. People can generate the same information content by formulating their sentences in different ways – that is why it is very difficult to predict exactly how a user will formulate his/her statements. This problem can be solved if the system has a dominant dialog initiative – i.e. the system poses the questions and "moderates" the dialog signaling based on how the user interacts with the system.

Research papers show that the use of system initiative helps increase task success by making the users more aware of any ASR mis-recognition and making it easier to correct them when they occur. Novice users prefer system initiative [Litman D. and S. Park, 1999]. On the other hand, system initiative can increase the total dialog length – especially when the amount of subtasks is relatively high – and, for expert users, this might be tedious. However, for user-friendly systems, a mixed dialog initiative strategy can be considerably better, because it permits both the users and the system to interfere in the dialog flow and thus creates the conditions for an optimal interaction.

3. Information load

3.1. Concerning the entire flowchart

The design of the first system prompt is, in fact, the most important for a good and smooth dialog evolution. The first prompt should contain **what** kind of information the system basically provides and **how** the user has to interact with the system.

3.2. Concerning the amount of information that should be provided in an interaction turn (questions - answer pair)*

a) The system prompts should not contain more information than required for the subtask they are designed for. Normally one question should handle one particular piece of information – e.g. a question about train departure time should contain only information related to departure and should not refer to ticket prices. Too many questions at the same time can confuse the user.

b) Prompts should be short, if possible. If in some particular contexts the system prompts cannot be short, then the designed prompt should contain a dialog focus at the end of the statement, e.g. the prompt should end with a question referring to the next dialog sequence.

c) Feedback: immediate feedback provides users with an opportunity to detect misunderstandings immediately. The sooner the misunderstanding can be corrected, the better. A problem with immediate feedback is that it tends to make the interaction somewhat "heavier" than corresponding human-human exchanges. There are three possibilities to provide feedback: **echo feedback** (the system echoes the key contents of the user's input and associates this key with the next question in a single prompt), **implicit feed-back** (the final answer includes the recognized user prompt) and **explicit feed-back** (the system directly ask for confirmation).

4. Truth and evidence*

It is obviously important that the user trusts what the system says. Users have good reasons to become annoyed if the system provides them with false information.

5. Manner*

The aspect of manner concerns the way in which the intended meaning is being expressed.

a) Too open and non-specific formulations, like "*System*: Do you want more?" should be avoided because it invites the user to take the initiative and say all sorts of things. Instead the designer should use prompts like "*System*: Do you want to know about other exam results?"

b) The system should address the task-relevant topics of interaction in an order which is as close as possible to the order expected by the user. If the user expects some topic to come up early during interaction, that topic's non-occurrence at its expected "place" may cause a user-initiated clarification sub-dialog which the system cannot cope with. The study of the structure of human-human conversation in the domain for which the system is being designed may support design of orderly interaction.

c) If the system needs some time to process the information, it should inform the user that it will take a few seconds to give the information requested.

6. Highlight partner asymmetry*
This principle refers to differences that exist between interlocutors which are likely to influence the course and eventual success of the interaction. When learning to speak, we implicitly learn, what is a “normal” or “standard” partner in a spoken interaction. Unless otherwise told, we assume that our partner, with whom we interact, is “normal” or “standard”. If it turns out that this is not the case, we are trained to adjust our manner of speaking according to the partner's abilities, such as when speaking to children or the hearing impaired or when conversing in noisy environments.

The computer is, in many respects, a non-standard partner in spoken interactions and should strongly need to make its users aware of this fact as the penalty of generating all sorts of miscommunication can be severe. An example that illustrates this principle is the system’s request to speak after the ‘beep’ tone – this tone signals to the user that the system started to record the user’s input.

7. Meta-communication*
Meta-communication situations are those in which users need clarification. Requests for clarifications generally tend to be difficult for the system to handle and can lead to total dialog failure. This is the reason why the system should be designed so that meta-communication cases occur as seldom as possible. However, if such situation occurs, the system needs to have the necessary capability to handle it.

8. Sympathy
This is a guideline which is ignored by most system designers even though the advantage that can be earned by applying it can be huge: an amiable relationship between user and computer, based on personality and speech custom similarities can make the user be more sympathetic to situations in which the machine fails, i.e. to tolerate simple potential recognition mistakes or other small flaws.

According to marketing principles, the success of a product increases if the features of the product appear familiar to the customers. If a product seems to be friendly, if the customers “identify” themselves with it, giving them a feeling of trust, then the customers might most probably buy it. There are a couple of simple rules to follow that can relate to the psychological needs of the users.

a) System’s prompts should be polite and friendly – statements like “O.k. great!” or “Thanks for calling and have a nice day!” make the system sound familiar and sympathetic.

b) Voice: model a female voice or male voice according to target user group (male users generally prefer female voices!)

3.2 Design realization

Most users expect, from their previous experiences with humans, to be misunderstood if they talk too fast. Exactly the opposite will occur when interacting with our dialog system, especially when the user is asked to input his/her student ID: the speech recognition is word-based, i.e. the student ID digit strings are treated as a single word. If the user speaks slowly, making breaks between the digits the system will not be able to recognize the right student ID. For this reason the system will advise the user not to slow down while he/she inputs the student ID.

As dialog strategy, we decided to use a quite rigid one, in order to prevent misrecognition. Considering the small amount of subtasks, this dialog strategy shouldn't affect the system friendliness. The dialog initiative will, in this case, be with the system most of the time, i.e. the system will impose on the user a pre-set flow of questions. The user can take the initiative when restarting a new query.

Regarding information load, we tried to limit the answer possibilities by offering the user restricted choices and letting him/her choose a particular answer:

Example:
System1: *I have the exam results for the following courses: G I T one, G I T two, Signal Processing one, Signal Processing two, Communication Acoustics, and Audio Acoustics.*
System2: *For which course do you want to know the results? Please speak after the tone."*

The prompts are generally large but they always contain, at the beginning of the statement, a reference to what the system expects the user to say, e.g. the system needs the user's student ID, but it also has to teach the user how to do it. The system will first ask the user to be prepared to provide his student ID and will explain how to correctly input this ID. The last prompt will contain the ultimate request for the student ID.

Example
System1*: O.K., great, now prepare your student ID. Say the digits one by one. And leave out the first four digits: --1-0-8-0. Try to speak normally and don't make too long a break between the digits.*
System2*: Now tell me your student ID.*

The dialog strategy we developed uses explicit confirmation. The confirmation occurs only at two important points of the dialog: in the beginning, when the system tries to make sure that it has correctly understood which course the user wants to know the results for and after the user has input his/her student ID. Even if this confirmation extends the dialog somewhat, it is advantageous since correction of possible misrecognitions is easier.

In case of doubts concerning speech recognition, i.e. the recognition score is under a certain level, the system repeats its question prompt. Thus, if no user input is given, the system uses this escape option: it repeats the enquiry options twice and finally breaks up the conversation, saying goodbye and requesting the user to send an email with his student ID and name to the secretary. It is desirable that the system doesn't hang up abruptly.

The system flowchart developed using these design principles is detailed below:

Welcome message (state name: Welcome):
Hello welcome to the exam results automatic information system of the Communication Acoustics Institute, my name is Baldi and I can only speak English.

Transition (state name: transition1):
I have the exam results for the following courses: G I T one, G I T two, Signal Processing one, Signal Processing two, Communication Acoustics and Audio Acoustics

Option (state name: make__choice):
For which course do you want to know the result? Please speak after the tone

Transition (state name: transition2):
I think you said (*recognized value*), is this correct?

Handling mistakes (state name: counter_state):
1_attempt: Sorry my mistake
2_attempt: Sorry my mistake again.
3_attempt: Sorry my mistake again

Dialog quit (state name: quit1):
I am not able to help you. Please send an email with your student ID to the secretary and you will soon get your exam result.

Transition (state name: transition3):
O.K., great! Now prepare your student ID. Say the digits one by one and leave out the first four digits: 1-0-8-0. Try to speak normally and don’t make too long a break between the digits.

Student ID input (state: runtime_tree):
Now tell me your student ID:

Confirmation (state: confirmation1):
O.k., I will now repeat your student ID:

ID output (state name: ID):
(*recognized digit string*)

Confirmation (state: confirmation2):
Is this correct?

Handling mistakes (state name: counter_state)
Mistake 1 occurred (state mistake1): Sorry my mistake. Please don't forget to say the digits without breaks
Mistake 2 occurred (state mistake1): Sorry my mistake again

Dialog quit (state name: quit2):
Sorry my mistake again. It might be possible that your student ID number is not on my database. Please send an email with your student ID to the secretary and you will soon get your exam result.

Student ID Output (state name: display_result1->5):
The score you got is (*retrieved score*)

Transition (state name: transition2):
Do you want to ask about other exam results?

Dialog quit (state name: quit3)
O.K., thank you for your call and please call me again whenever you need to know your exam results. Have a nice day!

Exit dialog (state name: goodbye):
Good bye!

Chapter 4
Implementation

Once the dialog flowchart was designed, we started to implement the system. We christened it **Baldi**, taking the name of the animated face provided by the RAD. **Baldi**, driven by the speech recognition and synthesis components, is capable of automatically synchronizing natural or synthetic speech to realistic lip, tongue, mouth and facial movements. However, as this animated component does not contribute anything by way of information for a telephonic dialog system and because it requires a lot of memory, reducing the processing speed of the application, we decided to switch it off.

For implementation purposes, the CSLU toolkit offers a wonderful GUI environment – the **RAD** graphical interfaces. In the RAD, dialog states (objects) are dragged and dropped onto a canvas. When an object is dropped, a small red triangle called port will appear beneath it. This port helps to connect the objects (by clicking it and dragging a line to the next object). The RAD offers 18 object types for different dialog configuration, such as generic, conditional, exit, start, media, list-builder or alpha-digits objects. However, for our application we used only three: generic, conditional and action objects.

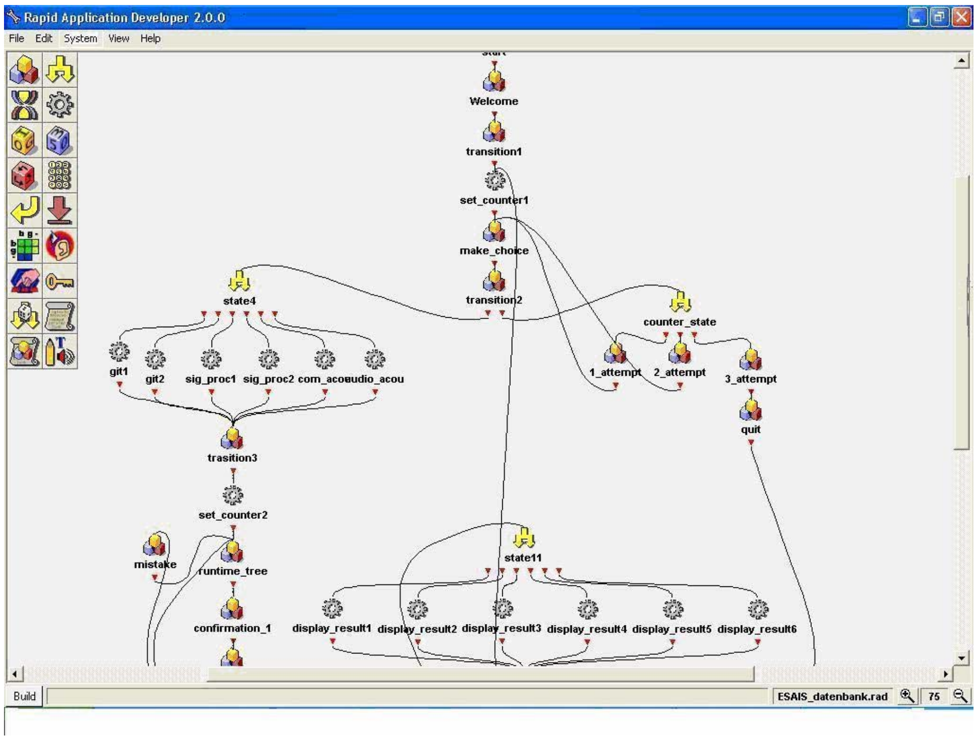


Fig. 2 RAD canvas

4.1 Generic Objects

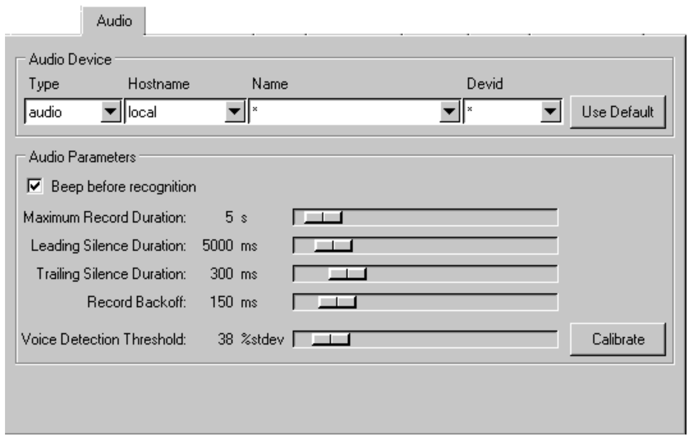
Generic objects are the most frequently used objects to create Baldi's dialog states. They perform speech synthesis and speech recognition. Clicking an object will open a widget, where the system prompts are entered. At runtime, prompts will be read out by the TTS module. Clicking the red port will open another widget that permits the entry of user vocabulary and grammar.

Due the simplicity of the task domain we reduce the user vocabulary to 11 words, representing course names ("GIT1", "GIT2", etc) and affirmations and negations with different pronunciation variants. The speech recognizers built into the RAD use phoneme-based pronunciation models as a basis for performing recognition. The pronunciation strings are automatically generated when a word is entered in the recognition window. The phonetic pronunciations can be viewed by selecting *Update All* in the recognition dialog box.

For each word in the vocabulary to be recognized, two sources are queried in order until a pronunciation is found. These are: a custom dictionary (containing speaker specific pronunciation models) and a system dictionary (the CMU[5] pronunciation dictionary). If the word is not found in any of the above dictionaries, a text-to-speech synthesizer is used to generate a pronunciation via letter-to-sound rules. This pronunciation is then stored in the local dictionary in case it is needed again. When speech recognition is successful in matching what the user says, a branching to the next object occurs.

There are two types of recognizers: Grammar and Tree recognizers. The grammar recognizer requires the creation of a recognition grammar while the tree recognizer needs only words or phrases in the generic object's box. Our system uses a tree recognizer.

To enter speech output for a generic object we need double-clicking on it. By selecting "Properties" another window will appear containing different tabs: the first tab is the TTS tab, where we can enter the text to be converted by the TTS in an acoustical output at run time. Apart from the TTS tab the property feature includes the setting of several parameters whose activation will only affect the one object[6].



Type: audio
Hostname: local
Name: *
Devid: *

☑ Beep before recognition

Maximum Record Duration: 5 s
Leading Silence Duration: 5000 ms
Trailing Silence Duration: 300 ms
Record Backoff: 150 ms
Voice Detection Threshold: 38 %stdev

Fig. 3 Audio tab

The most relevant tabs are the Audio and the Recognizer tab. These two tabs are explained in detail because they are of essential necessity to achieve a high recognition performance.

[5] CMU stands for Carnegie-Mellon University

[6] An individual object can override the global settings when the enabling checkbox is selected within the object's preferences menu

The **Audio tab** refers to audio input devices, audio-recording settings and microphone calibration. Of greater importance are the six Audio Parameters.

The *Maximum Record Duration* specifies the maximum length of time the speech recognizer will record the user's utterance.
The *Leading Silence Duration* specifies the maximum length of time the speech recognizer will continue to record if it is detecting only silence.
Adjusting the *Trailing Silence Duration* will affect the maximum length of time the recognizer will continue to record after the user stops speaking. The default value of this setting may need to be altered when the user is expected to say something that contains natural pauses, such as a telephone number. The trailing silence setting must be increased to prevent cutting off the speaker prematurely.
The *Record Backoff* is to specify the length of time between the beep and the start of recording.
Setting the *Voice Detection Threshold* can be done manually or by using the microphone calibration option. This parameter specifies the minimum sound threshold above which speech through a microphone or a telephone channel will be detected.
The *Calibrate Button* is required when either the acoustic signal delivered to the speech recognizer is altered or one of the following conditions is met:
1. different background noise levels
2. a change of the microphone position
3. a reset of the operating system’s audio properties

In the **Recognizer tab** only two options are of relevance for our application.

The *Out of Vocabulary Rejection Median* determines the recognition confidence required to reject an utterance as being “out of” the recognition vocabulary. A lower number rejects more and a higher number rejects less. This makes a high number more forgiving of incorrect pronunciations. Recommend is a rank of 9 for a 16 kHz adult recognizer and a rank of 22 for an 8 kHz adult recognizer (these are the values set by default).
The second relevant option is the *Word Spotting Median* which determines the recognizer's sensitivity to spot recognition vocabulary within an utterance. A low number spot less and a high number spots more.

Setting these parameters will affect only the selected object. To apply global changes to the entire application, the File->Preferences menu should be used.

4.2 Conditional Objects

The conditional object evaluates TCL Boolean expressions in its output ports and has at least two branches. The ports are evaluated from left to right. The leftmost port is evaluated first. If the expression is “true”, the dialog will immediately branch without considering the other ports.

We use conditional objects to select the right database (sorted according to the course name) and to build counters.

The counters are useful for repeating a certain action, a certain number of times. This is used, for example, to end a dialog after a certain number of repetitions/mis-recognitions. The counter is initialized in an action object and each time a misrecognition occurs, the counter is incremented.

4.3 Action Objects

Action Objects provide a text widget where the Tcl/Tk code is evaluated at runtime. The Tcl/Tk code is evaluated in a Tcl interpreter separate 5from RAD. We use action objects to assign values to some variables – like counters or display trees, to set up the connection with the database and to build the tree recognizer.

4.3.1 Display Trees

In the display tree box we enter the TCL code to look up the exam score for a student ID from a database. The display tree brings the output to the user in acoustical and visual form: it recites the exam result using TTS and concomitantly displays it in a special window. The information presentation in visual form is relevant only for the system evaluation part (see chapter 5) and can be left out for the real world application.

4.3.2 Database and Recognition tree

To connect the system to a database, we need to install the TclODBC package[7], which enables ODBC connections from Tcl/Tk scripting language. Once the package is installed, we can configure the databases. For our application we use an MS Access database, which we called "student". The database "student" contains six tables – one table for each course. Each table has three columns: an ID column- for the entry number, a student ID column and score column.

ID	studentid	Score
1	04243793	56
2	04244022	89
3	04244088	90

Fig. 4 Scrap of a data base table

The data can be directly entered in the database or it can be imported from another file, like an exel or a text data file. In order to be processed by the recognition modules, this data needs to have a special format, i.e. the student IDs are not allowed to contain blanks or other signs. The Tcl script code includes SQL to communicate with the database. All the results returned by the query are enclosed in curly brackets and, in this format, they are stored in a variable.

Thus, each the student ID data is first stored in a variable and, with the aid of adequate regular expressions, the blanks and the brackets will be removed. After this, the database will be updated with the new "clean" entry. The code snippet that does this is presented below:

```
set id [db "select ID from git_1"]
for {set i 1} {$i <= [llength $id] } {incr i} {
set details [db "select studentid from git_1 where ID = $i"]
regsub -all {\{} $details {} details
regsub -all {\}} $details {} details
regsub -all { } $details {} details
db "update git_1 set studentid='$details' where ID=$i"
}
```

The recognition tree is defined in the same action object. The recognition tree serves to dynamically create a recognition vocabulary at run-time by consulting the most recent entries in the database. To achieve this

[7]The package can be downloaded from the Sourceforge.net website. http://sourceforge.net/projects/TCLodbc.

objective, it will retrieve each digit string from the column “studentid” and will save it in a variable called “word”. The variable will contain the list of words for which a tree-based recognizer should be built:

set words [db "select studentid from signal_processing_2"]

The list will be passed to the CSLU toolkit standard function **createTreeVocab**. The function determines a pronunciation string for each word stored in variable “word” and adds that pronunciation string to the local dictionary. The function returns a new list containing all the words stored in “word” along with their respective pronunciation strings. The list will be saved in the variable “tree”:

set tree [createTreeVocab $words]

After that the recognition tree needs only one more function: **buildTreeRecognizer**. This function is also included in the CSLU package. **buildTreeRecognizer** creates a tree-based recognizer at the specified state – in our case the state name is *runtime_tree* – for the specified recognition vocabulary (i.e. $tree):

buildTreeRecognizer runtime_tree $tree.

4.4 Parameter settings

After the whole dialog is implemented we need to set the global and local preference values in order to ensure a good quality level of the system performance. The local parameters were set at four dialog states where user input was expected. These settings gave the best results for our system.

Make_choice		**Transition2**	
1. Recognition tab		**1. Recognition tab**	
Out of Vocabulary Rejection Median:	7 rank	Out of Vocabulary Rejection Median	9 rank
Word spotting median:	8 rank	Word spotting median:	9 rank
2. Misc tab		**2. Misc tab:**	
Maximum record duration:	6 s	Maximum record duration:	5 s
Leading Silence Duration:	5000 ms	Leading Silence Duration:	5567 ms
Trailing Silence Duration:	928 ms	Trailing Silence Duration:	464 ms
Record Backoff:	150 ms	Record Backoff:	111 ms

Runtime_tree		**Confirmation2**	
1. Recognition tab		**1. Recognition tab**	
Out of Vocabulary Rejection Median:	9 rank	Out of Vocabulary Rejection Median:	9 rank
Word spotting median:	9 rank	Word spotting median:	9 rank
2. Misc tab		**2. Misc tab:**	
Maximum record duration:	12 s	Maximum record duration:	14 s
Leading Silence Duration:	41134 ms	Leading Silence Duration:	9433 ms
Trailing Silence Duration:	10000 ms	Trailing Silence Duration:	1057 ms
Record Backoff:	505 ms	Record Backoff:	60 ms

Tab. 2 Local preferences

1. General tab	
Animated faced:	unchecked
Barge-in:	unchecked
Repair:	checked
2. Audio tab	
Maximum record duration:	100 s
Leading Silence Duration:	30839 ms
Trailing Silence Duration:	2797 ms
Record Backoff**:**	163 ms
3. Recog/DTMF	
Out of Vocabulary Rejection Median:	9 rank
Word spotting median:	9 rank

Tab. 3 Global Preferences

Once the application is implemented, it can be started by pressing the *Build* and *Run* button on the left corner. For future use, it is important to note that when the application is started for the first time it takes time to process the information from the data base. For this reason it should be started and left to process each data base before it is used.

Chapter 5
System Evaluation

To evaluate the usability of any system and to pinpoint the weak spots, the system needs to be tested. Accordingly, this chapter is concerned with the tests performed on our system and the results obtained.

5.1 Overview of evaluation methods

There are many evaluation methods which can be used, depending on the evaluation goal and development stage or level of the system. For example, [Hirschman and Thompson 1996] distinguish among three types of evaluation:

1. **Performance evaluation**, i.e. measurements of the performance of the system or its modules in terms of a set of quantitative and/or qualitative parameters
2. **Diagnostic evaluation**, i.e. detection and diagnosis of design and implementation errors
3. **Adequacy evaluation**, i.e., how well does the system or component fits its purpose and meets actual user needs and expectations

These three test modalities can be performed as integral parts of the development process to measure progress towards satisfaction of the requirement and design specifications.

From another point of view, evaluation methods can be classified as **blackbox** and **glassbox** tests. In a **blackbox** test only the input to and the output from the program are available to the evaluator. Test suites are constructed in accordance with the requirement specification and along with a specification of the expected output. Expected input and actual output are compared and any deviations must be explained. A **glassbox** test is an evaluation method, in which the internal system representation can be inspected. The evaluator should ensure that reasonable test suites, i.e. data sets, can be constructed that will activate all loops and conditions of the program being tested. Both **blackbox** and **glassbox** tests may be considered forms of **diagnostic evaluation**, but these tests are carried out on implemented components or **systems** only.

Other useful distinctions are those between **quantitative** and **qualitative** evaluation and **subjective** and **objective** evaluation. **Quantitative** evaluations provide information which is easy to analyze statistically and fairly reliable. **Qualitative** evaluations are ways of collecting data which are concerned with describing meaning, rather than with drawing statistical inferences. What qualitative methods (e.g. case studies and interviews) lose in terms of reliability they gain in terms of validity. They provide a more in-depth and rich description.

Subjective evaluation consists in judging some property with reference to users' opinions, like user satisfaction (based on TTS performance, ASR performance, task ease etc). **Objective** evaluation addresses objectively measurable performance parameters, like dialog duration, recognition score, task completion success, system and user turns ASR rejections, timeouts etc. Both quantitative and qualitative evaluations are objective evaluations.

Performance evaluation and **diagnostic** evaluation are forms of objective evaluation whereas **adequacy** evaluation includes both objective and subjective evaluation.

5.2 Test design

To test Baldi we decided to use an adequacy test combining objective and subjective criteria. Following the PARADISE framework of [Walker et al., 1997], we organized our evaluation measures along the following four performance dimensions:

- *System usability*: user satisfaction (based on the questionnaire regarding topics such as TTS performance, ASR performance, task ease, manner, expectation, desired improvements)
- *Transaction success*: transaction success rate
- *Dialog quality*: ASR rejections and input repetition request
- *Dialog efficiency*: turn correction ratio (TCR) and interaction time

5.3 Test preliminaries

5.3.1 Scenarios

For a uniform test performance we develop four scenarios. The scenarios are intended to be representative of system functionality and task domain coverage.

The first scenario presents the following situation: the user and two of his friends have taken three exams and the only possibility to find out their results is to call Baldi, the automatic exam information system of IKA[8]. The user has to request the system for the result scores for three student IDs from three different databases. No other additional requirements are needed for this scenario.

The second scenario "locates" the user in the Bahamas: s/he has only a telephone line to get in contact with the rest of the word and s/he is desperate to know her/his exam result. The user will call Baldi to seek out the exam results for her/himself and her/his friends for two different courses. Primarily, in this scenario the user is required to ask the system to repeat its statements. Unfortunately the system encountered huge difficulties in recognizing the word "repeat". After consulting the senior scientific programmer – Jacques de Villiers – from the CSLU, we got to know that a virtual "repeat" output port is used internally by RAD when the recognition confidence is low. It is configured to repeat the prompt and redo the recognition. The system recognizes "repeat" and then follows the internal, virtual output port instead of the one we've configured. However, this response came rather late to be of use as we had already started our evaluations. 19 out of 20 tests using the second scenario were performed without the repeat request.

The third scenario has a higher degree of difficulty; it tries to simulate what happens if a mistake occurs – the user has to intentionally mispronounce two different student IDs and to input an ID which is not in the database.

In the fourth and last scenario the user has the freedom to select any three student IDs from three different databases and to check the exam score. For this, he/she gets a list with the data base entries.

For each scenario, the user was asked to make a single "call" and to write down, on the evaluation score sheet, the results obtained.

5.3.2 Experiment flow

The test experiments took place in the laboratory of the Institute of Communication Acoustics. The test persons were conducted into a special sound-proof cabin, where the PC on which this system was running was located. Each person got a set of scenarios and the database lists. Each subject first had the test briefly

[8] IKA stands for "Institut für Kommunikationsakustik".

explained to him/her and was shown how to operate the system. Then, after completing the test, the subjects filled out an online questionnaire.

5.3.3 Online Questionnaire –technical details

The purpose of developing an online questionnaire was twofold: firstly, this presents a rather robust means of gathering data and, secondly, the framework of this questionnaire could be reused later.

For the questionnaire development we used the **Netbeans 4.1 IDE**. The questionnaire was built using **HTML** and contains two types of questions: descriptive ones, where the users can enter answers in special text boxes (text areas) and multiple-choice questions where the user selects one option/answer from a list of available choices. After filling out the form the test persons need only to press a send button. Once the button is pressed the data is sent to a **JSP** file running on a Tomcat Server.

On receiving the data the JSP file calls a **JavaBean** file to process it. JavaBeans is a java class with the following properties: it implements a **java.io.Serializable** interface and has a non parametric constructor. For passing parameters to a bean from the HTML file, there has to be a corresponding get/set method for every parameter. Besides, the get/set methods need a special format: the method name is built using the correspondent attribute names from the HTML (*String name* **->** *get***Name***/set***Name**)[9].

The JavaBean also contains a function called **void save()**, which saves the data, input by test users in the html form, into files using a **try/catch** block. For each user one file is generated at runtime, containing a short explanation of the input content – like "1. Test person no.: "2. Mother tongue:" – and the input value.

The file name is composed of the string "datei" and a timestamp delivered by the standard function **currentTimeMillis().** If the data file cannot be opened, the program will go to the catch block and will print out an error message. We choose to process the information with a JavaBean because it is an elegant solution to store the data and relieves the JSP file of additional load.

5.3.4 Automatic evaluation

After the experimental part was concluded we had 20 files containing the user data input. A Perl script was written to read the data from each file, to calculate the results for each question, and to print it to another file. The script opens each data file and searches for the input values using regular expressions. The input values can be numbers, strings units or text units. For saving number values for the multiple choice questions (consisting of integer values from 1 to 5) arrays have been created: to score the corresponding input, the array value at that index is incremented.

Strings inputs are expected only once – when the test user has to enter his/her mother tongue in the questionnaire. For saving these strings we have created a hash data structure. A hash – alternatively called an associative array – is a complex list of information linking a **key** (in this case user's the mother tongue) – to a **value** (the occurrence of a certain mother tongue among the users). When a mother tongue is found the script will increment the value at the corresponding mother tongue key.

Descriptive user comments comprise the text units. These are saved in arrays using the standard function **push()**.

For the tests, once all the 20 files were processed, another file – evaluation.txt – was created at runtime and the cumulative results of each data structure were printed out into this file.

[9] For example if in the HTML file one of the buttons, text areas or radios contains an attribute name "**scenario1**", in the Java Bean we will declare a get/set method get**Scenario1**/set**Scenario1**

5.4 Subjective measurements

The subjective measurements should show how the users perceive the quality of the developed dialog system. The subjective criteria measured in our questionnaire concern the fourth performance dimension, namely the system usability.

5.4.1 Questionnaire design

The questionnaire contains a set of 10 questions and utilizes both, quantitative and qualitative evaluation methods for data analysis. We decided to use these two evaluation models together in order to get a meaningful overview that combines the advantages of both methods. Each question, with the exception of the first one referring to the mother tongue, corresponds to a usability factor as shown in the following table:

Question	**Usability factor**	**Evaluation method**	
		quantitative	**qualitative**
2. Was easy to communicate with the system?	ASR	*	
3. What do you think about the voice synthesis of the system?	TTS	*	
4. Was it easy to find the exam results you wanted? Comments are optional!	Task ease	*	*
5. Did you know what you could say at each point of the dialog?	Manner	*	
5.1 If you have selected **"No"**, where exactly did you have problems with the system? Give a short description of your problem!	Manner		*
6. Did the system work the way you expected it to?	Expectation	*	
7. What is your assessment of the general quality of the system?	General system quality	*	
8. Is there anything you didn't like about the system? If yes, please describe briefly.	General system quality		*
9. What kind of improvements would you like?	Desired improvements		*
10. Other comments:	User defined		*

Tab. 4 Usability factors

Questions that yielded statistical data were multiple-choice and had a response on a scale of one to five, with one being “very good” and five being “very bad”.

	1	*2*	*3*	*4*	*5*	
very good	○	○	○	○	○	**very bad**

Fig. 5 Scale

The questionnaire includes also questions that give the user the possibility to describe, in greater detail, their experience with the system and to propose future improvements. These qualitative types of questions cannot be statistically evaluated, but they are rich sources of information concerning the system-user interaction and user expectation.

5.4.2 Evaluation results

We tried to select the test subjects from the target user population: students studying electrical engineering, a majority of males, and mostly native German speakers with a smattering of foreigners. Consequently the test user group was comprised of 20 persons: 65% Germans, 10% Polish, 10% Spanish, 5% Korean, 5% Tamil and 5% English. Furthermore, among these, 90% of the test subjects were male and 10% were female[10].

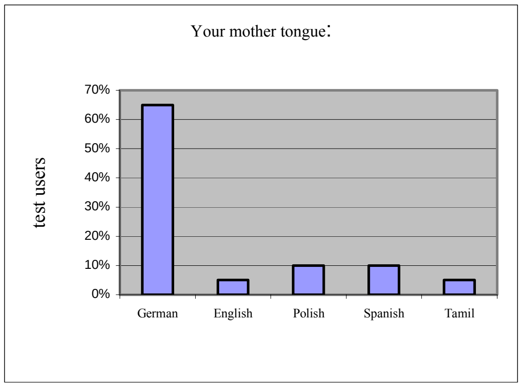


Fig. 6 Mother tongue

5.4.2.1 ASR

75% of the test subjects appreciated the communication with the system as being “easy” or “very easy”. This leads us to conclude that the speech recognizer did a good job. Only 5% of the users complained about a difficult interaction.

very easy	easy	so-so	difficult	very difficult
30%	45%	20%	5%	0%

Tab. 5 ASR

5.4.2.2 TTS

The assessment of the TTS did not get as good scores as the ASR: only 55% of the subjects categorized the Text To Speech Tool as being “good” or “very good”. 35% assessed it as “so-so” and 10% as “bad”.

very good	good	so-so	bad	very bad
5%	50%	35%	10%	0%

Tab. 6 TTS

[10] Due to the limited number of female test users, the distinction between male and female users will not be considered for the statistical evaluation.

5.4.2.3 Task ease

The task ease was rated differently for each scenario. The most difficult task to complete was the third scenario – the one where pronunciation and data base mistakes are simulated. Due to its length and to the high probability of speech recognition mistakes, the third scenario leaves a bad impression: only 40% of the users considered the task as being "easy" or "very easy".

very easy	easy	so-so	difficult	Very difficult
20%	20%	35%	25%	0%

Tab. 7 Task ease – Scenario 3

The third scenario also contains the most comments (10 in total). 15% of the subjects expressed their discontent regarding the way the system handles the input of non-existing student IDs in the data base: due to the word recognition principle implemented, the system can only recognize digit strings.

Although such methods normally have very high recognition scores, they also have a huge disadvantage: each time when the user inputs non-existing data, the recognizer will try to find the best match for it, even if the confidence level is lower. Therefore, the dialog duration gets unnecessarily prolonged and the user gets confused. 25% of the users reported that the system faced speech recognition problems during the third scenario.

The other three scenarios have about the same degree of difficulty. This can be seen in the way the users evaluated their task ease: 75% (scenario1), 85% (scenario 2) and 80% (scenario 4) of the users classified the task completion as "easy" or "very easy".

Scenario	very easy	easy	so-so	difficult	very difficult
1	55%	20%	15%	10%	0%
2	60%	25%	10%	5%	0%
4	60%	20%	5%	15%	0%

Tab. 8 Task ease – Scenarios 1, 2 and 4

Comparing the results between the scenarios, it is obvious that the second scenario has the best rating. This can have two reasons: the second scenario has the shortest dialog duration (it contains only two subtasks) and this might give the user the impression of having attained the goal sooner. Secondly, the users already have experience from the first scenario and know better how to interact with the system. User comments like "at the beginning it's a bit difficult", "a lot of mistakes" or "the first attempt failed" in the comment box of the first scenario corroborate the fact that some degree of familiarity is necessary before the users can handle the system. In general around 15% of the subjects needed to get used to the system.

5.4.2.4 Manner

The statistics for this usability factor shows that 75% of the subjects knew what they could say at each point of the dialog. However the dialog structure still needs improvements. Of the 25% of the subjects who had problems with the system, 5% weren't sure how to answer in the affirmative to the system question "Is this correct?" and used, as an alternative to "yes", the word "correct" – which is not in the recognition vocabulary. This often led to speech recognition failure.

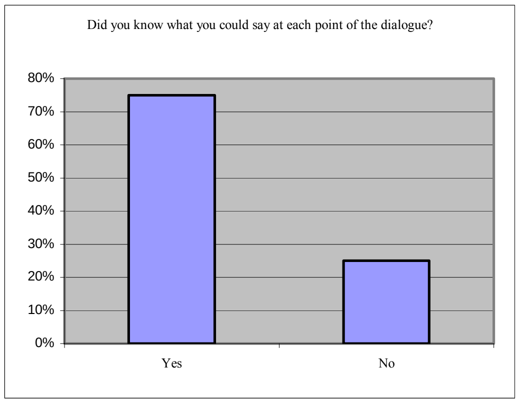


Fig. 7 Dialog structure

5.4.2.5 Expectations

Concerning the expectations regarding the system behavior, it seems that 90% ("totally agree" + "agree") of the users could correctly anticipate the way the system works.

totally agree	agree	somewhat agree	disagree	totally disagree
30%	60%	10%	0%	0%

Tab. 9 Expectation

5.4.2.6 General System quality

Even if 75% of the subjects appreciated the general quality of the system as being "good" or "very good", most of them (75%) used the opportunity to express what they did not like it about it.

very good	good	so-so	bad	very bad
5%	70%	20%	5%	0%

Tab. 10 General System Quality

We summarized the comments made in this section together with them at the **Desired Improvements**, as the most remarks are complementary.

5.4.2.7 Desired Improvements

35% of the subjects complained that the system does not offer an "escape" option: in case of failure the system should be more cooperative; in effect, being able to handle commands like "stop", "exit", "go back" or explain how to cancel the current run.

Another 35% perceived as annoying, the recurrent system instruction not to mention the four digits of the student ID, to speak normally and only after the tone. They suggested that the system should distinguish between an expert and a novice user, making the introduction and other hints shorter in the former case.

The way the system handles non-existing student IDs was also disliked by 20% of the users. Therefore, the system should exhibit a more cooperative behavior when non-existing database input is given, by indicating that the desired information cannot be found.

The vocabulary for affirmation/negation should be enlarged to include words like “wrong” and “correct”. Furthermore, the system should be able to repeat its statements if the user has misunderstood the output.

The TTS was characterized as being too fast (30%) – especially when the system is repeating the student ID for verification - too artificial (30%), “somewhat awkward”, having different time interval between the words, giving the impression of a non-uniform intonation and word speed.10% of the users suggested that the current TTS voice be replaced with a female one as female voices “sound trustworthier” and “nicer” than a “speaking tin-box”! One of the more arcane suggestions was to use the voice in Unreal Tournament 2004 – a computer game.

5.4.2.8 Other comments

The subjects used this section to make some additional comments. While some of the users stated they enjoyed the test, others were rather put-off by the idea of talking to machines. We also got positive feedback from 25% of the users, who mentioned “excellent recognition of course names”, “well done” or “the system, in general, is pretty nice”.

5.5 Objective measurements

The objective measurements address the last three performance dimensions of our adequacy test: transaction success, dialog quality and dialog efficiency. To do this, we analyzed the log files generated by each system call during the interaction. To each log file there belong: a text file, in which the system will record information regarding the time when the user passed a state and other audio files containing the user input. The system stores the audio input provided by the user in separate files, each file containing the input for the corresponding state of the dialog.

These files are extremely important for evaluation. Without them no objective measurement would be possible. The log files such as the text and the audio files should first be processed manually. We analyzed each scenario from each log file according to the following model:

Scenario No.:
Turn correction ratio:
Transaction success:
  -succeeded:
  -partially succeeded
  -succeeded in spotting that no answer exists
  -failed:
Dialog duration:
  -start:
  -end:
  -total:
Input repetition request:
Recognition mistakes:

Fig. 8 Test user protocol file

Additionally, all the subjects were requested to complete one call per scenario, but not all of them satisfied this requirement. Some of the users did more then one call per scenario. In order to have the correct statistics, we shall need to take these additional calls into account. The list below shows the total number of dialogs per scenario.

Scenario	No. of dialogs
Scenario 1	23
Scenario 2	20
Scenario 3	22
Scenario 4	26
TOTAL	**91**

Tab. 11 No. of dialogs per scenario

Lastly, before we analyzed the users' log files, we generated a "reference" log file for the four scenarios. The user results are compared with this standard log file.

5.5.1 Transaction success

Transaction success is one of the most frequently used metrics in deciding whether a system is acceptable according to the preordained standards. This metric measures the success of the system in providing users with information they require, if such information is in the database. The result of the transaction can take four values (as seen in Tab. 13): **succeeded** (**S**) – for tasks for which answers exists, **partially succeeded** (**PS**) – when at least one subtask is completed, i.e. one exam score of three, **succeeded in spotting that no answer exists** (**SN**) – for tasks in which users ask for information about non-existent data, and **failed** (**F**) – when the dialog ends without providing any requested information [Simpson and Fraser, 1993][11].

Comparing the reference log file with the user log files we have the following results:

Scenarios	S	SN	PS	F
Scenario 1	100%			
Scenario 2	100%			
Scenario 3	100%	100%		
Scenario 4	100%			

Tab. 12 Reference file

Scenarios	S	PS	SN	F
Scenario 1	82.60%	13.04%		4.34%
Scenario 2	95.00%			5.00%
Scenario 3	59.09%	31.81%	50.00%	9.09%
Scenario 4	73.07%	3.84%		23.07%

Tab. 13 User log files

As may be seen, scenario 2 has the highest score for transaction success (95%). This corresponds to the user perception that this scenario was easy to complete. The lowest score is obtained by the third scenario, as expected (59.09%). As explained previously, this is because of its different subtasks: (**S**) – for the first two subtasks and (**SN**) **–** for the last subtask, which required the input of a non-existent object in the data base. The difference of 9.09% between the **S** and **SN** can be explained in that some users forgot to complete the last subtask. The third scenario has also the highest rate of **partially succeeded** transactions (31.81%), probably due to the difficulty associated with it.

The failure rate of the fourth scenario is rather surprising: even though this scenario was ranked by 80% of the user as being "easy" or "very easy", it has the highest failure rate (23.07%). The reason for this could be that the users perceived scenario 1 as being more difficult to complete as compared to this scenario as it was the first one they evaluated.

5.5.2 Dialog quality

The dialog quality level is given by measuring the error rate (**ER**) of the automatic speech recognition. In addition, we measured another parameter, called the input repetition request (**IRR**), which pertains only to those cases in which the recognition score is very low and the system asks the user to repeat the input.

[11] We have replaced the value "succeeded with constraint relaxation", described by [Simpson and Fraser, 1993], with "partially succeeded", as this was more suitable for our evaluation test

The error rate gives the percent of misrecognized words from the whole amount of input words. We considered for the recognition error statistics, only those user statements containing at least one valid input in the database or vocabulary were misrecognized. Non existing objects input intentionally as in scenario 3 were not considered as errors generated by the system.

Word Total	**Error Total**	**Error Rate**	**Recognition Rate**
1758	**123**	**6.99%**	**93.01%**

Tab. 14 Error Rate

As can be seen from the table above, the error rate is 6.99%. This means we have a recognition rate of 93.01%, which is a rather good result. The table below shows the average of the recognition errors per scenario:

Scenarios	**Amount of dialogs**	**Total errors**	**Average per scenario**
Scenario 1	23	**35**	1.52
Scenario 2	20	**11**	0.55
Scenario 3	22	**33**	1.50
Scenario 4	26	**44**	1.69
TOTAL	**91**	**123**	**1.35**

Tab. 15 Recognition error average per scenario

Using the audio files at our disposal, we then studied, in detail, the different errors. Understanding why an error occurs is a basic step for improving the recognition. In our case we could distinguish between ten possible causes for error. The first four are localized on the side of the user and concern errors which can occur due to negligent user input, the fifth addresses errors emerging from deficient vocabulary design and the last four handle technical errors on the part of the system – appearing due to insufficient system development or wrong parameter adjustment. These errors are detailed below

1. Wrong pronunciation due to language interference (**LI**): user mistakes due to combination of languages, e.g. some of the users combined English with German words, thereby pronouncing the words erroneously

2. Wrong input (**WrongI**): user introduces wrong input, anticipating the next dialog state or ignoring the system requirements, e.g. inputting the first four digits of the ID although expressly warned against it

3. Spontaneous speech phenomena (**SP**): hesitations, repetitions, incomplete utterances, word fragments, unclear pronunciation, etc

4. Utterance complexity (**UC**): too many statements at one time, e.g. user greeting the system etc.

5. Insufficient lexical coverage (**LC**): occurs when a word is not in the language model lexicon, like “wrong”, “correct”, “thank you” or “exit”, or when, for example, the first four digits of the student ID (1080) is used in addition to a valid word, like “yes” and “no”

6. Background noise (**BN**): when the tool is very sensitive to background noise, e.g. a simple paper rustling can be interpreted as speech input

7. Inadequate speech feature (**SF**): too slow or too loud speech input

8. Insufficient training (**ITrain**): the HMM parameters are not correctly estimated

9. Wrong parameter set up (**Wsetup**): occurs when parameters are not correctly adjusted for the application in use. Some of the audio input is strongly distorted, so that the speech input cannot be recognized at all. In other cases beginning or end of the speech input is missing

10. Technical failures (**TF**): in some of the cases the tone of the recorded input becomes suddenly very low, or no input is recorded

The last two parameters deal with the physical interfaces of the system and we shall refer to them as "technical deficiencies". The table below lists the distribution of the recognition errors attributable to the above listed causes:

Spontaneous speech phenomena	**Inadequate speech feature**	**Wrong parameter set up**	**Language interference**
5 (7.04%)	25 (35.21%)	7 (9.85%)	8 (11.26%)

Insufficient lexical coverage	**Wrong input**	**Utterance complexity**	**Background noise**	**Tech. failures**
3 (4.22%)	5 (7.04%)	3 (4.22%)	7 (9.85%)	8 (11.26%)

Tab. 16 Technical deficiencies

A total of 71 recognition mistake could be identified in this way, which represents a 52.84% of the total recognition mistakes:

Total recognition errors	**Identified error**		**NOT identified recognition error**	
	No.	**%**	**No.**	**%**
123	**71**	**52.84%**	**52**	**47.15%**

Tab. 17 Identified recognition errors

For the last two parameters – **Wsetup** and **TF –** we checked the audio files individually to draw up the statistics of their occurrence at each state and the influence they had on the speech recognition performance. We found that these parameters have a relatively low frequency of occurrence - we found them in 17.69% of the states- and did not always cause recognition errors.

States	No. of states per user	Signal strongly distorted	Word beginning missing	Word end missing	Low tone	No input recorded	Total
confirmation2	449	5,34%	6.01%	0.22%	14.25%	2.00%	27.83%
transition2	565	7.25%	6.54%	0.17%	7.61%	6.01%	27.61%
make_choice	324	0.00%	2.16%	0.92%	3.08%	3.08%	9.25%
runtime_tree	423	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
TOTAL	**1758**	**3.69%**	**4.03%**	**0.28%**	**6.65%**	**3.01%**	**17.69%**

Tab. 18 Percentage technical deficiencies per scenario

An interesting remark is that the "confirmation" and "transition" states were the ones most affected by these phenomena while the "runtime_tree" was unaffected.
However, when the recorded input was strongly distorted, fragmented or completely lacking in information, it leads the system to ask the user for repetition. This phenomenon is measured by the input repetition request parameter (**IRR**).

Scenarios	No. of states per user	Input repetition request	IRR %	No. of dialogs per scenario	IRR average per scenario
Scenario 1	449	**11**	2.44%	23	0.47
Scenario 2	565	**16**	2.83%	20	0.80
Scenario 3	324	**26**	8.02%	22	0.84
Scenario 4	423	**28**	6.61%	26	1.07
TOTAL	**1758**	**81**	**4.60%**	**91**	**0.89**

Tab. 19 IRR

The input repetition request rate is an indicator of the system strategy to avoid misrecognition. The measured value of this parameter is quite low (4.60%) for the entire test interaction. This is due to the system calibration regarding the word accuracy: it accepts more and rejects less, even if the input has distorted features or the recorded tone is low.

5.5.3 Dialog efficiency

The last performance dimension we considered was the dialog efficiency. This is given by measuring the turn correction ratio (**TCR**) and the dialog duration per scenario. The TCR is a very useful parameter measuring the ratio of those turns containing anomalous behaviour of both, the user, and the system to all turns in the dialog. The TCR is obtained by adding the results of the turn correction by the system (**STC**) and the turn correction by the user (**UTC**) in one variable.

Scenarios	No. of states per scenario	No. of TCR	TCR %
Scenario 1	449	66	14.69%
Scenario 2	565	25	4.42%
Scenario 3	324	40	12.34%
Scenario 4	423	79	18.67%
TOTAL	**1758**	**210**	**11.94%**

T
Tab. 20 TCR

The highest rate of TCR occurs in the fourth scenario – 18.67%, followed by the first scenario with 14.69%. As expected, the second scenario has the lowest rate, only 4.42%. The third scenario has, in this case, a lower percentage because we measured only the real recognition errors that occurred and not the errors produced intentionally as required in the scenario.

The dialog duration is the last parameter measured. It shows the average duration for each scenario in comparison with the reference file. The dialog duration is captured in the log file that the system generates for each call. The file contains the time when the user reaches a dialog state. To determine the correct end of a call we need to add the time the system uses to exit the dialog. If the dialog is finished and terminated successfully by the user, the system will still need 10 seconds to finish the dialog covering the quit states. If the system terminates the dialog prematurely due to recognition mistakes or erroneous input, an additional 15 seconds are to be added in order to determine the dialog length. In the case the user hangs up, no additional time needs to be added.

Scenarios	**Average duration (min)**	**Reference file duration (min)**	**Average difference (min)**
Scenario 1	3.36	2.58	0.38
Scenario 2	2.16	2.01	0.15
Scenario 3	4.36	4.26	0.10
Scenario 4	3.14	2.49	0.25

Tab. 21 Dialog duration

As may be seen, the average dialog duration is quite close to the reference value. This could be taken as an indicator of the (relatively) smooth interaction between the user and the system.

5.6 Summary

The subjective and objective criteria were analyzed to determine the degree of adequacy of the system for its designed purpose.

From the subjective measurements, it may be concluded that the general quality of the system, the ASR and the dialog manner are acceptable. Furthermore, the system satisfies the expectations of the majority of the users. The tests also show that it is quite easy to communicate with the system, although improvements could be made on the TTS.

The results of the objective measurements show a close correlation with the values in the reference set.

In general, we can conclude that the system has a high degree of adequacy and can fulfill its purpose. However, some improvements need to be done before its deployment.

5.6.1 Improvements and future work

The analysis of the audio input files showed that the origin of more than a half of the recognition mistakes occurred can be easily found. The mistakes can be broadly categorized as mistakes on the user part and mistake on the system side.

The mistakes on the system part are much easier to handle because we can directly influence the system configuration by adjusting the corresponding parameters.

During the test many of the users input transition words like “wrong”, “correct” or “thank you” which are not in the designed vocabulary. A solution for this would be to enlarge the vocabulary model, introducing the missing words in the vocabulary box of the corresponding statement. A more flexible alternative would be to create a grammar in which such words can be grouped together, describing a set of rules and variables for possible utterances. The RAD offers the possibility of building finite state grammars by clicking the red output port and selecting the "Grammar" check box.

Background noise, such as heavy breathing or paper rustling, led to recognition mistakes in around 10% of the cases. To resolve this problem, a better microphone calibration is required. This can be done by selecting *Preference* from the *File* menu. The calibrate button can be found in the *Audio Tab*. The calibration is extremely important, not only for the current application used with microphone but also for its future implementation over a phone line.

Around 11% of the audio files contain technical deficiencies: the input is extremely distorted, containing only one letter sound; others are missing the word beginning or the end and, in some of them, the input is completely absent. This is due to inadequate parameter adjustment. We observed these phenomena especially at the confirmation and transition states where a "yes" or a "no" was expected. If the beginnings of the words are missing, the **Voice Detection Threshold** (*File* > *Preferences...* > *Audio*) should be set to a lower level. If the ends of the recorded words are missing, the **Trailing Silence Duration** option in the *Properties* menu (right click on the object icon) should be increased -the set up of this parameter should be done for each state separately as the expected input has a different length at each state. Further tests need to be done with different combinations of these thresholds to determine the optimal value for the application.

Around 7% of the files were recorded having a lower tone. We could not find any explanation for that. Although this deficiency didn't cause any recognition mistakes, it is an indication that some inputs are not properly recorded. This needs to be analyzed.

Most of the errors identified (35%) were caused by inadequate speech features, like too slow or too loud speech input. The loudness of the speech input can be handled by appropriate calibration of the microphone. A more difficult to solve is the problem of slow input of the digit string. We increased the value of the audio parameter **Leading Silence Duration** specifies to 30839 ms but a higher level seems to be required. Future development work could test for the optimal value of this parameter.

Around 10% of the recognition errors were due to German language interference: the user mixed the English input with German words, especially the digit strings. A solution to this problem could be to improve the speech recognition module by training the parameters with German digit pronunciation. A very well written HMM training tutorial, describing the procedures for training both, neural network and HMM-based speech recognizers with the Toolkit is available on the CSLU web-site[12]
Errors generated by utterance complexity can be handled by increasing the rank of the **Word median spotting** parameter. This allows the recognizer to detect vocabulary words within an utterance, with a higher sensitivity. At the same time the **Maximum record duration** parameter should be increased.

Wrong user input due to anticipation of next dialog state or ignorance of the system requirements cannot be handled by the speech recognition module. Here the dialog structure can be improved by modelling a control state to verify if the user input has the required form. If not, the system could branch to a short dialog where the user gets more detailed hints on how to interact with the system.

Spontaneous speech phenomena like hesitations, repetitions, or word fragments are extremely difficult to handle even for very highly developed systems used in commercial applications. More research work on the area of acoustic and language models need to be done in order to improve the capabilities of dialog system to handle this type of erroneous input.

[12] http://cslu.cse.ogi.edu/toolkit/docs/researchers.html

Some of the users complained about the TTS engine as being too fast and too artificial, expressing their wish to be attended to by a female voice engine. The RAD toolkit comes with six voices, including male and female versions of American English and Mexican Spanish but they are not all very good. Our subjective impression was that they not only sound like monotone, bored speaker droning through a tin box. Also the concatenation of the phonemes lacked naturalness. We tried all and we chose the most suitable one, in our opinion. However, the TTS can be substituted in the RAD with any recorded voice. The process of creating a natural speech prompt is explained in detail on the tutorial web site of the CSLU RAD toolkit[13].

The dialog structure needs some improvements as well: a barge-in functionality should be activated (*File > Preferences... > General*). This serves to shorten the dialog duration for experienced users. However, the caveat using this functionality is that it increases the system vulnerability to recognition errors. Therefore, in the beginning of our implementation, this feature was switched off.

The instruction regarding the omission of the first four digits of the student ID should be prompted only during the first round of the interaction. This can be done using a counter.

The state design should be improved to include some kind of "emergency brake", in case the user wants to get back to the dialog beginning. This can be achieved by adding an extra port to each state through which the user can reach the *make_choice* state.

A better solution to handle non existent student ID must be found. In the actual configuration the system tries to recognize the input and uses the best match for it; if after four inputs the system is still not able to find the required student ID it concludes that it is handling non-existent objects and exits the dialog. This solution is not suitable and need to be replaced. A possible solution could be to analyze the confidence score and to determine under which level an input can be consider as non-existent in the database.

The dialog structure needs to be provided with a repeat-option. In the first version of the system such an option was designed but unfortunately, due to a system intern configuration. A quick solution to this problem, provided by the CSLU senior developer Jacques de Villiers, would be to change the word "repeat" to something else (e.g. "repeat*" or "repeatX") while keeping the pronunciation exactly the same.

The system is envisaged as a telephonically-accessible system. For this it needs to be interfaced with a telephone line. The support for telephony interfaces for RAD is rather limited in the current version of the CSLU Toolkit[14]. According to the system developers, the system was tested only with the Intel's **Dialogic** range of cards such as D/21x, D/41x, Proline/2V, D/41ESC and D/240SC-T1 of the Dialogic Native Architecture software suite and works only on Windows NT. The first two cards connect to analogue phone lines and the latter to a T1 circuit[15]. A price list can be seen at http://www.tti.net/computer-telephony/dialogic.html.

Once the Dialogic hardware and software are installed, we can use a phone line instead of the default microphone/speaker combination by selecting "*File->Preferences->Audio* and, in the *Audio tab*, setting *Devid* to the required Dialogic channel ID (e.g. dxxxB1C1 for channel 1 on board 1). In the *Type* field, the default "**audio"** has to be changed to "**line"**. If "line" is not an option in the drop-down list then the card was not correctly installed. After saving the settings, the application can be operated with a phone line: clicking on the *Run* button the application will wait for a call on the designated line and will hang up once the dialog is complete.

An alternative to the Dialogic card, which seems to be quite expensive (around $500 and more), is offered by a Greek researcher, who has developed a simple interface between the phone line and the serial port of

[13] http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/tutorials/tutorial013/index.html
[14] The information presented in this paragraph was obtained from the CSLU Toolkit Discussion forum available on http://www.cslu.ogi.edu/forum/search.php.
[15] A price list can be seen at http://www.tti.net/computer-telephony/dialogic.html.

the PC, using the sound card for audio I/O. Apparently it should cost only a few dollars in parts, being built from commonly available electronics parts such e.g. as an hybrid transformer from an old telephone. As described, the circuit can be built on a board and housed in the computer speaker, taking its power from the speaker's power supply.

Once the interface is built, it needs to be connected to the serial port #2 (COM2) and the telephone. The audio from the interface goes to the microphone input of the sound card. The audio out to the interface is taken from the earphone jack of the computer speakers (the audio card does not have enough volume to drive the telephone line adequately).

The interface needs only a driver, **wincom.dll**, that has been modified to correspond to the need for control of the individual serial port lines and to be compatible with the Tcl 8.05 language used by CSLU[16]. Then, the interface needs only to be connected to a phone number.

For the application to work well over the phone line, it might be better to train the recognizer using data collected over the phone.

[16] For more details the interface' author, Mr. V. Georgiou can be contacted at vgeorgiou@eie.gr.

Chapter 6
Conclusions

Spoken dialog systems are emerging as an effective means for humans to access information through natural spoken interaction with computers. The development of such systems requires a wide range of speech and language technologies like automatic speech recognition (ASR) for converting human speech into text strings, natural language and dialog processing to determine the meaning and intentions behind the recognized utterances and to generate a cooperative response to them, and text-to-speech synthesis (TTS), to convert the system utterance into actual speech output.

This work was concerned with the design, implementation, test and evaluation of **Baldi**, a spoken dialog system developed at the Institute of Communication Acoustics, for assisting students in the dissemination of exam results. The dialog design follows a list of principles described in the literature, extended by our own experiences and observations from previous projects. For the implementation we chose the CSLU toolkit from Oregon Graduate Institute of Science and Technology. The toolkit is free of charge for research purpose and can be downloaded from the website of the institute.

After the system was implemented we tested it with 20 subjects. To evaluate the system adequacy for the intended purpose we performed tests to measure both objective and subjective criteria, grouped along four performance dimensions: *system usability* (measuring the user satisfaction regarding topics such as TTS performance, ASR performance, task ease, manner, expectation and desiderated improvements), *transaction success* (given by the transaction success rate), *dialog quality* (characterized by ASR rejections and input repetition requests) and *dialog efficiency* (counting the user turns and the interaction time). The tests comprised of four scenarios, developed in order to ensure a uniform test corpus.

The subjective criteria were measured using an online questionnaire whereas the objective criteria were measured based on the log file the system creates for each call performed. From the results of these tests we conclude that while the system is acceptable, it still needs some improvements (concerning the speech recognition, dialog structure and speech synthesis) before it can go online.

BIBLIOGRAPHY

Bersen, N. O., Dybkjæer H. & Dybkjæer L. (1998). Designing Interactive Speech Systems. London: Springer Verlag.

Bernsen, N. O., Dybkjær, H. and Dybkjær, L**.** (1996): Principles for the design of cooperative spoken human-machine dialog. Proceedings of the International Conference on Spoken Language Processing (ICSLP) '96, Philadelphia, S. 729-732.

Danieli, M., Gerbino, E. (1995). Metrics for evaluating dialog strategies in a spoken language system. Proceedings of the 1995 AAAI Spring Symposium on empirical methods in discourse interpretation and generation .Stanford, Menlo Park, CA: AAAI Press, 34-39

Delogu, C., Carlo, A. D., Sementina, C. &Stecconi, S. (1993). A Methodology for Evaluating Human-Machine Spoken Language Interaction (Proceedings 3rd Europ. Conf. An Speech Communication and Technology, EUROSPEECH ’93, Nr.3) (S.1427-1430). Berlin.

Gleiss, N. (1992). Usability-Concepts and Evaluation. TELE (English edition), Swedish Telecommunication Administration, 2, 24-30.

Goodine, D., Hirschman, L., Polifroni, J., Seneff, S., & Zue, V. (1992). Evaluating Interactive Spoken Language Systems Proceedings of the 1992 International Conference on Spoken Language Processing, ICSLP '92 , S. 197-200). Banff.

Hirschman, L., and Thompson, H.S. (1997) Overview of evaluation in speech and natural language processing, In: R. Cole et al. (eds.) “Survey of the State of the Art in Human Language Technology", Cambridge University Press,

Litman, D. and S. Pan (1999): Empirically Evaluating an Adaptable Spoken Dialog System. In Proceedings of the 7th International Conference on User Modeling http://www.cs.usask.ca/UM99/Proc/litman.pdf

Loderer, G. (1998). Evaluierung von Dialogstrategien eines natürlichsprachlichen Dialogssystem durch Wizard-of-Oz Experimente. Diplomarbeit. Institut für med. Kybernetik und Artificial Intelligence, Universität Wien.

Simpson, A., Fraser, N.M. (1993): Black-Box and Glass-Box Evaluation of the SUDIAL-System, Proceedings 3rd Europ. Conf. An Speech Communication and Technology, EUROSPEECH ’93, Nr.3, Berlin, S.1423-1426

Sadek, D., De Mori, R. (1997). Dialogue Systems. In R. De Mori (Hrsg.), Spoken Dialogues with Computers (S. 523-561). San Diego: Academic Press.

Walker, M. A., Litman, D., Kamm, C. A., Abella, A. (1997). PARADISE: A general framework for evaluating spoken dialogue agents, Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics, ACL/EACL 97, S.271-280.

INFORMATIVE WEBSITES

1. http://www.nis.sdu.dk/~nob/publications/ICSLP-14.5.96-F.pdf
2. http://www.ika.ruhr-uni-bochum.de/ika/lehre/praktika/v10/v10_intr.pdf
3. http://staff.cs.utu.fi/courses/ddsi/05/Assignment1_Quenel.pdf
4. http://www.nis.sdu.dk/~nob/publications/MIT-PRESS-6.7.99-NOBx.pdf
5. http://www.cc.gatech.edu/classes/cs6751_97_winter/Topics/dialog-speech/#Synthesis
6. http://www.disc2.dk/tools/d2.7b-15.11.99.html#sec6.2
7. http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/quickstart.html
8. http://cslu.cse.ogi.edu/toolkit/old/old/documentation/cslurp/wincslurp/newmanual.html
9. http://cslu.cse.ogi.edu/toolkit/docs/researchers.htm

APPENDIX A: Evaluated Toolkits

CSLU Toolkit

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	Small / Middle -size
Grammars	Phrases / Context free grammar
Speaker	Independent
Supported languages	English, Spanish, Italian, Portuguese
Word Spotting	Yes
Barge-in	Yes
B. Other Technical Features	
TTS	Yes (using Festival system)
Recognizer	Yes
Developer Tools	-Graphically-based authoring environment (RAD) for designing and implementing spoken dialog systems; speech recognition is automated for the developer who simply lists the words to be recognized at each state -Several tools for audio-visuals and natural language processing, speech processing, tutorials and examples, like vocabulary tutor, spectrogram reading tutorial, Tcl/Tk Manual pages, Tcl/wish Shells
Supported Standards	-4 levels of interface for application developers: -RAD, the speech recognition tools, as well as the programming environment, have an underlying implementation as a set of **C libraries** -Natural language processing modules are developed in **Prolog** -Core functionalities are then re-implemented in an extension to **Tcl** to facilitate writing, debugging, and modification. -Graphical user interfaces are written in **Tcl/Tk** -Festival TTS system included in the Toolkit, and its OGI extensions, is written in **C++** and **Scheme**. -No support for general speech application interfaces (such as Microsoft SAPI or Java Speech API)
C. Support, development and education services	
Supported platforms	Windows 9X, 2000, XP, NT
Telephony	Dialogic driver software included in the CSLU Toolkit uses the **Dialogic** Native Architecture3.3 release of the drivers for Windows NT/2K.
Requirements	PC, 200 MHz processor, 64 MB Ram (128 preferred), Windows XP or NT, Microphone / Speakers (sound-blaster compatible.), Video card with OpenGl
D. Other features	
Price/license type	Free of charge for educational, research, personal or evaluation purposes. Separate license for commercial use
Other	Facial animation: animated 3D talking faces; capable of automatically synchronizing speech with realistic lip, tongue, mouth and facial movements; the face can be made transparent revealing the movements of the teeth and tongue while producing speech; the orientation of the face can be changed so it can be viewed from different perspectives while speaking; the basic emotions of surprise, happiness, anger, sadness, fear etc can be communicated through facial expressions.
Area of Application	Academic research projects with small vocabulary. Not suitable for large scale projects or for multilingual applications.

APPENDIX A

Microsoft Speech Server

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	Large
Grammars	- n-gram - Grammar file – XML format or an inline (static) script;
Speaker	Independent
Supported languages	English, French and Spanish
Word Spotting	-
Barge-in	barge-in, support mixed-initiative applications
B. Other Technical Features	
TTS	Yes
Recognizer	Yes
Developer Tools	Microsoft Speech Application SDK (SASDK) + Visual Studio .Net
Supported Standards	HTML, SALT, Script via Web - Handles DTMF and speech applications
C. Support, development and education services	
Supported platforms	Windows Server 2003 and above
Telephony	Supported telephony card (Telephony Application Services (TAS) (MSS component). Each computer running TAS also requires Telephony Interface Manager (TIM) software).Handles up to 96 telephony ports
Requirements	Pentium 2.5 GHz or + with 2 to 4Gb of RAM, .Net Framework, Microsoft Enterprise Instrumentation, at least 22 Gb HD, Video card, Pointing device (mouse), cd or dvd player Internet Information Services (IIS), ASP.NET 1.1 with Service Pack 1 or later
D. Other features	
Price/license type	$845 or Evaluation package for 180 days
Other	Enables companies to unify their Web and telephony infrastructure, and extend existing or new ASP.NET Web applications for speech-enabled access from phones, mobile phones, Pocket PC and Smartphones; - telephony simulator within the SASDK
Area of Application	For project requiring large grammars and vocabulary, as support for telephony interaction. In addition and a large set of available supported languages

APPENDIX A

Nuance Speech products

Feature	**Values/Notes**
A. Speech Recognition Features	
Vocabulary Size	Very large
Grammars	n-gram
Speaker	Independent
Supported languages	25 languages
Word Spotting	yes
Barge-in	barge-in
B. Other Technical Features	
TTS	Yes – (Nuance Vocalizer)
Speaker Recognizer	Yes (Nuance Verifier)
Developer Tools	2 components: Nuance V-Builder (graphical, drag-and-drop design environment) & Nuance V-Server
Supported Standards	VoiceXML, VoIP, HTTP & HTTPS, SNTP, TCP/IP
C. Support, development and education services	
Supported platforms	Windows 2000, Linux, Solaris
Telephony	Supported interfaces: T1/E1, IDSN PRI, CAS, SIP/RTP, Lucent 4ESS/5ESS
Requirements	Pentium 3 with 512 Mb RAM
D. Other features	
Price/license type	Commercial – Patented application
Other	Dynamic language detection: caller begins speaking in preferred language, and system understands and interacts accordingly
Area of application	Complete speech solutions; such as voice-activated dialing, customer care, information lines, e-mail reading

APPENDIX A

VoCon Games & Mobile Speech SDKs

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	Medium
Grammars	Words or phrases (phoneme-based speech recognition)
Speaker	independent
Supported languages	English (US & UK), French, German, Spanish, Italian, Japanese, Mandarin, Korean + specific model for children's voice
Word Spotting	-
6. Barge-in	Yes
B. Other Technical Features	
TTS	Yes (concatenated speech synthesis - segments of actual recorded human speech assembled to speak the text (SpeechWorks RealSpeak Solo)
Speaker Recognizer	Yes (SpeechSecure Embedded)
Developer Tools	Development tools, an evaluator for PC, sample program code and detailed documentation
Supported Standards	XHMI, an XML based language, VoiceXML, MRCP (Media Resource Control Protocol), Aurora Standard, SOAP (Simple Object Access Protocol), SALT
C. Support, development and education services	
Supported platforms	-Primarily targeted for game platforms; -Available on the most popular embedded platforms
Telephony	Mobile devices such as cellular, ATM (VoCON Mobile)
Requirements	-Superior performance on platforms with constrained memory and CPU resources -RealSpeak Word requires no recordings and just 3MB of ROM for 60,000 words. -Require specific hardware depending on development.
D. Other features	
Price/license type	Commercial / available for free download for qualified PC, PlayStation 2, and GameCube developers.
Other	-Available for Sony PlayStation 2 as well as for Nintendo, PlayStation GameCube and mobile devices.
Area of Application	-For embedded recognition, like games, on cellular phones -Name dialing, navigation destination entry, and device command and control

APPENDIX A

Voice Signal Technology

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	Active vocabulary => Small to large
Grammars	Finite state grammar
Speaker	adaptive / independent
Supported languages	English (US & UK), French, Spanish, Portuguese (Brazilian &Iberian), German, Italian, Finnish, Korean, Mandarin, Cantonese, Italian, Dutch, Polish, Russian, Finnish, Swedish, Norwegian, Korean, Japanese
Word Spotting	Yes
Barge-in	Yes
B. Other Technical Features	
TTS	Yes
Speaker Recognizer	-
Developer Tools	VoiceSignal software comes as a precompiled, pre-tested library so there is no need to develop using an SDK.
Supported Standards	Markup languages
C. Support, development and education services	
Supported platforms	O.S.: Symbian, Linux, Microsoft, Hardware: Agere systems, Intel, Philips, Qualcomm
Telephony	Wireless mobile devices
Requirements	
D. Other features	
Price/license type	Commercial
Other	-
Area of application	Speech solutions for wireless mobile devices, to look up or call any person in their contact list, dial any number, address and dictate a message, and access any feature or operator service on the phone (like games, ring tones, music downloads or the web)

APPENDIX A

TellMe Studio

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	Small to large (up to thirty thousand entries)
Grammars	Phrases / CFG / n-gram; GSL or GRXML syntax - support Nuance GSL grammars only.
Speaker	dependent / adaptive / independent
Supported languages	English;
Word Spotting	-
Barge-in	Yes
B. Other Technical Features	
TTS	Yes (Tellme Voice Appl1ication Network's TTS –an AT&T Natural Voices TTS engine)
Speaker Recognizer	-
Developer Tools	-Scratchpad, Editors, Debug Log, Syntax Checker, Grammar Tools, VoiceXML Terminal, Record by phone tool -VoiceXML Tutorials Code and Audio Library, Audio Conversion tools
Supported Standards	VXML , JavaScript , SSL , HTTP, DBI, ODBC, OLE/DB, WAV
C. Support, development and education services	
Supported platforms	Standards pc -
Telephony	-Session Initiation Protocol (SIP) for call signaling and Real-time Protocol (RTP) for voice media -Soft phone- freely available Pingtel instant xpressa. SIP hard phone; Pingtel' and Cisco
Requirements	Web browser and a soft or hard telephone
D. Other features	
Price/license type	Free of charge
Other	-
Area of application	-Development and demonstration of voice applications. -Allows to quickly build and test VoiceXML applications without buying or installing any special hardware or software

APPENDIX A

PublicVoiceXML

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	small to large
Grammars	phrases / CFG / n-gram
Speaker	independent
Supported languages	American and British English, Arabic, Brazilian Portuguese, Breton, Canadian French, Croatian, Czech, Dutch, Estonian, French, German, Greek, Korean, Hebrew, Indonesian, Hindi, Italian, Japanese, Lithuanian, Polish, Portuguese, Romanian, Spanish, Swedish, Telugu, Turkish
Word Spotting	-
6. Barge-in	Yes
B. Other Technical Features	
TTS	Yes
Speaker Recognizer	
Developer Tools	Graphical Development, grammar development tools.
Supported Standards	TTS: MS SAPI compliant (Windows), IBM Via Voice, CMU Festival + MBROLA
C. Support, development and education services	
Supported platforms	All 32-bit MS Windows (95/98/NT/2000/XP) , All POSIX (Linux/BSD/UNIX-like OSes) , Linux , Win2K , WinXP
Telephony	VoIP, telephone, SAPHIR Cards
Requirements	Memory footprint; PC .
D. Other features	
Price/license type	GNU Public License
Other	-
Area of Application	Free Software toolkit for developing voice services. Using the VoiceXML 2.0 standard, phone based applications can be easily built and connected to web or database services.

APPENDIX A

IBM Voice Server

Feature	Values/Notes
A. Speech Recognition Features	
Vocabulary Size	Active vocabulary => Small to large
Grammars	Finite state grammar ; Grammar-based speech recognition, including support for dynamic grammars
Speaker	independent
Supported languages	English (U.S, U.K. + Australian), Brazilian Portuguese, French (Canadian + European), Dutch, German, Italian, Spanish (Latin American + European), Japanese, Korean, Chinese (Cantonese + Simplified)
Word Spotting	Yes
Barge-in	Yes
B. Other Technical Features	
Speech Synthesis	Yes (text to speech)
Speaker Recognizer	-
Developer Tools	WebSphere Studio Site Developer and **Voice Toolkit V5.1, which** includes the functionality in Voice Toolkit V5.0 (graphical CallFlow™ Builder, call control support using CCXML, VoiceXML 2.0 editor, grammar editor, a pronunciation builder, VoiceXML 1.0 to 2.0 conversion utilities, grammar conversion utilities, and an integrated simulator for application testing and debug).
Supported Standards	JavaTM, VoiceXML 2.0, Support for VoiceXML 2.0, Speech Recognition Grammar Specification (SRGS) 1.0, and Speech Synthesis Markup Language (SSML) 1.0 , Semantic Interpretation for Speech Recognition (SISR) and W3C Working Draft 1, , dated April 2003
C. Support, development and education services	
Supported platforms	AIX, Windows, and Linux
Telephony	Support many telephony platforms, including WebSphere Voice Response for AIX® V3.1 and WebSphere Voice Response for Windows V3.1, Intel Dialogic, Cisco or Siemens HiPath, and Voice Server Speech Technologies for Windows and Linux
Requirements	Third-party IVRs or VoiceXML2.0 gateways that interoperate with the WebSphere Voice Server V5.1 using MRCP V1 draft 4, Processor: Pentium III 1GHz (minimum), Memory (RAM): 2 GB (minimum), Available, Disk Space: 2 GB, Network: TCP/IP, Other: CD-ROM
D. Other features	
Price/license type	Trial version /Commercial
Other	-
Area of Application	Several Voice applications accessed via a Web browser, a telephone or a mobile device

APPENDIX B: Scenarios

Dear participant ☺ !

Below are some pointers for you to take note of before you start the tests:

1) Please read the scenarios carefully and try to follow them exactly as described.

2) Kindly pay attention to the system instructions - it is essential for a clear and successful communication between you and the system.

3) You may take note of the exam score at the end of each tested scenario - the system displays the score for each requested student ID.

4) After you have finished the tests you will be asked to complete an online questionnaire regarding your experience with the system. Please give us your frank opinions / suggestions

On behalf of the IKA, thank you very much for your participation in this test!

Scenario 1

You and your four friends have sat for your exams, and are eager to know your results. It is a Saturday afternoon, which gives you two options: to wait till Monday, or to call Baldi - the Automatic Exam Information System of the Communication Acoustics Institute. You decide to call Baldi!

Your student ID (Matrikel number) is: 1080 00115427
You want to know the exam results for: GIT 1
Score:

1st friend's student ID is: 1080 04243793
He wants to know his result for: GIT 2
Score:

2nd friend's student ID is: 1080 03229943
He wants to know his result for: Signal Processing 1
Score:

Please call Baldi and ask about all the exam results!

Scenario 2

You are in the Bahamas enjoying the best holiday you're ever had together with three other friends (the holiday was a birthday present from your parents ☺). But one thing bothers all of you: you still don't know if you've passed your exams. You decide to call Baldi, an exam results automatic machine, which can inform you of your exam results.

Your student ID is: 1080 04203469
You want to know you exam result for: Signal Processing 1
Once the system recognises your student ID and asks you to confirm, ask the system to repeat your student ID number instead of choosing 'Yes' or 'No' immediately.
Score:

Your friend's student ID is: 1080 03232095
He wants to know his result for: Communication acoustics
Ask the system again to repeat his student ID!
Score:

APPENDIX B

Scenario 3

You and your friend have sat for two exams but both of you are really afraid to know the results. So you ask your little sister, Anna to call Baldi to find out your results. Both of you give her your student IDs.

Your student ID: 1080 04203415
You sat for Communication Acoustics
Score:

Your friend's student ID: 1080 00202298
She sat for Audio Acoustics
Score:

Anna can't speak English very well and she makes a mistake reading out the student IDs – she confuses 2 with 1 (see WRONG).

WRONG	CORRECT
0 4 2 0 3 4 2 5	0 4 2 0 3 4 1 5
0 0 2 0 1 2 9 8	0 0 2 0 2 2 9 8

If Baldi doesn't understand her the next time Anna will pronounce the digits correctly (see CORRECT).

Patrick asks Anna to find out his exam result as well. He doesn't know that his student ID is not on the data base. Let see how Baldi deals with non-existing student IDs!

Patrick's student ID: 1080 095 232958.
He sat for Audio Acoustics
Now imagine you are Anna!
Good luck ☺!

Scenario 4

Pick any student IDs for the following courses: GIT 1, Signal Processing 2 and Audio Acoustics. Write down the student IDs and the score obtained.

GIT 1
Student ID 1:
Score:
Student ID 2:
Score:

Signal Processing 2
Student ID 1:
Score:
Student ID 2:
Score:

Audio Acoustics
Student ID 1:
Score:
Student ID 2:
Score:

APPENDIX C: Questionnaire

Your mother tongue is:

1. Was easy to communicate with the system?
   Very easy ☐ ☐ ☐ ☐ ☐ **Very difficult**

2. What do you think about the voice synthesis of the system?
   Very good ☐ ☐ ☐ ☐ ☐ **Very bad**

3. Was it easy to find the exam results you wanted? Comments are optional!

   Scenario 1 **--->** **Very easy** ☐ ☐ ☐ ☐ ☐ **Very difficult**

   Comments

   ____

   Scenario 2 **--->** **Very easy** ☐ ☐ ☐ ☐ ☐ **Very difficult**

   Comments

   ____

   Scenario 3 **--->** **Very easy** ☐ ☐ ☐ ☐ ☐ **Very difficult**

   Comments

   ____

   Scenario 4 **--->** **Very easy** ☐ ☐ ☐ ☐ ☐ **Very difficult**

   Comments

   ____

4. Did you know what you could say at each point of the dialogue? If not, where exactly did you have problems with the system?

   Yes ☐ **No** ☐

4.1 If you have selected **"No"**, where exactly did you have problems with the system? Give a short description of your problem!

5. Did the system work the way you expected it to?
   Yes ☐ ☐ ☐ ☐ ☐ **No**

6. What is your assessment of the general quality of the system?
   Very good ☐ ☐ ☐ ☐ ☐ **Very bad**

7. Is there anything you didn't like about the system? If yes, please describe briefly
____

8. What kind of improvements would you like?
____

9. Other Comments:
____

APPENDIX D: Subjective Measurements

```
####################################
#                                  #
#      DATA   EVALUATION           #
#                                  #
####################################
```

1. Number of tested person: 20

2. Native speaker:
German: 13
English:1
Polish: 2
Spanish:2
Tamil: 1
Korean: 1

3. Assessment of the communication with the system:

very easy:	6
easy:	9
so-so:	4
difficult:	1
very difficult:	0

4. Assessment of the tts:

very good:	1
good:	10
so-so:	7
bad:	2
very bad:	0

5. Assessment of Scenario 1:

very easy:	11
easy:	4
so-so:	3
difficult:	2
very difficult:	

6. Comments to Scenario 1:
1.- excellent recognition of course names, good recognition of matricel # (1x wrong)
2-
3-
4- at the beginning its a bit difficult
5- system doesn't understand first Number --> it was not possible to exit
6-
7-
8-
9- It is not necessary to repeat the complete instructions every time.
10- Die Information daß der Beepton aussage darüber trifft wann man reden soll könnte in einer Erklärung zu Anfang des Telefonats erfolgen.
11-
12-
13-
14-
15-
14-
16- a lot of mistakes
17-
18- erster Versuch fehlgeschlagen (zu stockend vorgelesen?);
19-
20-

7. Assessment of Scenario 2:

very easy:	12
easy:	5
so-so:	2
difficult:	1
very difficult:	

8. Comments to Scenario 2:
1- repetition came more often than wanted (yes->repeat)
2-
3-

APPENDIX D

4- it was ok
5- null
6-
7-
8-
9- Recognition failed for the first one.
10 Wie oben
11-
12-
13- no possibility to quit - see below
13-
14-
15-
16- it was better than scenario 1, but some mistakes too
17-
18- keine Probleme
19-
20-

9. Assessment of Scenario 3:

very easy:	4
easy:	4
so-so:	7
difficult:	5
very difficult:	

10. Comments to Scenario 3:
1- 1) ok, wrong spelling-> wrong #, corrected, ok 2)wrong spelling but correct # recognized yes->no ... so I had to call again 3)8x "no" recognized as "repeat"
2- When the ID number isn't correct, it's a little bit annoying to repeat the number before the system doesn't realize the it's a false one.
3-
4-
5-
6- The system doesn't understand the wrong number
7-
8-
9- Correction of first id: Recognition failed.
10- Bei diesem Szenario ist der Abgleich zwischen der falsch vorgelesenen und durch das System berichtigten Matrikelnummern Angabe beeindruckend. Beim Test mit einer nicht vorhandenen Matrikelnummer wurde in meinen Augen zu oft eine Komplett andere Matrikelnummer vom System vorgelesen dies kann zur verwirrung führen der Abbruch nach mehrmaliger Eingabe ist gut gelungen.
11- The system didn't recognize the correct number after 3 trials (1#st student).
12-
13- The system seems to find the correct numbers automatically - a bit embarrassing
14-
15-
16-
17- only some mistakes
18-
19- Fall "Wrong" und "Correct" nicht erkannt, leider auch auf die Frage "Is this correct" mein "no" als „yes „verstanden. Prozedur für die nicht vorhandene Matr.nr. etwas länglich, vielleicht eher ein Versuch weniger?
20- problems with "no"

11. Assessment of Scenario 4:

very easy:	12
easy:	4
so-so:	1
difficult:	3
very difficult:	

12. Comments to Scenario 4:
1- again problems with "yes" recognized as "repeat" beides that, all numbers recognizes on the first try, except for 1
2-
3-
4-
5-
6- The hang up in between
7-
8- die Nummer 04200364 wurde nicht erkannt (viele versuche)eine andere id wurde sofort erkannt
9- recognition failed for second id
10- Wie oben bei 1
11-
12-
13-
14-

APPENDIX D

15-
16-
17-
18-
19- keine Probleme
20- problems with "no"

13. Assessment of Dialogue structure:
yes: 15
no : 5

14. Comments to Dialogue structure:
1- null
2-
3-
4-
5- when telling me my matr.number the system was too fast sometimes
6- I could not stop the system after the course name was recognized.
7- null
8- welche antwort ich geben muss (ob ich „correct“ wiederholen sollte oder „yes“ sagen soll
9-
10-
11- Canceling the dialog is not possible if the computer understands "yes" when he asks you, if you want to know another result. -> hang up
12-
13- no possibility to quit or to get back to a "root" ... (I said "No", the system understood "yes" and I got stuck in a loop.)
14-
15-
16-
17-
18- At the End, the word "No" was not always detected!
19- Wusste nicht, wie ich das System abbrechen kann, nachdem es in eine falsche Richtung gelaufen war (s. 1. Szenario)
20-

15. Expectation:

totally agree:	6
agree:	12
somewhat agree:	2
disagree:	0
totally disagree:	0

16.System quality:

very good:	1
good:	14
so-so:	4
bad:	1
very bad:	

17. Discomfort:
1- 1) see afore mentioned "no"->"repeat" and "yes"->"repeat" 2) explanation dialogue before saying the number is too long / has only to come once (when asking for the first #)
2- Just what I say in the point 4
3- Text to speech is somewhat awkward. I am not sure that this could be improved, but otherwise, the system, in general, is pretty nice.
4- if the my number is not on the list the system repeats the question more times
5- sometimes too fast
6- The repetition all the time how to say the number. In one telephone call one explanation is enough.
7- Die Wiederholung der eingesprochenen Matr.Nr. war zu schnell
8-
9-
10- Die Stimme des Systems muss verbessert werden als Beispiel für eine gute Stimmensynthese kann z.B. UnrealTournament2004 herangezogen werden. Eine weibliche Stimme als Sprecherin macht das System vertrauenswürdiger und netter im Umgang als eine redende Blechkiste.
11-
12- repetitions
13- The voice: a bit too fast (in my opinion)
14-
15- it confuse yes and no
16- Repeating of "don't mention 1080" is perhaps to often
17- in scenario 3 the mistakes were ignored
18- null
19- Die Künstlichkeit der synthetischen Stimme missfällt.
20- Ein Vermerk könnte nützlich sein, wenn die ID gar nicht in der Datenbank vorhanden ist

18. Improvements:
1- see 7)

APPENDIX D

2- to fix the sited problem in the scenario 3
3- Include barge in capability. When I need to ask the score for the second time, i don't need to be given the whole sermon of how i should leave out the first 4 digits and so on. Dialogues in the second round and above should be made shorter.
4- woman voice
5- opportunity to exiting the system, when the system doesn't understand me
6- It should be possible to answer during the system is asking to shorten the process
7- Matr.Nr. langsamer wiederholen. Einige "Yes" wurden als "NO" verstanden
8-
9- The spaces between the words of the instructions are misleading and it seems to speak with varying speed. Word recognition has to improve.
10- Zu Anfang eine Einführung mit sprechen Sie immer nach dem Piepton ...Und definitif eine andere Stimme.
11- The digits are repeated too fast.
12- more possibilities to interrupt the system
13- a possibility to stop and get back to a "root point"
14-
15- better understanding of yes and no, shorter introduction if you know the system
16-
17- a repeat-possibility
18- Better reaction after mistakes
19- natürlichere Stimme, Abbruchmöglichkeiten
20-

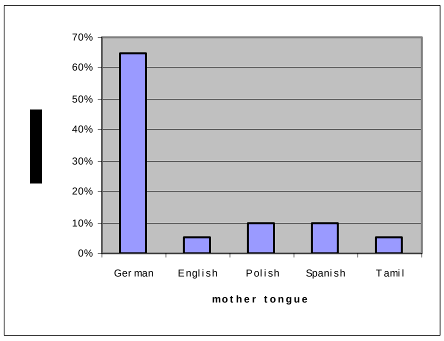
19. Other Comments:
1- well done ;-) hope you get some sleep sometime ...
2-
3- include a general statement at the start of the dialog that one should speak only AFTER the tone
4-
5-
6-
7-
8-
9- Personally, I'd prefer to look up my results on the net. I don't like talking to machines.
10- Keeeeeeekseeeeee
11-
12-
13 -
14- The system has problems to recognize the numbers, when you aren't speak fast enough.
15- Viel Erfolg ;)
16-
17-
18-
19-
20- Wenn nichts erkannt worden ist, hat das System von sich einen Vorschlag gemacht

APPENDIX E: Subjective criteria

1.Preliminaries

Mother Tongue

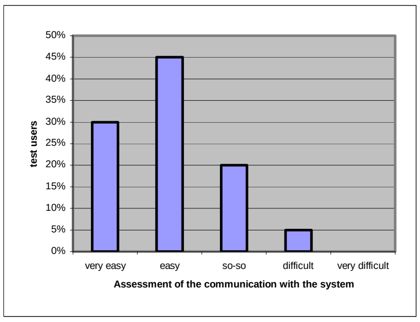
Language	%
German	65%
English	5%
Polish	10%
Spanish	10%
Tamil	5%
Korean	5%



2. Subjective Criteria

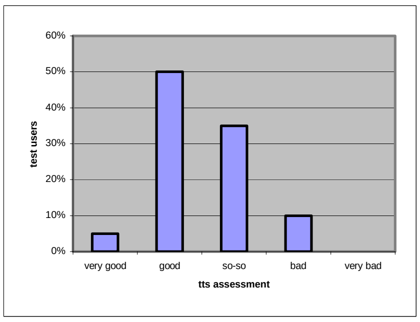
1. ASR

very easy	easy	so-so	difficult	very difficult
30%	45%	20%	5%	0%



2. TTS

very good	good	so-so	bad	very bad
5%	50%	35%	10%	0%

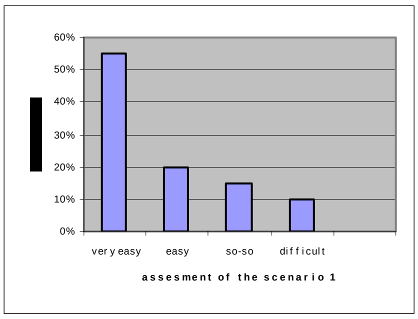


APPENDIX E

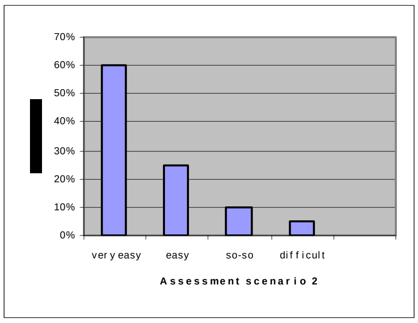
3. Task ease

Scenario	very easy	easy	so-so	difficult	very difficult
1	55%	20%	15%	10%	0%
2	60%	25%	10%	5%	0%
3	20%	20%	35%	25%	0%
4	60%	20%	5%	15%	0%

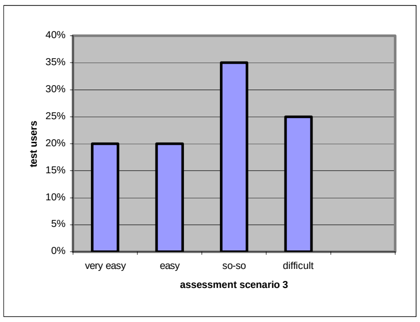
Scenario 1



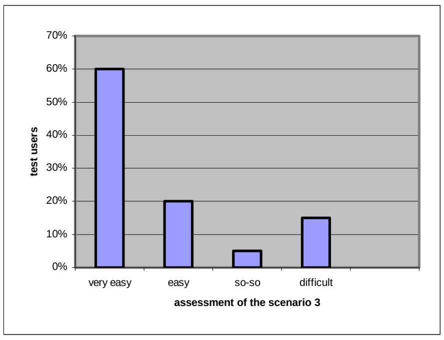
Scenario 2



Scenario 3



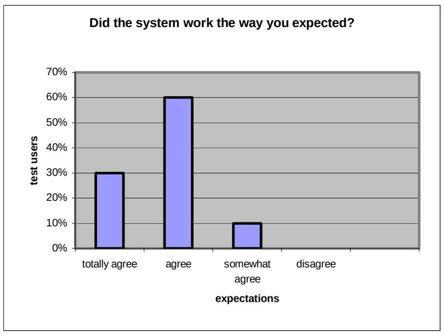
Scenario 4



APPENDIX E

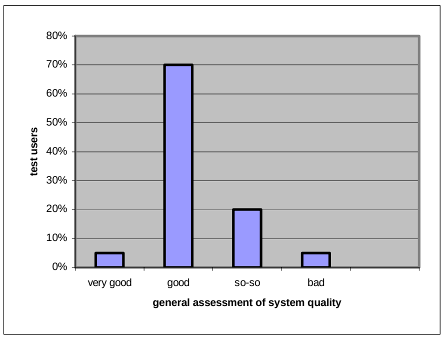
4. Expectation

totally agree	agree	somewhat agree	disagree	totally disagree
30%	60%	10%	0%	0%



5. General System quality

very good	good	so-so	bad	very bad
5%	70%	20%	5%	0%



6. Comments

Problems /Comments	**Amount of user**	**%**
Recognition	13	65%
Escape option and barge-in	6	60%
Dialog duration too long	8	40%
Handling with non existent object	5	25%
Vocabulary enlargement	2	10%
Not clear how to answer	1	5%
TTS	9	45%
Positive feedback	5	25%
Other comments	1	5%

APPENDIX E

Remark concerning	Person No.	Remark content
Recognition	1	- repetition came more often than wanted (yes->repeat) - yes->no recognition problems... so I had to call again (**scenario 1**) - 8x "no" recognized as "repeat" (**scenario 2**) - again problems with "yes" recognized as "repeat" besides that, all numbers recognizes on the first try, except for 1
	5	- system doesn't understand first number (**scenario1**)
	7	- Einige "Yes" wurden als "NO" verstanden
	8	die Nummer 04200364 wurde nicht erkannt (viele versuche) eine andere ID wurde sofort erkannt (**scenario 4**)
	9	- erster Versuch fehlgeschlagen (zu stockend vorgelesen?) (**scenario1**) - Word recognition has to improve. - Recognition failed for the first one (**scenario 2)** - Correction of first id: recognition failed (**scenario 3**) - recognition failed for second id (**scenario 4**)
	11	- the system didn't recognize the correct number after 3 trials (1#st student) **(scenario 3)**
	12	-problems with "no" **(scenario 3)**
	13	- I said "No", the system understood "yes" and I got stuck in a loop
	15	- it confuse yes and no -better understanding of yes and no,
	18	- At the End, the word "No" was not always detected!
	17	-A lot of mistakes **(scenario 1)** -it was better than scenario 1, but some mistakes too **(scenario 2)** - only some mistakes **(scenario 3)**
	19	-leider auch auf die Frage "Is this correct" mein "no" als „yes“ verstanden.**(scenario 3)**
	20	-problems with "no"
Escape option and barge-in	5	- opportunity to exiting the system, when the system doesn’t understand me
	6	-I could not stop the system after the course name was recognized.
	7	-Cancelling the dialog is not possible if the computer understands "yes" when he asks you, if you want to know another result. -> hang up
	12	-more possibilities to interrupt the system
	13	- a possibility to stop and get back to a "root point" - no possibility to quit - Abbruchmöglichkeiten
	19	- wusste nicht, wie ich das System abbrechen kann, nachdem es in eine falsche Richtung gelaufen war
Dialog duration too long	1	-explanation dialogue before saying the number is too long / has only to come once (when asking for the first #)
	3	- Include barge in capability. When I need to ask the score for the second time, I don't need to be given the whole sermon of how i should leave out the first 4 digits and so on. Dialogues in the second round and above should be made shorter. - include a general statement at the start of the dialog that one should speak only AFTER the tone
	6	-The repetition all the time how to say the number. In one telephone call one explanation is enough. It should be possible to answer during the system is asking to shorten the process
Dialog duration too long	9	-It is not necessary to repeat the complete instructions every time.
	10	-Zu Anfang eine Einführung mit sprechen Sie immer nach dem Piepton Die Information, daß der Beepton Aussage darüber trifft wann man reden soll könnte in einer Erklärung zu Anfang des Telefonats erfolgen.
	15	-shorter introduction if you know the system
	12	-repetitions
	16	-Repeating of "don't mention 1080" is perhaps to often
Criticizing	2	- when the ID number isn't correct, it's a little bit annoying to repeat the number before the system doesn't realize the it's a false one
	4	-if the my number is not on the list the system repeats the question more times

the way of handling with non-existent object	10	-Beim Test mit einer nicht vorhandenen Matrikelnummer wurde in meinen Augen zu oft eine Komplett andere Matrikelnummer vom System vorgelesen dies kann zur Verwirrung führen der Abbruch nach mehrmaliger Eingabe ist gut gelungen.
	19	-Prozedur für die nicht vorhandene Matr.nr. etwas länglich, vielleicht eher ein Versuch weniger?
	20	-Ein Vermerk könnte nützlich sein, wenn die ID gar nicht in der Datenbank vorhanden ist
Vocabulary enlargement	17	- a repeat possibility
	19	-Fall "Wrong" und "Correct" nicht erkannt
Not sure how to answer	8	-welche Antwort ich geben muss (ob ich „correct“ wiederholen sollte oder „yes“ sagen soll
TTS	3	- Text to speech is somewhat awkward. I am not sure that this could be improved
	4	- woman voice
	5	- when telling me my matr. number the system was too fast sometimes - sometimes too fast
	7	- die Wiederholung der eingesprochenen Matr.Nr. war zu schnell - Matr.Nr. langsamer wiederholen.
	9	- The spaces between the words of the instructions are misleading and it seems to speak with varying speed.
	10	-Die Stimme des Systems muss verbessert werden als Beispiel für eine gute Stimmensynthese kann z.B. UnrealTournament2004 herangezogen werden. Eine weibliche Stimme als Sprecherin macht das System vertrauenswürdiger und netter im Umgang als eine redende Blechkiste
	11	- The digits are repeated too fast.
	13	- The voice: a bit too fast (in my opinion)
	19	- die Künstlichkeit der synthetischen Stimme missfällt. - natürlichere Stimme
Positive feedback	1	- excellent recognition of course names, good recognition of matricel # (1x wrong) - well done ;-)
	3	- the system, in general, is pretty nice
	4	- it was ok.
	11	- keine Probleme
	19	-keine Probleme
Other comments	18	-Better reaction after mistakes

APPENDIX F: Objective Measurements

```
#################
# Test person  #
#################

Scenario 1:
-----------
Turn correction ratio:
Transaction success:
succeed:
partial succeeded :
succeed in spotting that no answer exists:
fail:

Dialog duration:
Start:
End:    + 10 sec (quit state) =
Total:
input repeat request:

Recognition mistakes:



Scenario 2:
-----------
Turn correction ratio:
Transaction success:
succeed:
partial succeeded:
succeed in spotting that no answer exists
fail:

Dialog duration:
Start:
End:    + 10 sec (quit state) =
Total:
input repeat request:

Recognition mistakes:



Scenario 3:
-----------
Turn correction ratio:
Transaction success:

succeed:
partial succeeded:
Succeed in spotting that no answer exists
fail:
Total:
input repeat request:

Dialog duration:
Star:
End:    + 15 sec (quit state - failure)=

Recognition mistakes:
```

Appendix G: Objective Criteria

1. Turns correction ratio

TRC per test user (TU) and scenario type

Scenarios	TU1	TU2	TU3	TU4	TU5	TU6	TU7	TU8	TU9	TU10	Total A
Scenario 1	2	4	0	6	6	0	0	6	0	2	**26**
Scenario 2	5	4	0	0	0	0	0	0	0	0	**9**
Scenario 3	5	0	0	2	0	0	2	0	2	0	**11**
Scenario 4	6	2	0	2	2	4	4	18	6	0	**44**

Scenarios	TU11	TU12	TU13	TU14	TU15	TU16	TU17	TU18	TU19	TU20	Total B
Scenario 1	4	4	2	8	4	0	4	7	3	10	**46**
Scenario 2	2	0	10	2	0	0	0	2	0	0	**16**
Scenario 3	5	0	0	2	4	0	2	0	0	16	**29**
Scenario 4	0	2	0	4	19	2	2	2	0	4	**35**

Scenarios	Total A	Total B	No. of turns
Scenario 1	26	46	**72**
Scenario 2	9	16	**25**
Scenario 3	11	29	**40**
Scenario 4	44	35	**79**

TCR - Total average per scenario type

Scenarios	No. of states per scenario	No. of TCR	TCR %
Scenario 1	449	72	16.03%
Scenario 2	565	25	4.42%
Scenario 3	324	40	12.34%
Scenario 4	423	79	18.67%
TOTAL	**1758**	**210**	**11.94%**

Appendix G

2. Transaction success

User files

Scenarios	No. of dialogues	Succeeded (S)		succeed in spotting that no answer exist (SN)		Partial succeeded (PS)		Fail (F)	
		No.	Ratio	No.	Ratio	No.	Ratio	No.	Ratio
Scenario 1	23	19	82.60%			3	13.04%	1	4.34%
Scenario 2	20	19	95.00%					1	5.00%
Scenario 3	22	13	59.09%	11	50%	7	31.81%	2	9.09%
Scenario 4	26	19	73.07%			1	3.84%	6	23.07%

Reference file

Scenarios	No. of dialogues	Succeeded (S)		succeed in spotting that no answer exist (SN)		Partial succeeded (PS)		Fail (F)	
		No.	Ratio	No.	Ratio	No.	Ratio	No.	
Scenario 1	1	1	100%						
Scenario 2	1	1	100%						
Scenario 3	1	1	100%	1	100%				
Scenario 4	1	1	100%						

Appendix G

3. Dialogue Duration

Dialogue duration (min) per test user and scenario type

Scenarios	TU1		TU2	TU3	TU4	TU5		TU6	TU7
Scenario 1	3.38		4.12	3.01	3.00	2.11	3.31	4.19	3.04
Scenario 2	4.07		2.33	2.20	2.03	2.04		2.04	1.04
Scenario 3	3.16	4.24	4.10	4.44	7.22	5.09		3.04	2.52
Scenario 4	2x4.23[17]		3.18	3.18	3.19	2.22	3.23	4.61	4.33

Scenarios	TU8				TU9	TU10	TU11		TU12
Scenario 1	3.27		3.06		3.05	3.32	3.27		4.14
Scenario 2	2.09				2.14	1.59	2.29		1.58
Scenario 3	4.50				4.50	4.49	2.17	2.49	4.33
Scenario 4	2.14	2.14	2.18	3.40	4.35	3.09	2.57		3.20

Scenarios	TU13	TU14	TU15			TU16	TU17	TU18	T19	T20
Scenario 1	3.24	4.26	3.29			3.00	3.54	4.29	5.00	4.04
Scenario 2	3.06	2.34	2.00			2.00	2.05	2.29	2.01	2.01
Scenario 3	4.10	4.12	5.20			4.29	5.24	2.26	4.84	9.29
Scenario 4	2.48	5.36	2.52	2.43	3.31	5.59	3.30	3.21	2.58	3.01

Total average dialogue duration per scenario type

Scenarios	Amount of dialogues	Total duration	Average duration
Scenario 1	23	77.13	3.36
Scenario 2	20	43.20	2.16
Scenario 3	22	96.03	4.36
Scenario 4	26	81.85	3.14
TOTAL	**91**	**292.21**	**3.21**

Comparing the reference file

Scenarios	Average duration (min)	Reference file duration (min)	Average difference (min)
Scenario 1	3.36	2.58	0.38
Scenario 2	2.16	2.01	0.15
Scenario 3	4.36	4.26	0.10
Scenario 4	3.14	2.49	0.25

[17] The test person did the double amount of dialogues required for that scenario. We did the calculation using the half value (4.235)

Appendix G

4. Input repeat request

Input repetition request per test user and scenario type

Scenarios	TU1	TU2	TU3	TU4	TU5	TU6	TU7	TU8	TU9	TU10	Total A
Scenario 1	1	2	0	5	1	2	0	0	0	0	**11**
Scenario 2	12	0	1	0	0	0	0	0	1	0	**14**
Scenario 3	19	0	0	1	0	0	0	0	2	2	**24**
Scenario 4	24	0	1	0	0	0	0	0	0	2	**27**

Scenarios	TU11	TU12	TU13	TU14	TU15	TU16	TU17	TU18	TU19	TU20	Total B
Scenario 1	0	0	0	0	0	0	2	0	0	0	**2**
Scenario 2	0	0	2	0	0	0	0	0	0	0	**2**
Scenario 3	0	0	0	0	1	1	0	0	0	0	**2**
Scenario 4	0	0	0	0	0	1	0	0	0	0	**1**

Scenarios	Total A	Total B	No. Input repetition request
Scenario 1	11	2	**13**
Scenario 2	14	2	**16**
Scenario 3	24	2	**26**
Scenario 4	27	1	**28**
TOTAL	**76**	**7**	**83**

Total Average of Input repeat request per statement

Scenarios	Amount of dialogues	Input repeat request rate
Scenario 1	449	2.89%
Scenario 2	565	2.83%
Scenario 3	324	8.02%
Scenario 4	423	6.61%
TOTAL	**1758**	**4.72%**

Appendix G

5. Recognition errors

Recognition errors per test user (TU) and scenario type

Scenarios	TU1	TU2	TU3	TU4	TU5	TU6	TU7	TU8	TU9	TU10	Total A
Scenario 1	1	2	0	0	4	0	0	4	0	1	**12**
Scenario 2	2	2	0	0	1	0	1	0	0	0	**6**
Scenario 3	8	1	0	3	0	0	1	1	2	0	**16**
Scenario 4	3	1	0	1	1	1	2	12	4	0	**25**
TOTAL	**14**	**6**	**0**	**4**	**6**	**1**	**4**	**17**	**6**	**1**	**59**

Scenarios	TU11	TU12	TU13	TU14	TU15	TU16	TU17	TU18	TU19	TU20	Total B
Scenario 1	1	1	1	5	2	0	4	2	2	5	**23**
Scenario 2	0	0	2	1	0	0	1	1	0	0	**5**
Scenario 3	3	0	0	0	3	0	1	1	1	8	**17**
Scenario 4	1	1	0	2	9	1	3	0	0	2	**19**
TOTAL	**5**	**2**	**3**	**8**	**14**	**1**	**9**	**4**	**3**	**15**	**64**

Scenarios	Total A	Total B	Total errors
Scenario 1	12	23	**35**
Scenario 2	6	5	**11**
Scenario 3	16	17	**33**
Scenario 4	25	19	**44**
TOTAL	**59**	**64**	**123**

Total average recognition errors per scenario type

Scenarios	Amount of dialogues	Total errors	Average per scenario
Scenario 1	23	**35**	1.52
Scenario 2	20	**11**	0.55
Scenario 3	22	**33**	1.50
Scenario 4	26	**44**	1.69
TOTAL	91	**123**	1.35

Word accuracy

Scenarios	TOTAL Words	Total errors	Error Rate	Recognition Rate
Scenario 1	449	**35**	7.79%	92.21%
Scenario 2	565	**11**	1.94%	98.06%
Scenario 3	324	**33**	10.18%	89.82%
Scenario 4	423	**44**	10.40%	89.60%
TOTAL	**1758**	**123**	**6.99%**	**93.01%**

Appendix G

Recognition mistake identification

User	Total recognition errors	Recognition error	Fail type	Category
TU1	14	1	wrong digit pronunciation	**SP**
		1	input to slowly	**SF**
		1	input distorted	**Wsetup**
		1	hesitation	**SP**
		Σ = 4		
TU2	6	4	wrong pronunciation	**LI**
		1	input to slow	**SF**
		1	hesitation and too slowly input	**SP**
		Σ = 6		
TU3	0			
TU4	4	2	input too slowly	**SF**
		1	wrong input containing 1080	**WrongI**
		1	input distorted	**Wsetup**
		Σ = 4		
TU5	6	3	inputs to slowly	**SF**
		Σ = 3		
TU6	1	1	no input recorded	**TF**
		Σ = 1		
TU7	4	1	due to a very noisy breathing	**BN**
		2	input distorted	**Wsetup**
		Σ = 3		
TU8	16	2	non vocabulary words used ,"correct/wrong"	**LC**
		2	pronunciation mistakes	**LI**
		2	inputs too slowly	**SF**
		Σ = 6		
TU9	6	1	due to too much background noise	**BN**
		1	wrong input	**WrongI**
		Σ = 2		
TU10	1	1	input distorted	**Wsetup**
		Σ = 1		
TU11	5	1	non vocabulary words used , "no thank you"	**LC**
		1	due to too much background noise	**BN**
		3	inputs too slowly	**SF**
		Σ = 5		
TU12	2	1	no input recorded	**TF**
		1	hesitation and too slowly input	**SP + SF**
		Σ = 2		
TU13	3	1	input too slowly	**SF**
		1	input distorted	**Wsetup**
		Σ = 2		
TU14	9	1	wrong input	**WrongI**
		3	inputs too loud	**SF**
		2	inputs too slowly	**SF**
		Σ = 6		
TU15	14	1	caused by too much background noise	**BN**
		3	inputs too slowly	**SF**
		1	input distorted	**Wsetup**
		1	no input recorded	**TF**
		Σ = 6		
TU16	1	1	no input recorded	**TF**
		Σ = 1		
User	**Total recognition errors**	**Recognition error**	**Fail type**	**Category**
TU17	9	3	mistake caused by too much background noise	**BN**
		2	input containing too long utterances and "1080"	**UC + WrongI**
		1	input too loud	**SF**
		1	no input recorded	**TF**
		Σ =7		
TU18	4	1	no input recorded	**TF**
		1	unclear pronunciation	**SP**
		Σ =2		
TU19	3	1	no input recorded	**TF**

		1	input too slowly	**SF**
		Σ =2		
TU20	15	1	input containing too long utterances	**UC**
		2	wrong digit pronunciation using German	**LI**
		2	inputs too slowly	**SF**
		1	no input recorded	**TF**
		Σ =6		
TOTAL	**123**	**65**		

Recognition error statistics

Not identified recognition mistake	% to total recognition amount	Identified recognition mistake	% to total recognition amount	Total recognition errors
58	**47.15%**	**65**	**52.84%**	**123**

Legend

Category	**Explanation**
LI	Wrong pronunciation due to language interference
WrongI	User enters wrong input ignoring system requirements
SP	Spontaneous speech phenomena, like hesitations, repetitions, incomplete utterances, unclear pronunciation
UC	Utterance complexity: too many statements at the time, greetings etc
LC	Insufficient lexical coverage: it occurs when a word is used, which is not in the language model lexicon
BN	Background noise
SF	Inadequate speech feature: to slow or to loud speech input
Wsetup	Wrong parameter set up: it occurs when parameters are not correctly adjusted for the application in use.
TF	Technical fails like no input recorded

Appendix G

6. Technical fails per dialogue state

States	**No. of states per user**	**strongly distorted**		**word beginning missing**		**low tone**	
		No.	**%**	**No.**	**%**	**No.**	
confirmation	449	24	5,34%	27	6.01%	64	14.25%
transition2	565	41	7.25%	37	6.54%	43	7.61%
make_choice	324	0	0.00%	7	2.16%	10	3.08%
runtime_tree	423	0	0.00%	0	0.00%	0	0.00%
TOTAL	**1758**	**65**	**3.69%**	**71**	**4.03%**	**117**	**6.65%**

States	**No. of states per user**	**No input recorded**		**Word end missing**	
		No.	**%**	**No.**	**%**
confirmation	449	9	2.00%	1	0.22%
transition2	565	34	6.01%	1	0.17%
make_choice	324	10	3.08%	3	0.92%
runtime_tree	423	0	0.00%	0	0.00%
TOTAL	**1758**	**53**	**3.01%**	**5**	**0.28%**

States	**No. of states per user**	**TOTAL fails**	
		No.	**%**
confirmation	449	125	27.83%
transition2	565	156	27.61%
make_choice	324	30	9.25%
runtime_tree	423	0	0.00%
TOTAL	**1758**	**311**	**17.69%**

Appendix G

7. User turns per state

States	**TP1**	**TP2**	**TP3**	**TP4**	**TP5**	**TP6**	**TP7**	**TP8**	**TP9**	**TP10**
confirmation	65	23	21	22	22	19	11	33	21	18
transition2	29	33	26	32	22	25	20	32	29	24
make_choice	16	18	15	18	12	13	11	17	13	12
runtime_tree	23	25	21	24	23	21	11	32	21	16
TOTAL	**133**	**99**	**83**	**96**	**79**	**78**	**53**	**114**	**84**	**70**

States	**TP11**	**TP12**	**TP13**	**TP14**	**TP15**	**TP16**	**TP17**	**TP18**	**TP19**	**TP20**
confirmation	17	16	15	20	35	15	16	17	18	25
transition2	24	23	25	29	40	23	30	30	26	43
make_choice	13	16	17	17	24	11	21	19	15	26
runtime_tree	17	18	15	21	37	15	18	18	18	29
TOTAL	**71**	**73**	**72**	**87**	**136**	**64**	**85**	**84**	**77**	**123**

User files

TOTAL States	**Total no. of dialog**	**Average statements per dialog**
1758	**91**	**19.31**

Reference file

TOTAL States	**Total no. of dialogues**	**Average statements per dialogue**
63	**4**	**15.75**