Auditory User Interfaces

for Desktop, Mobile and Embedded Applications

by

MARTIN KALTENBRUNNER

Diploma Thesis

Department of Media Technology and Design
Polytechnic University of Upper Austria at Hagenberg

June 2000

Supervisor: Dr. Christoph Schaffer

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, Juni 2000 Martin Kaltenbrunner

I. Contents

II. Preface

During my studies at the Polytechnic University of Hagenberg in Austria, I have been working on several virtual reality projects incorporating speech technology. Although there are already some quite convenient tools available, which allow the easy development of speech enabled applications, I always was confronted with the general design problem of such an interface. Then last year during my internship at the Centre for Electronic Arts in London, I had the opportunity to gain some insight into the general research issues concerning the design of auditory user interfaces. There the idea for this thesis was born, which is one of the reasons why I decided to write it in English as well as I hope to reach a larger possible audience instead of writing it in German.

I' d like to thank several people at this opportunity, who helped and supported me during the last few years. This includes some of my teachers in Hagenberg who taught me the basics of computer science and multimedia design, but first of all Dr. Christoph Schaffer who guided me firmly through the process of writing this thesis. I especially thank Dr. Avon Huxor, senior research fellow at the Centre for Electronic Arts, for introducing me to the scientific research methods in a very open minded way during my internship. Then of course I'd like to thank my family, first of all my mother and my sister, who supported me and believed in me during the last years. Without them it would have been significantly harder for me to reach the point where I am now. Also to mention is Hugh Huddy, a Ph.D. student at the Centre for Electronic Arts, who gave me some basic insight into the way how blind people observe their sonic surroundings and who also provided me with several initial ideas for this paper.

Martin Kaltenbrunner Hagenberg, June 2000

IIIa. Abstract

Current modern graphical user interfaces are ergonomic and convenient and the desktop metaphor allows virtually any novice computer user to become familiar with the intuitive concept of pointing and clicking with the mouse. But Graphical User Interfaces (GUI) are not suitable in all circumstances. Sight disabled users still prefer text based console applications because those interfaces can be more easily spoken by their voice synthesiser devices. Mobile devices with their limited screen sizes and insufficient keyboards demand for a completely different solution. And the solution is the auditory user interface. Speech synthesis and recognition as well as the necessary processing power are already available for both workstations and mobile personal digital assistants. But since the engineering goals in speech technology are almost sufficiently achieved and only need some more perfection in accuracy in noisy environments, the major task is now the design of an auditory interface as convenient and powerful as current desktop GUI applications.

The first part of this thesis will present an overview on current speech technology products on the market, it will give some insight into the basic technology concepts of speech recognition and synthesis as well as in current standard programming technologies.
The second section will describe current research activities around auditory user interface design. There are several pioneers like T.V. Raman [RAM97] who already developed very sophisticated concepts for the design of auditory interfaces, such as the Emacspeak auditory desktop [RAM00]. Finally these basics will be brought together in the concept of Y-Windows, an approach to create a general auditory interface environment similar to the way X-Windows [X1198] provides GUI functionality for UNIX™ systems. There is an evident need for a concept like this to provide the basic underlying infrastructure for the breakthrough of standardised applications with Auditory User Interfaces (AUI), preventing application developers from inventing the wheel again and again.

IIIb. Kurzfassung

Moderne graphische Benutzerschnittstellen sind bequem und ergonomisch, die Desktop-Metapher erlaubt es sogar unerfahrenen Computerbenutzern sich schnell und ohne allzu großen Lernaufwand mit Hilfe des Konzeptes vom Zeigen und Klicken mit der Maus zurechtzufinden. Aber Graphische Benutzerschnittstellen (GUI) eignen sich nicht für alle Anwendungen. Sehbehinderte Benutzer bevorzugen daher etwa immer noch textorientierte Anwendungen, weil deren Benutzerschnittstelle einfacher von den verwendeten Sprachsynthesizern gesprochen werden kann. Mobile Geräte mit ihren klein dimensionierten Bildschirmen und unbequemen winzigen Tastaturen verlangen ebenfalls nach einer vollkommen anderen Lösung: Und diese Lösung ist die akustische Benutzerschnittstelle (AUI), die sowohl moderne Sprachtechnologie als auch bewährte Audiokonzepte integriert. Sprachsynthese und -erkennung und auch die notwendige Prozessorleistung sind heute bereits für Personal Computer wie auch für mobile Assistenten (PDAs) verfügbar. Aber obwohl die technischen Anforderungen im Bereich der Sprachtechnologie allmählich ausreichend umgesetzt sind und nur mehr einiger Optimierungen wie etwa bezüglich der Erkennungsgenauigkeit in lauten Umgebungen benötigen, liegt die große Herausforderung im Design der akustischen Benutzerschnittstelle selbst. Diese sollte genau so vielseitig und bequem sein wie die bewährten graphischen Benutzerschnittstellen.

Der erste Teil dieser Diplomarbeit präsentiert eine Marktübersicht über das derzeitige Angebot an Sprachtechnologie. Es werden darin sowohl die grundlegenden Konzepte der Spracherkennung und -synthese beleuchtet als auch alle in diesem Bereich verfügbaren technischen Standards präsentiert. Der zweite Teil beschreibt den derzeitigen Stand der Forschung rund um das Konzept der akustischen Benutzerschnittstelle. Es gibt wenige Pioniere wie T.V. Raman [RAM97] die bereits sehr ausgereifte Konzepte für die Gestaltung derartiger Benutzerschnittstellen, wie etwa den Emacspeak Auditory Desktop [RAM00] entwickelt haben. Im letzten Teil dieser Arbeit werden diese Grundlagen im Konzept von Y-Windows umgesetzt. Dies ist ein Versuch eine standardisierte akustische Benutzerumgebung zu spezifizieren, ähnlich wie etwa X-Windows [X1198] die graphische Benutzerschnittstelle für verschiedene Unix™ Betriebssysteme definiert. Es besteht eine klare Notwendigkeit für Konzepte wie dieses, die die grundlegende Infrastruktur für die vermehrte Entwicklung von Anwendungen mit akustischer Benutzerschnittstelle zur Verfügung stellen. Dies soll vor allem Anwendungsentwickler davor bewahren, das Rad immer wieder von Neuem erfinden zu müssen.

1. Basic Technologies

1.1. Speech Synthesis

1.1.1. General Structure of Speech Synthesis

Speech synthesis is a technology for generating artificial speech from a textual input. It is therefore also often referred as a Text to Speech (TTS) System. In general a modern TTS system has to implement a specific set of modules, which are processed sequentially in order to generate the most natural speech output [TAY98b]:

- Normalisation: This module has to prepare more complicated text input such as numbers, dates and time into a separable form. So for example 21/01/72 will be transformed to "twenty first of January nineteen hundred and seventy two"
- Pronunciation: Here the list of words is converted to their phonetic equivalent. Normally this is done by a simple dictionary, while for unknown words common pronunciation rules are used.
- Phrasing: The Phrasing modules adds pauses where necessary, for example after the end of sentences or after a comma.
- Intonation, Duration, Rhythm: Speech has a special intonation melody, the so called f0-contour which controls the pitch of the sentence. A good example is the pitch raise at a question mark or the pitch fall at the end of a normal sentence. It also considers the special accentuation of emphasised words.
- Waveform Generation: Given the phonetic transcription complete with its duration and pitch information, the waveform can be produced either by concatenative synthesis using a speech database, where subsequent signal processing is used to modify the pitch and duration, or by complete synthesis using a physical formant model.

1.1.2. Evaluation of Speech Synthesis Systems

There are three main factors to evaluate the quality of synthesised speech: *naturalness*, *intelligibility* and *expressiveness* [RAM97]. Today most synthesised speech still sounds definitely artificial, although modern synthesisers already achieve a high standard of quality. Many people consider naturalness as an important feature of an auditory user interface, although objectively it isn't that important as intelligibility. Imagine a drunken person whose speech may sound perfectly natural but hardly understandable. Therefore current

speech synthesisers can all be considered as perfectly intelligible without being natural. Nevertheless naturalness is a major task for the future development of TTS systems. Another important future challenge is the expressiveness of synthesised speech output, such as emotions like sadness or excitement. Expression in general allows the transport of important meta-information along with the simple speech. In fact this meta-information plays a significant role in human communication. Only the right intonation can reveal if a statement was meant to be serious or maybe cynical.

1.1.3. Waveform Concatenation

The easiest type of Speech Synthesis is the simple concatenation of pre-recorded sentences, words and numbers. This technique is often used in trains, airports and similar announcement systems, which only require a limited set of pre-recorded voice samples such as numbers, station names and a few additional words. The advantage of such systems is their natural sound, the samples are often spoken by trained actors, and the low consumption of processing power. On the other hand these solutions are very inflexible and memory consuming. This principle was also used in early analog speech synthesis applications such as the speaking clock by British Telecom in the 1930's.

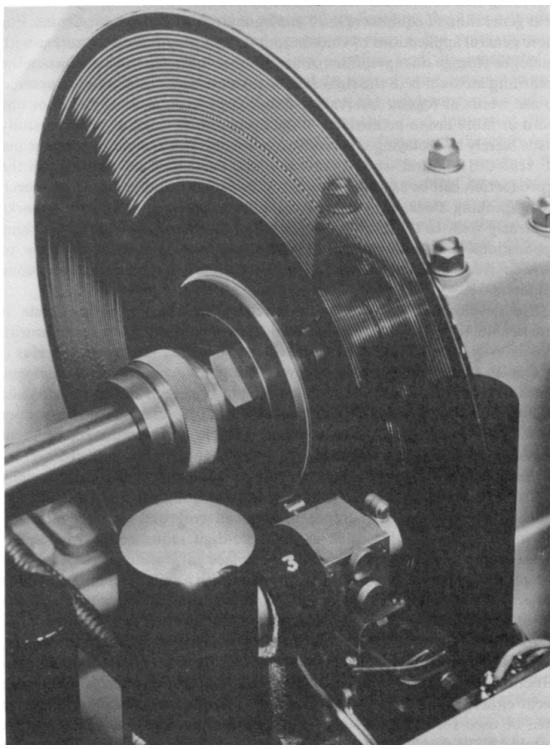

Fig. 1.1.: The glass disk of the 1936 BT speaking clock [HOL93].

1.1.4. Phoneme Synthesis

Phoneme Synthesis works similar to the concatenation of pre-recorded words described above. But here the pre-recorded samples of spoken language are divided into so-called phonemes, which represent the basic atomic structures of a language. Generally a phoneme is "the smallest unit in speech where substitution of one unit for another might make a distinction of meaning" [HOL93, p.4].

The amount of phonemes in a language is definitely larger than the simple number of vowels and consonants in the written language. This results in the different pronunciation of these in various contexts, such as the vowel "a" which is pronounced differently in the words "hat" and "waste", therefore these vowels are clearly different phonemes. The exact number of phonemes is characteristic for each language, generally English consists of about 44 phonemes, which also depends on the dialect that is used.

This type of speech synthesis is an elegant way to create more flexible systems which allow the synthesis of any desired text. Therefore all existing phonemes have to be extracted from a recorded language sample and stored in a phoneme database file. There has to be a specific phoneme database for any desired language or dialect and of course different genders as well.

To synthesise speech from any given text this text has to be analysed to create a phoneme representation of itself. Because the pronunciation of many words isn't exactly like the standard rules there is also a pronunciation dictionary necessary to improve the quality of the spoken output. These dictionaries are generally large lists of words with their according pronunciation. Once the text is successfully translated into its phonetic equivalent the synthesising process can be started. In general only the corresponding phonemes from the language database are concatenated to a continuous audio stream.

1.1.5. Diphone Synthesis

Although the phoneme synthesis allows very flexible and easy to implement speech synthesis applications this approach has a fundamental problem: The synthesised output sounds too artificial and monotonous. This happens because the simple phoneme concatenation doesn't consider the various different appearance of phonemes in combination with others. To solve this problem the diphone synthesis was created. A diphone is a pair of phonemes which represents the transition between them rather than the phonemes themselves. The concatenation of these phoneme pairs results in a

synthesised speech output of much higher quality. In a signal processing step the context of the given text is considered, such as question marks etc. to create a less monotonous and even more natural speech by adjusting the pitch of the signal. This step significantly increases the amount of processing power required. The effort to create a diphone database though is much higher than the creation of a simple phoneme database. The number of diphones is slightly lower than square of the number of phonemes because not all possible combinations are used. English for example uses with its 44 phonemes about 1600 diphones, which results in a much larger database of about 10 Megabytes, depending on the sample rate of the digitised language recording.

A brilliant example of the quality of diphone synthesis is the *Mbrola* [MBROL] synthesiser. This program accepts the input of text in phonetic notation and synthesises the speech with a given diphone database. There already exists a vast number of diphone databases for this application. In order to create the phonetic text a special pre-processor and dictionary for every language is also required.

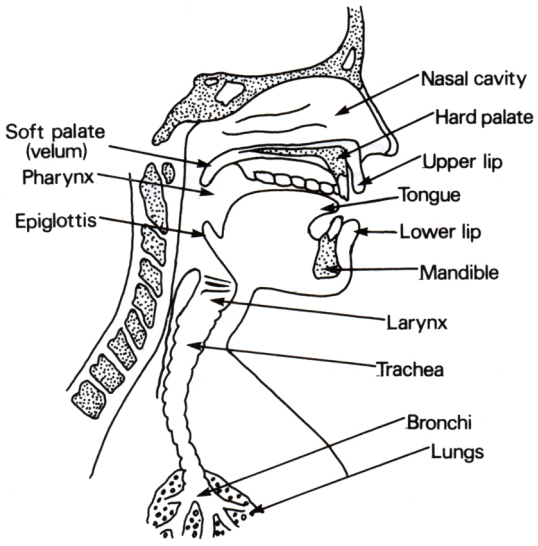

Fig. 1.2: Model of the human speech production organs [HOL93].

1.1.6. Formant Synthesis

This is a complete synthesis of artificial speech without the use of any digitised human speech samples. Therefore a physical model of the human speech production apparatus is mapped onto a series of signal processing algorithms. This results in a set of digital filters which modify the signal of a simple waveform production source like the human vocal cords in the larynx. In order to create those algorithms a large number of human speech

production organs have to be considered, such as the trachea, tongue and lips etc. Once such a basic speech production system is created the appearance of different voices such as gender and age can be easily changed by modifying several parameters of the physical model such as the head size or the pitch of the waveform generator.
Additionally unlike with concatenative speech synthesis it is much easier to include more sophisticated features to the physical model, such as the intonation or even language defects. Generally formant synthesis consumes much more processing power than other technologies, although with today's strong CPUs this is already a negligible figure. That's why systems which were formerly implemented in hardware with their own CPU, today already run well on an ordinary desktop system without any significant CPU load.

1.2. Speech Recognition

1.2.1. History and Future Challenges

This technology already has a quite long history of more than 50 years. After early primitive analog speech recognition systems including even some commercial applications in the 1950's and 1960's the first reliably working software recognition systems were built in the early 1970's [LOW80]. These systems were already able to perform word spotting for simple command and control applications.
The first successful and reliable research prototype was the *HARPY* system [LOW80]. This replaced the speech-understanding approach of the less successful artificial intelligence techniques with more sophisticated and effective signal processing methods for pattern recognition. In the 1980's the speech recognition research introduced many new methods such as *Vector Quantisation* [KHE94] and *Hidden Marcov Models* [ROD99], which resulted in today's mature commercial applications for speech recognition on the desktop. Recent research uses *Neural Networks* [RAN99] to improve the speech recognition reliability and accuracy even in noisy environments. Although speech recognition is today already highly accurate in perfect conditions (calm environment, good microphone, co-operative speaker) the future challenge for this technology is to improve its performance within less perfect environments especially for mobile applications.

1.2.2. Evaluation of Speech Recognition

All in all speech recognition is not a trivial task. In fact the technology requires the combined knowledge of several sciences such as signal processing, acoustics, pattern recognition, communication and information theory, linguistics, physiology, computer science and psychology [RAB93].

The performance of a recogniser engine is mainly affected by the *variability* of the speech signal, which can be classified in four major groups [WOO98]:

- Intra-speaker (same speaker): physical or emotional state
- Inter-speaker (different speakers): accent or dialect, age, gender
- Speaking style: reading or spontaneous, formal or casual, redundant utterances
- Acoustic channel: noise, telephone quality, background speech

There are commonly five measures to evaluate the performance of a speech recognition system [RAM97]:

- The *accuracy* is measured in the number of errors made by the system under perfect conditions. A common value is a recognition accuracy higher than 98%, which can already easily compete with a non-trained typist.
- The *reliability* of a recogniser even in non-perfect conditions such as noisy environments or with low level equipment such as a bad microphone or a telephone headset.
- Many speech recognition products perform better when they are trained for a specific user. *Speaker independent* performance therefore is a good measure for the recognition quality.
- The *vocabulary size* determines the amount of recognisable words. Depending on the application its size and compilation may vary from a few utterances for command and control systems to several thousand words with specialised vocabulary for legal applications for instance. The larger the vocabulary however the longer the search algorithm takes to find the right corresponding words.
- Speech recognition consumes a lot of processing-power and memory. Therefore its *consumption of computer resources* has to be balanced with the desired performance.

1.2.3. Types of Speech Recognition

In general there are two types of speech recognition technologies and applications today: *Command and Control Systems* and *Continuous Speech Recognition*. Historically there also existed the type of *Discrete Speech Recognition*, which is now obsolete due to the improvements in the performance of today's speech recognition engines. Early products forced the user to speak a sentence with a brief pause after each word, which required the user to adapt to the system's needs. Modern speech engines allow the speaker to speak in natural and continuous but clear speech. This type of speech recognition requires large dictionaries and a larger amount of processing power.

For various simpler applications such as basic speech commands, continuous recognition engines would be a waste of resources. Therefore any current product also comes with a command and control engine, which uses the simpler *word spotting* technology. This approach simply uses a predefined small vocabulary of just a few words. If one of these words is recognised in a common sentence, a specific programmable action is triggered.

1.2.4. General Speech Recognition Architecture [WOO98]

Current speech recognition architectures use a statistical model to find the most likely word **W** which corresponds to the input acoustics **Y** and the given language models.

Therefore a basic speech recognition algorithm would have to compute the equation:

$$\begin{aligned} \max_W p(W|\mathbf{Y}) &= \max_W \frac{p(\mathbf{Y}|W)p(W)}{p(\mathbf{Y})} \\ &= \max_W p(\mathbf{Y}|W)p(W) \end{aligned}$$

Fig. 1.3.: Word recognition probability [WOO1998].

To build a speech recogniser it is therefore necessary to find suitable representations for the *acoustic model* from which the probability p(Y|W) is calculated and a *language model* from which p(W) is calculated. Given these representations an algorithm is needed to perform the decoding of the most likely word. Speech recognition systems based on this statistical modelling are successful because the parameters of the models can be estimated from large amounts of real speech and language data, such as databases from training sessions and pronunciation dictionaries. In nearly all current speech recognition systems the acoustic modelling component is based on hidden Marcov models (HMM), which will be described

more detailed in section 2.2.5. To summarise there are three crucial components needed for building a speech recognition system:

- Acoustic models of individual speech sounds or phonemes (e.g. HMMs).
- A pronunciation dictionary.
- A language model representing the probability of certain word sequences.

1.2.5. Dynamic Time Warping (DTW)

Dynamic Time Warping is a signal processing technique which basically deals with one of the problems of speech recognition: The variable speed and length of spoken input especially in continuous speech. DTW therefore attempts to find a match by distorting the utterance. Lets assume the system should recognise the word *labrador*. Therefore we record a reference model of this word which might be represented as *llaabraadoor*. In our example the speaker pronounces the word as *lllaaabrraaador*. While a simple comparison of the two words wouldn't result in a match, DTW now tries to compare them part by part. After the first two matching *L*'s we get a mismatch *L-A*. By considering the following letter *A* of the model we realise that the third *L* of the utterance still corresponds to the second *L* of the model, so we have compressed the beginning to enforce a match. The following two *A*'s result in a match followed by a mismatch which also can be resolved considering the surrounding letters. So we follow the model to the end where the final *R* produces a match for the two *R*'s of the model. A slightly different word such as *laboratory* would produce a higher number of mismatches. To create a speech recognition system with DTW we have to set a threshold of matches and mismatches to determine when an utterance has to be considered as recognised. DTW based recognisers are relatively simple and therefore perfectly suitable for command and control applications with a limited set of predefined commands.

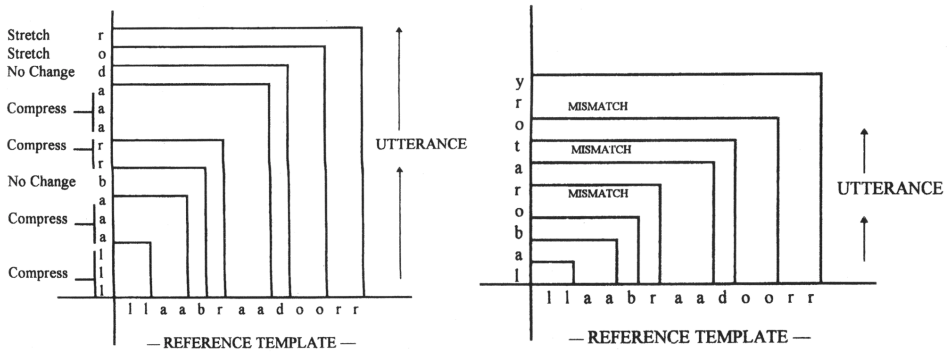

Fig. 1.4.: Principle of dynamic time warping [ROD99].

1.2.6. Hidden Markov Models (HMM)

Hidden Marcov Models are a statistical approach to determine the probability of a recognised speech pattern [ROD99]. Therefore the system needs a database of previously trained speech models. The training is a quite computationally expensive process which fortunately only has to be done once by the manufacturer of speech recognition software. To adapt and optimise these models for a certain speaker, HMM based recognisers work better when the users train the models with their specific type of pronunciation in a first enrolment session. The result is a database of acoustic parameters, say codebook entries from a vector quantisation process, with a certain probability associated to each of them. HMM are networks that can generate speech in the form of triphones (sets of three phonemes) by taking a path from the entry state to the exit state. The likelihood of a particular sequence can be calculated from the probability of the state transitions from the language model.

A key feature of HMMs is that they may also be cascaded. There can be separate models operating on different levels. Models operating on the acoustic level can determine the probability of the triphones, their output will be fed to a higher level HMM which determines the most likely word. On the highest level there also can be installed a syntactic HMM which incorporates some grammatical rules of the language.

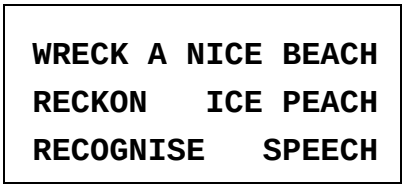

Fig. 1.5.: A word sequence showing three possible combinations of words, that fit the same model [ROD99].

1.2.7. Artificial Neural Networks (ANN)

Artificial Neural Networks are a quite recent method in computer science and are derived from the way our human brain works [RAN99]. “A Neural Network is a computing system which is made up of a number of simple, highly interconnected processing elements, which process information by its dynamical state response to external inputs” [Hecht-Nielsen R.]. These systems are particularly used for pattern recognition applications, a task in which the human brain is quite good as well. Therefore neural networks are very suitable for speech

recognition as well as for visual recognition (characters, faces) applications. They mainly play the same role as HMMs in a current speech recognition system to compare previously trained speech patterns with the spoken utterances. While HMMs try to achieve this with statistical and probability models as explained above, ANNs deal with the problem from a classification point of view: they can easily classify a given pattern into one of a hundred categories, each associated with an allophone (phoneme variation) of the language. Neural nets are basically a collection of primitive processing units (neurodes), where each of them can only perform some basic operations. When the system is trained with specific patterns they are able to build up links between each other and therefore are able to "remember" those patterns, when fed with the same or a similar pattern again later. Recent research by the biomedical engineers Berger and Liaw at the University of Southern California [BER99] shows that neural networks are far more reliable in even noisy environments. In fact this research prototype obviously outperforms the human ability of speech recognition. Also commercial manufacturers such as Lernout & Hauspie are already using neural network technology in some of their products.

1.2.8. Speaker Recognition

Basically there are two flavours of speaker recognition:

- *Speaker Identification* can identify which speaker from a registered set of users said the recognised utterance.
- *Speaker Verification* can verify if the speaker is really the person he pretends to be.

Whereas both speaker verification and identification are likely to be used for authentication in future speech enabled applications such as phone banking or other security based applications, speaker identification could be an essential feature of tomorrows multi-user speech recognition products. Imagine of not only one person can talk to the auditory application anonymously but a group of people can use the same application at the same time. Basically the identification uses the same technologies like ordinary speech recognition - the statistical modelling with hidden Marcov models. Both systems of course require the user to train the recognition algorithm in order to create a user specific database to compare the patterns. Generally an excellent voice model consumes less than 500 bytes, therefore speaker identification systems can be easily implemented in hardware as well. In fact there are already some successfully working prototypes in use by the American immigration authorities [ROD99]. Further basic information on the topic is available online at [GAU95].

1.2.9. Language Identification

Additionally there is also the possibility of language identification, though there exist no commercial applications of this technology yet. But there is ongoing research in this area in various US research laboratories such as Bell Labs, Texas Instruments and Oregon Graduate Institute [ROD99]. Imagine a telephone information system where the system can automatically identify the caller's language and route him to the appropriate operator speaking his particular language. Other applications of this technology will of course be telecommunication monitoring systems by intelligence organisations like the American National Security Agency (NSA).

1.2.10. Voice Recognition

Another future challenge in speech technology will be voice recognition. The human vocal organs can do much more than produce speech. We can sing, shout, imitate animal sounds and do other funny things with our voice. On the other hand also speech can be produced in various forms, depending on the emotional state of the user. There already exist some basic musical applications such as Digital Ear™ by Epinoisis Software Corporation [EPI00] - a pitch-to-MIDI converter that allows the control of MIDI devices with the voice pitch. Nevertheless all existing speech products still have to struggle enough with their common purpose - the recognition of speech. Nevertheless future auditory user interfaces could be far more versatile, if they would not only react to spoken input but also to a simple click of the tongue for example. Also much more meta-information could be transported if the computer were able to understand the emotional state of the user, and not only the simple words he spoke.

1.3. Natural Language Processing

Natural Language Processing (NLP) is the science of computational linguistics, which overlaps like the name already suggests with linguistics, computer science and artificial intelligence [RAD99]. Although today's computers are still far away from the ability to understand human languages, NLP tries to provide tools for the generation of understandable and grammatically correct language and the analysis of written or naturally spoken and maybe informally structured language. Applications of NLP techniques are quite versatile and range from machine translation systems to the automatic generation of technical manuals. NLP is also important for the realisation of dialogues for speech enabled auditory user interfaces.

1.3.1. Natural Language Generation

"Natural Language Generation (NLG) is the process of generating text or speech from some originally non-linguistic source of data" [MEL98]. Its applications range from the production of radio weather reports from numerical data to the automatic generation of technical documentations. It therefore still needs specially structured data sources of course, but concerning larger quantities of text it has a few advantages compared to human authoring. The basic processes of a NLG system can be structured into four stages:

- *Content Determination:* choosing what to say
- *Sentence Planning:* building abstract representations of sentences
- *Surface Realisation:* using a grammar for detailed syntax
- *Morphology:* formatting and signal generation

One of the NLG key problems is making the right choices since language often offers many possible ways a sentence or text can be organised. If wrong syntactical choices are made this can therefore easily result in different meanings of the generated text.

1.3.2. Natural Language Analysis

Natural Language Analysis (NLA) tries not to understand human language unlike the artificial intelligence approach of Natural Language Understanding (NLU) [MOE98]. It rather takes written or spoken language input and maps it into a representation which can be used and processed by an application. Computers need not really to understand language in order to be able to do useful things with the provided data.

Today's existing NLA applications range from machine translation, extracting keywords for a database to automatic summarising of larger texts. Most of these applications just involve hand-made pattern matching algorithms or statistical processing, whereas NLU tries to map the input to a logic semantic representation. Generally the processed data should provide a structure which then can be used for database queries.

2. Speech Technology Standards

Since speech recognition and synthesis are already quite mature and all existing products already offer highly accurate recognition engines it is very likely that speech enabled applications will be a standard component in tomorrows operating systems. Unfortunately all manufacturers have been using their own proprietary technology in order to bind licensees of their speech engines to their specific product. The need for a standard interface and related technologies is obvious in order to allow the users to choose their preferred speech products for their desktop applications.

Also developers need a common application-programming interface for developing their own speech applications without the need for learning different proprietary technologies. The Microsoft Speech API™ [MSSAPI] was already an early solution for these problems and today most of the existing speech products for the Windows platform are supporting this standard. The only problem with the Microsoft's solution is, as its name suggests, that it is only available for operating systems manufactured by Microsoft. Therefore Sun Microsystems created the Java Speech API [SMJSAPI] that should be platform independent due to its implementation in the Java programming language. Additionally the World Wide Web has shown us how powerful networked information systems can be. VoiceXML [VFVXML] is an approach to adopt this concept and its ease of use for speech enabled remote controlled applications.

2.1. Microsoft Speech API

2.1.1. Concept

The Microsoft Speech API (SAPI) [MSSAPI] is the current standard application programming interface for the Microsoft Windows platform. In its current version 4.0 it brings together with the API libraries some sample continuous speech recognition and command and control engines, a speech synthesiser and also an interface for telephony applications - the TAPI. The delivered engines at the moment only support American English and have to be considered as a very basic sample implementation. They are not as highly accurate and sophisticated as other existing products on the market, which all currently implement the SAPI as a quasi standard. You will learn more about these products in [sect. 3.1.].

The SAPI is currently available for the Intel x86™ and DEC Alpha Windows platforms but not yet for Windows CE™ devices. A software developer's kit is delivered separately, which provides sample code and wrapper classes for various programming languages including C, C++, Visual Basic, Java and technologies such as COM [MSCOM] and ActiveX [MSACTX].

2.1.2. Structure

As shown below the SAPI is organised in several levels. The Voice Command, Voice Dictation and Voice Text APIs provide the highest and most abstract access to the SAPI functions. Their functions are somewhat more limited and also slower, but therefore they offer automatic resource and memory sharing between processes. The Direct Speech Recognition and Direct TTS APIs give full access to the speech engines on a lower level. The lowest Audio Objects level provides direct access to the audio hardware.

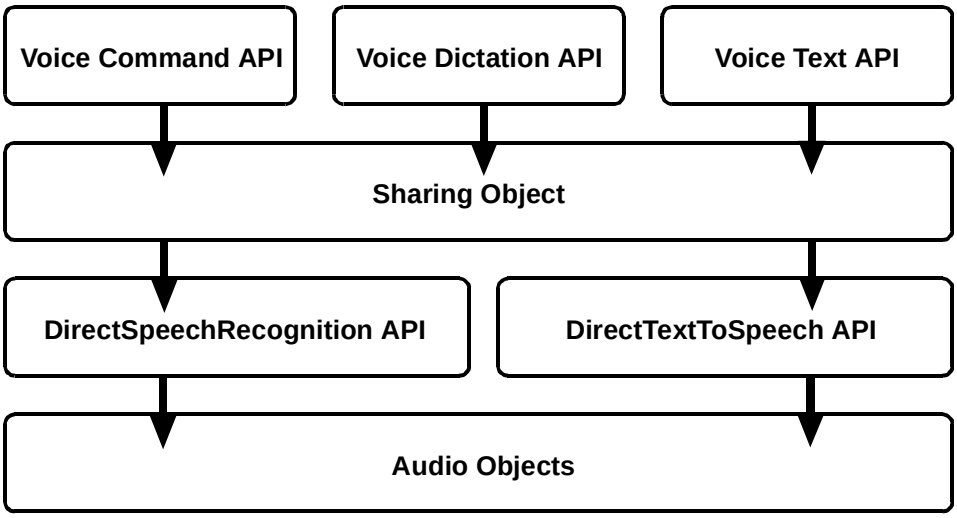

Fig. 2.1.: Structure of the Microsoft Speech API [MSSAPI].

- Voice Command

This interface provides a high level access to the command and control functions of the SAPI. Its design mimics a Windows menu in behaviour, providing a "menu" of commands users can speak to access common Windows GUI elements.

- Voice Dictation

The Dictation API has been available since SAPI version 3.0. It allows the developer to instantiate a dictation session attached to a specific text window using continuous speech recognition. As soon as this window has the keyboard focus the dictation engine accepts spoken input for this window. There are also a variety of predefined objects available, such as a correction window and some macros for text editing commands.

- Voice Text

This TTS interface basically accepts a character string input, which can be both synthesised and played over the computer soundcard or a telephone line. Therefore it defines a voice text object with several properties to define voice, gender, age, pitch or speed of the synthesised voice.

- Direct Speech Recognition and Direct Text To Speech

This component allows direct low level access to the speech recognition engine. It gives the developer much more control over the recognition process, but also involves more work.

- Audio Objects

Every implementation of the SAPI needs to provide access to the audio hardware via this interface in order to guarantee that the recognition and synthesis engines don't occupy the soundcard. This is necessary if the application simply wants to play a sound sample or record anything from the sound input.

2.1.3. Upcoming SAPI v5.0

A new release of the Microsoft Speech API 5.0 is due to be released in autumn 2000. It will be a complete rewrite from the SAPI 4.0, reducing its complexity from over 140 different API objects to one single COM interface. SAPI 5.0 features a new Context Free Grammar (CFG) namespace that can load grammars in multiple ways, including recognising XML from the Web and using new semantic tags that support nested properties and eliminate reparsing of recognised text. SAPI 5.0 also includes a new voice object for handling audio conversion formats.

2.2. Java Speech API

The *Java Speech API* (JSAPI) [SMJSAPI] was defined by *Sun Microsystems* in 1998 to provide access to state of the art speech technologies for the Java programming language. Since an early access version 0.9 allowed developers and implementers to take part in its design process, the current version 1.0 is already a quite mature package of classes which offer platform independent high level objects for speech recognition, synthesis and much more. Implementations need not to be created in pure Java, in fact all implementations currently existing [see section 3.2.2.] only had to attach their native engines to Java via the *Java Native Interface* [SMJNI]. This concept guarantees high portability of applications using the JSAPI as well as high performance. Besides these existing native implementations Sun also proposes pure Java solutions as well as hardware implementations in versatile telephones for example.

2.2.1. Structure

The JSAPI was defined as an extension to the Java standard specification, therefore its packages are organised in the javax tree. Its main packages are:

javax.speech	the core JSAPI classes
javax.speech.synthesis	the speech synthesis package
javax.speech.recognition	the speech recognition package

Due to the object-oriented nature of the Java programming language the JSAPI packages provide a set of objects to access the speech engines. Additionally to the basic synthesiser and recogniser objects the JSAPI supports internationalisation and external dictation and rule grammars for the programmer's convenience. These grammars have to be formatted according to the Java Speech Grammar Format (JSGF). There are still some limitations in the API, such as the lack of input from, or output to audio files for both the synthesiser and the recogniser engines.

- Synthesiser

This object allocates the speech synthesis engine for use within Java applications. Related objects like *Voice* allow the configuration of gender, age, speed etc. of the synthesised voice. The JSAPI also offers its own mark-up language - the Java Speech Mark-Up Language (JSML) - in order to support more natural pronunciation such as emphasising words. It is still open if future versions of the specification will implement VoiceXML.

- Recogniser

The recogniser object allocates the speech recognition engine. It can either be used for command and control purposes or direct continuous speech recognition, this depends on the type of event handler that is registered with an instance of the recogniser object.

- Rule Grammar

These objects allow for the definition of rules for speech command and control by external configuration files. In a rule-based speech recognition system, an application provides the recogniser with rules that define what the user is expected to say.

- Dictation Grammar

A dictation grammar is typically larger and more complex than rule-based grammars. Dictation grammars are typically developed by statistical training on large collections of written text. Fortunately, developers don't need to know any of this because a speech recogniser that supports a dictation grammar through the Java Speech API has a built-in dictation grammar. An application that needs to use that dictation grammar simply requests a reference to it and activates it when the user might say something matching the dictation grammar.

- Events

Speech engines, both recognisers and synthesisers, generate many types of events. Applications are not required to handle all events, however, some events are particularly important for implementing speech applications. For example, some resulting events must be processed to receive recognised text from a recogniser. A Speech Event can be either an Engine Event, which indicates a change in speech engine state, an Audio Event, which indicates an audio input or output event or an Engine Error Event.

2.2.2. Implementations

There already exists a handful of implementations of the JSAPI though none of them currently represents the full specification. A full list of all available implementations can be obtained at the JSAPI home page [SMJSAPI].

The most complete implementation at the moment is IBM's product using the ViaVoice™ Runtime Environment as its engine. Although not all classes and objects are fully implemented yet, any developer who wishes to gain some insight into this technology is highly recommended to use it. There is not only the largest amount of different languages available for this product but it also supports Linux in addition to the Windows package. Due to this fact, developing JSAPI applications using this product really makes sense since it allows for the writing of platform independent and internationalised speech applications. A demo runtime environment for both platforms in several languages as well as the JSAPI package can be obtained at IBM's speech developer web site [IBMJSAPI].

Two other products, Lernout & Hauspie's ASR1600 and TTS3000 [LHJSAPI] for the Solaris™ Unix platform and the Festival project by the University of Edinburgh [UEFEST] implement only the speech synthesis packages. An interesting and very promising approach is the recently added support for the JSAPI within the voice controlled web browser Conversa™ Web [CONVW]. This product allows developers to take full advantage of speech technologies within applets.

2.3. VoiceXML

2.3.1. Concept

VoiceXML was published in spring 2000 by the VoiceXML Forum [VXMLF] an industry organisation founded by AT&T, IBM, Lucent and Motorola, all of them major companies in the telecommunications and speech technology sector. As its name already suggests VoiceXML in its current version 1.0 [VFVXML] was designed as a subset of the XML 1.0 standard [W3CXML], published by the World Wide Web Consortium [W3C]. An early release of version 0.9 was presented to the public in August 1999 to invite developers and the industry to take part in its final design process.

It is therefore generally a mark-up language designed to be used with distributed server-client applications. The success of the World Wide Web has shown that network applications which separate computing logic and the actual display on the client's browser software is a versatile concept which will also suit voice only applications. Documents written in VoiceXML can be delivered by any existing HTTP server. User data resulting from form input can be sent back to the server and processed by CGI scripts similar to

ordinary web applications. Additionally developers of such VoiceXML applications are shielded from low-level and platform specific details.

The voice browser itself is responsible for providing the speech recognition and synthesis environment on the client side which reduces network traffic significantly because only ASCII text has to be transmitted as well as some smaller auditory icons in common audio formats. Such voice browsers can either be a specific desktop application or implemented in hardware such as a modern telephone.

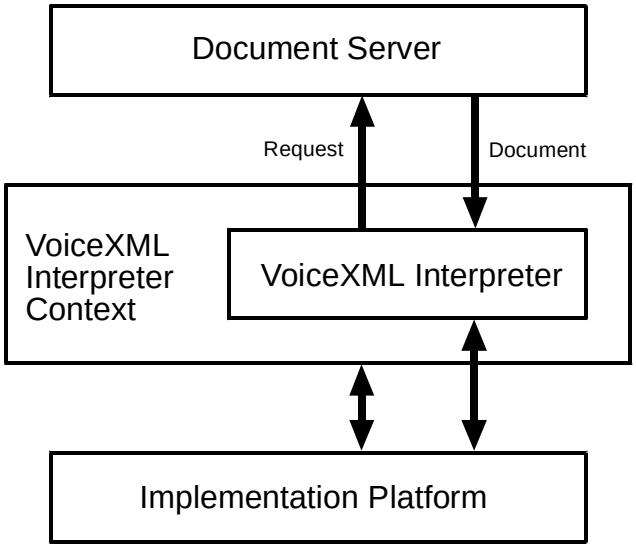

Fig. 2.2.: Structure of a typical Voice Browser environment [VFVXML].

2.2.2. Structure

Generally the language and browser specification provides the following key features:

- Output of synthesised speech and audio files
- recognition of spoken and DTMF input
- recording of spoken input
- telephony features such as call transfer and disconnect

An implementation platform therefore has to provide the features listed above as well as the ability of the acquisition of XML documents via the network including their parsing and proper interpretation.

The VoiceXML structure can be described by six general concepts:

- Dialogs and Subdialogs

There are two kinds of dialogs: forms and menus. Forms define an interaction that collects data input by the user, menus present a choice of options to the user.
Subdialogs can be compared with function calls in other programming languages and can be called up for several similar dialogs.

- Sessions

A session begins when the user starts to interact with a VoiceXML interpreter context, it continues as further documents are loaded and processed, and ends when requested by the user, the document or the context.

- Applications

An application is a set of documents sharing the same application root document. Any interaction with a document of a specific application also requires its root document. All variables of the root document are available within the whole application, they are called application variables.

- Grammars

A grammar can be described as specific methods attached to each dialog, which are only active when the user is currently in that specific dialog, which are *machine directed applications*. In *mixed initiative applications,* grammars can be marked as active even when the user is in a different dialog. In this case when the user says something matching another dialog's active grammars, executions jumps to that other dialog.

- Events

Like in other programming languages VoiceXML defines a set of events which are triggered by the user or the browser context.

- Links

If a user matches the grammar of a link in a dialog the control is transferred to the link's destination URI. This generally is the request of a new VoiceXML document or the submission of form data to a server script.

2.3.3. Implementations

Again IBM was one of the first major companies to present an early implementation of the VoiceXML specification. Already with the pre-release version 0.9 they presented an implementation based on Java using the Java Speech API. The package currently supporting a major subset of the actual version 1.0 can be downloaded at IBM's Alphaworks [IBMVXML]. It comes with some very helpful demo applications including their Java and VoiceXML source code. Additionally IBM supplies a VoiceXML-Server SDK, which is a collection of various tools such as a ViaVoice runtime, JSAPI and some demo applications.

2.3.4. W3C Voice Browser Concept

The World Wide Web Consortium currently also considers the standardisation of a concept similar to VoiceXML. This activity also targets applications in the telephony and mobile sector. They are currently examining various products including VoiceXML for the standardisation process. The latest activity was the release of a requirement draft in autumn 1999. This draft defines the common architecture of a future voice browser model with technologies such as voice dialogs, speech grammars, natural language representation, speech synthesis and recognition. It will be interesting to see where this standardisation process will lead.

2.4. Other APIs

There have been two other attempts to create a speech API standard, but both of them seem to have been abandoned by their creators. The *Advanced Speech API* (ASAPI) [ATTASAPI] by AT&T was designed to provide some extensions to the Microsoft Speech API. The other project was the *Speech Recognition API* (SRAPI) [SRAPIC] by the SRAPI Committee, a non-profit corporation with the goal of providing solutions for interaction of speech technology with applications. Its core members were similar to those from the VoiceXML Consortium and it might be that they abandoned this technology in favour of the VoiceXML concept.

Motorola, also a member of the VoiceXML Forum, has created its own speech mark-up language VoxML 1.2 [MOTVOX], which is an approach very similar to VoiceXML. Upon request Motorola said that it will maintain VoxML as well as support VoiceXML, because they already have a large installed VoxML basis.

Another approach is the VocAPI by the Philips Speech Processing group [PHVA]. This technology will be soon standardised as the German DIN Standard 33880. Its main targets are lean command and control applications for embedded solutions in cars or other industry environments.

Additionally there also exists a variety of telephony APIs for dedicated telephony applications. These interfaces not only define dialog structures but also provide abstracted access to the telephony hardware and telephony call control. Together with the existing speech interfaces they form the basis of modern automatic telephony applications as already used in many call centres or other phone based information systems. Examples are the Microsoft Telephony API included in the SAPI and the Java Telephony API (JTAPI). The JTAPI in its current version 1.3 released in July 1999 is a set of modular, abstract interface specifications for telephony integrated call control. This specification was developed by a consortium of companies, including Dialogic, IBM, Intel, Lucent, Nortel, Novell, Siemens, and Sun Microsystems.

3. Product and Market Overview

3.1. Consumer Software

As shown in chapter one, speech recognition and speech synthesis have already a long history in research but it took quite a while until this technology was mature enough for the mass market. There had been some initial first steps in hardware and software speech synthesis for home computers such as the *Texas Instruments TI99* [TI99SP] or the popular *BBC Microcomputer* [BBCMC] in the early 1980's, though these synthesisers could only be considered more as toys. Nevertheless they already allowed blind users to use the full benefits of home computing at this time. Due to the increasing processing power on the desktop and the general advance in research during the 1990's the market in speech technology began to grow very fast. Today there are three main players in this market for consumer speech recognition products, such as continuous dictation software and voice controlled application tools. On the one hand these three market leaders whose products will be reviewed in the following section can look back at a long history of experience with their quite mature and already highly accurate products. On the other hand there are also some smaller start-up companies with a new range products such as the voice controlled Internet browser *Conversa™ Web* [CONVW]. All in all there is still major competition in the market, which manifests in the recent take-over of *Dragon Systems* by the Belgian manufacturer Lernout *&* *Hauspie* [LHDRAG]. Since there are ongoing research advances in this area such as new powerful neural network recognition engines, new players can be expected to appear soon on the stage.

3.1.1. Philips Free Speech

The Philips Speech Processing Group [PHSPG] is the speech technology division of the Philips group for general speech technology products such us telephony applications and desktop dictation products. Philips' current major product for the consumer market is *Philips Free Speech 2000* [PHFS], which is available in 15 languages including some local dialects and support for even more languages is in development. It is currently only available for the x86 Windows platforms and requires at least a Pentium class machine with 200MHz and 64MB of memory. The product's main features are similar to the competitive products and include command and control for all major applications and dictation for word processors as well as a couple of predefined macros. In addition to its

Free Speech 2000 dictation software Philips also offers the Free Speech Browser which is a special command and control application for the Microsoft Internet Explorer. Its standard vocabulary for continuous recognition, developed in conjunction with Oxford University Press, includes 290.000 words and can easily be extended by 37.000 own words and phrases. For the command and control feature Free Speech offers a database of 11.000 voice commands. In addition Philips has licensed IBM Text to Speech engine to be included in its Free Speech product range.

Free Speech has already two predefined recognition patterns for male and female speakers and should be trained after installation by reading a couple of predefined personal and business letters. The whole installation and training process takes about an hour and is necessary in order to achieve best results with a particular speaker. It also includes an online learning feature, which allows the application to adapt continuously to the speaker's pronunciation. Its multi-user features support further users who still need to train the software to achieve the best results.

There are two types of microphones delivered with Free Speech 2000, one of them is the high quality *Plantronic SR1* headset, which is considered to be better than most of the other microphones sold in the box with a speech recognition product. The other is the *Philips Speechmike*, a rather interesting device that combines a microphone, trackball and mouse buttons in order to achieve higher convenience during the dictation process.

3.1.2. IBM ViaVoice

IBM is the most innovative key player in today's speech application market. It not only serves the consumer market with its ViaVoice product range [IBMVV], but is also one of the major supporters of the upcoming standards besides the Microsoft Speech API such as the Java Speech API or VoiceXML. It is the only company that is offering early implementations of these technologies on its Alpha Works web-site [IBMAW] allowing independent developers not only to test these new technologies but to also let them take part in the standardisation process. ViaVoice therefore plays an important role in this process not only because it provides an engine for all these new technologies but also due to its support for various major operating system platforms such as Windows, MacOS and Linux. The speech synthesis package ViaVoice Outloud, a software formant synthesiser, is

impressive with its highly natural and intelligible voices. In May 2000 IBM also presented an on-line Java version of this product, which has a high performance even in an ordinary web-browser context.[IBMTTS] The latest addition to the ViaVoice product range is the speaker identification package ViaVoice Identified. The current dictation product range ViaVoice Millennium is available for the Windows, Macintosh and Linux operating systems and supports eight languages (English, German, Spanish, Portuguese, Italian, French, Japanese and Chinese) with some local variations.

3.1.3. Lernout & Hauspie Voice Xpress

Lernout & Hauspie acquired Dragon Systems in the end of March 2000, which reduced the number of the major competitors in the consumer market from four to three. It is not clear yet if the *Naturally Speaking* speech product range of Dragon systems will be maintained or not. Due to this new situation and since most of the products described above have similar features the former Dragon Systems packages will not be described here in detail, although they offered the first commercially available speech recognition product available in the 1980's.

L&H Voice Xpress [LHVX] in its current version 4.0 is available in different categories starting from simple command and control applications up to specialised dictation software with medical and legal special vocabularies. The standard vocabularies include 230.000 words that can be extended by 20.000 new words and phrases as you talk. Its minimum System requirements for the x86 Windows platforms are a Pentium 233 processor with at least 48MB of memory.

RealSpeak, their Text to Speech product offers outstanding and very natural speech language quality. This is achieved with a combination of concatenative and diphone synthesis. Additionally sophisticated linguistic processing and prosody models contribute to the natural appearance of the spoken output. The product is available for several mainstream languages and is supported under Windows NT.

3.1.4. Others

Apple Computer Inc. already supplied a speech synthesis and a basic command and control engines with its MacOS since version 7.5 [APPPT]. Though *PlainTalk* was a quite early step to integrate speech into the operating system this technology never had a real

breakthrough. Unfortunately this product still only supports English and Spanish text-to speech, and the command and control engine is far to immature for serious applications. Due to the lack of a working continuous dictation software and the missing support for further languages, real speech technology has not been an issue for Macintosh users until recently. Today IBM's ViaVoice and Dragon Systems Naturally Speaking are available for the MacOS.

3.2. Development Packages

3.2.1. Commercial Runtime Engines and Development Environments

All of the major manufacturers listed in section 4.1 also license their speech engines to third party developers. These engines are already used in some interesting applications such as the speech to speech translation software *Talk & Translate* by Linguatec [LINGTT]. IBM offers its speech synthesis and recognition engines for Windows and Linux developers, trial versions of the runtime and a software development kit (SDK) can be downloaded for free [IBMVVDK], the bundling of the product with one's own software is subject to the licensing conditions of course. As mentioned above the ViaVoice runtimes support their own proprietary API as well as the Windows SAPI and the Java JSAPI. The command and control engine or the speech synthesis can be downloaded separately if only these features are required. Lernout & Hauspie also licence their speech products to developers but also charge developers for their SDK. A demo version of the VoiceXpress SDK, which also supports the Microsoft Speech API, can be downloaded from [LHVESDK]. Lernout & Hauspie also offer a range of phonetic speech recognition engines for embedded processors such as the Hitachi SH-3 [LHEMB] which allows basic command and control speech applications on portable and embedded devices.

Also the Philips Speech Processing Division offers a software development Kit for its FreeSpeech 2000 product line as well as a the VocOn SDK for the upcoming VocAPI DIN standard [PHVASDK].

There also exist some basic speech technology extensions for the Macromedia Director authoring tool as well, like the text-to-speech *Xpress Xtra* by DirectXtras Inc. [DXXX] which allow multimedia developers to add speech technology to their applications.

3.2.2. Research and Open Source Projects

There exists a variety of freely available speech synthesis packages for many different operating systems. Some of these freeware applications are not very mature or on the top of today's technology, but there are several highly sophisticated packages available based on the latest research results. One of them is *Mbrola* created by the Belgian Multitel TCTS Lab [MBROL]. Its diphone based speech synthesiser produces high quality output and there is a vast amount of different languages available including some European minority languages. The package is also freely available for non-commercial use for almost any existing operating systems. The other is the *Festival* project [UEFEST] by the Centre for Speech Technology Research at the University of Edinburgh. This product is available with its source code and therefore allows developers and researches a deep insight into today's state of the art speech synthesis technology.

On the other side with *Sphinx2* [SPHX2] there also exists a very interesting speech recognition engine which is available including its source code. The speech group at the Carnegie Mellon University maintains this product. Because of its long history in speech recognition research *Sphinx2* provides a rich basis for developers of speech technology. An interesting feature for example is the possibility of building your own language models for specialised applications.

There are a variety of other open source speech recognition projects available, which are quite useful for developers who want to see how some particular speech recognition methods work. Hidden Marcov models are implemented within *Myer's Hidden Marcov Software* [MYHMMS] whereas the basics of neural networks can be learned from the *Nico Artificial Neural Network Toolkit* [NICANN].

3.3. Accessibility Products

3.3.1. Software for the Visually Impaired

Today there already exist a few dozens manufacturers of accessibility software for the visually impaired computer user [GILSR]. Most of the software products available are screen readers for text based console applications or windows based GUI applications.

Generally the purpose of these packages is to auralise the graphical desktop, which most of them do in a very useful and sophisticated manner. The access is limited though to applications which use the standard user interface widgets. Any self-made graphical user interfaces like those of the rendering software *Bryce* [MCBRY] are not speakable by those products. Therefore Microsoft has its own accessibility policy to ensure accessibility to their operating systems and applications. [MSMA]. In spite of all these efforts these applications only can be considered as crutches for the graphical desktop.
For this survey the advanced screen reader *JAWS for Windows* by Henter–Joice Inc. [HJJFW] was tested. The desktop is controlled with the keyboard to switch to various icons on the desktop or widgets in a window. As soon as a widget is selected its type and name is read by a speech synthesiser, which can be either a hardware solution such as DECTalk or a software product such as ViaVoice Outloud. Interface widgets such as scrollbars for example are represented by their name and their corresponding value. As soon as any value is changed by certain keyboard commands the new value is spoken. Jaws supports scripts and macros for various major desktop applications such as Microsoft Word, Outlook or Internet Explorer, which make their handling much easier and faster.

After a period of training with screen reading applications most of the everyday desktop tasks can be easily and effectively done with this aid. Experienced users can already adjust the rate of words per minute of their synthesisers to a very high level in order to speed up the interaction process. Some other products also support Braille interfaces (tactile reading systems for the blind) for textual output, but will not be discussed here in further detail.

3.3.2. Blinux

Blinux [ZOEBL] is a distribution of the Linux operating system [LINUX] specially adapted to the needs of sight disabled users. This project is particularly interesting because of its revolutionary approach to a new operating system including speech technology within its core design. This process has been made easier by the GNU open source model [FSFGPL] that allows developers full insight and control of virtually all of the features of the operating system.
Generally the Blinux distribution consists of a standard Red Hat Linux [LINRH] with some additional packages for speech synthesis and recognition, drivers for specific hardware such as speech synthesises and Braille terminals and a variety of other speech enabled

applications. An important main feature is a kernel module, which allows the automatic speech synthesis of any console output. Advanced AUI desktop applications such as the Emacspeak [RAM00] desktop allow visually impaired users full access to all major everyday tasks, such as word-processing, e-mail and so on. Due to is standards-setting architecture for auditory user interfaces Emacspeak will be discussed in detail in [sect. 4.5.].

There are already some other projects around which are similar to the Blinux approach. ZipSpeak [LINZS], a talking mini-distribution of Linux for example fits onto a small portable media such as a zip cartridge and allows its users to boot into their own operating system environment at any workstation. Other initiatives like the Blind Penguin project aim to create a standard helper application for the sight disabled to allow access to the standard Unix GUI interface X-Windows.

Due to the open source nature of Linux and the enthusiasm of many programmers Linux already has exceeded the accessibility features of other operating systems according to the Blind Penguin project [BLPP]. It is therefore very likely that these products will be the first choice for sight disabled users in the future.

3.4. Speech Hardware

3.4.1. Embedded Speech Recognition

A handful of manufacturers offer some basic hardware based speech recognition products for embedded and mobile devices. These are basically designated chips for simple command and control applications. Others offer ready made embedded solutions with special software for digital signal processing (DSP) chips. Due to the limited memory size of such applications dictionary sizes only vary between a few hundred or a few thousand words. Nevertheless these products are ideal for applications in consumer devices such as toys or entertainment electronics. Some products like the *MSM6679A-110* processor by OKI Semiconductors [OKIMSM] offer basic speaker independent command and control for around 25 words with an accuracy of more than 95% in perfect conditions. Additionally this chip also supports speech synthesis, voice recording and voice playback. Generally, integrated circuits like this or the similar RSC-364 recognition only chip by Sensory Inc. [SENSRSC] are sold for between three and ten US Dollars. Complete ready made modules for individual embedded speech controlled hardware kits [IMGSRK] are also available from around 50$.

The more advanced DSP based software solutions allow larger vocabularies and higher recognition reliability and performance even in noisier environments such as in cars or industrial applications. Market leader Lernout & Hauspie also has ASR1600/C for the Hitachi SH-3 RISC [LHEMB] processor in its catalogue. Fluent Speech Technologies Inc. offers ready made embedded speech recognition solutions for a variety of microprocessors and DSPs. [FSTDSP]. Another interesting noise-robust command and control system comes from Motorola. Clamor is running on various DSPs such as the Motorola DSP 56L811 and achieves a 96% accuracy even in noisy car environments [MOTCLM].

Due to the growing market in this sector there are several other embedded speech technology manufacturers which can't be listed here. Therefore this list is far from being complete and should be understood as a general overview about what different types of embedded speech products currently are available.

3.4.2. Hardware Speech Synthesiser

There exists a variety of hardware based speech synthesis solutions, which are either available as an internal ISA or PCI card or as external boxes connected to the serial or parallel port [GILSPS].Years ago these hardware solutions were mainly chosen because they did not misuse the limited processing power of older personal computers. Today the external boxes are still convenient because of their portability and versatile use. There are also PCMCIA cards for notebooks and PDAs available. These products are generally using formant speech synthesis, like the popular DECTalk synthesiser which was developed by DEC and now is manufactured and distributed by Force Computers [FCDEC].

3.4.3. Mobile Devices

In the last few months there have been some exciting movements in this growing market segment. There have already been a number of mobile phones before, which provided simple voice control features such as voice dialling. There are also some other products like Dragon Naturally Speaking Mobile, which allow dictation on the move with later speech to text conversion on the desktop.

Now there are already some real mobile speech recognition projects in sight though none of them are available as a final product yet. Lernout & Hauspie for example presented in spring 2000 a prototype of a Linux based handheld code-named *Nak* [LHNAK] offering

full continuous speech recognition. IBM also announced a promising hardware add-on for the popular 3COM Palm Pilot PDAs [PALM] based on its ViaVoice technology [IBMPDA]. The product will simply plug into the PDA's serial expansion port, enabling PalmOS application with full continuous speech recognition and synthesis.

Microsoft also announced its speech enabled version of a WindowsCE based PDA, though no further details were published. Probably their press release has to be considered as the usual "vapour ware" in order to catch up with the recent developments in the market [MSMIPAD]. Advanced Recognition Technologies Inc. [ARTCOM], a manufacturer of handwriting and speech recognition products already offers command and control speech recognition software for handheld PCs such as ARTcommander for WindowsCE™ devices and SmartSpeak for WindowsCE™ and EPOC32 based pocket organisers. These products already allow speech enabled applications on today's existing mobile devices.

Last but not least Compaq's research prototype Itsy [ITSY] should also be mentioned in this context. This miniature StrongARM™ Linux handheld computer, which is not available on the market yet, presents a good picture how tomorrow's mobile applications could look like. It also incorporates the HMM based speech recognition engine by Talksoft Corporation [TALKS]. This software product supports various DSPs as well as mobile CPUs such as StrongARM™, MIPS and Hitachi SH3 which are widely used in current PDAs. The product is specially designed for the hands free operation on mobile devices using an open microphone. and performs real-time speaker independent speech recognition. The Itsy is specially interesting because of its open hardware and software design specifications, therefore it could be used for an initial mobile implementation of the Y-Windows concept as in [sect. 5].

4. Principles of Auditory User Interface Design

4.1. Human - Computer Interface

According to the Oxford English Dictionary (OED) an interface is "A place or region, or a piece of equipment, where interaction occurs between two systems, organisations or persons." The human-computer interface (HCI) provides the basis for the real-time interaction between the human user and the computer. It not only includes input and output devices such as monitor, keyboard, mouse, speaker and microphone, which represent the technical infrastructure of the interface, but it is mainly formed by software that defines the structure and protocol of the human-computer dialogue. Human computer interface design involves a variety of disciplines such as information theory, cognitive psychology, information design, sociology, engineering and ergonomics and is therefore not a trivial task. In the following section we will discuss the structure, software design and metaphors of a general HCI.

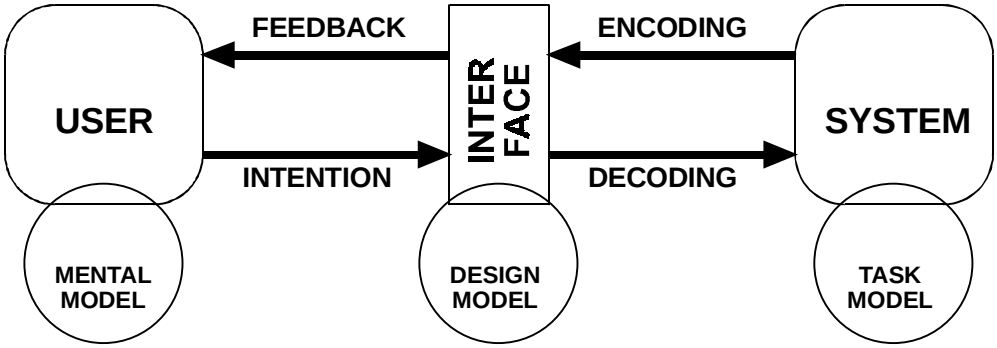

Fig. 4.1.: General communication model of human-computer interaction.

4.1.1. General Interface Dialogues

- Question-Answer Dialogues

Question and answer dialogues are the most basic form of human-computer interaction. Here the computer leads the session, guiding even the inexperienced user through the interface. This doesn't require any cognitive skills from the user, he just has to do what he is told. A good example is the interface of most cash-machines where the client has to follow the machine dialogue step by step. For complex systems though or for experienced users these interfaces are much too narrow and do not offer a sufficient amount of flexibility because of the soon predictive linear structure.

- Menu Dialogues

A range of selections is presented to the user from which the required choices can be made. Menus are usually cascaded in a hierarchical tree, where the user refines his choices until he reaches the desired information level. To maintain an overview the number of choices presented by a menu should not be to high and the hierarchy should not be too deep. With a menu tree with eight choices and three hierarchical levels there are already 512 different options available. The tree of course needs not to be purely hierarchical where particular nodes can also be reached by more than one route. A menu dialogue is also suitable for novice users though experienced users, who already know the presented hierarchy, may find it annoying to navigate through the hierarchy again and again.

- Form-filling Dialogues

The key benefit of form-filling is that all information is simultaneously available, giving the user full control over the selection of his priorities. Forms also allow the repetitive input of larger amounts of data. Most users are quite familiar with the concept of form-filling because it is a direct adaptation of the paper form. Additionally a form interface can demand the input of predefined data types for the various fields.

- Command Interfaces

Unlike the previous dialog types command language dialogues are user-initiated. The user has to feed the system with syntactically correct words without any system support. The number of available commands and their flexibility depends on the grammar of the command language. The user is expected to know the commands and their syntax. Any accepted command or command sequence triggers the associated tasks and gives instant feedback to the user. While this dialog type provides the maximum flexibility for experienced users it is unsuitable for the novice user because it involves long training times.

- Direct Manipulation

The system presents symbolic representations of a large number of available tasks to the user. The desired command or operation is performed by directly manipulating these symbols. Direct manipulation interfaces make intense use of metaphors in order to support the cognitive process. They are primarily designed to create a mental model of *objects* with their associated *actions*.

4.1.2. Graphical User Interfaces

Graphical User Interface (GUI) dominate all of the mainstream operating systems, such as MacOS™ or the Microsoft Windows™ family. A GUI is also often referred as WIMP interface [RAM97], which is an abbreviation for its standard design elements: Windows, Icons, Menus and Pointers. Other sources define it as Windows, Icons, Mouse and Pull-down menus which means generally the same. These elements are part of the virtual desktop metaphor that was first developed at the Xerox Palo Alto Research Centre [PARC] in the late 1970's and appeared in their Alto and Star computers. The breakthrough of the GUI desktop in today's form came with the introduction of the Apple Macintosh Finder in 1984. Graphical interfaces are the classic direct manipulation interfaces, supporting the user with a large set of metaphors.

- Desktop

The desktop is the central GUI metaphor representing the user's workspace on the screen. Similar to the objects and tasks on a real desktop the user can manipulate various objects on the screen which are associated with predefined tasks. A desktop is often realised within a central shell application that allows to control the basic tasks of the operating system.

- Windows

Windows are rectangular areas on the screen which contain logically separated information items. They extend the limited space of the two-dimensional screen to multiple layers which can be cascaded on the desktop. These windows can be freely moved and resized on the screen. When displaying information that is larger than the available space the whole content, which is invisibly extending over the window edges, is accessible by scrolling the window body.

- Icons

An Icon is a small picture (usually 32x32 pixels) which represents objects on the desktop such as folders, files or applications. Usually a graphical representation of the object itself (e.g. a folder) or any suitable metaphor is chosen in order to support the easier association with the object or the corresponding action. By clicking on the icon the object is either selected to perform further tasks with it or the attached action is started.

- Menus

Menus present a hierarchical list of choices which are associated with a specific command or action. This gives an instant overview of all available choices and also allows the free selection of any desired option.

- Pointers

The mouse or similar pointing gadgets such as trackballs and pens are the central interaction devices for today's GUIs. In fact it is often already impossible or inconvenient to perform any task without such a pointing device. With the mouse movements the user controls a pointer on the screen, often represented by a tiny arrow. The user can express his attention by moving the mouse-pointer into a specific region of the screen and then express his exact intention by clicking the button on the mouse.

- Widgets

These are hybrid dialogues which combine the abstract elements as described above. There are basic widgets such as push buttons and radio buttons, but also more complex structures such as form input dialogues or file selection boxes. Most GUI operating systems come with a predefined set of widgets which maintains the same interaction procedures within all applications.

Using direct manipulation, the user can interact with the objects presented by the interface. Incorporating the see-and-point principle he can point onto the objects he sees on the screen. This is a good solution for the blank-screen problem of command-line interfaces, which are hardly accessible for the inexperienced user. Nevertheless the possible actions are limited to what the interface is offering such as opening folders, moving files by drag-and-drop or starting applications. While GUIs offer intuitive interaction and do not require the learning of complex commands, their major weakness is that repetitive tasks can hardly be combined into one command. This relies mostly on the very primitive communication bandwidth of pointing, which does not allow the construction of more complex command grammars. Another important feature of GUIs is the WYSIWYG (What you see is what you get) principle, giving an instant visual feedback of the document manipulation results. On the other hand this visual feedback only informs the user how documents will look like, but doesn't provide any cues about their logical structure.

4.1.3. Incorporating further senses

There are a variety of other user interface types which try to incorporate further human senses in order to extend the bandwidth of the human computer interaction. One example which is widely used in virtual reality systems are haptic feedback devices. In tele-medicine applications a surgeon remote controls a robot which performs the actual operation. A task which would be virtually impossible without any tactile information. Similar devices are already available in the gaming industry, such as the vast variety of force-feedback joysticks. Other solutions such as Braille terminals – a tactile reading system for the blind – also have already a long tradition in haptic user interface design.

Recent research successes which appear rather futuristic and somehow scaring resulted in interfaces which directly connect to the human brain, bypassing the human sensory system. These interfaces are particularly important for the disabled. Completely paralysed people for example can already make use of the *Biomouse* [SCA96], a device which allows the direct control of the cursor on the screen by analysing the brainwaves. The user just has to concentrate on a particular direction to control the device. To transport information in the other direction there also already exist some devices which can directly stimulate the brain cortex [DOB00]. These systems allow for example blind people to actually *see* the abstract picture of an electronic camera connected to the cortex via brain implants.

Another interesting research in cognitive psychology showed that the brain substitutes the lack of input from other sensory channels. An experimental study on multisensory displays at the M.I.N.D. Lab, Michigan State University [BIO99] proved that users of virtual environments actually could hear non-existing sounds which only were suggested by visual cues. It is unclear if those cognitive effects, already widely known from optical illusions, also work in the other direction where acoustic cues trigger visual perception. If so, this would be another interesting challenge for the design of auditory user interfaces, especially for data auralisation applications.

Noncommand user interfaces [NIE93] adapt the dialogue to the user's needs by observing his behaviour. This includes for example eye-tracking which can determine where the user currently is looking at. Additionally such an interface could incorporate agents, autonomous processes that act on behalf of the user in some specified role, by automatically performing everyday tasks without any user interaction for example.

4.2. Structure of an Auditory User Interface

There are two major parts of an auditory user interface: *speech* and *non-speech* audio. A broader definition as in [KRA94] groups the auditory information channel on a scale from *analogic* to *symbolic* audio. Analogic audio in its simplest form is for example the ticking of a geiger counter, which directly maps physical data to a corresponding audio signal. On the other end of the scale is speech which is highly abstract. Between these two extremes there are various forms of auditory information structures such as the rather symbolic auditory icons and music, and the more analogic data auralisation. Most of these structures will be discussed in further detail in this section.

4.2.1. Dimensions of Sound

A central question for the design of auditory interfaces is, how many different dimensions and variables are available in an auditory universe? What are the major differences in the way we visually perceive our surroundings and is the audio channel really inferior compared to the visual system? When we look we only can see those objects which are in our visual field, while we can hear everything which surrounds us, we even can hear things we can't see. Hearing is a passive process, we can easily listen to the radio while being occupied with other tasks. But when we read a book, watch television or sit in front of the computer monitor, all our visual attention is required for this task. On the other hand environmental noise might be distracting, we even can't close our ears like we can do with our eyes. Auditory perception is linear in time, therefore sound objects exist in time, but over space. Whereas visual perception is linear in space, so visual objects exist in space, but over time [GAV89]. That means we cannot hear everything at the same time, and we cannot see everything by looking at the same place.

Sound can be categorised in two different ways, from a physical acoustic point of view and from a psychological psycho-acoustic viewpoint [GAV97]. While physics defines the parameters of sound as *base frequency*, *amplitude* and *spectrum*, in psycho-acoustics there are *pitch*, *loudness* and *timbre*. These are not only two different viewpoints in the definition of sound, but there are some significant differences to consider. There is no linear connection between frequency and pitch for example, it needs a doubling of the frequency for raising the pitch by an octave. Loudness is also changing exponential against changes in the amplitude as therefore measured in decibel (dB). Finally there is the spectrum which influences the general appearance and characteristics of a sound. Generally a sound can be split into its basic waveforms, which results in its spectrum of different frequencies. There

are three different characteristics of sounds: *harmonic*, *inharmonic* and *noise*. Harmonic sounds appear quite pure because their spectrum consists only of integer multiples of their base frequency. Most natural sounds even most music instruments are inharmonic, but especially this slightly inharmonic structure gives them their specific characteristics. Noise in its physical meaning is a random spectrum of frequencies, but in psychoacoustics noise are already inharmonic structures which are annoying for the listener.

In addition to these basic concepts their are various additional characteristics of sound. *Masking* for example describes the ability of sound to make others inaudible. Louder sounds clearly mask sounds of lower volume, you can't hear a whispering person in a noisy environment. The same happens for lower frequencies masking higher ones and sounds which follow them. This phenomenon is used for the psycho-acoustic compression algorithms in the MPEG Audio Layer 3 standard developed by the German Fraunhofer Institute [MPEG3]. Masking is less probable the more complex and different the used sounds are. Another effect is the *streaming* which describes the tendency of the perception to group various different sound into one virtual sound source. These effects both should be considered and also can be consciously exploited for the design of auditory interfaces.

4.2.2. Speech Interaction Theory

The central structure in speech-enabled interfaces is the spoken human-computer dialogue. Though it seems that this might be the most natural form of communication, the problem lies in the fact that the computer is only in a limited way an intelligent communication partner. Nevertheless the structure of these dialogs should be as close as possible to the natural human speech communication. Today there already exists a variety of speech interfaces, such as in automatic telephone inquiry systems. Most of these systems are designed using menu interfaces or simple question-answer dialogues by guiding the user through a list of questions, choices and decisions. Many other speech applications incorporate simple command interfaces which only allow the triggering of various tasks by a predefined set of spoken commands. The real challenge lies in the formation of *natural language interfaces* which allow the user to speak the way he wants, and not how the system expects him to speak. This mainly involves techniques of natural language processing as described in [sect. 1.3.], which avoid unnecessary learning processes and annoying, redundant and artificial computer dialogs for the user.

- Linguistic Structure

How are the classic concepts of interface dialogues, such as question-answer, form-filling and menu dialogues constructed with a speech enabled system? Generally all these dialogue types are summarised in complex question-answer dialogs. Both, menus and forms, guide the user through a series of adequately formed questions. Linguistics defines several basic semantic question forms which each demand a corresponding answer type [LAU95]. A basic *verification* question for example can only be answered with *yes* or *no*. A *disjunctive* question inquires, which of a set of alternatives is the case. A *concept completion* question is formed with question words such as *What? Who?* or *When?* and excepts the further specification of an entity. *Quantification* questions demand for some numerical input. Further types such as *causal* questions (*Why?*) are difficult to implement and therefore to avoid within human-computer dialogues because of their complex answer structure.

Speech interaction theory defines the concept of *focused* and *unfocused* questions [BER98]. While "What is your name" focuses on a specific expected answer, the unfocused question "How can I help you?" leaves a variety of possible answers open. While conducting a question-answer dialogue the user needs constant feedback if his answers were understood correctly. The design of *implicit feedback,* which immediately continues the conversation, guarantees a more fluent and natural dialogue. While the statement "You want to fly to London?" just provides *explicit feedback* for the last user input, the implicit question "When do you want to fly to London?" does the same while already performing the next step. It is also not very natural to state an immediate feedback after each input, but *summarising* after the successful collection of related data also provides the necessary feedback. With *tapering* [SMJSAPI], repetitive structures are presented in a shorter form after each occurrence to avoid lengthy and redundant dialogues. The initial sentence "Start recording after the tone. Pause for several seconds when done!" becomes "Record after the tone, then pause!" when displayed a second time and simply is spoken as "Record then pause" any further time.

- Initiative

The initiative defines which communication partner currently has the control over the dialogue. An interactive speech system is called *system-directed*, when the system has the control through the whole interaction, while in a *user-directed* system the user controls. In a mixed initiative dialogue both the user and the system can influence the direction of the dialogue.

- Attentional State

The *attentional state* defines the elements that concern the interaction at a certain point of time [BER98]. Speech dialogs are a dynamic process, therefore a *focus set* is defined which include the currently important or probable objects and sub-tasks which could be needed for the next interaction step. The *focus* is the topic which is most likely to be addressed by the next user input. This includes for example any possible answers and the corresponding functions related to the last system utterance.

- Intentional Structure

The intentional structure defines the desired tasks, communication types and interaction levels of the speech system [BER98]. *Well-structured* tasks have a simple stereotypical structure , while *ill-structured* tasks are rather complex with a large number of sub-dialogs. *Domain communication* refers to the central topic what the dialogue is all about, while *meta-communication* provides additional information, such as help or clarification feedback.

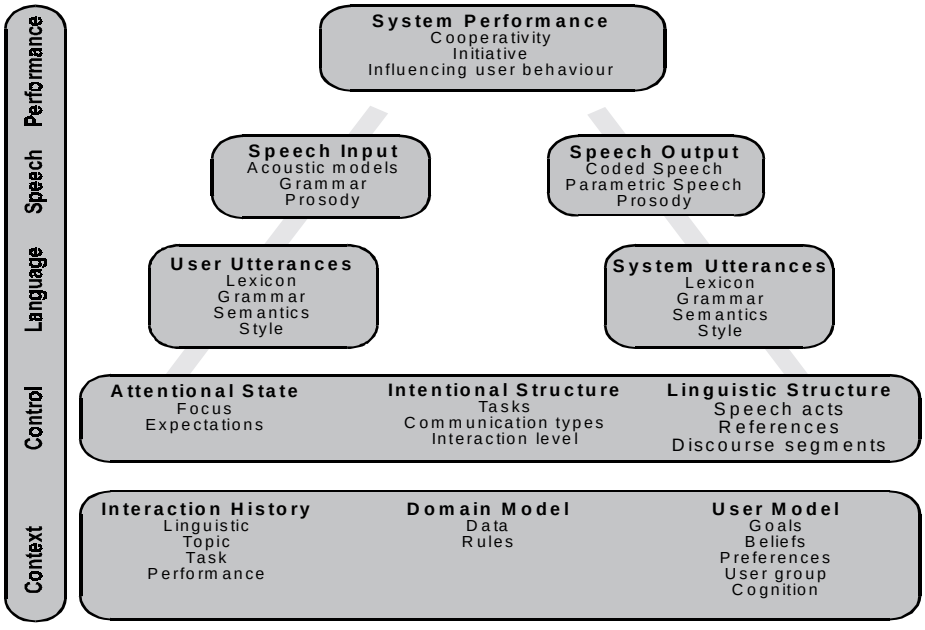

Fig. 4.2.: Elements of an interactive speech theory [BER98].

A useful tool for the evaluation of speech interfaces is the *Wizard of Oz simulation* [BER98], a method from the artificial intelligence research, where the user-computer dialogue is supervised and conducted by a human. This method also allows the fast prototyping of spoken dialogue concepts.

4.2.3. Data Auralisation

The sound parameters as explained above provide the basis for the auralisation of complex data. In general the data parameters are mapped to as many as possible dimensions of sound. This allows the modelling of rather complex auditory structures which provide sufficient feedback to the user for the construction of a mental model of the displayed data. The Kyma system [SCA94] defined a general set of tools and algorithms for generalised and versatile data auralisation:

- The *Shifter* directly represents a data stream as a waveform. If the source data produces frequencies in the infrasonic or ultrasonic range, the frequencies are transformed to an audible range. The resulting data streams can also be used for the control of other parameters.
- A *Mapper* is one of the basic methods for mapping the various data sources to suitable sound parameters.
- *Combiners* perform arithmetic operations, such as summing, subtracting or multiplying on two or more sound streams.
- The *Comparator* plays two audio signals at the same time on the left and the right audio channel to allow the direct comparison of the data. Using this tool differences in timing and structure are instantly audible.
- A *Marker* triggers sounds as soon as specified conditions are met, for example if a signal exceeds a specific amplitude or frequency.
- The *Histogram* auralises classes of data with different frequencies. Their current values are mapped to the amplitude of each frequency.

Data sonification is especially suitable for time-varying data, since sound is a temporal medium [GAV97], which allows to detect changes in the data patterns over time. The essence of good auralisation is the convergence of the multidimensional data in a way that it is perceived as an integral whole. Only this careful design, mostly relying on psycho-acoustics, allows the perception of patterns in multidimensional data. *Interactive auralisation* enables the user to explore specific data regions by having direct influence on the data auralisation process. The tuning of a radio receiver to find the right channel is a good metaphor for the interactive exploration of patterns in auralised data streams.

4.3. Common Design Elements

4.3.1. Voice

Since Text to Speech synthesis is one of the most important channels to transport clearly formed and non-redundant information within an auditory user interface, all its possible configuration options should be exploited for that purpose. The modification options include different gender, age, speed and pitch etc. As already mentioned above the human communication includes more than just the ordinary exchange of spoken words. Important meta-information is transported through the body language, facial expressions and of course the intonation of speech. Since the first two options are not available within an auditory user interface, the expression of speech can be extensively used in order to achieve a more natural quality of the speech communication.

While on the personal computer still versatile keyboard commands can be used in addition or instead of voice commands, within mobile auditory applications the human voice is the only available input for the construction of dialogues. That's why these principles should also be incorporated on the voice recognition side. As described in [sect. 1.2.] here are also more parameters to consider than the common speech recognition. A universal engine is desirable, which combines speech, speaker and voice recognition under the same roof, in order to allow more versatile voice control of auditory applications. This extends the possible bandwidth for the linear spoken interaction significantly.

4.3.2. Auditory Icons

Auditory icons are short sound samples which are played to indicate or accompany certain events on the auditory desktop. They are some of the minor elements which are available in today's graphical user interfaces as well. There they are also generally used to notify the user about certain events or to auralise some user actions such as clicking on objects and so on.

The simplest form of an auditory icon is an ordinary system beep to attract the user's attention. But while this may be sufficient within a graphical user interface, auditory icons for audio interfaces demand to be more complex. In modern everyday life there are already a huge set of auditory icons around wherever we are: The ringing of the phone, the announcement chime on a train platform or the hooting of a car. These everyday sounds can be digitised and used within auditory interfaces to indicate similar events and therefore they are easily recognisable for the user without any learning effort. Other real life sounds

such as an opening door or a coughing person can be chosen for events which are easily to associate with the particular sound, although if badly chosen this already involves a short period of learning and recalling for the user. The third group of auditory icons are artificial sounds with no real world equivalent. Nevertheless these artificial sounds can be designed using musical composition techniques in order to associate them with a certain expression such as gentle, firm or excited. The *Sonic Finder* [GAV89], an auditory extension to the MacOS, which was never released commercially. is a good example for the concept of auditory icons within a mainstream operating system. Apples latest release MacOS 9 now includes some of this functionality as well.

4.3.3. Earcons

Earcons are a variation of auditory icons. While those are mostly digitised sounds associated with real life events or artificial sounds which express a certain condition with psycho-acoustic means, the approach of Earcons is slightly more complex. In general Earcons are short harmonic sequences with a dedicated characteristics, such as a common chord. This characteristics can be inherited by related Earcons in order to form hierarchical groups. The design allows the construction of complex hierarchies, such as the browsing through a tree structure [BREW98]. A transposed common chord for instance indicates a higher hierarchical level, whereas a four note chord stands for a different branch of the tree. This supports the maintaining of a mental model of the tree structure as well as the reasonable localisation of the current position in the hierarchy. An advantage of the structure of earcons from the implementation point of view is that they can easily be created on the fly using MIDI hardware or software.

4.3.4. Music

As shown above Earcons already incorporate some basic musical structures for the construction of the auditory interface. The concept of music within auditory user interfaces is still in its early stages but has a huge potential due to its highly communicative nature [LEP99]. Generally the design of basic AUI components as well as the interface as a whole should also be considered to be realised in a more musical and harmonic manner. This generally avoids the accidental construction of noisy and annoying interfaces and therefore contributes to the user's convenience.

Music extends the physical dimensions of sound again with a variety of additional properties which can be exploited for auditory design. These properties include rhythm,

harmonies, themes, melodies and so on. While music is a complex field which better should be handled by trained composers, it is also quite mathematical which allows the automatic generation of musical structures. While artists have been exploring the field of algorithmic composition during the last decades [Stockhausen], the results of the ongoing research in computer generated and algorithmic music [ALP95] can also be incorporated into the design of auditory interfaces.

4.3.5. Multiple Audio Channels

A way to evade the assumed one-dimensional linearity of auditory interfaces is the use of several simultaneous audio streams. This allows to group various auditory applications on the auditory desktop in a similar way like windows extend the two-dimensional desktop into multiple layers. The often mentioned *Cocktail Party Effect* [ARO92] describes a person's ability to focus on a specific conversation in a noisy party environment, where usually multiple conversations and other background noise take place. This effect can be consciously used for the design of auditory interfaces. In fact this effect somehow is superior than the two dimensional visual display. Recent research by [KALHUX] has shown that it is easier to follow the various events from different sources with multiple auditory cues, than to monitor the movements on several overlapping windows on the screen. Additional use of spatial audio and other audio effects as described below helps to distinguish between the various audio sources.

In auditory desktop environments therefore either the output of several simultaneously running applications as well as of various threads of a single application can be played at the same time. Imagine a web browser, where a download is running in the background, displayed with a standard auditory progress bar as in [sect. 4.4.]. This thread can be displayed at the same time while the user continues to browse the web. The additional background sound does not interfere with the dialogue in the foreground. Additionally there could be another application running in the background which constantly displays the current CPU and memory usage. If such a background task requires some input, the user can switch it to the foreground if desired, which suspends the current dialogue.

Of course many running applications can easily create a noisy environment which might make it harder to follow the foreground task. Therefore background tasks should be significantly lower in volume and should not be designed to aggressive. This would simply be bad design, which is also often unbearable on the visual desktop.

4.3.5. Audio Effects

When working with more than one application at the same time, it can be a problem to distinguish from which application the current output came from. The incorporation of selected audio effects is one of the possible solutions to that problem. Therefore an auditory interface should implement a set of easily distinguishable and predefined audio effects such as echo, reverb and other distortions. An echo for example can be applied to something which is happening in the distance, whereas a heavily distorted sound can be produced to indicate some sort of malfunction. This allows for example to use the same set of auditory icons for various applications in different contexts. On the one hand a smaller set of auditory icons are easier to remember and to associate with their attached events while on the other hand the attached audio effect either binds them to a specific application or gives them a slightly different meaning. If attached to a specific application all the auditory output - synthesised speech and sound - will be distorted in a final signal processing step before being displayed.

4.3.6. Spatial Audio

This is another possibility to distinguish between several concurrent applications. Equipped with a stereo headphone or a good speaker system it is also possible to use spatial audio within auditory applications. This allows the placement of a task's auditory output at a specific location in the audio space. The ability of spatial hearing allows us to specify the distance and the direction of a sound source. We perceive the distance through variations in the intensity of the signal, which is a *monaural* process, which means we basically need only one ear for this task. For the distinction of the direction of a sound source we need both ears, it is a *binaural* process. Similar to the visual stereo depth perception by the combination of two different images from the left and right eye, the spatial position of a sound source can be located by analysing the level, time and phase differences of the two signals received by the left and the right ear [ING00]. The *interaural time delay* (ITD) is usually not longer than 0.65 milliseconds. The level difference as a result of the shading of the further ear is called *interaural intensity difference* (IID). In general the human audio perception performs quite well in the localisation of the direction of audio signals independent of a personal Head Related Transfer Function (HRTF) [WEN91]. The HRTF defines the physiological characteristics of a particular listener. This was an important insight since it makes the generation process of the 3D audio much easier.

Nevertheless there are several problems such as the distraction by other nearby sounds [WIG97] or the confusion of backward and front audio signals [ARR92]. Referring to these problems only a limited number of different points in the space surrounding the front of the head remains available. Given a half sphere surrounding the front of the head there still can be defined nine different application positions in addition to the centre. This number is sufficient for the placement of various applications in the auditory space, though it needs to be evaluated in a future experiment also considering the distraction by concurrent sounds of each application.

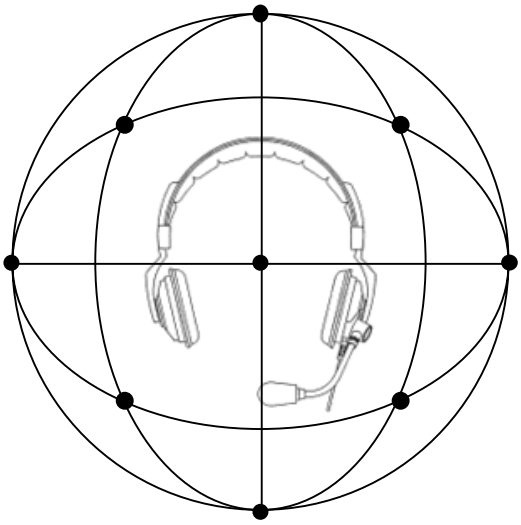

Fig. 4.3.: Location of nine easily distinguishable points in the auditory space.

Besides the integration of 3D audio functionality in the game industry and in virtual reality concepts such as VRML and Java3D [SMJ3D], there are a number of research projects which incorporate spatial audio as integral part of the user interface. The Nomadic Radio project [SAW98] for example, a wearable auditory interface by the Speech Interface Group at the Massachusetts Institute of Technology (MIT) also utilises spatial audio for the positioning of various information sources. There the audio sources are virtually placed in the horizontal plane surrounding the head. While there is also ongoing research in the field of real-time auralisation of virtual environments [STA00], the spatial audio algorithms for auditory interfaces need not to be too sophisticated and can also be implemented with some existing spatial audio libraries [RSX3D], which require less CPU power and also provide sufficient spatial information.

4.4. Auditory Widgets

This section describes the concepts of a few auditory widgets which can be used for the design of dedicated auditory applications. Similar to some available GUI widget toolkits such as Swing for Java [SMJFC] these components could be collected in a reusable library. The ideas listed here are designed for the use with dedicated auditory applications and should not be considered as an attempted auralisation of some corresponding GUI widgets. Nevertheless some of these concepts could be used additionally for screen reader software as well. The following compilation is a collection of initial ideas of which components a collection of auditory widgets could consist and how their basic functionality could look like.

4.4.1. Basic Dialogue Widgets

- Statement

A simple informative statement, usually not longer than a sentence, is read. Additionally an auditory icon corresponding to the actual content of the statement can be attached to capture the user's attention. It is also possible to attach any specified voice (see below) in order to modify the gender, age or expression of the speaking voice.

- Confirmation Dialog

A confirmation dialog is an extension to a statement with similar features, but the given sentence has to be formulated as a Yes-No question. The user then can either answer by a spoken "yes", "no" or "cancel", or alternatively provide his answer by typing y(es) or n(o) or ESC on the keyboard. The resulting Boolean value is then given back to the application for further processing.

- Single Selection Menu

This is already a more complex dialog, since there are a number of options presented to the user from which he can choose one. The user then can choose the desired option by saying the corresponding keyword. The dialog widget itself will manage the appropriate semantic rendering of the question. If only two options are available, a simple question like "Do you want to X or Y?" will be rendered. If there are multiple options to choose from, a spoken list of them will be presented after the question. Additionally the user can scroll through

the list of options using the keyboard arrow keys, while each highlighted option is spoken again. A hit of the return key will select this highlighted option as well. A spoken confirmation then returns the selected value to the calling method.

- Multiple Selection Dialogue

This form of dialog is closely related to the single selection dialog. Its only difference is that it allows to select several items from the presented options.

- Input Label

A minor version of the dictation frame (see below) which only accepts a predefined number of words, and therefore allows to limit or validate the desired input. It can for example be configured only to accept a date input or any other similar data structure such as strings or integers. Upon call a short statement is read after an optional auditory icon. The user then can dictate the desired input word or phrase. A validation process then performs an initial syntax check if the input suits the required data structure. If not the user is asked to repeat his input in the proper way, with an additional example spoken example input. As soon as the input suits the requirements the resulting data is spoken and the user is asked to confirm with a confirmation dialog. After this successful validation and confirmation of the input the resulting data is then given back to the calling application.

- List

This widget displays a one dimensional list of items such as strings, numbers or dates. It can be either spoken automatically presenting one entry after each other, or by browsing it interactively with voice or keyboard commands, where only the selected item is spoken. Corresponding to each entry there can be some header data attached which is then spoken together with the list item.

- Earcon

This object implements the basic earcon functionality as discussed above. It allows the generation of earcon objects, whose characteristics can be inherited be derived objects. This allows the construction of self organising hierarchical tree structures that can be used for various browsing tasks for example. By defining a basic harmonic structure this widget should automatically compute the possible variations for the different levels and branches of the tree structure, but also provide several parameters for the more detailed modification

of the particular sounds. This should speed up the development process while still providing most flexibility in design.

- Reader

This allows the interactive reading of any provided text. Upon call the content is read automatically and can be stopped and started at any time. Additionally the user can scroll through the documents using voice or keyboard commands if he wants to hear some parts again.

- Progress Bar

An auditory progress bar could be realised by playing a looping sound, which changes its pitch according to the current percentage of the progress. A reference sound with a constant pitch corresponding to 100% allows to estimate the actual percentage. After completion an final auditory icon is played. Another useful feature as in [CRE98] is the indication of the progress rate by playing another low level rhythmic sound, where the beat frequency corresponds to the current download rate for example. Predefined voice and keyboard commands are attached to speak the current value and name or to switch the audio progress off and on. Whereas the keyboard and voice commands are predefined, the actual sound loop and reference sound can be chosen from a set.

4.4.2. Complex Dialogue Components

- Table

The table widget is an extension to the simple list widget and accepts any two dimensional array along with the corresponding header data for the rows and columns. It then allows the user to browse the table entries using the appropriate voice or keyboard commands. The actual table data is then spoken to the user in an understandable and structured way. If additional meta-data for the proper speaking of the table entries is required the widget also allows to specify the semantic structure of the spoken output.

- Form

Ideally a form dialogue widget only needs the specification of the required data fields. This includes the data types and their label. Given the form structure this component automatically generates the appropriate form input dialogue. This dialogue should be designed to allow the easy repetitive input of larger amounts of data, while initially guiding

the user through a question-answer dialogue. With the continuing data input first the questions become shorter and less redundant, by only speaking the field labels. Later the user can decide to dictate a whole data set by first defining his desired input sequence by speaking the label names followed by the input. Finally it is only required to dictate the list of data to the form. The dialog provides constant feedback and correction possibilities, and allows the user to modify his input variant.

- File Dialog – Directory Browser

A file dialogue can appear in various forms depending on the way it was called. Once registered the dialog listens to a specified set of commands which trigger different actions. If the user says for example "Save the file!" while working on an already named and existing file the user is prompted with a confirmation dialog after which the file can be saved. When saying "Save the file as" the user is prompted to speak or spell a filename which is then saved in the current working directory. "Save the file in" starts a dedicated component for browsing the complete directory structure. This directory component provides a speech enabled interface to the complete file management. As soon as the right directory is chosen or created, the user provides the filename again in the usual way. Additional inclusion of earcons supports the orientation in the hierarchical directory structure, by assigning a specific pattern to each directory. A file open dialog works the same way according to its current context.

- Dictation Frame

This is a widget which temporarily switches the speech recognition engine to its continuous recognition mode. This allows the user dictate any text freely, as well as the component accepts any text input from the keyboard at the same time. The recognised input is constantly spoken to provide instant feedback and allow immediate correction. The voice feedback stops similar to Emacspeak as soon as further input is provided. This widget also includes some basic word pad functionality such as find and replace. Once the dictation session is finished, the resulting input is provided to the application.

- VoiceXML Frame

This component displays any given VoiceXML document according to its content. It is a core VoiceXML rendering engine, which can be included in applications to allow the reusability of existing local or remote VoiceXML content.

4.4.3. Meta Objects

- Voice

This object is similar to the voice object in the Java Speech API. It modifies the gender, age and expression of the desired voice output. A voice object can be attached to any auditory widget which is then spoken with the specified voice.

- Audio Properties

Any audio property of an auditory component can be controlled in detail. This includes volume settings, speed and pitch etc. This object also incorporates a selection of predefined audio effects such as echo, chorus, reverb etc. that can be used to create various flavours of the same sound sample. In addition to these common properties the audio source can also be virtually positioned at one of nine predefined spatial locations, as well as any freely selectable location by providing the corresponding coordinates. This object then can be applied to any audio source such as digital audio or synthesised speech in order to modify and position the sound in the way as it was discussed above. In a final signal processing step all the selected properties are then applied to the default audio stream.

- Locale

The locale object defines the desired interaction language of a dialog widget. If no dedicated locale is specified the application uses the system's default locale setting to determine the required language. Each widget uses external grammar configuration files for the definition of its speech commands and dialogs. Via this common configuration option the adaptation to some additional languages is facilitated without any changes to the actual widget code.

4.3.4. Auxiliary Functions

- Speech pre-processor

This module provides a set of auxiliary functions to enable the understandable speaking of several language independent structures. These include the proper formatting of date and time statements, the resolving of common abbreviations as well as the correct speaking of complex numbers and simple equations. The component accepts various data structures as explained above and then renders them to a speakable expression. The resulting string is then given back to the application to be included in further dialogs.

4.5. Evaluation of the Emacspeak Auditory Desktop

Emacspeak is currently one of the most mature speech-enabled applications which provides complete eyes-free access to daily computing tasks. Although it is an extension of the popular Unix editor Emacs it should not be considered as a simple screen reader software, but rather as an independent auditory desktop application. Emacspeak doesn't speak the visual screen output of the incorporated applications but provides a set of extensions to these applications which render the basic data output to a speakable format. With Emacspeak blind users can already perform almost any computing task, such as text processing, web browsing, or file management. Other applications include e-mail and news readers as well as complete integrated development environments. Further applications can be speech-enabled by implementing an appropriate extension.

Emacspeak provides a device independent speech interface, which allows the use of various hardware and software speech synthesisers. It comes with a core module for basic spoken feedback, which can be extended with application specific modules. Applications which are speech-enabled by Emacspeak provide continuous context specific feedback, which is immediately stopped upon further user actions. The use of auditory icons augments the interaction. Since no speech recognition functionality is provided by Emacspeak, all the user interaction is performed with the keyboard by typing rather complex key-combinations which are only familiar for experienced Emacs users. Nevertheless after a short learning period these keyboard commands allow the fast and easy access to the complete functionality.

While providing sufficient flexibility to speech-enable existing Emacs applications, the design of Emacspeak is a rather monolithic concept. It doesn't provide any speech or voice recognition functionality yet, as well as it doesn't support other auditory features besides auditory icons. This design allows blind people the powerful speech-enabled access to various standard applications, but doesn't seem to be the right environment for developing dedicated auditory applications. Although Emacspeak provides a large amount of reusable code which can be implemented for further application extensions, it lacks a modular concept for the construction of complex auditory applications which not only use speech as the major communication channel. Surprisingly Emacsspeak incorporates the screen interface as well, although this is primarily not necessary for auditory applications.

5. Y-Windows: Proposal For a Standardised AUI Environment

5.1. Introduction

5.1.1. Basic Idea

Y-Windows (say: "why Windows?") is an attempt to define a multi-platform standardised auditory desktop environment. While providing a fully functional replacement for GUI environments on the personal computer, Y-Windows is also designed as a platform for mobile and embedded applications. This concept is not meant to be another amongst various speech APIs, but will rather implement some of these technologies as described in section two at a higher abstraction level. It also tries to incorporate most of today's concepts of auditory user interfaces as covered in [sect. 4].

Today most desktop operating systems boot by default directly into their GUI desktop environment, such as the Microsoft Windows Explorer™, the MacOS Finder™ or any X-Windows window manager. On the one hand these environments provide the main desktop shell for all major tasks such as file handling or the start of further applications. On the other hand these GUIs manage the graphical interface and overall appearance of all running applications. Additionally every GUI based operating system provides a set of abstract functions and libraries for the composition and display of GUI components, such as windows, dialog boxes, buttons and so on.

The aim of Y-Windows is to provide the same level of functionality for AUI applications. Unlike current screen readers it is not designed to auralise the graphical desktop, but to primarily provide an interface for dedicated auditory applications. A pure Y-Windows scenario would therefore directly boot into its AUI desktop environment and start an auditory desktop application, which could be called *Yshell*. This shell would also allow basic tasks such as file-management and the start of Y-Windows compatible applications.

Additionally there will be supporting auditory widget libraries, which will allow the easy and fast standard compliant implementation of Y-Windows compatible AUI applications. But the most important feature will be that this design allows the simultaneous running of many concurrent auditory applications at the same time. Due to its higher level of abstraction all running applications can use the audio functions and speech engines at the same time, since the Y-Windows server manages their access to the audio hardware. Independent AUI applications, such as current dictation software, occupy all audio resources and therefore limit the use of other auditory applications.

5.1.2. The Unix X-Windows Concept

The Y-Windows design is loosely related to the X-Windows concept, the quasi standard windows GUI environment for various Unix platforms. X-Windows was first designed and developed in the 1980's by the MIT X Consortium and later by X Consortium Inc. [X1198] a non profit membership organisation, which was dissolved in 1996 and all assets such as trademarks and copyrights were transferred to the Open Group Inc. Its current version X11 Release 6.4 was released in January 1998, which was implemented for Linux with the latest version of the open source X Server XFree86 4.0. Other implementations also exist for the various Unix flavours as well as for Windows and MacOS.

The main feature of the X-Windows system is its conceptual and physical separation of the application logic from the actual GUI display. Therefore a dedicated application, the X-Server manages all the graphical output to the screen as well as any user input. The server also provides the low level drivers to the graphics hardware as well as input devices such as the mouse. The actual applications, the so called X-Clients communicate with the X-Server via the X-Protocol, which transports the display commands to the server as well as any user input back to the client. This architecture allows the display of the application's GUI on any computer running an X-Server, whereas the application itself is executed on a remote machine. Of course the server and its clients can also run on the same computer.

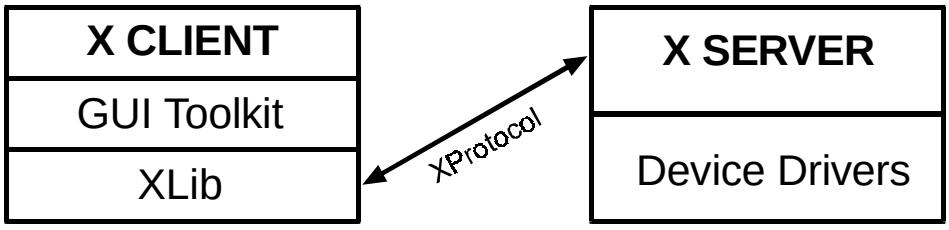

Fig. 5.1.: X-Windows client-server architecture.

An X Server itself only can display basic graphical structures such as lines, points and bitmaps. Since the construction of a modern GUI is far too complex to be constructed from scratch only with these elements, third party developers provide a variety of GUI toolkits, which support programmers with a powerful set of GUI widgets such as windows, menus, buttons and so on. Popular toolkits for that purpose are Motif™ [OGMOT], QT™ [TROQT] or GTK [GNUGTK]. These toolkits then build their widgets out of the basic elements and communicate with the X-Server by using the Xlib which provides the basic functions for the communication with the X-Server.

5.1.3. Common Y-Windows Architecture

Similar to the X-Windows approach the central element of Y-Windows is the Y-Server. Any Y-Windows applications - called the Y-Clients - require a running instance of a Y-Server in order to be able to display their auditory interface. A client can either use a locally running server for its auditory output or can also redirect its display to a remote server running on a different computer in a network. The communication between the server and its clients is managed by a combination of a session protocol, VoiceXML and streaming audio, where the Ylib library provides the necessary set of functions. While the Y-Server only performs basic functions such as speech and audio I/O, the higher level AUI functions are provided and computed by dedicated AUI toolkits, exactly the same way as the corresponding GUI toolkits do. These libraries then encode their interface using VoiceXML and communicate with the server by calling the appropriate Ylib functions.

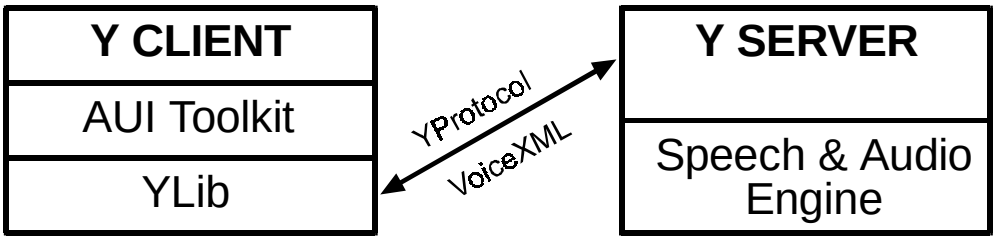

Fig. 5.2.: Y-Windows client-server architecture.

This model also encourages the strict separation of the application logic from the actual user interface display. It not only allows the design of versatile and reasonably platform independent applications but also supports the development of auditory environments in general by providing virtually every infrastructure which is needed for this particular purpose.

5.1.4. Integration into current Operating Systems

Due to its open source nature, which allows access to virtually every operating system function, the primary implementation of Y-Windows should be realised for Linux. This Unix clone already offers all basic functions and libraries needed for a complete Y-Windows implementation:

- Besides the commercial ViaVoice runtimes there are a number of open source speech recognition engines available.
- Drivers exist for most common hardware speech synthesisers and there are various high quality software synthesisers for a number of different languages running on Linux.
- The Linux kernel and further sound architectures provide low level sound and midi drivers for many mainstream soundcards.
- For many of today's audio codecs open source implementations are available.
- At last there is now growing support for proprietary multimedia technologies such as RealAudio or QuickTime.

Besides its easy expandability, Linux also allows the removal of any unnecessary components, such as any GUI applications and libraries in this case. This results in a lean architecture especially designed for AUI applications, which other operating systems would not allow. Linux is also highly scalable and runs on almost any hardware platform starting from PDAs to mobile and desktop personal computers up to large main frame servers. Therefore an AUI application designed for the use with mobile devices could also be used by a blind user on his personal computer.

Additionally Linux offers a system of various run-levels for different states of the operating system. While run-level three for example starts up to a text console only, run-level five starts directly into the graphical X-Windows environment. Luckily the run-level four is unused until now, and reserved for future ideas. Y-Windows could make use of this concept by using the run-level four for the direct start-up into the auditory desktop environment (ADE).

Due to the availability of the Java Speech API a platform-independent Java based implementation of the Y-Windows concept could be realised as well. Java already comes with a variety of other multimedia APIs such as the Java Media and Sound APIs, as well as further Java extensions for mainstream architectures such as QuickTime or RealAudio. Similar to the native Linux implementation the Java solution would have the advantage of running on various different platforms from PDAs to personal computers. On dedicated desktop GUI operating systems such as Microsoft Windows™ or Apple MacOS™ a Y-Windows implementation would have no option other than running on top of the existing GUI environment. A Windows implementation could make extensive use of the Microsoft

SAPI and the DirectX multimedia features, whereas a Macintosh version would mostly rely on technologies such as QuickTime or PlainTalk and some proprietary speech APIs.
For the everyday use of this architecture there are three different scenarios possible: When running on a personal computer, a mode especially designed for use by the sight disabled, the AUI could also incorporate the keyboard as a major input device. Whilst running on a mobile device, where no proper keyboard or similar input device is available, the whole user input should also be manageable with voice commands and dictation only. The third possible scenario can be the co-operational GUI mode, where Y-Windows is running at the same time with a standard GUI.

5.1.5. Problems and Future Challenges

A main question concerning the design of Y-Windows may be: Why all this effort when there are versatile concepts such as VoiceXML and many other speech or multimedia APIs?
VoiceXML is definitely a very powerful concept. In fact the Y-Windows design is somehow an implementation of a VoiceXML browser. Whereas the Yshell is basically also useable as a VoiceXML browser, the Y-Windows framework as a whole also provides an environment for developing and distributing VoiceXML applications.
Since not all graphical GUI applications can be replaced by faceless XML applications which are displayed within a web browser, VoiceXML is also not (yet) the answer to all the issues of auditory user interfaces. Therefore Y-Windows with its libraries and basic applications can be considered more than the sum of the existing speech technologies and APIs: It is a fully functional auditory desktop environment. VoiceXML has clearly its advantages for usage with lean mobile devices and it will possibly evolve substantially in the future which could make the Y-Windows system obsolete. But then still a major part of the proposed concepts can be recycled for future technologies.
On the other end the speech and multimedia APIs themselves also do provide almost everything which is needed for developing complete AUI applications. But they provide no higher level AUI functions and no instance for managing multiple concurrent auditory applications. It is here where Y-Windows again finds its *raison d'être* by providing this additional high level functionality.

5.2. Y- Server

5.2.1. Integrated Audio and Speech Engine

One of the main components of the Y-Windows system is the management of the audio resources. Many Linux desktop managers already follow the concept of a central audio engine, which only allows applications the indirect access to the audio hardware. If a particular application address the soundcard directly, no other application could gain access to the audio hardware at the same time. An AUI additionally make extensive use of speech recognition and synthesis, which then would permanently occupy the main sound resources such as input and output.

Therefore Y-Windows provides an integrated audio and speech engine, which exclusively manages the access to the computer's audio resources. Any Y-Client can only call some higher level audio functions, which the Y-Server then maps to the appropriate audio in- and output. Y-Client applications should not make use of any audio resource directly.

The speech recognition engines used can either be some existing product like IBM ViaVoice™ or an independent implementation using open source projects like Sphinx2. Speech synthesis can also either be performed by hardware synthesisers or some software solutions. A modular design will allow the easy replacement of any of these components with a different or improved solution. The common audio part handles the playback and recording of digital audio and MIDI data. The hardware devices themselves are managed by the appropriate system drivers and accessed via the standard multimedia APIs of the host operating system.

Communication Layer		
YProtocol		RTSP
Rendering Layer		
VoiceXML		Audio Codecs
Engine Layer		
Audio	Speech Synth.	Speech Reco.
Hardware Layer (OS)		
Audio I/O	MIDI I/O	Speech HW

Fig. 5.3.: Y-Server modular layer architecture.

5.2.2. Rendering Component

The heart of this component is a voice browser based on standard VoiceXML extended with some additional functionality. This part will parse the incoming VoiceXML content, control the speech recognition and synthesis engines as well as play and record digital audio. Additional functionality means the implementation of real time streaming and some more sophisticated audio and signal-processing functions such as spatial audio and various sound effects, which can be applied to plain audio signals. This design allows the voice browser component both the handling of standard VoiceXML content as well as the display of the full AUI functionality.

5.2.3. Communication & Protocols

Upon start-up a Y-Server will listen to a specified port and wait for Y-Client applications to connect to use its services. The communication between the server and its clients will consist of three major components:

- A designated Y-Windows session and transport protocol (Y-Protocol).
- VoiceXML for the AUI content encoding.
- Real-time streaming audio protocols (RTSP) for remote displays.

This communication architecture is a combination of open standards to ensure maximum compatibility with existing technologies such as RTSP and VoiceXML and some independent extensions to achieve maximum flexibility for more versatile AUI applications. Real time streaming is only planned for the instant display of longer audio streams on remote displays, since delayed network transfer would interfere with some time critical applications. For shorter audio files as well as for locally running Y-Servers direct access to the audio data will be provided. To enhance performance for frequently used auditory icons, the server incorporates a caching system, which also significantly reduces the use of communication bandwidth.

5.3. Libraries & APIs

5.3.1. Ylib

This is the most basic component, which is at least necessary to build a Y-Windows client application. The Ylib library primarily defines functions for the communication between the client applications and the Y-Server. This mainly includes the management of the client-server session by implementing the Y-Protocol and includes the transport of voice dialogs and audio to the server and the feedback of user interactions and other events back to the client.

5.3.2. VoiceXML Library

The VoiceXML Library provides everything which is needed to create, validate and process VoiceXML documents. They are used by the Y-Clients in particular the AUI toolkits to encode their voice dialogs into this format. The Y-Server also needs some of its functionality to validate, decode and render the interface. Actually these libraries should consist of two major parts, one which implements the actual VoiceXML standard, and a second component that incorporates some proprietary extensions. Therefore the extensible nature of VoiceXML will be used. Probably a third party VoiceXML library can be used, or any XML library extended for the primary part. If on the other hand these necessary libraries are implemented from scratch for the Y-Windows environment they should be designed to be commonly usable for third party projects as well.

5.3.3. AUI Widget Toolkit

Besides the actual Y-Server implementation most effort should be put into the design of a mature AUI widget toolkit. Whilst for an initial release only one particular library set will be created, the Y-Client design should also allow third party developers to create their own independent AUI toolkit. This should lead to some freedom of choice for the actual *hear & feel* of auditory applications, while a common interaction standard for the user should be maintained. In general these toolkits should incorporate most of the elements as presented in [sect. 4.4.].

5.3.4. Data Auralisation Toolkit

This library provides a set of functions and classes for the display of complex data. It should include tools and algorithms as presented in [sect. 4.2.3.] which make the general data auralisation easier for the programmer. The user also would appreciate a common way of data representation in various applications in order to reduce the learning effort.

5.3.4. Internationalisation

An important issue of Y-Windows and of application design in general is the internationalisation (I18N). To allow easy language independent development of Y-Windows applications standard AUI widgets should already come in a predefined variety of different mainstream languages configurable within external grammar files. These grammars should be easily extendable for any desired language. Depending on the system's locale settings and the installed or selected speech synthesis and recognition dictionaries the appropriate language for the output will be chosen.

5.4. Example Y-Windows Applications

5.4.1. Login Application: Ylogin, Ydm

After starting directly into run-level four the user will be presented with a login dialog, with similar functionality like its X-Windows counterpart *xdm*. Its main purpose is the authentication of the user who wants to log into the system. Instead of the traditional way of asking for a login name and a password this application should use a speaker recognition feature to verify if the speaker is the person he pretends to be. This could be achieved by speaking a simple passphrase. This solution has the advantage that if other people overhear the user's passphrase the system still won't let an unauthorised person use the account if it doesn't recognise his voice. A quite similar application like this is already implemented in the latest Macintosh operating system MacOS 9.

5.4.2. Shell Application: Yshell, Ybrowser

This is normally the first application to start up after a successful login dialog. Though other applications do not require a running *Yshell* to be fully operational, this should be the main program of a Y-Windows session. The Yshell provides all of the functions a desktop environment needs such as file handling and the start and management of other Y-

Windows applications. It also gives direct access to the VoiceXML browser functionality of the Y-Server. In fact the Yshell is a 100% standards compatible VoiceXML browser with full access to every VoiceXML content online. All other desktop functions are similar to those of Emacspeak, except the additional support of voice commands and dialogs as well.

5.4.3. HTML Browser: Yweb

This application is designed to allow auditory access to the standard world wide web. Its main task is to retrieve ordinary HTML pages and convert them to a speakable format as well as it can. Generally HTML pages should be well structured documents which could easily be transformed into a reasonably speakable form. Unfortunately web design is today not only standard HTML formatting, but rather the abuse of the available functions for graphical design. Even with an ordinary text browser such as *Lynx* [LYNX] most web-pages are not readable anymore. The W3C therefore started an accessibility initiative [W3CWAI] in order to encourage developers of web content to pay more attention. to this issue Hopefully there will be more accessible web documents after the expected transition from HTML4 to XHTML [W3CXHTML], which is much stricter than the HTML model and therefore strictly separates the content structure from the actual visual rendering. Additionally the W3C published a working draft for Aural Cascading Style Sheets [W3CACSS], co-authored by T.V. Raman, which proposes a standard method for the aural rendering of web documents. This concept later was included in the current Cascading Style Sheets 2 specification [W3CCSS2]. Yweb should incorporate these technologies as well in order to improve its aural rendering quality. Like all other Y-Windows applications Yweb accepts input controls from the keyboard as well as spoken commands.

5.4.4. Editor: Yedit

This is the main word processing application of the Y-Windows environment. With its two main operational modes it is either a text editor which accepts keyboard input with speech feedback, or it can also work using fully functional continuous speech recognition in the dictation mode. Similar to Emacspeak instant input feedback using TTS is provided by speaking any typed letter, word or sentence.

5.4.5: Unix Shell Utility: Yconsole

Unix comes with a vast amount of console based and text only applications, which are far more easier to handle within an auditory environment than GUI applications. With its simple but powerful concepts of a command structure like "command –{option}", and its versatile redirection possibilities of the standard in- and output those commands can handle tasks from converting sound files up to burning CD-ROMs. While they allow easy and fast input via the keyboard, the speaking of such commands can be rather exhausting because of their cryptic syntax. Since the spelling of complex command structures cannot be the best solution, Yconsole provides configuration scripts for the most important console commands in order to make them speakable. These scripts also define rules for the proper auditory rendering of the command's output. Of course, as with any open source project, the structure of these configuration files is publicly available so any console application can therefore be easily made speakable within Yconsole. Applications which use the full capability of the terminal such as *Pine* or *Lynx* are not suitable to auralise because they already provide their terminal interface.

5.4.6. GUI Reader Utility: Yreader

When operating together with a GUI environment Yreader takes the same approach like other screen readers such as *Jaws for Window*s or Mercator [MERC], the X-Windows equivalent. Accordingly it therefore tries to auralise all GUI component in a similar manner. It could be implemented with two operational modes by either redesigning the concept of *Jaws* to be Y-Windows compatible which has the advantage that current users of such packages does not need to learn a new concept or by using the standard Y-Windows AUI widgets to represent their GUI equivalents. This second mode would be rather more convenient for experienced Y-Windows users. Of course special versions of Yreader would have to be designed for every different GUI system either as an independent application or with a more versatile plug-in architecture.

5.4.7. Other Application Ideas

This space is far too limited to describe all imaginable auditory application ideas. In fact virtually any existing desktop application should also be eligible to be implemented with an auditory user interface only. Exceptions might be designated graphical applications, though these might be a challenge as well [RAM89]. Generally everything from office applications such as spreadsheets to internet tools like e-mail and chat clients are possible as well as auditory tools for composers or hands free programs for other professionals.

5.5. Hardware Requirements

A Y-Windows implementation platform of course primarily needs a sufficient audio hardware equipment. For a desktop personal computer this means at least a state of the art soundcard, preferably with full-duplex and MIDI support as well as a very good signal quality. Presumably this is the case for most of today's standard soundcards. In order to achieve better performance it could also be an option to install a second soundcard to provide a dedicated sound I/O channel for the speech recognition and synthesis components. Some external or internal speech synthesis hardware would also do the job. For the audio playback and recording a premium quality stereo headset including microphone is highly recommended, since this is the main human-computer interface. Mainly for optimal speech recognition performance a fast processor and sufficient memory supply should be considered, a Pentium MMX 233 with 64 MB is the absolute minimum, the more the better.

An embedded implementation platform generally will be equipped with similar hardware as above, but the main difference is the lack of a headset. Therefore an embedded system has to come with built-in microphone and speakers, of course with sufficient quality. Since embedded solutions should be lean and cheap the use of speech recognition chips for command and control purposes instead of large software based recognition systems should be considered. A mobile scenario will come again with a headset, but probably a one-ear mono version only. Here also the use of dedicated speech hardware should be considered for the implementation platform.

6. Conclusions and Future Work

This thesis has shown that there are three major tasks to be completed towards a standardised auditory user interface or desktop environment. First of all there are still several open problems to investigate around the basic AUI research, including some empirical studies to validate most of the concepts as collected in this paper. The second challenge is the implementation of these concepts in the form of powerful and convenient auditory widgets in order to speed up the development of AUI applications. Last but not least speech and audio engines need to be integrated as a standard component into the core of today's operating systems.

Main target of the future work will be the further and more detailed design of the Y-Windows concept with a special focus on the implementation of the auditory widget components. An early prototype should be implemented on a desktop platform while the main target of the development should definitely focus on the design of mobile auditory applications, since this clearly is the future application area for auditory user interfaces. Another important aspect shown by this thesis is the need for the simultaneous development of auditory user interfaces which are both suitable for the blind user on the personal computer and for the users of mobile applications as well.

The concepts presented in this paper still should be considered as an early draft version. Their main purpose is the formation of an idea, which is open to discussion and comments and also may still contain some contradictions. After an initial presentation of this thesis at an appropriate conference (topics such as Speech Technology, User Interfaces or Linux), a second more detailed version may be published which already should incorporate some third party input and corrections. If this concept proves to be worth being implemented, an open Y-Windows project group will be founded, in its structure similar to other open-source Linux projects. This will include the installation of a project web-site, a mailing list, a CVS server and similar services for the joint development. Any volunteer is welcome to take part in the various design and implementation processes.

IV. References

[ALP95] Alpern A.: *Techniques for Alogorithmic Composition of Music*. Hampshire College. URL, http://hamp.hampshire.edu/~adaF92/algocomp/ (2000/06/01).

[APPPT] Apple Computer Inc.: *PlainTalk*. URL, http://www.apple.com/speech/ (2000/06/01).

[APPQT] Apple Computer Inc.: *Quicktime*. URL, http://www.apple.com/quicktime/ (2000/06/01).

[ARO92] Arons B.: *A Review of the Cocktail Party Effect*. In *Journal of American Voice I/O Society*, Vol. 12 p.35-50, July (1992).

[ARR92] Arruda M., Kistler D. J., Wightman F.: *Origin and analysis of front-back confusions in human judgements of apparent sound direction. In Abstracts of the 15th Midwinter Research Meeting*, Association for Research in Otolaryngology (1992).

[ARTCOM] Advanced Recognition Technologies Inc.*: Voice and Handwriting Recognition for Mobile PCs*. URL, http://www.artcomp.com/ (2000/06/01).

[ATTASAPI] AT&T Corporation: *Advanced Speech API*. URL, http://www.att.com/aspg/ (2000/06/01).

[BBCMC] Schmidt R.: *BBC Microcomputer Documentation*. URL, http://www.nvg.ntnu.no/bbc/docs.shtml (2000/05/19).

[BER98] Bernsen N.O., Dybkjaer H., Dybkjaer L.: *Designing Interactive Speech Systems*. Springer, London (1998).

[BER99] Berger, Liaw: *Berger-Liaw Neural Network*. URL, http://www.usc.edu/ext-relations/news_service/releases/stories/36013.html (1999/09/30).

[BIO99] Biocca F., Nowak K.: *I feel as if I'm here, inside the Computer. Toward a theory of presence in Advanced Virtual Environments*. International Communication Association, San Francisco (1999).

[BLPP] O' Sullivan K., Teahan C.: *Blind Penguin Project*. URL, http://snet.wit.ie/BlindPenguin/ (2000/06/01).

[BREW98] Brewster S.A.: *Using non-speech sounds to provide navigation cues*. In ACM Transactions on Computer-Human Interaction, 5(2), pp 224-259 (1998). URL, http://www.dcs.gla.ac.uk/~stephen/papers/TOCHI98.pdf (2000/06/01).

[CAR96] Carey C. (Ed.): *Human Factors in Information Systems - The Relationship Between User Interface and Human Performance*. Ablex Publishing Cooperation, London (1996).

[CONVW] Conversa Computing Corporation: *Conversa Web v3.0*. URL, http://www.conversa.com/Products/Web.asp (2000/06/01).

[CRE98] Crease, M.G., Brewster, S.A.: *Making Progress With Sounds - The Design and Evaluation Of An Audio Progress Bar*. In Proceedings of ICAD'98 (Glasgow, UK), British Computer Society (1998). URL, http://www.dcs.gla.ac.uk/~stephen/papers/ICAD98_MC.PDF (2000/06/01).

[DEM98] De Mori R. (Ed.): *Spoken Dialogues with Computers*. Academic Press, London (1998).

[DOB00] Dobelle W.H.: *Artificial Vision for the Blind*. URL, http://www.artificialvision.com/vision/ (2000/04/12).

[DOW91] Downton A. (Ed.): *Engineering the Human-Computer Interface*. McGraw-Hill, London (1991).

[DXXX] DirectXtras Inc.: *Xpress Xtra*. URL, http://www.directxtras.com/ (2000/06/01).

[EPI00] Epinoisis Software Corp.: *Digital Ear v2.0*. URL, http://digitalear.iwarp.com/info.htm (2000/06/01).

[FCDEC] Force Computers Inc.: *DECtalk Speech Synthesis*. URL, http://www.forcecomputers.com/product/dectalk/dtalk.htm (2000/06/01).

[FSFGPL] Free Software Foundation: *GNU General Public License v2*. URL, http://www.gnu.org/copyleft/gpl.html (1991/06/01).

[FSTDSP] Fluent Speech Technologies: *Embedded Speech Interfaces*. URL, http://www.fluent-speech.com/embedded.htm (2000/06/01).

[GAU95] Gauvin J.L., Lamel L.F., Prouts B.: *Speaker Identification and Verification*. URL, http://www.limsi.fr/Recherche/TLP/reco/2pg95-sv/2pg95-sv.html (1999/03/01).

[GAV89] Gaver W.: *The Sonic Finder – An Interface that Uses Auditory Icons. In The Use of Non-Speech Audio at the Interface* pp. 5.85-5.106, ACM Press, CHI '89 Austin, TX (1989).

[GAV97] Gaver W.: *Auditory Interfaces*. In Helander M.G, Landauer T.K., Prabhu P. (Ed.): *Handbook of Human Computer Interaction*. Elsevier Science, Amsterdam (1997).

[GET96] Gettys J., Scheifler R.W.: *Xlib –C Language X Interface*. X Consortium Inc. (1996).

[GILSPS] Gill Dr. J.: *Speech Synthesisers*. URL, http://www.tiresias.org/eb8.htm (2000/05/05).

[GILSR] Gill Dr. J.: *Screen Reading Software*. URL, http://www.tiresias.org/eb9.htm (2000/05/11).

[GNUGTK] Free Software Foundation: *GTK –The GIMP Toolkit*. URL, http://www.gtk.org/ (2000/06/01).

[GOO99] Goose S., Möller C.: *A 3D Audio Only Interactive Web Browser: Using Spatialization to Convey Hypermedia Document Structure*, in the Electronic Proceedings of the ACM 99 Conference. URL, http://woodworm.cs.uml.edu/~rprice/ep/goose/ (2000/11/5).

[HJJFW] Henter-Joice Inc.: *Jaws for Windows*. URL, http://www.hj.com/JFW/JFW.html (2000/06/01).

[HOL93] Holmes J.N.: *Speech Synthesis and Recognition*. Chapman & Hall, London (1993).

[IBMAW] IBM Corporation: *Alphaworks Technology*. URL, http://www.alphaworks.ibm.com/ (2000/06/01).

[IBMPDA] IBM Corporation: *IBM Embedded ViaVoice*. URL, http://www-4.ibm.com/software/speech/enterprise/technologies_5.html (2000/06/01).

[IBMSAPI] IBM Corporation: *Java Speech API implementation*. URL, http://www.alphaworks.ibm.com/tech/speech/ (2000/03/36).

[IBMTTS] IBM Corporation: *Online Text to Speech*. URL, http://www.alphaworks.ibm.com/tech/tts/ (2000/05/04).

[IBMVV] IBM Corporation, *ViaVoice Millennium*. URL, http://www-4.ibm.com/software/speech/ (2000/06/01).

[IBMVVDC] IBM Corporation: *ViaVoice Developer's Corner*. URL, http://www-4.ibm.com/software/speech/dev/ (2000/06/01).

[IBMVXML] IBM Corporation, *VoiceXML implementation*. URL, http://www.alphaworks.ibm.com/tech/voicexml (2000/04/17).

[IMGSRK] Images Company: *Speech Recognition Kit*. URL, http://www.imagesco.com/catalog/hm2007/SpeechRecognitionKit.html (2000/06/01).

[ING00] Ingham N.: *An Introduction into Spatial Hearing*. URL, http://www.physiol.ucl.ac.uk/wwwphysiol/research/spheres/intro.htm (2000/02/19).

[ITSY] Compaq Western Research Lab: *The Itsy Pocket Computer*. URL, http://research.compaq.com/wrl/projects/itsy/ (2000/06/01).

[JUL94] Jullien J.P., Warusfel O.: *Technologies et perception auditive de l'espace*. In *Cahiers de l'Ircam*, 5, (March 1994). URL, http://mediatheque.ircam.fr/articles/textes/Jullien94/ (1997/06/20).

[KALHUX] Kaltenbrunner M., Huxor A.: *Marvin - Supporting Awareness Through Audio in Collaborative Virtual Environments*, In *The Proceedings of the 2000 Conference on Digital Content Creation*, Bradford (2000/04/13).

[KEL94] Keller E. (Ed.): *Fundamentals of Speech Synthesis and Speech Recognition*, John Wiley & Sons, Chichester (1994).

[KHE94] Kheong-Chee Y., Edgar H., Myers D., Docherty T.*: An Introduction into Vector Quantisation*. In *The Proceedings of the Second International Interactive Multimedia Symposium*. URL, http://cleo.murdoch.edu.au/gen/aset/confs/iims/94/bc/chee.html (1994/01/28).

[KRA94] Kramer G.: *An Introduction to Auditory Display*. In Kramer G. (Ed.) *Auditory Display*. Addison-Wesley, New York (1994).

[LAU95] Lauer T.W.: *Development of a Generic Inquiry Based Problem Analysis Method*. Oakland University. URL, http://lattanze.loyola.edu/frames/research/wp0595.028.html (95/05/28).

[LAN94] Lansdale M.W., Ormerod T.C.*: Understanding Interfaces – A Handbook of Human-Computer Dialogue*. Academic Press, London (1994).

[LEP99] Leplâtre G., Brewster S. A.: *Perspectives on the Design of Musical Auditory Interfaces.* Focus conference on Anticipation, Cognition and Music. Liege, Belgium, Aug. 1998. In Dubois D. (Ed) *International Journal of Computing Anticipatory Systems*, Feb. 1999. URL, http://www.dcs.gla.ac.uk/~stephen/papers/casys98.pdf

[LHDRAG] Lernout & Hauspie: *Lernout & Hauspie signs definitive agreement to acquire Dragon Systems*. URL, http://www.lhsl.com/news/releases/20000328_dragon.asp (2000/03/28).

[LHEMB] Lernout & Hauspie: *Phonetic Speech Recognition for RISC Processors*. URL, http://www.lhsl.com/srec/asr1600risc.asp (2000/06/01).

[LHJSAPI] Lernout & Hauspie: *Java Speech API Implementation*. URL, http://www.lhsl.com/news/releases/19981019_Sun.asp (1998/10/19).

[LHNAK] Lernout & Hauspie, *Nak Speech Recognition Handheld*. URL, http://www.lhsl.com/news/releases/20000208_dictation.asp (2000/24/02).

[LHRS] Lernout & Hauspie: *Real Speak Speech Synthesis*. URL, http://www.lhsl.com/realspeak/ (2000/06/01).

[LHVE] Lernout & Hauspie: *Voice Xpress v4.0*. URL, http://www.lhsl.com/voicexpress/ (2000/06/01).

[LHVESDK] Lernout & Hauspie: *Voice Xpress SDK*. URL, http://www.lhsl.com/voicexpress/sdk (2000/06/01).

[LINGTT] Linguatec GmbH: Talk & Translate. URL, http://www.linguatec.de/products/tt/ (2000/06/01).

[LINRH] Red Hat Inc.: *Red Hat Linux Distribution*. http://www.redhat.com/ (2000/06/01).

[LINUX] Torvalds L.: *Linux Operating System*. URL, http://www.linux.org/ (2000/06/01).

[LINZS] Campbell M.: *ZipSpeak - A Talking Mini-Distribution of Linux*. URL, http://www.crosswinds.net/~mattcamp/zipspeak.html (2000/03/19).

[LOW80] Lowerre B., Reddy R.: The Harpy speech understanding system. In *Trends in Speech Recognition* (W. A. Lea, ed.), Prentice Hall (1980).

[LYNX] Dickey T.: *Lynx Web Browser*. URL, http://lynx.browser.org/ (2000/04/23).

[MAR95] Martin K.: *A Computational Model of Spatial Hearing*. Master's thesis, EECS dept, MIT (1995). URL, ftp://sound.media.mit.edu/pub/Papers/kdm-ms-thesis.ps.Z (2000/06/01).

[MBROL] Multitel TCTS Lab: *Mbrola Speech Synthesiser*. URL, http://tcts.fpms.ac.be/synthesis/ (1999/12/17).

[MCBRY] Meta Creations Corporation: *Bryce v4*. URL, http://www.metacreations.com/products/bryce4/ (2000/06/01).

[MEL98] Mellish C.: *Natural Language Generation*. In *Speech and Language Engineering –State of the Art*, IEE Informatics, London (1998).

[MERC] Edwards K.: *Mercator - Providing Access to Graphical User Interfaces for Computer Users Who Are Blind*. URL, http://www.cc.gatech.edu/gvu/multimedia/mercator/mercator.html (2000/06/01).

[MOE98] Moens M.: *Natural Language Analysis*. In *Speech and Language Engineering –State of the Art*, IEE Informatics, London (1998).

[MOTCLM] Motorola Inc.: *Clamor Speech Recognition Technlology*. URL, http://www.mot.com/MIMS/lexicus/technologies/speech/clamor.html (2000/06/01).

[MOTVOX] Motorola Inc.: *VoxML*. URL, http://www.voxml.com/voxml.html (2000/06/01).

[MPEG3] Fraunhofer Institut für Integrierte Schaltungen: *MPEG Audio Layer-3*. URL, http://www.iis.fhg.de/amm/techinf/layer3/ (2000/06/01).

[MSACTX] Microsoft Corporation: *ActiveX*. URL, http://www.microsoft.com/com/tech/ActiveX.asp (2000/06/01).

[MSCOM] Microsoft Corporation: *Component Object Model*. URL, http://www.microsoft.com/com/tech/com.asp (1999/03/30).

[MSMA] Microsoft Corporation: *Microsoft Accessibility*. URL, http://www.microsoft.com/enable/ (2000/05/18).

[MSMIPAD] Microsoft Corporation: *MiPad - A Multimodal, Interactive Note Pad*. URL, http://research.microsoft.com/srg/mipad.asp (2000/06/01).

[MSSAPI] Microsoft Corporation: *Microsoft Speech API v4.0*. URL, http://www.microsoft.com/IIT/projects/sapisdk.htm (1999/04/26).

[MYHMMS] Myers R. Whitson J.: *Hidden Marcov Models for Automatic Speech Recognition*. URL, http://svr-www.eng.cam.ac.uk/comp.speech/Section6/Recognition/myers.hmm.html (1996/09/26).

[NICANN] Ström N.: *The Nico Artificial Neural Network Toolkit v1.0*. URL, http://www.speech.kth.se/NICO/ (1999/02/14).

[NIE93] Nielson J.: *Noncommand User Interfaces*. In *Communications of the ACM* 36 (1993), p. 83-99. URL, http://www.useit.com/papers/noncommand.html (2000/06/01).

[OKIMSM] Oki Semiconductors Inc.: *MSM6679 Voice Dialling*. URL, http://www.okisemi.com/communicator/public/nf/docs/Intro-4280.html (2000/06/01).

[OGMOT] The Open Group Inc.: *Open Motif*. URL, http://www.opengroup.org/openmotif/ (2000/06/02).

[PALM] Palm Computing Inc.: *Palm Pilot PDA*. URL, http://www.palm.com/ (2000/06/01).

[PARC] Xerox Corporation: *Xerox Palo Alto Research Centre*. URL, http://www.parc.xerox.com/ (2000/06/01).

[PHFS] Philips Speech Processing Group: *Free Speech 2000*. URL, http://www.speech.be.philips.com/ud/get/Pages/072fr.htm (2000/06/01).

[PHSPG] Philips Speech Processing Group: *Homepage*. URL, http://www.speech.be.philips.com/ (2000/06/01).

[PHVA] Philips Speech Processing Group: *VocAPI*. URL, http://www.speech.be.philips.com/ud/get/Pages/h0642.htm (2000/06/01).

[PHVASDK] Philips Speech Processing Group: *VocAPI SDK*. URL, http://www.speech.be.philips.com/ud/get/Pages/h0644.htm (2000/06/01).

[RAB93] Rabiner L., Biing-Hwang J.: *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs (1993).

[RAD99] Radev D.R.: *Natural Language Processing FAQ*. URL, ftp://rtfm.mit.edu/pub/usenet/comp.ai.nat-lang/Natural_Language_Processing_FAQ (1999/09/16).

[RAN99] Ranjan A.: *Artificial Neural Networks – An Introduction*. In *Hello World* 3/99. URL, http://helloworld.nexus.carleton.ca/1999/03-mar/artificial_neural_networks.html (2000/06/01).

[RAM00] Raman T.V.: *Emacspeak Auditory Desktop 12.0*. URL, http://www.cs.cornell.edu/home/raman/emacspeak/ (2000/04/30).

[RAM89] Raman T.V.: *Congrats: A System for Converting Graphics to Sound*. Cornell University (1989).

[RAM97] Raman T.V.: *Auditory User Interfaces: Towards the Speaking Computer*, Kluwer Academic Publishers, Norwell (1997).

[ROD99] Rodman, R.D.: *Computer Speech Technology*, Artech House, Norwood (1999).

[RSX3D] Rad Game Tools Inc.: *Miles' RSX 3D Audio*. URL, http://www.radgametools.com/rsxmain.htm (2000/02/01).

[SAW98] Sawhney N.,Schmandt C.: *Speaking and Listening on the Run - Design for Wearable Audio Computing*. In *The Proceedings of ISWC'98, International Symposium on Wearable Computing*. URL, http://www.media.mit.edu/~nitin/projects/NomadicRadio/ISWC98/ISWC98.html (1998/10/20).

[SCA96] Lusted H.S., Knapp R.B.: *Controlling Computers with Neural Signals*. In *Scientific American*, Issue 10/96. URL, http://www.sciam.com/1096issue/1096lusted.html (1996/10/01).

[SCA94] Scaletti C.: *Sound synthesis algorithms for auditory data representations*. In Kramer G. (ed.) *Auditory Display*.New York (1994).

[SCHE94] Scheifler R.W.: *X Window System Protocol – X Consortium Standard*. X Consortium Inc. (1994).

[SCHU94] Schukat-Talamazzini E.G.: *Automatische Spracherkennung*. Vieweg & Sohn, Braunschweig/Wiesbaden (1995).

[SEP99] Sephton T., Black J., El Naggar G., Fong A.: *A Mobile Computing System for Testing Wearable Augmented Reality User Interface Design*. In The *Proceedings of ISWC'99, International Symposium on Wearable Computing*, URL, http://funhousefilms.com/MARS/ISWC99e.htm (1999/10/20).

[SENSRSC] Sensory Inc.: *RSC 300/363 – Speech Recognition Microcontroller*. URL, http://www.sensoryinc.com/html/products/rsc364.html (2000/06/01).

[SMJ3D] Sun Microsystems: *Java 3D API 1.2*. URL, http://java.sun.com/products/java-media/3D/ (2000/05/24).

[SMJNI] Sun Microsystems: *Java Native Interface*. URL, http://java.sun.com/products/jdk/1.3/docs/guide/jni/ (2000/02/18).

[SMJFC] Sun Microsystems: *Java Foundation Classes - Swing*. URL, http://java.sun.com/products/jfc/ (2000/05/15).

[SMJSAPI] Sun Microsystems: *Java Speech API v1.0*. URL, http://java.sun.com/products/java-media/speech/ (1998/10/26).

[SMJTAPI] Sun Microsystems: *Java Telephony API v1.3*. URL, http://java.sun.com/products/jtapi/ (2000/02/28).

[SPHX2] Carnegie Mellon University*: CMU Sphinx –Open Source Speech Recognition*. URL, http://www.speech.cs.cmu.edu/sphinx/Sphinx.html (2000/01/30).

[SPTMAG] *Speech Technology Magazine*. URL, http://www.speechtechmag.com/ (2000/06/01).

[SRAPIC} Speech Recognition API Committee: *Speech Recognition API Homepage*. URL, http://www.srapi.com/ (unavailable).

[STA00] Staudinger J.: *Echtzeitfähige, binaurale Auralisation geschlossener, virtueller Räume*. Diplomarbeit. Fachhochschule Medientechnik und -design, Hagenberg (2000).

[TALKS] TalkSoft Corporation: *Embedded Speech Recognition Technology*. URL, http://www.talksoft.com/ (1998/10/22).

[TAY98a] Taylor P., Black A.W., Caley R.: *The Architecture of the Festival Speech Synthesis System*. In The Third ESCA Workshop in Speech Synthesis, pp. 147-151 (1998).

[TAY98b] Taylor P.: Speech Synthesis. In *Speech and Language Engineering –State of the Art*, IEE Informatics, London (1998).

[TI99SP] Nouspikel T.: *The TI99 Speech Synthesiser Module*. URL, http://www.stanford.edu/~thierry1/ti99/speech.htm (1999/01/31).

[TROQT] Trolltech AS: *QT™ GUI Software Toolkit*. URL, http://www.trolltech.com/products/qt/qt.html (2000/05/31).

[UEFEST] Centre for Speech Technology Research, University of Edinburgh*: The Festival Speech Synthesis System*. URL, http://www.cstr.ed.ac.uk/projects/festival/ (1999/12/01).

[VFVXML] VoiceXML Forum: *VoiceXML 1.0 Specification*, URL, http://www.voicexml.org/specs/VoiceXML-100.pdf (2000/05/10).

[VXMLF] VoiceXML Forum: *VoiceXML*. URL, http://www.voicexml.org/ (2000/05/10).

[W3C] World Wide Web Consortium: *Homepage*. URL, http://www.w3c.org/ (2000/06/01).

[W3CACSS] Lilley C., Raman T.V.: *Aural Cascading Style Sheets (ACSS)*, W3C Working Draft. URL, http://www.w3.org/TR/WD-acss-970630 (1997/06/30).

[W3CCSS2] W3C: *Cascading Style Sheets, Level 2*. URL, http://www.w3.org/TR/PR-CSS2/ (1998/03/24).

[W3CVBC] W3C: *Voice Browser Concept*. URL, http://www.w3c.org/Voice/ (2000/03/17).

[W3CWAI] W3C: *Web Accessibility Initiative*. URL, http://www.w3.org/WAI/ (2000/06/01).

[W3CXML] W3C: *XML: Extensible Mark-Up Language*. URL, http://www.w3.org/XML/ (1998/02/10).

[W3XHTML] W3C: *XHTML: Extensible Hypertext Mark-Up Language*. URL, http://www.w3.org/TR/xhtml1/ (2000/06/01).

[WEN91] Wenzel E.M., Wightman F.L., Kistler D.J.: *Localisation of non-individualised virtual acoustic display cues*. In *Proceedings of the CHI'91 ACM Conference on Computer-Human Interaction* (pp. 351-359). ACM press, New York (1991).

[WEST98] Westall F.A., Johnston R.D., Lewis A.V. (Ed.*): Speech Technology for Telecommunications*. Chapman & Hall, London (1998).

[WIG97] Wightman F.L., Kistler D.J.*: Sound localisation in the presence of multiple distracters*. In *Journal of the Acoustical Society of America,* 101, 3105 (1997).

[WOO98] Woodland P.: *Speech Recognition*. In *Speech and Language Engineering –State of the Art*, IEE Informatics, London (1998).

[X1198] Open Group Inc.: *X11 R6.4*. URL, http://www.x.org/last_release.htm (1998/01/30).

[ZOEBL] Zoebelein H.: *Blinux*. URL, http://leb.net/~blinux/ (2000/03/10).

V. Glossary

ACSS	Aural Cascading Style Sheets
ADE	Auditory Desktop Environment
AI	Artificial Intelligence
ANN	Artificial Neural Networks
API	Application Programming Interface
ASAPI	Advanced Speech API
ASR	Automatic Speech Recognition
AUI	Auditory User Interface
CGI	Common Gateway Interface
CLI	Command Line Interface
COM	Component Object Model
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CVS	Concurrent Versions System
DARPA	Defense Advanced Research Projects Agency
DSP	Digital Signal Processing
DTMF	Dial Tone Multiple Frequency
DTW	Dynamic Time Warping
GNU	GNU is not Unix
GPL	GNU General Public License, open source licensing model
GUI	Graphical User Interface
HCI	Human-Computer Interface
HMM	Hidden Marcov Models
HRTF	Head Related Transfer Function
HTML	Hypertext Mark-Up Language
HTTP	Hypertext Transfer Protocol
I18N	Internationalisation
IC	Integrated Circuit
IID	Interaural Intensity Difference
ISA	Industry Standard Architecture
ITD	Interaural Time Difference
JAVA	Object oriented programming language
JSAPI	Java Speech API
JTAPI	Java Telephony API
JSML	Java Speech Mark-Up Language
JTAPI	Java Telephony API
Linux	Open source operating system, similar to Unix
MIDI	Musical Instrument Digital Interface

NLA	Natural Language Analysis
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
RTSP	Real Time Streaming Protocol
SAPI	Microsoft Speech API
SDK	Software Developers Kit
SRAPI	Speech Recognition API
TAPI	Microsoft Telephony API
TTS	Text to Speech
VocAPI	Voice API
VoiceXML	Extensible Voice Mark-Up Language
VoxML	Voice Mark-Up Language
W3C	World Wide Web Consortium
XHTML	Extensible Hypertext Mark-Up Language
XML	Extensible Mark-Up Language
WYSIWYG	What You See Is What You Get

VI. List of Figures