AN ANIMATION SYSTEM FOR USER INTERFACE AGENTS

Marc Alexa, Uwe Berner, Michael Hellenschmidt, Thomas Rieger

Technische Universität Darmstadt, Interactive Graphics Systems Group, 3D Graphics Computing
Rundeturmstraße 6, 64283 Darmstadt, Germany
Email {alexa, uberner, rieger}@gris.informatik.tu-darmstadt.de
http://www.gris.informatik.tu-darmstadt.de
Fraunhofer Institute for Computer Graphics, Dept. Cooperative HyperMedia Systems,
Rundeturmstraße 6, 64283 Darmstadt, Germany
Email mhellens@igd.fhg.de
http://www.igd.fhg.de/a6

ABSTRACT

With the advent of software agents and assistants, the concept of so called conversational user interfaces evolved, incorporating natural language interaction, dialogue management, and anthropomorphic representations. Today's challenge is to build a suitable visualization architecture for anthropomorphic conversational user interfaces, and to design believable and appropriate face-to-face interaction imitating human attributes such as emotions. The system is designed as an autonomous agent enabling easy integration into a variety of scenarios. Architecture, protocols, and graphical output are discussed.

Keywords*:* Agents, Anthropomorphic Conversational Interfaces, Avatars, Facial Animation, KQML, XML

1. Introduction

Everyday technology, such as office and home equipment, has become more and more powerful – and the corresponding interfaces more and more complex. Consequently, most people have problems to control the various systems and appliances and hardly anybody knows how to access all features of a device. Think for instance of a VCR and its programming features.
One reason for this problem is that traditional interfaces, mostly based on the WIMP metaphor (windows, icons, menus, pointers), are not sufficient to control all aspects of today's technology. One possible way out of this dilemma is to provide an assistance function with a universal, easy to use, and efficient interface.
In this context a vision is that the new human computer interaction paradigm "assistance" is going to replace the so far valid paradigm of the "computer as a tool" [Maes94].
The current goal of our developments are so called *Social User Interfaces,* which are related to social rules in reality, using speech and nonverbal communication methods. Dialogs and other forms of human communication are imitated to make handling a system more natural and personalized, which, in turn, decreases the learning efforts and increases the user-acceptance of the system. This new kind of interface includes the design and control of the conversation, not only to mediate the information, also to establish a relationship with the user. Furthermore the system is adapting on the current user, personal preferences, the history of the conversation, and the current context.
Conversational User Interfaces are representing such an interface and provide complex dialogue structures. Thereby, the use of additional communication channels is tested, new techniques to support nonverbal communication forms like mimics, gesture, and body language are developed. Conversational interfaces evolve beyond natural speech interaction, including nonverbal behavior such as facial expressions and gestures towards new kinds of face-to-face interactions. [CSPC00].
In the context of the focus projects EMBASSI, which faces the above mentioned challenges with the distribution of tasks to several specialized software agents, a unique approach is undertaken to suggest and evaluate prototypes and solutions for face-to-face interaction with virtual characters (here called "user interface agents" or "avatars"1), integrated in traditional interaction concepts. This approach is interdisciplinary and addresses the following challenges:

1. Human factors research towards an academic basis for nonverbal communication: Evaluation of human-human interaction as a

1 Avatar: Originally used as a virtual body representation for a remote user in Internet chats, we extend the term as also being a body for intelligent agents that are part of the computer system.

starting point for the design and generation of human-computer interaction.
2. Technological platform for animated behavior: Lean behavior animation platform with emphasis on real-time interaction and flexibility.
3. Design for appropriate usage and acceptance: Integrated design of face-to-face scenarios regarding human and technical requirements as well as context of interfaces and contents.

2. OVERVIEW

Concrete implementations as a part of conversational interfaces are User-Interface Agents/Avatars, anthropomorphic representatives on the base of artificial 2D or 3D characters. They are representing a human-computer interaction with the explicit presence of emotional aspects contained in every communication using mimics, gestures, and poses.
In contrast to approaches such as Cassell et al. [Cass99], we target at a UI control module appropriate for rendering animated characters with speech output and lip sync on standard PC platforms while communicating over low bandwidth connections. For realization, these technological constraints as well as considerations for future design of successful conversations have been regarded for the conception of a new and flexible Avatar Platform.
The overall human-machine dialogue is controlled by a preceding dialogue manager, which manages all user-interface components and modalities. It also decides on explicit sentences to be generated by a speech synthesis software, and delivers them to the Avatar Platform.
All modules are being designed as agents. The reasons for this decision are discussed in the upcoming section.

3. AGENTS

The theory of agents developed mainly in artificial and shared intelligence. In the following, the features of agents and systems of agents and their benefits for the coverage of shared tasks shall be discussed. Then, we give an overview of this technology for the use in distributed systems, where several independent programs act together on the same task.
An agent, in this case a software-agent, offers an interface for communication with other agents and their environment. It is able to accept orders from other agents or the user. The agent might store an act of communication in an inner representation in order to process it later.
If an agent gets a task, it is able to perform the action autonomously, i.e. independent from outer intervention. Every agent might communicate by means of the interfaces of other agents asking them to work on the given task.
The mentioned autonomy causes an agent to be more than a module for execution of tasks – it can act. An agent not only gets tasks from outside, but can also communicate with other agents or the user. For instance, that it has recognized a designated situation or that it needs the work of other agents to solve a task. Consequently, a cooperation of agents is possible without an outside control system.
For a formal definition of software-agents several variations can be found in the literature, e.g.:

- The FIPA-Foundation [FIPA97, Fini93, LiFi97] defines an agent as the fundamental actor in a system of agents, which offers several services to other agents or the user. Software-agents are programs, which can negotiate and can coordinate the transfer of information.
- IBM uses the term of software-agents for programs, which can be classified through a combination of mobility, intelligence and agent qualities [Brad97].
- According to Pattie Maes [Meas94] autonomic agents are systems, which observe a dynamic environment, in which they are part of, and can act independently with given information to deal with ordained tasks or to reach default goals.

The basic principles of agents can be characterized according to [Jenn97] in the following way:

- Autonomy: Agents act (at least partially) without a direct intervention of other actors and are under full control of their own actions and conditions.
- Reactivation: Agents perceive their environment and react in destined time on their changes (events).
- Determination: Agents do not only act reactively, but also determined.
- Communication: Agents are able to communicate with other agents or their environment to fulfil their tasks.

Furthermore, agents of different functionality and scopes have to run on the same agent-platform to interact with each other.
In our application, processing different tasks of several fields in computer-science is crucial, and we decided to use agent-technology. Therewith, many partners are able to cooperate and can contribute their knowledge to form powerful systems. The Agents we use, deployed in the EMBASSI-project, communicate with each other respectively with their environment using the Knowledge Query and Manipulation Language (KQML) as Agent Communication Language (ACL) in the published version of 1997 ([Fini93] and [LiFi97]).
The declaration of a common standard for an outside language is needed for the communication

of agents with each other and their environment. KQML distinguishes within one act of communication between so called Performatives and Parameters. Performatives destine, what impact a message should have. We use the Performatives *register* and *unregister* to make an agent known within the common agent-platform. After a message is sent with the Performative *connect* an agent is reachable for other agents and the environment to communicate with.
By employment of Parameters an agent is able to send different information within one communication act (message). The Parameter „:sender“ specifies, which agents start off the communication. A reply-id offers the possibility to refer to previous messages and provides self organization. The essential content of the message (communication act) resides in the Parameter „:content“. For this inner communication we use the standard of XML (Extensible Markup Language) [WBLK00].
Through this strategy, combination of KQML as outside language, for the act of communication between agents and between agents and their environment, and XML as the internal language, for the interchange of information, we succeed to segregate the management of messages (communication acts) and the management of the content and information of messages.
By courtesy of an interface for KQML-communication our agents offer the possibility for other agents using the KQML standard to interact with. For instance it is possible to pass information by means of the Performative *tell*. The intrinsic information lies within the Parameter *content* represented by the internal language XML. Using the Performative *achieve*, one implies an order to another agent to perform the content given in XML respectively an agent gets the task to make the message given in XML becomes true.
The described Performatives should only outline the given possibilities of using the outside language KQML in combination with XML as internal language.

4. XML

XML is selected for conveying the information among the modules of our system. Other possible choices might be derived from languages being used in the AI community such as LISP-S expressions. An advantage of expressions is their integration into the supporting language. However, in our context it is more important to be independent of proprietary solutions and specific platforms, ideally using deployment of web services. XML is, therefore, the natural choice as it is widespread, independent of any platform, and supported by most web enabled services. Note that KQML and XML as such are completely independent and are handled by different parts of the modules.
For an application in the Internet an appropriate XML page is changed into a stream and passed on to the XML parser. For an application in an agent system the intrinsic message is embedded in XML in the KQML content.

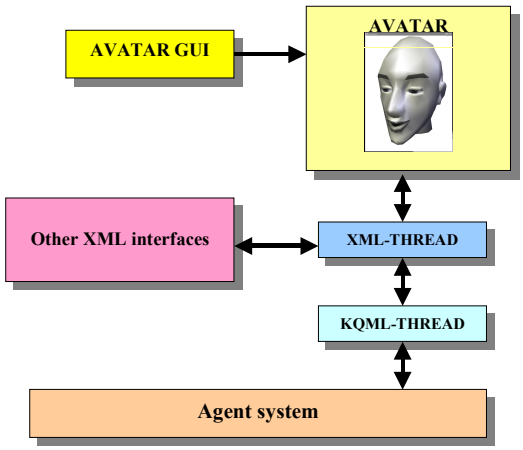

Figure 1: architecture picture with XML and KQML thread.

The whole communication with other components operates in this way.
We have developed a set of XML tags to control the Avatar. The defined instructions are together chamfered in a DTD (Document Type Declaration). This requires a clear separation between the KQML and the XML parser.
Using XML tags the Avatar is *adjusted*. These adjustments are valid until they are set again, or the Avatar is terminated.
It is not necessary to do a setup for an Avatar. However, if several Avatars are among the agents they have to be set up to assign unique identifiers to each of them.
The most important tags are the following:

<Setup>…</Setup>
Starts and terminates the setup. All setup adjustments must be made within this area.

<AvatarBehavior *name*="helper"/>
Set the Avatar behavior name as an identifier. The name can be arbitrary, since no analysis is made. However, the name has to be unique.

<Emotion_System *category*="happy"/>
Set Avatar emotion. The emotion influences the behavior of the Avatar. A merry Avatar is polite and indulgent. Note that an Avatar should change the mood - otherwise it appears as unnatural.

<Coordinates *x*="300" *y*="200"/>

Set Avatar direction. The line of sight can be directed on an interlocutor or towards any article.

<DL *speechact*="message_greeting"> </DL>
Set Avatar context. The *speech-act* is of particular importance here. The speech-act conveys the current system intention like greetings, say goodbye, to approve or to provide information.

Tags outside of the setup:
<Execute>…</Execute>
Avatar executes the content. That could be saying a record, to nod or a shake the head. An ID might be used, which would refer an internal message.

<Turntaking *direction*="to_user"/>
Set turn-taking direction to user or to system. A turn-taking increases the attention of the Avatar and mediates to the Avatar to seize or deliver the word.

The most important part of the communication lies in the speech-acts [AuLL88], since they influence the overall behavior of the Avatar. In human communication every mimic or gesture depends on the context, the preceding dialogue, and underlying assumptions. Speech-acts define this context for the avatar. In this way we aim at making the avatar more understandable.
By combination of *speech-acts* and the *emotion* tag a multiplicity of possible behavior patterns may be specified. A speech-act is divided into a status and an internal message. The status is provided as a parameter of a certain internal message. The internal message: *message_inform* can possess the following status.

- *[status: warning]*
- *[status: busy]*
- *[status: idle]*
- *[status: error]*
- *[status: ok]*
- *[status: failed]*
- *[status: offer]*

Further important speech-acts in a discussion between two persons are:

message_accept
Is used in cases of agreement.
message_reject
Is used in cases of denial, whereby a fact is called false.
message_command
Can be used for suggestions or requests.
message_cancel
Mediates a break using turn-taking.
message_acknowledge
Acknowledges the successful processing of a job.
message_correct
Corrects a false acceptance of the user.
query_input
Is used if the system expects any inputs from the user, normally concerning certain parameters.
query_y/n
Produces an expectation for a simple yes or no response.
query_selection
Implies a listing of alternatives, like film titles for example.
query_request_acknowledge
Is used in case of uncertainty during the language analysis or the processing of the user intention. The cause might be incorrect voice recognition and/or repeated misunderstandings between users and system.
query_request_repair
Is likewise meant for false recognition results or corrections, which were stated by the user. Repair strategies are repetitive demands to the user or rigid yes/no questions.

The speech-acts represented above are partly based on the META dialog hierarchy. Additionally, the same speech-acts are introduced for the opposite direction, i.e. for description of the user's intentions.

5. AVATAR-PLATFORM

The avatar platform represents the graphical user interface agent. By means of a shell it is represented as an autonomous agent inside the conversational UI. Commands delivered to the avatar are processed in different units. An overview of the avatar platform is given in Figure 2.

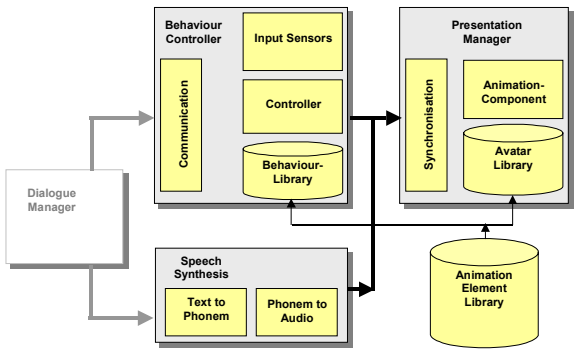

Figure 2: Architecture of the Avatar Platform (with preceding Dialogue Manager)

This platform consists of the following components:
Presentation Manager: This module provides the functionality to present animated artificial

characters, to perform facial animations, and to achieve lip sync. Animated characters are represented in a structure conforming to H-ANIM [H-AN98] for a standardized and easy access of body parts and joints. H-ANIM joints are augmented by a facial structure based on Morph Targets and efficiently realized as a Morph Node (Alexa et. al. 2000) [AlBM00]. Hereby, a wide space of facial expressions can be provided, while the parameterization of the face is kept simple and can be realized with just a few key values. Key-frames defining the state of the animated character may consist only of a small number of such values and playback is possible even over low-bandwidth networks and on small portable devices. Moreover, facial animations can be easily mapped to different geometries further extending the possible choices for the User Interface Designer. In this way, photo-realistic avatars or comic-like characters can be displayed and animated. Figure 3 presents a more comic-like character model.

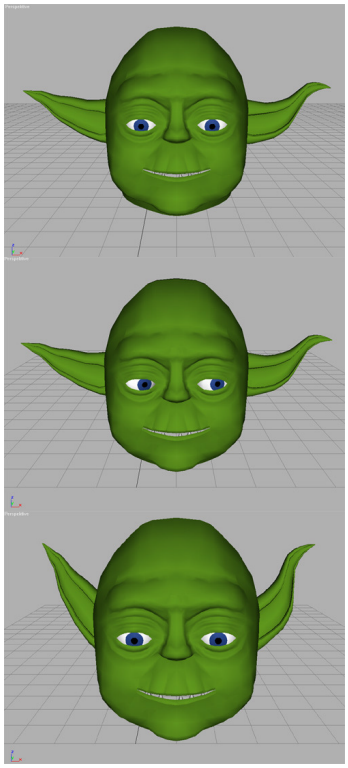

Figure 3: Comic-like character

Behavior Controller: While the Presentation Manager allows for the control of the avatar on the fundamental geometric levels, the Behavior Controller provides an interface on a more abstract level. Tasks and even motivations can be specified and the corresponding actions are performed automatically. Examples for such actions are gestures and movements of the avatar, but also more complex facial expressions with temporal aspects. In addition, behavior patterns for specific motivations or moods can be activated. This corresponds to a sometimes rule-based, sometimes non-deterministic activation of behavior elements. Possible applications for this are simple Avatar actions (e.g. accidental looking around or smiling) to avoid repetitive behavior, which is easily detected as artificial by the human observer. Another possible application are emotional expressions to emphasize systems states, such as a puzzled look in case of an unexpected user input.

One goal is to make the avatar more natural and less artificial or stiff. Even if the underlying application layer does not invoke any action the avatar presents randomly generated behaviors like eye blinking or head movement. The different animation sequences representing an explicit gesture or an unconscious behavior are synchronized via a blackboard mechanism, which also controls the appearance of the emotional state of the avatar.

Speech Synthesis: An important requirement for the Avatar to achieve a realistic appearance is speech output with lip-sync. Since the dialogue with the user is not known in advance, prerecorded animation sequences with lip animation can not serve as a solution. Instead, a real-time lip sync with the audio output generated by a speech synthesis module have to be realized. For this, the speech synthesis system is expected to generate phoneme information, which will be mapped to appropriate visemes. Phonemes are communicated using the international standard SAMPA [SAMP95]. The mapping to visemes or sequences of visemes is based on timing and frequency information available in SAMPA. Heuristics are used to also generate nonverbal facial expressions from phoneme information. This concept offers several advantages:

Lip-sync is realized automatically assuming that speech synthesis works in real time.

Interactive applications with speech generation during and based on the interaction are possible. This is in strong contrast to systems based on a set of pre-generated animations.

A transparent system working with a speech synthesis for several languages is relatively easy to implement.

New developments in the area of speech synthesis systems are readily usable. Even new features of such systems seem to be easy to include.

A general (system independent) problem is the fact that most commercial speech synthesis systems are not open. This means the intermediate phoneme level is not accessible from outside. In our prototype implementation we use MBROLA [MBRO99], which is the result of an EU project and publicly available.

We are also trying to emphasize the content of the speech with gestures and mimic. For this purpose we analyze the prosody of the text based on the generated phonemes. Depending on the emphasis in the spoken text the avatar performs supporting actions such as moving the eyebrows and the eyelids.

To give a better impression of the existing system some screenshots will be given. Figure 3 depicts a selection of frames of one animation sequence generated in real-time, showing the overlay of general facial expressions with the lip-sync information produced from a speech system. At the beginning the avatar is speaking an "o", then smiles, and at the end closes the eyes.

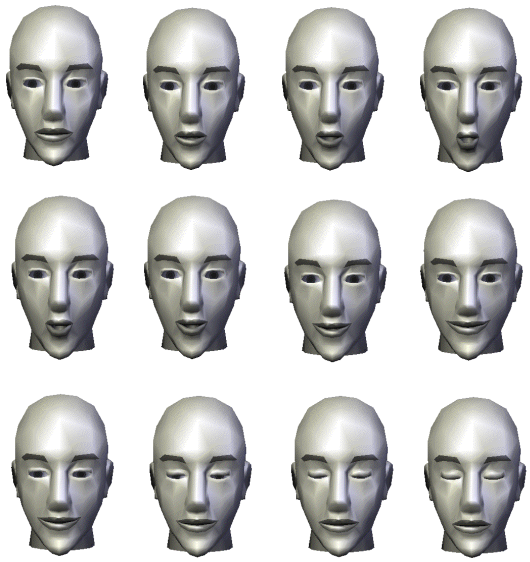

Figure 4: Animation Sequence produced by the system

Figure 5 shows a screen shot of the system. On the left side one can see some sliders to define the values of the basic geometries. Combining this values leads to different facial expressions. The values can be stored as a key frame (upper window). Several key frames comprise an animation sequence of the avatar, which can be played and refined interactively. Some pictures of a resulting animation are shown in Figure 2. Another possibility is to type a sentence in the text field and after pressing the speak button the text will be spoken with a corresponding facial animation.

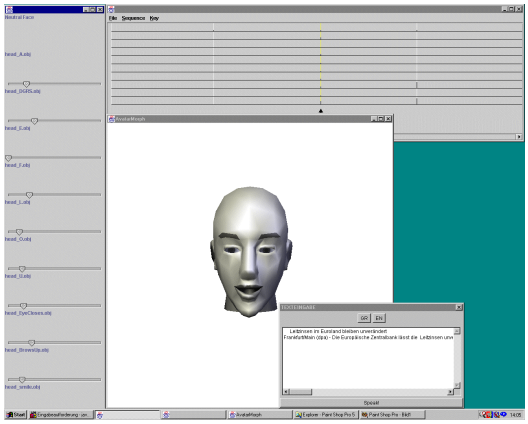

Figure 5: Screenshot Avatar System

6. FIELDS OF APPLICATION

In principle our avatar system is a supplement for the traditional point-and-click interfaces. The usage of the User-Interface Agents is very interesting for systems with one-time or inexperienced users. These people can use techniques of natural communication to use an unknown system.
An example application developed at GRIS (Interactive Graphics Systems Group) is a virtual newsreader for web-sites (see Figure 6).
Besides providing a general-purpose software platform and players for Avatars and User-Interface agents, we are also supporting the application-specific design of such conversational user interfaces.

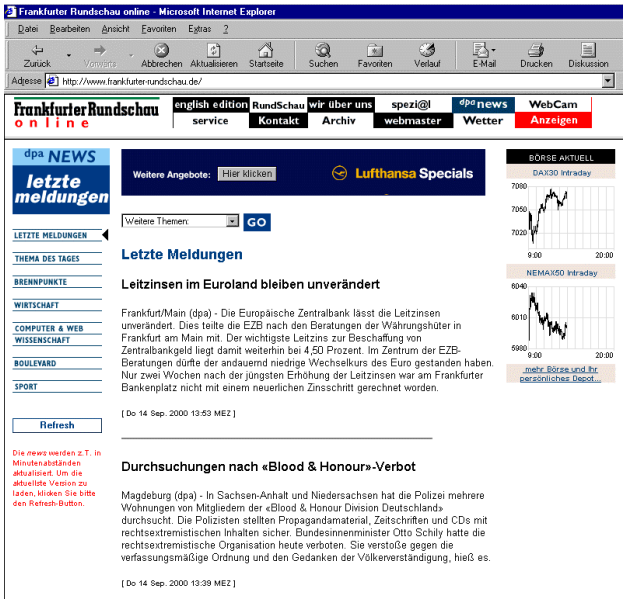

Figure 6: Screenshot of the virtual newsreader with the corresponding web page

7. CONCLUSION

We have presented an agent-based system to enable natural user interactions in a variety of environments. The implementation is based on KQML and supports the performatives *register*, *connect*, *unregister*, *disconnect*, *tell*, *achieve*, *sorry* and *error*. A connected Avatar is informed about the system state with *tell* and performs actions communicated with *achieve*.
However, KMQL is only needed within a router based agent system. It is also possible to send the only the XML tags using other protocols such as HTTP. In contrast to other user-system or system-system communication applications our implementation provides context analysis based on speech acts.

8. FUTURE WORK

A different development goal is the use of simple psychological models to control the behavior of the avatar depending on the emotional state. This will modify for example the amount of head movement, the lifting of the eyebrows, the presented pupil size, the duration of an excitement, or the prosody of the spoken text. Also, the general unconscious behavior of the avatar will be controlled by a database of characters. With respect to the character chosen by the user the avatar will react in a slightly different manner.

9. ACKNOWLEDGEMENTS:

This work was partially funded by the German "Bundesministerium für Bildung und Forschung, BMB+F through the Focus Project EMBASSI (BMB+F-No. FKZ 01 IL 904 U8) [http://www.embassi.de]. We would like to thank all project partners for their helpful ideas and support, especially Thorsten Herfet, Thomas Kirste, and Gary Bente.

10. BIBLIOGRAPHY

[AlBM00] Alexa, M., Behr, J., Müller, W. (2000). The Morph Node. Proc. Web3d/VRML 2000, Monterey, CA., pp. 29-34

[AuLL88] Esa Auramäki, Erkki Lehtinen and Kalle Lyytinen "A speech-act-based office modeling approach", ACM Transactions on Information Systems, 6,2, 1988

[Brad97] Bradshaw, Jeffrey M.: An Introduction to Software Agents, Published Paper, 1997

[CSPC00] Cassell, J., Sullivan, J., Prevost, S., Churchill, E. (ed.): *Embodied Conversational Agents*, MIT Press, Cambridge, MA 2000

[Fini93] Tim Finin, et al, „Draft: Specification of the KQML Agent-Communication Language“, DARPA Knowledge Sharing Initiative, Jun 1993.

[FIPA97] Specification of the Foundation for intelligent, physical Agents, FIPA, 1997

[H-AN98] Humanoid Animation Working Group (H-ANIM). http:// ece.uwaterloo.ca:80/~h-anim/, 1999

[JeWo97] Jennings, Nicholas R., Wooldridge, Michael J.: Agent Tochnology, 1. Edition, Springer Verlag 1997

[LaFi97] Yannis Labrou and Tim Finin, „A Proposal for a new KQML Specifiaction“, Technical Report TR CS-93-03, February 3, 1997

[Maes94] P. Maes, Agents that Reduce Work and Information Overload. Comm. of the ACM, 37 (7). July 1994

[MBRO99] The MBROLA Project – Towards a Freely Available Multilingual Synthesizer, http://tcts.fpms.ac.be/synthesis/mbrola.html, 1999

[SAMP95] SAMPA - Computer readable phonetic alphabet, http://www.phon.ucl.ac.uk/home/sampa/home.htm, 1995

[WBLK00] Tim Weitzel, Dr. Peter Buxmann, Frank Ladner, Prof. Dr. Wolfgang König "Konzept und Anwendungen der Extensible Markup Language", 2000