Semi-Automatic Generation of Dialogue Applications in the GEMINI Project *

Stefan W. Hamerich, Volker Schubert, Volker Schless
TEMIC Speech Dialog Systems, Ulm, Germany
{stefan.hamerich|volker.schubert|volker.schless}@temic-sds.com

Ricardo de Córdoba, José M. Pardo, Luis F. d'Haro
Grupo de Tecnología del Habla, Universidad Politécnica de Madrid, Madrid, Spain
{cordoba|pardo|lfdharo}@die.upm.es

Basilis Kladis, Otilia Kocsis
Knowledge S.A. (LogicDIS group), Patras, Greece
{bkladis|okocsis}@logicdis.gr

Stefan Igel
Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW), Ulm, Germany
sigel@faw.uni-ulm.de

Abstract

GEMINI (Generic Environment for Multilingual Interactive Natural Interfaces) is an EC funded research project, which has two main objectives: First, the development of a flexible platform able to produce user-friendly interactive multilingual and multi-modal dialogue interfaces to databases with a minimum of human effort, and, second, the demonstration of the platform's efficiency through the development of two different applications based on this platform: EG-Banking, a voice-portal for high-quality interactions for bank customers, and CitizenCare, an e-government platform framework for citizen-to-administration interaction which are available for spoken and web-based user interaction.

1 Introduction

GEMINI[1] exploits experience gained from previous projects (see e.g. (Ehrlich et al., 1997; Lehtinen et al., 2000)) and from real-world use of similar systems, to create a generic platform for the development of user-friendly, natural, high quality, intuitive, platform independent and multi-modal interactive interfaces to a wide area of databases employed by information service providers.

The main idea of GEMINI is that, given a database, a description of its structure and how to access the data as well as a list of the kinds of requests the user may make, the system should be able to automatically generate the necessary dialogue scripts to run the service. In a sense, this is exactly what a human call center agent does when being trained for the job. Within the project we strive to get as close as possible to this ideal.

Specifically, the application generation platform of the GEMINI project contains generic dialogue components available for adaptation to new services and languages. Thus, generation of multilingual and multi-modal interfaces is achieved by incorporating the lexical and semantic relations of the databases contents, reducing the development time and facilitating the system's maintenance and transportability to different applications and languages. Furthermore, the platform enables a high degree of personalisation (i.e. user modelling, speaker verification, etc.).

This paper is organised as follows: First we describe the application generation platform (AGP) of the GEMINI project. Afterwards we introduce the two pilot applications developed with our platform. Next we compare our approach with other proposals made by different research groups. Finally we conclude our major findings.

2 Application Generation Platform

The main target of the GEMINI project is the development of a platform for generating interactive, multilingual and multi-modal dialogue interfaces to databases with a minimum of cost and human effort. The AGP is an integrated set of assistants to generate multi-modal dialogue applications in a semi-automatic way. Its open

*This work was partly supported by the European Commission's Information Society Technologies Programme under contract no. IST-2001-32343. The authors are solely responsible for the contents of this publication.

[1]Refer to the GEMINI Project Homepage on www.gemini-project.org for further details.

and modular architecture simplifies the adaptability of applications designed with the AGP to different use cases. Connecting to a different database, adding a new modality or changing a scripting language can be achieved by adding or replacing the appropriate component without touching the other aspects of dialogue design again.

The AGP consists of assistants, which are tools (partly with a GUI) producing models. All these models generated within the AGP are described in GDialogXML (GEMINI Dialog XML), which is an object-oriented abstract dialogue modelling language. It was created during GEMINI for use with the AGP. See Figure 1 for an example of the GDialogXML syntax. For a detailed description of GDialogXML refer to (Hamerich et al., 2003).

```
<Var id = "xPersonName">
   <xType><Type refr = "String"/></xType>
</Var>

<Var id = "xPersonList">
   <xType><Type refr = "List">
      <xItemType>
         <Type refr = "ObjEmbed"/>
            <xClass><Class refr = "Person"/></xClass>
      </xItemType></Type>
   </xType>
</Var>
```

Figure 1: *Definition of variables in GDialogXML*

All models in the AGP may be saved as libraries for future applications.

As shown in Figure 2 the AGP is not supposed to complete its task without any human interaction. This is because there will always be different ways for retrieving specific information. Consequently, the designer of dialogue applications has to select the preferred flow of dialogue manually by confirming the proposals of the AGP components. Most of these operations are simply drag & drop actions between various windows that contain all relevant fields, which are automatically created from the previous tools of the platform.

2.1 AGP Architecture

All components of the AGP are integrated into one framework. This eases the use of the platform and enables the designer to switch back and forward to different tools in case she or he wants to add or modify certain dialogues.

In Figure 2 the architecture of the AGP is illustrated. The whole AGP consists of three layers. These layers are described in more detail in the following sections.

2.1.1 Framework Layer

The framework layer is the first layer of the AGP (refer to Figure 2). It includes the application description assistant (ADA), the data modelling assistant (DMA), and the data connector modelling assistant (DCMA). As indicated by the black arrow in the upper left corner of Figure 2, all assistants are controlled manually.

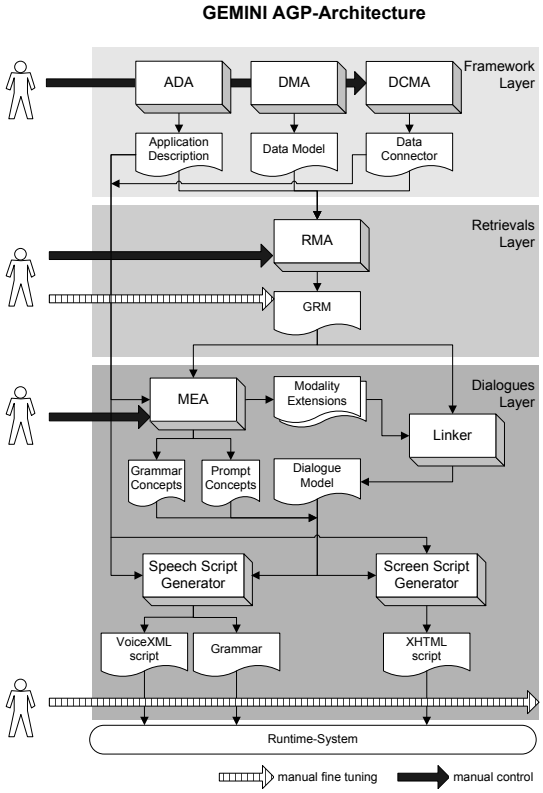

Figure 2: *Schematic view of the AGP architecture.*

The designer has to provide the application description, which mainly consists of the modalities for which the AGP should generate dialogue scripts, the languages for which the dialogues should be available, the dialogue strategy for the resulting system, some settings for error handling and a rough application description containing the major dialogue steps and their respective slots.

The DMA helps creating the data model, which consists of class descriptions. Also, the attributes and elementary types of the data are specified here. In this process, the GUI guides the designer, and there is the possibility to load libraries of previously created classes.

Furthermore the DCMA helps creating APIs and implementation references for application specific data access functions.[2] These functions could then be used in the runtime system without any knowledge of the existing database.

[2]The implementation of data access functions has to be done outside of the AGP context, since special knowledge about the database itself is needed for this.

2.1.2 Retrievals Layer

The retrievals layer (shown as the second layer in Figure 2) mainly consists of the retrieval modelling assistant (RMA). This layer is modality and language independent, therefore no language or modality specific data is included here.

The designer uses the RMA to create the abstract dialogue flow. It provides a user-friendly interface where the design process is accelerated. Two main sources of information are used to automate the process: the data model and the data connector. Using the information in the data model, several dialogues are automatically generated: (1) candidate dialogues for attributes that the user should be asked for (we call them 'get information dialogues') and (2) another dialogue where that specific attribute is presented by the system ('say information dialogues'). At the same time, all procedures from the data connector are available to the designer, who can drag & drop any of the dialogues mentioned so far.

In the ideal situation, where a dialogue only depends on items from the data model, it can be modelled with just three drag actions: (1) drag & drop a get information dialogue, (2) drag & drop a call to the database (from the data connector), and (3) drag & drop a say information dialogue. All the values exchanged by these three functions are assigned automatically by the assistant, so the designer just has to press 'Accept' for all assignments.

When the dialogue depends on data not contained in the data model (as questions to the user that do not correspond to an object from the data model), the designer can use a set of four different types of dialogues: dialogue based on user input / on a variable / on a sequence / on a loop. In all of them, conditional, switch-case and loop constructs can be inserted. So, the designer has both automation and a great flexibility in dialogue design.

The resulting output is called generic retrieval model (GRM), which consists of the modality and language independent parts of a dialogue, which is mainly the application flow. The GRM is modelled in an object-oriented way using GDialogXML and mainly consists of dialogue modules. A dialogue module can call other modules as subdialogues or can jump to another top level module. This way, the application flow of dialogues in GDialogXML is modelled.

As indicated by the dashed arrow, it may be necessary to do some manual fine tuning on the GRM, as the complexity of the RMA depends on the application and may be rather high and often there exist several ways to implement the application.

2.1.3 Dialogue Layer

The dialogue layer is modality and language dependent as now the modality extensions from the modality extension assistant (MEA) are added to the retrieval model. In the extension files the input and output behaviour of an application is described for a specific modality. The current implementation of the AGP supports the generation of voice (speech modality) and web-based applications (web modality). For the speech modality the extensions consist of links to grammar and prompt concepts, which are language and modality independent. For each language, there is a separate concept file, containing the wording for the prompts and the names of the grammars used. Additionally the modality extension consists of special subdialogues which are specific for one modality only.

All grammars and prompts of the AGP are handled in a global library, which eases the quick and easy reuse of several components.

The GRM is enriched by the modality extensions in the Linker. The resulting model is called dialogue model, which is processed by the speech script generator and/or the web-page script generator depending on the selected modalities in the application description. For the speech modality VoiceXML scripts with some additional CGI scripts are generated. The grammars are taken from the AGP grammar library or have to be generated with the MEA. For the web modality a web-page script is generated out of the dialogue model which enables dynamic web pages.

For the speech modality, some more tools are relevant, namely the language modelling tool and the vocabulary builder.

To have the runtime system ready for use, little effort has to be spent on manual fine tuning again. For example the recogniser dependent settings have to be adjusted for the VoiceXML platform.

2.2 Implementation of the AGP

The initial prototype of the AGP of the GEMINI project was finished in summer 2003. This version's architecture is shown on Figure 2. In spring 2004 an extended and improved version of the AGP will be implemented. This version covers additional features like mixed initiative dialogues with over-answering, advanced user-modelling, natural language generation, and language-identification. As well, multilingual dialogues are possible with this final version.

All platform components have been implemented using Qt. Due to this fact, the AGP is applicable on different operating systems.

3 Applications

Two pilot applications have been generated using the AGP for evaluation and validation. All these applications are generated in a very user friendly way, taking into account the automatic multi-modal error handling capabil-

ities of the AGP, refer to (Wang et al., 2003) for more details about the error handling in GEMINI.

3.1 EG-Banking

The voice banking application called EG-Banking application constitutes a voice portal for user-friendly, high-quality interactions for bank-customers. The main functionality of EG-Banking includes a general information part (covering credit cards, accounts, loans infos) available to the public and a transaction part (covering account flow, account balance, statements, etc.) available to customers of the bank only. The multi-lingual application is accessible via a cellular or fixed network telephone.

A manually refined version of the generated application is installed at Egnatia Bank in Greece and is used as a commercial product for phone banking.

3.2 CitizenCare

CitizenCare is an e-government dialogue system for citizen-to-administration interaction (via multiple channels like internet and public terminals), filled with content for an exemplary community. The main functionality is an interactive authority and information guide, providing different views like an administrative view, based on the hierarchical structure of the authorities, and a concern-oriented view, giving the citizen all the information needed to make use of services offered by public administration authorities.

4 Comparison to Other Approaches

The GEMINI approach for setting up new dialogue applications differs in a lot of points from other proposals. In this section we compare our AGP with other existent approaches.

Compared with the REWARD system from (Brøndsted et al., 1998) the GEMINI AGP allows the generation of dialogues for several modalities. Additionally in GEMINI we generate dialogues in standardised description languages (VoiceXML and XHTML), so we have no need to develop a special runtime system. As done for the REWARD system, we focused a lot on reusability.

In (Polifroni et al., 2003) a rapid development environment for speech dialogues from online resources was described. The development process there first takes knowledge from various web applications and composes a database from it. This is one of the differences to our approach. Our AGP requires a filled database and allows the development of speech and web applications from it. Because of this, we do not need to extract any knowledge, which makes the GEMINI approach more domain independent. Another important difference is, that the speech dialogue applications generated by the AGP will be implemented in VoiceXML, which allows the generated dialogues to be executed with every VoiceXML interpreter.

5 Conclusion and Future Work

In the GEMINI project we aim at the design and implementation of an application generation platform, which generates state of the art speech and web applications. The platform architecture is open to generate multilingual dialogue applications from different databases in several modalities. We can consider the platform a success, as we have streamlined the design process of the applications thanks the help of our assistants. The use of standards (e.g. VoiceXML) places us in a good position in the market of voice applications.
To facilitate the communication between all modules, an abstract dialogue description language, called GDialogXML, was defined, which is another important result from the project.

Future work will cover the realisation of the improved AGP, which will allow multi-lingual applications with mixed initiative, overanswering, and user modelling. Additionally a graphical control flow will be available to the designer. Furthermore the AGP will be evaluated against other approaches of dialogue design.

References

T. Brøndsted, B. N. Bai, and J. Ø. Olsen. 1998. The REWARD Service Creation Environment, an Overview. In *Proceedings ICSLP*, pages 1175–1178, Sydney, Australia.

U. Ehrlich, G. Hanrieder, L. Hitzenberger, P. Heisterkamp, K. Mecklenburg, and P. Regel-Brietzmann. 1997. ACCeSS - Automated Call Center through Speech Understanding System. In *Proceedings EUROSPEECH*, pages 1819–1822, Rhodes, Greece.

S. W. Hamerich, Y.-F. H. Wang, V. Schubert, V. Schless, and S. Igel. 2003. XML-Based Dialogue Descriptions in the GEMINI Project. In *Proceedings of the 'Berliner XML-Tage 2003'*, pages 404–412, Berlin, Germany.

G. Lehtinen, S. Safra, M. Gauger, J.-L. Cochard, B. Kaspar, M. E. Hennecke, J. M. Pardo, R. de Córdoba, R. San-Segundo, A. Tsopanoglou, D. Louloudis, and M. Mantakas. 2000. IDAS: Interactive Directory Assistance Service. In *Proceedings of the international Workshop 'Voice Operated Telecom Services'*, pages 51–54, Ghent, Belgium. COST 249.

J. Polifroni, G. Chung, and S. Seneff. 2003. Towards the Automatic Generation of Mixed-Initiative Dialogue Systems from Web Content. In *Proceedings EUROSPEECH*, pages 193–196, Geneva, Switzerland.

Y.-F. H. Wang, S. W. Hamerich, and V. Schless. 2003. Multi-Modal and Modality Specific Error Handling in the GEMINI Project. In *Proceedings of the ISCA Workshop on 'Error Handling in Spoken Dialogue Systems'*, pages 139–144, Chateau d'Oex, Switzerland.