Developing Interactive Voice Response Interfaces for Large Information Systems

Gunar Fiedler
Institute for Computer Science and Applied Mathematics,
Kiel University
Olshausenstr. 40, 24098 Kiel,
Germany
Fiedler@is.informatik.uni-kiel.de,
www.is.informatik.uni-kiel.de/~fiedler

Peggy Schmidt
Institute for Computer Science and Applied Mathematics,
Kiel University
Olshausenstr. 40, 24098 Kiel,
Germany
pesc@is.informatik.uni-kiel.de
www.is.informatik.uni-kiel.de/~pesc

Abstract : *The usage of mobile technology for eGovernment applications is not only restricted to (hyper-) text or multimedia services like WAP, SMS, or MMS. Nowadays, due to various improvements in speech technology it is practicable to use natural speech as an input source for information systems. Typically, when using this interface technology it will coexists with other ones within an enterprise environment. While a well founded modeling of information structures is widely accepted during information system development interface applications are often designed ad hoc. Because information systems evolve this ad hoc procedure complicates the management of application versions and variations. Especially in eGovernment scenarios with changing legal conditions or political tenors a structured development and change management of multi channel interfaces can reduce costs for IT projects without reducing functionality or public acceptance.*
We present a methodology for a structured interface development based on the website description language SiteLang. The design of this methodology was heavily influenced by our experiences in several co-operations with German public administrations. One of these projects, the community management system SeSAM, is used as a running example within this paper.

Keywords: interface development, SiteLang, information system, VoiceXML

1. Introduction

Speaking about information systems in mGovernment includes the discussion of using phone technology based on natural speech processing as an input source. Processing natural speech is a complex task. That's why only a few methodologies for a structured development of such interfaces exist (see Schwanzara-Bennoit, 2004). Influenced by our web site development projects, e.g. by our eGovernment project SeSAM introduced in section 1.2 we designed a development methodology for voice applications on the basis of the website modeling language SiteLang. Section 1.1 gives a short overview over natural speech and phone technology together with important restrictions for voice enabled systems in opposite to graphical interfaces. Section 1.2 introduces the community platform SeSAM as the running example of this paper. Section 2 describes the modeling process of voice applications, introduces SiteLang and explains how this language can be used for modeling voice based interaction. Section 3 shows how SiteLang specifications can be automatically translated to executable code either for running simulations or for creating consistent source code for multi-channel systems.

1.1. Interactive Voice Response Systems

Interactive Voice Response Systems differ from classical HTML based web interfaces since they use natural speech and DTMF[1] input instead of graphical manipulations or text input within a web browser. Processing natural speech is a complex process and still an open field of research. Typically, a speech enabled system consists of the following components:

- a speech recognizer, which converts acoustical input into text strings,
- a speech analyzer for extracting the meaning of the recognized text,
- a dialog controller that performs predefined actions based on the extracted commands,
- and a speech synthesizer that generates acoustical output from the system's answers.

There exist many different implementations on the market especially for speech recognizers, speech analyzers, and speech synthesizers. An overview can be found in (Schwanzara-Bennoit 2004). In this paper we focus on developing the dialog controller. We do not consider speech recognition or speech synthesis although we have to take care about their restrictions.

In a normal man machine conversation the partners form dialogs by changing their activities between speaking and listening. An interface designer has to model these dialogs by specifying possible user commands and the corresponding system answers. Several languages with different paradigms were defined for this task. VoiceXML is a W3C recommendation for defining dialogs featuring synthesized speech, digitized audio, recognition of spoken commands, DTMF input, recording of spoken input, telephony, and mixed initiative conversations. The architectural model of VoiceXML applications is depicted in figure 1.

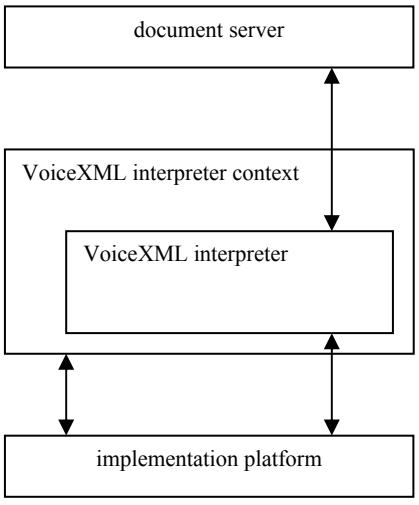

Figure 1: Architectural Model of VoiceXML Applications

The VoiceXML interpreter acts like a common web browser and plays the role of an interface between the internet and the telephony environment. The interpreter is embedded in a context responsible for administrative tasks like detecting incoming calls. Dialog definitions are stored as VoiceXML (.vxml) files on a document server, typically a normal HTTP enabled web server. The interpreter controls the implementation platform during its reactions on the users input.

[1] DTMF (Dual Tone Multi Frequency) – Pressing a key (0-9,*,#) on a telephone produces a sequence of two tones with well defined frequencies.

In the last years various improvements in speech technology can be found. Nevertheless, speech enabled applications suffer from some problems when compared to graphical companions:

- *Finances:* Telecommunication hardware, speech recognition components, VoiceXML browsers, and Text-to-Speech (TTS) voices cost a lot of money.
- *Naturalness:* Calling users expect to talk to a human.
- *Reliability:* Speech recognition components are not as good as human call center agents when dealing with unexpected situations.
- *Patience:* Speech based communication is always in "real time". It takes a lot of time to listen to the system. This is also a problem within the development cycle because debugging an application in real time is less efficient. That's why visual development methodologies are necessary.
- *Navigation:* The human perception is very limited as well as the input facilities. This causes deep navigation structures through the application space.
- *Limited Conversation:* VoiceXML applications can only handle sentences and commands they are programmed for.

1.2. Application Scenario

In co-operation with the city administration of Cottbus we developed the eDemocracy portal SeSAM which enables politicians as well as interested citizens to be always up-to-date in current parliamentary discussions. Depending on his/her rights the user has a role based access to the data of the members of parliament, the committees, or parties. He/she can read bills and motions that are currently under discussion and he/she can retrieve agendas of certain meetings. Additionally, communication facilities are provided. A detailed description of the SeSAM functionality can be found in (Fiedler et. al., 2004).

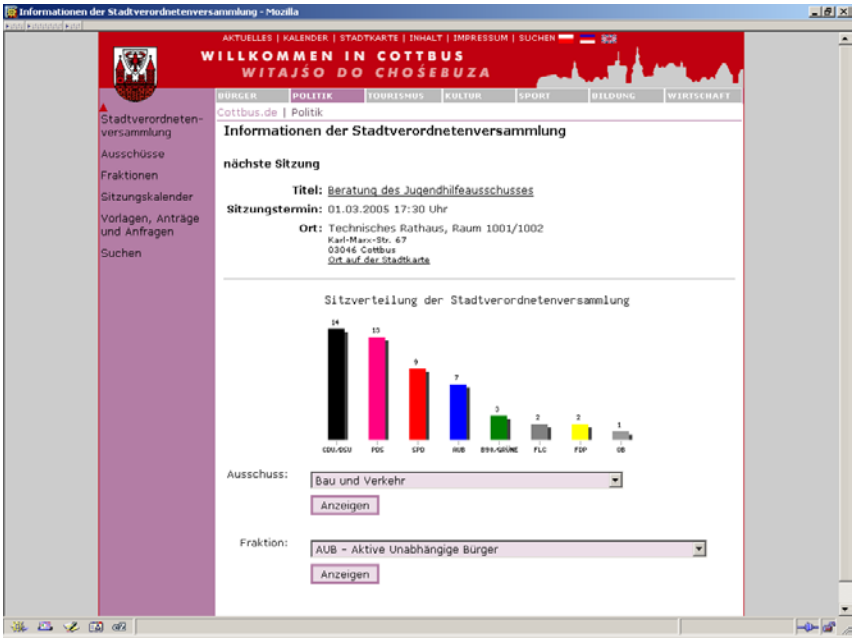

Figure 2: Screenshot of the Public Accessible Part of SeSAM (www.cottbus.de).

The main interface for SeSAM is classically web based as shown in figure 2. Additionally to this web interface we developed a prototype for a voice based access to the SeSAM functionality. Interesting use cases for such an interface is communication and coordination support for traveling parliamentarians. With

a simple phone call they can track changes of appointments or agendas; they can fetch personal messages, or send messages in a role based way.

2. Modeling Interactive Voice Response Systems

A general framework for developing and visualizing interactive voice response systems was presented in (Schwanzara-Bennoit & Fiedler, 2004). Beginning with a general system and process analysis the developer derives significant use cases. For each use case a finite state machine (FSM) representing the navigational structure of the corresponding dialog sequence is created. The states of this FSM are enhanced by HTML representations of typical dialog elements like spoken text or menus. This static visualization allows simulations of the dialog structure. In terms of rapid prototyping the designer can decide in this very early stage whether his concept is suitable or not. This static prototype is then connected to real business data by creating views over the applications database. This dynamic prototype allows logical simulations to decide whether the dialog structures can handle real data. The dynamic prototype can be transformed in a nearly one-to-one mapping to the resulting VoiceXML application for further timing simulations.

This framework defines only a general procedure without any suggestion for specific modeling languages. A designer familiar to UML can use UML use cases and activity diagrams for deriving the navigational structure. (Illustrated) Statecharts may be an alternative, too. In principle, every modeling language is suitable, if

- the language supports the representation of finite state machines,
- the language has a well defined semantics at least within the development team.

In this paper we use the modeling language SiteLang which integrates use case modeling and interaction specification. Due to the formalized SiteLang semantics SiteLang models can be easily used as executable specifications for various target platforms (HTML, VoiceXML, WML, etc.) in parallel.

2.1. SiteLang

SiteLang was introduced in (Thalheim & Düsterhöft, 2001) and (Thalheim, 2003). It uses storyboard and staging concepts to model the interaction between several actors. An actor is defined as a group of system users so “actor” corresponds to the commonly known concept of “roles”.

The main flow of control is represented by *scenes* and transitions between scenes forming *scenarios*. Every scene is associated with a content type and actors that are active within the scene. The content type represents the connection to the business data that is used within the scene. It defines a complex view over the applications database together with possible functionality.

A scene can be refined by *dialog steps*. A dialog step represents an atomic unit within the interaction. It is associated with a set of actors that can perform this dialog step. A dialog step is characterized by a precondition that must hold before the dialog step can be performed, a postcondition that must be fulfilled to finish the dialog step, and transactions that are processed during the dialog step. Additionally, every dialog step is associated with a representation of the system’s state in the target platform after the transactions of the dialog step are performed. This representation is called “*dialog*”.

Beside the algebraic specification there are graphical representations of SiteLang constructs. In scene diagrams, scenes are represented by boxes. For example, the general SeSAM voice interface can be drawn as shown in figure 3.

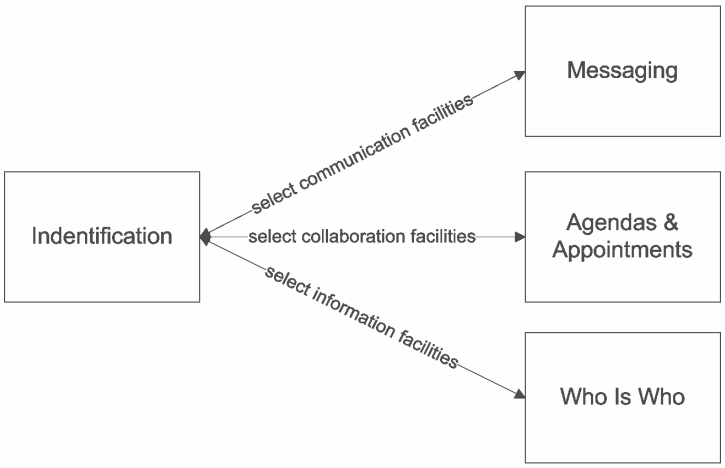

Figure 3. SeSAM scene diagram.

This is a very abstract definition nearly on the level of use cases. The diagram can be refined by replacing scenes by (sub-)scenarios. In a last refinement step each remaining scene is enhanced by a dialog step diagram. Dialog step diagrams represent dialogs steps as ovals and transitions between them as curves. For example, the flow of control for reading or sending personal messages can be modeled as depicted in figure 4. Dialog step specifications can be refined by replacing single dialog steps by new sub graphs representing a finer granularity.

SiteLang supports also actor characterizations by specifying the actors profile and task portfolio. Additionally, the definition frame for content types as views over the applications database extended by functionality, navigational micro behavior, and associations is given in SiteLang. In these terms SiteLang combines facilities for modeling structure, functionality, and interaction in one language.

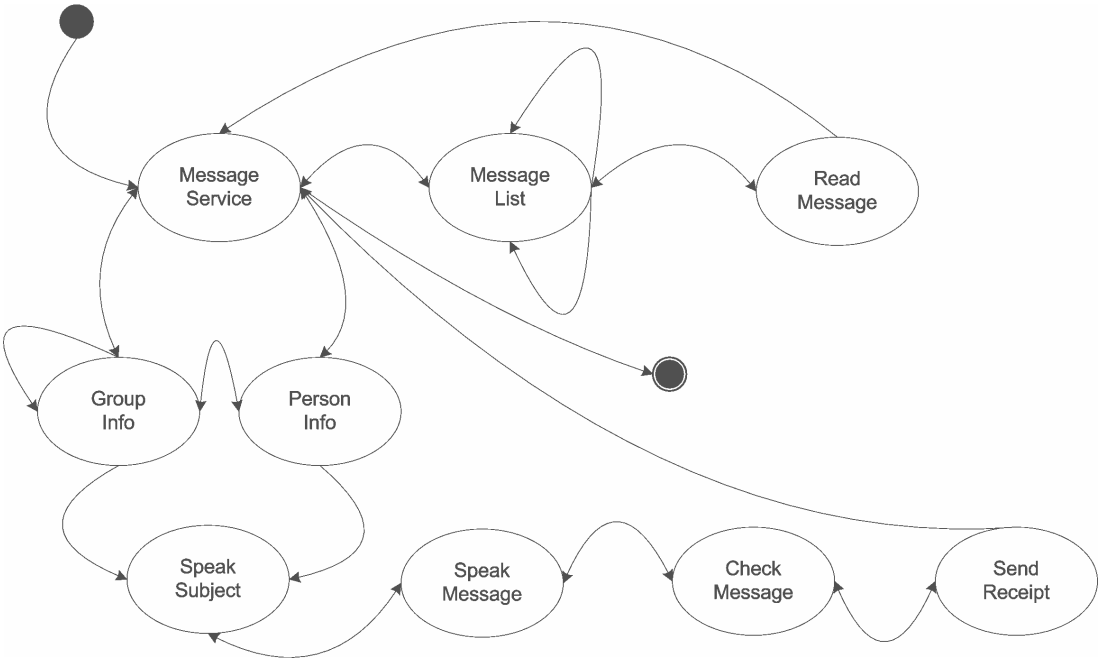

Figure 4: Dialog Steps for Message Reading.

2.2. Applying SiteLang to Voice Based Dialogs

SiteLang was designed for modeling arbitrary interaction scenarios independent from the target platform. The staging approach using active and passive actors fits the need for modeling speaking and listening partners during a natural speech dialog. SiteLang can be seamlessly integrated into the development cycle presented at the beginning of section 2:

- In a general system and process analysis the designer identifies the general stories of the application as well as possible actors and underlying processes.

  In the SeSAM example, the application should support the tasks "information", "collaboration", and "communication" for parliamentarians and interested citizens.

- Afterwards the designer decides which use cases are relevant. For each use case a broad scene is introduced together with possible actors.

  In SeSAM, interested citizens want to get personal information about the parliament (which members, which parties, which committees, which connections between them, etc.). Citizens want to know which topics are on the agendas of certain committee meetings and which motions are under discussion. Additionally, citizens want to communicate to parliamentarians by sending messages.

- The scenes representing the use cases are refined by introducing finer scene specifications as well as transitions between scenes. This step is an iterative one.
  In SeSAM, the use case "sending a message to a parliamentarian" can be modeled as depicted in figure 5.

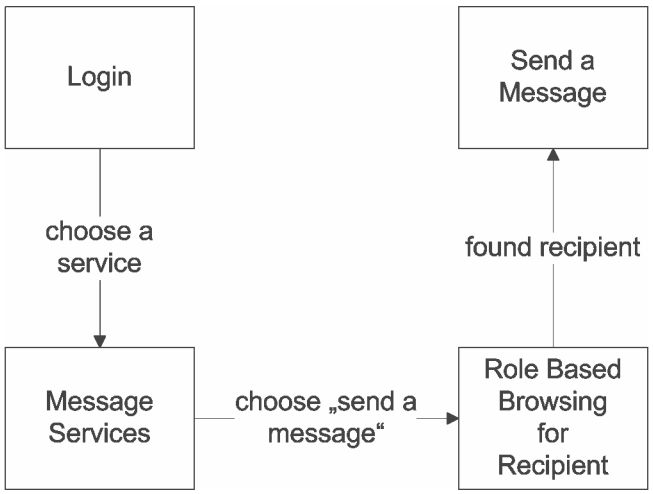

Figure 5: Scene Sequence for Use Case "Citizen Sends a Message"

- The designer has to identify which data is needed in each scene. This leads to a modeling of the scene's content types.

In SeSAM, the Scene "Sending a Message" needs the contact information for the recipient, the contact information for the sender, the subject of the message, and the message body.

- Each scene introduced in the previous step is now associated with a dialog step specification. Each dialog step corresponds to a single piece of human machine interaction where the machine says something and offers several options to the user. Afterwards the user chooses from this list by speaking a command. This command leads to a dialog step transition. The decision whether a piece of interaction should be modeled as a sequence of scenes or as a sequence of dialog steps is rather fuzzy. It should be based on aspects like interface technology independence or reusability.

The scene "Sending a Message" can be refined as depicted in figure 6.

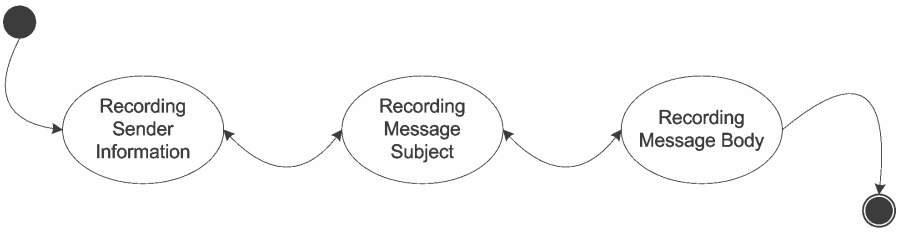

Figure 6: Dialog Steps for "Sending a Message"

- At this point the complete navigational structure of the interface is present. This SiteLang specification can be automatically transformed into executable code. The designer has to enhance this navigational specification by the dialog step ornamentations, e.g. spoken text, lists, menus as will be shown in section 3.
- Now the designer can perform logical and timing simulations in the described manner with possible refinements of the SiteLang model.

Beside the integrated modeling concepts the main advantage of using SiteLang within the development cycle is the automated generation of executable code based on SiteLang's formalized semantics. In the next section we present the transformation process for servlet and JSP based web applications following a Model-View-Controller (MVC) styled architecture which is very similar to common enterprise application architectures.

3. Deriving a System Architecture for Rapid Application Development

Within the development cycle it is important for the designer to create different kinds of prototypes. In early development stages visual prototypes are necessary for a fast discovery of logical errors and misunderstandings. In later stages voice based prototypes are needed for timing improvements. All prototypes are extensions of earlier ones and agree at least in the navigational structure. For this reason a strict separation between controller logic generated from the SiteLang specification and the visual or acoustical representation of dialogs is mandatory. That's why the Model-View-Controller pattern is a good starting point. The general architecture of the generated application is depicted in figure 7.

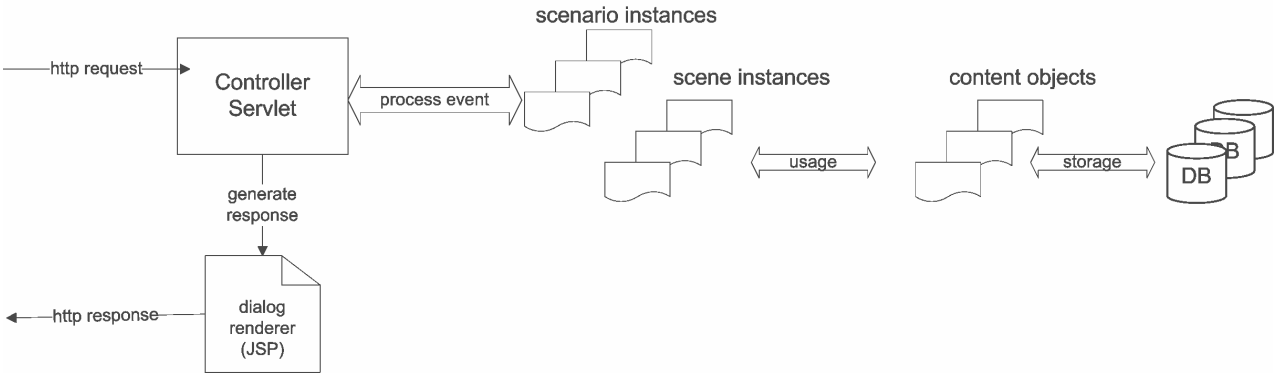

Figure 7: The MVC-styled architecture of generated SiteLang applications

The controller servlet interprets incoming requests, e.g. http requests, and creates an event object. An event is an identifiable data structure, e.g. a name together with some (possibly complex) escort data. The request's parameters are mapped to this data structure according to the SiteLang specification. The controller maintains a set of currently active scenario instances together with their scene instances. Depending on the address part of the request, the target scenario and scene instances are chosen. The target scene instance processes the event using the retrieval and modification functionality of content objects as specified in the SiteLang model and updates the system's state. According to the specification the scene instance chooses a dialog which renders the response using the data that is provided by the scene instance and the associated content objects.

The ornamentation of the dialog representation has to be written by the designer. In the case of designing voice based systems he/she needs a HTML as well as a VXML representation during logical and timing simulations. Because we use only very simple HTML structures (paragraphs for spoken text, lists for menus, etc.) the mapping between HTML and VXML tags is nearly one-to-one. Only some VXML specialties like recorded wave input cannot be directly expressed in HTML. Because of this one-to-one mapping the designer has to specify only one representation, the other one is generated automatically. An overview over connections between HTML and VXML output can be found in (Schwanzara-Bennoit & Fiedler, 2004).

To fit the presented architecture the compiler has to create the following classes from the SiteLang specification:

- *Management of scenarios:* The scenario class manages meta information for a specified SiteLang scenario (which scenes, which possible actors, etc.). The extension of the class contains all active scenario instances. A scenario instance provides information which scene instance is currently active and which actor is currently associated with which user. Additionally, a scenario instance contains local variables as part of the system state.
- *Management of scenes:* For each scene a separate class derived from a common super class is created. These classes provide meta information for the scene (which dialog steps, which actors, access to the content type, links to dialog definitions, etc.). The extension of the class contains all active scene instances of this class. These instances provide the state of the scene, current actor-to-user mappings, and the content object.
- *Management of dialog steps:* Dialog steps are represented as members of scene instances. Events are propagated to the corresponding scene instance. For this instance the trigger conditions for each dialog step are evaluated and the dialog step is activated if necessary.
- *Management of content types:* Each content type is translated to a specific class derived from a common super class. The content type classes provide meta information about the structure, the

functionality, the micro behavior, and the associations of the corresponding content type. Additionally, the generic database load / store mechanisms are provided. Each class instance contains the values for a concrete content object usable in dialog step actions.

- *Management of actors and users:* (Thalheim, 2003) defines an actor and user model based on profile and task portfolio management. For each defined actor a class derived from a common super class is defined which implements this actor model. This class can be used as an oracle that can be asked for certain properties of the actor. Additionally, this class manages mappings between actors and users.

4. Conclusions

In our paper we presented a modeling methodology for interactive voice response systems on the basis of the web site description language SiteLang. By using only one modeling language with just a few easy-to-learn concepts it is possible to support use case modeling and interaction refinement. Due to the well defined SiteLang semantics an automated translation to executable code can easily be achieved to support a rapid prototyping strategy.

SiteLang is a language generally suitable for interaction modeling. Modeled SiteLang specifications can be shared between different interaction media in a multi channel environment; especially on higher abstraction levels. For example, the classical HTML web interface of the presented eDemocracy portal SeSAM differs from the voice interface only at the level of dialog steps. This allows a consistent management of different interface versions and variations within a large information system.

Interactive Voice Response Systems are an interesting technology for supporting mobility in modern IT projects. But using this technology needs additional thoughts about its restrictions. Not every use case can be usefully expressed by voice dialogs. Voice processing and telephony technology is more expensive than traditional web technology. But one main advantage remains: all you need is a phone. This enables people that are not able to or feared of using classical web based portals to participate in modern (mobile) eGovernment projects.

References

Ceri, S. et.al., (2003), Designing Data-Intensive Web Applications, Morgan Kaufman, Boston.

Fiedler et.al., (2004), SeSAM – Support on Profile, Portfolio, and Demand for (e-)Parliamentarians, Proceedings of e-Society 2004.

Fraternali, P. & Ceri, S., (1997), Designing Database Applications with Objects and Rules: The Idea Methodology, Series on Database Systems and Applications, Addision-Wesley.

Fleming, J., (1998), Web Navigation – Designing the User Experience, O`Reilly, Cambridge, 1998.

Freiheit, A. et.al., (2003), Voice XML (Programmierung und Applikationen), VDE Verlag.

Forbrig, P., (2002), Objektorientierte Softwareentwicklung mit UML, Fachbuchverlag Leipzig.

Krautz, Udo, (2003), Entwurf einer eGovernment Anwendung, bachelor thesis, Brandenburg University of Technology at Cottbus.

Schewe, K.-D. & Thalheim, B., (2004), Web Information Systems: Usage, Content, and Functionality Modelling, Report 0405, Kiel University, Institute for Computer Science and Applied Mathematics.

Schwanzara-Bennoit, Th., (2004), Entwurf und prototypische Implementierung einer Sprachdialog-Schnittstelle für ein eGovernment Informationssystem, Diplomarbeit Brandenburg University of Technology at Cottbus (in German).

Schwanzara-Bennoit, Th. & Fiedler, G., (2004), State and Object Oriented Specification of Interactive VoiceXML Information Services, Proceedings of NLDB 2004.

Thalheim, B., (2003), Co-Design of Structuring, Functionality, Distribution, and Interactivity of Large Information Systems, Report 15/03, Brandenburg University of Technology at Cottbus.
Thalheim, B. et.al., (2004), Website Modeling, Website Orchestration, and Website Management, Proceedings of LIT 2004.
Thalheim, B. & Düsterhöft, A., (2001), Conceptual Modeling of Internet Sites, Proceedings of ER 2001.

http://www.cottbus.de (2005/02/20)
http://www.w3.org/TR/voicexml20/ (2005/02/20)
http://www.w3.org/TR/speech-synthesis/ (2005/02/20)
http://www.w3.org/TR/speech-grammar/ (2005/02/20)