

EdFest Voice

Aspects of Usability and Context Awareness
in a Voice-based Information System

Jan S. Rellermeyer

Semester Thesis

July 5th, 2004 – October 4th, 2004

Supervising Professor: Prof. Moira Norrie
Supervising Assistant: Ljiljana Vukelja

Contents

Introduction	1
1 User Centered Design	3
1.1 User Centred Design	3
1.2 User Centred Web Design	3
1.3 User Centred Design of Voice Interfaces	4
1.4 Criticism on UCD	5
2 Context Awareness	7
2.1 A Little Theory of Context Awareness	7
2.2 Internal and External Context	8
2.3 Context Awareness in OMSwe	8
3 EdFestVoice	11
3.1 Design Issues	11
3.2 Architecture	15
4 EdFestLight	25
4.1 Design Issues	25
4.2 Architecture	26
5 User Study	29
5.1 User 1	29
5.2 User 2	30
5.3 User 3	30
5.4 EdFestLight	31
6 Conclusions	33
A Task	35
B Installation of EdFestVoice	37
B.1 Server Installation	37
B.2 Client Installation	37

C	Technical Documentations	39
C.1	Hardware	39
C.2	GrammarBuilder	39
C.3	Navigation	40
D	Interviews	44
D.1	User 1	44
D.2	User 2	46
D.3	User 3	49
E	The least	53
	Bibliography	54

List of Figures

3.1	EdFestVoice Architektur	15
3.2	EdFestVoice Design	17
3.3	EdFestVoice Features	18
3.4	Voicemail	19
3.5	Personal Agenda	20
3.6	Location	21
3.7	Events	22
3.8	OMS Schema	23
4.1	EdFestLight Architecture	25
4.2	Example macro	26
4.3	Example XSLT stylesheet	27
C.1	OMS prolog module	40
C.2	GrammarBuilder Architecture	41
C.3	Main loop of GrammarBuilder	42
C.4	Navigation schema	43
C.5	Navigation algorithm	43

Introduction

The tourist industry is a major market for mobile devices and integrated information systems. People in foreign countries or cities are interested in getting information about the environment and specially in Edinburgh, a guide to all the different events and venues can be very helpful. Classic printed information sources have the disadvantage that they are not always up to date and do not provide any possibility of feedback or interaction. An electronic information device is more flexible and can do even complicated tasks like guiding the tourist to several places in the sense of a navigation system. A personal agenda can help to plan the visitation of events.

To build a successful information system, usability is one of the most important aspects for the design, specially for a system like EdFestVoice that is purely voice based. Virtual user interfaces can disappoint and irritate users if they are badly designed, no optical effects can convince the user, information must be provided easily and in an understandable way.

The other important aspect of successful information systems is the quality of provided information. One way to improve the quality is context awareness. The system can use additional data collected by sensors like GPS receivers or from a user database to get a better idea of what the user actually wants. Finding a restaurant can mean finding one close to the current position and also getting information on how to get there. If the user usually prefers theatre plays it is most likely that a query about today's events should have a special focus on theatre plays. Of course, this is not always the case so the challenge is to work with context awareness without limiting the possibilities by assuming things that might not meet the current requirements. A well designed system will leave the user the choice if additional information gains by context awareness should be used in a specific situation or not.

EdFestVoice is an approach to integrate current state of usability guidelines for voice interfaces as well as context awareness and it is a technical design to use easily available resources like GPRS to form a system that can help tourists during their visit to festivals in Edinburgh. EdFestVoice is a concept for a device that also integrates social aspects and shows the possibilities of current micro devices and IT-infrastructure. It has been tested in August 2004 in Edinburgh

in several user studies. The report analyses the real world experiences with the system and shows how EdFestVoice could be further improved.

Chapter 1

User Centered Design

1.1 User Centred Design

User centred design (UCD) is on one hand a philosophy based on placing the user as person instead of the product as thing in the centre. On the other hand it is a process, based on cognitive factors like perception, learning, problem solving etc. The idea is to let the user participate in the design process as early as possible to meet the users' actual needs and expectations. Scientists like J. Nielsen, D. Norman and J. Spool have developed several guidelines for usability issues in design process and how to implement them. Although UCD covers any kind of human-computer interface, many recent publications focus on user centred web design.

1.2 User Centred Web Design

The idea behind user centred web design is that most users actually don't read web sites, but scan them for the information they need. UCD improves usefulness and usability.

In [10], ten guidelines for UCD are offered:

1. **Visibility:** Important screen elements like navigational aids should signal the user what they can and what they cannot do. The user should be able to predict the effects of an action.
2. **Reduce memory load** by keeping screen elements meaningful and consistent across the site. By this, users can recognise certain patterns instead of remembering what the elements mean from one page to the other.
3. **Provide immediate feedback** when a user performs an action to signal the user that the system has registered the action.
4. **Accessibility:** Guide the user to find the desired information quickly and easily, e.g. by navigation or a search function but don't use too many elements at once to avoid confusing the user. Provide the information in small digestible pieces using hierarchies that are meaningful to the user.

5. Orientation: Help users to orient themselves by site maps, descriptive links, offer an obvious way to exit every page etc.
6. Minimise user errors by avoiding ambiguity.
7. Satisfaction: Make the site pleasant to use and view.
8. Legibility: Make text easy to read by using sans serif fonts rather than serif fonts and avoid the use of ornamental fonts.
9. Use every day language instead of technical terms, be careful with metaphors, idioms and puns because they could have a different meaning in different cultures.
10. Visual Design: The aesthetic of the interface plays an important role, create interesting and uncluttered pages, group related elements, etc.

Apart from concrete design guidelines, UCD is a methodology to concentrate on product design and develop user interfaces that meet the user requirements. [4] divides the design process into five different phases:

1. Collect data about user tasks, their behaviour and their current work processes.
2. Analyse data to determine how to redesign the work-flow and to identify usability requirements and goals.
3. Design and develop scenarios and information architectures.
4. Test the design and conduct iterations of usability assessments.
5. Deploy and follow up with measures of usability and user satisfaction.

Many classic approaches for web design start with a concept and expect the users to accept this concept. UCD tries to learn from users and iteratively design web sites that meet the users demands.

1.3 User Centred Design of Voice Interfaces

While XHTML is the markup language for desktop browsers and WXML for small mobile devices using WAP, VXML was designed to serve voice browsers. The main difference between web interfaces and voice interfaces is the total virtuality of voice based interfaces, so design issues are crucial for the success of a voice based application. Navigation between pages and presentation of data must be well-designed to help the user find the way through voice pages and finally get the desired information. User centred design can help to avoid design failures. [4] points out the major reasons why user will abandon a voice driven application:

- They are confused by their initial contact with it.
- They think the application has stopped working.

- They encounter errors.
- They don't quickly perceive the application's value.
- They become bored with long prompts or unwieldy application navigation.

Navigation must be considered carefully as users have no visual feedback if a link has been taken or what links are available. The structure of pages must be simple and logical, users will build up a mental model of the application and use their memory to navigate. Because the mental model is a subjective perception, it heavily depends on the user's expectation of what the application does and how it works. Thus, user centred design is especially helpful for voice applications. Only if the application can meet the requirements and the expectations, it can be successful.

1.4 Criticism on UCD

In recent days, some scientists came up with the thesis that UCD might constrict innovation. If a design process only conforms to users' needs, this might tighten the space for new elements, users might not directly see the advantages of a new element and dismiss it.

Second, UCD only knows two contradicting influences in design process, site needs and users needs. In this arrangement, UCD guarantees the maximum profitability. But other requisites may figure strategic placement on a market or the developer's corporate identity. These aspects are generally not covered by UCD. Another point of criticism on UCD is the idea of a user that it depends on. UCD implicitly requires an abstract essential user with a certain set of requirements that do not contradict and therefore can be met by an iterative design process in finite but unbounded time.

Existentialism postulates that existence precedes essence and a model motivated from this cannot be based on the essence of a user per se. Aside from custom tailored applications, designer cannot have an idea for what kind of users they are designing a system; in research, this would already be a restriction. A motivation can be taken from human-computer interaction, no matter how large a group of representative users is, there will be a subgroup of users who prefer to use the mouse for navigation between GUI input fields and a subgroup that prefers to use the keyboard. But both subgroups will exist almost equally distributed, so a well-designed application should better support both methods of navigation.

Chapter 2

Context Awareness

2.1 A Little Theory of Context Awareness

Information devices are no longer just enterprise sized servers or desktop sized workstations. They have found their way into cellular phones or PDA's, smart devices or even smart buildings help the user to manage complex tasks and ambient screens provide the user with information. Users expect systems that can deliver desired information as easy and as quickly as possible. One important approach on the way to the user-friendly information systems is context awareness. On one hand, real objects can play roles in human-computer interaction, e.g. the user can point at objects to get information about them. Systems that are aware of the user location or other users or environmental parameters can use this knowledge to make assumptions what kind of information the user wants. A simple example is the current time: a query of restaurants in the city of Edinburgh might implicitly mean that the user is looking for restaurants that are currently opened. More advanced, the user might expect opened restaurants close to the current location or the system knows that the user prefers vegetarian food and therefore will not recommend a steak house. If an ordinary query can be generally expressed as some kind of function $f(x_1, x_2, x_3, \dots, x_n)$ of given input parameters $x_1, x_2, x_3, \dots, x_n$, non-context-aware system requires the user to specify all n parameters and will therefore always return the same result for a fixed set of input parameters. A context-aware system however will assume at least some of the parameter according to preferences, sensor values or other data sources. From the design point of view it should be made clear where these assumptions are used and what impact they might have. In general, a context aware query can have different results for the same input parameters because the assumed parameters might have changed. The user should be aware of context awareness of the system, otherwise users might find the system confusing or even patronising. Another aspect of user awareness is privacy. Smart systems tend to collect data about users, so do context aware systems and so user data should be protected against abuse. Systems must be well-designed to show the user what kind of context can be used to interact with the system. They should not hide their functionalities behind context that is a virtual presence from the users' point of view.

2.2 Internal and External Context

The classic theory of context awareness only deals with external context, environmental and user information that can be used to improve the quality of results, but especially for voice driven systems internal context is also of importance. Voice recognition in VXML systems requires precisely defined grammars. Navigation elements can live with static grammars defined at design time but often VXML elements depend on data from the underlying database, e.g. choice of a museum with the names of the museums as keywords, this grammar must be build up at runtime to reflect the current selection of information. This spans an internal context for data, for example in a collection of landmarks in Edinburgh, the word "Scot" will be a significant keyword for the "Scots' Monument" but within the phone-book of the University of Edinburgh several persons with the name "Scot" might appear and in this context the word is no longer significant. This means that the system cannot make an unambiguous decision. Additional information like the family name is required to select a specific value, the term "Scot" is no longer unique. Even worse, the importance of a word for recognition can change when the data changes, therefore the internal context is also volatile if the data is not static. In the EdFest Voice System an on-the-fly Grammar Builder is used to deal with internal context of data and get better recognition rates. With a very efficient algorithm, the Grammar Builder creates a grammar that allows the system to decide on every significant word and on every unique combination of insignificant words. With every request, the grammar is newly created to reflect the current internal state of context.

2.3 Context Awareness in OMSwe

OMS Web Elements is a multi-language, multi-version, rapid prototyping web extension of the well-known object oriented OMS database suite. The main approach is to separate content from presentation and to allow multiple versions of the same object. Due to the circumstance that in OMS everything is an object, this does not only allow to have multiple pieces of content for different situations that fit into the same content slot, but also to have different forms of presentation, e.g. the same content can be viewed on a desktop screen, on a WAP device or via VoiceXML, it can be heard on an appropriate audio device. Normally, the browser that forms the HTML request already adds some context information to the request, e.g. the type of browser, the user's operating system or the locale. Database designers can label different versions of objects with properties and OMSwe will perform pattern matching on this properties to find the best possible fitting language of a piece of content or the proper XSLT template to transform the requested page into the kind of presentation that the browser can interpret. But apart from this, additional properties can be added manually to the HTML requests and this already provides a good basis for context awareness. EdFestVoice, however, also uses context in database queries and needs additional server side logic to perform this. But, as OMS

is designed to be extensible, even complicated tasks like the navigation system for the city of Edinburgh can be embedded into the database system in form of Prolog modules. The navigation system is implemented as a draft to show possible applications in the tourist industry. Starting from the current location, a simple greedy algorithm finds a path to the destination. As optimisation, at every junction the choice is always on the street that brings the user closer to the destination. As further optimisation, a more sophisticated routing algorithm like Adaptive Face Routing, developed at Distributed Computing Group of ETH Zurich, could be implemented. The model behind the navigation is simple: maps are modelled by parts and junctions, where parts is a connection between two junctions and streets are collections of parts. So every street has a name and consists of one or more parts that connect different junctions that have coordinates.

Chapter 3

EdFestVoice

3.1 Design Issues

EdFestVoice was designed to be a representative of a new generation of information systems. Usability and social aspects were taken into account as well as technological issues from mobile computing. In addition, the system shows the possibilities of OMSpro / OMSwe.

Recognition Rates

A crucial point in the usability of a voice based system is the recognition rate. If the rate is too low the system will misunderstand the user's input and will navigate to wrong pages or search for wrong keywords. This will cause the user to get frustrated. EdFestVoice has the special problem that many search items have similar names, e.g. many festivals have the words "Edinburgh" and "Festival" in it. A classic grammar will lead to unacceptable recognition rates for festival titles because the recognition algorithm mainly relies on pattern recognition. If the grammar elements are unstructured pattern containing several identical words, the distance between two grammar elements can become that small that recognition leads to unpredictable results. The GrammarBuilder is a component that has been developed for the OMSpro / OMSwe database suite to get grammars that allow best possible recognition rates by separating significant from insignificant words. This enlarges the distance between the grammar elements and implements the most natural approach thinkable: unique words within grammars lead to direct hits, no matter of any surrounding ambiguous words. A full description of the GrammarBuilder is given in the appendix. Generally, the algorithm is based on an optimistic "go-back-once" strategy. In a first phase, the data is parsed and tokenised. It is expected that the data consists of phrases that are a composition of one or many tokens. Every token that has not appeared yet, is marked as significant. This is the optimistic element of the algorithm. Although the marking is correct for every subset processed so far, the algorithm cannot decide if a word will still be significant for the whole set. Additionally, a reference between the token and the phrase in which it appeared is saved in a hashtable. So checking for duplicate appearance means querying the hashtable. In case a token has already appeared before, it

must have been marked as significant before and this must be corrected now. By definition a significant word must be unique, a second appearance violates significance. But as a result of the architecture, the hashtable returns a reference to the first appearance of the token. The invariant of the algorithm is preserved, at any time, all marked words are correct for the processed subset, correctness means that they are either marked as insignificant because they already appeared more than once or are unique up to that point. So a correction can only happen once. Therefore, the algorithm can be described as "go-back-once". After the first phase, all tokens are marked and the second phase can generate the grammar.

Multipath Navigation

As mentioned earlier, EdFestVoice tries to follow a way of Users Centred Design instead of designing a system for an idealistic user. There are several plurivalent ways of getting the same kind of information, because different user use different ways to get the same information.

As an example, the location functions are designed for different needs. In some cases, a user might already know exactly the name of the location that he wants to have detailed information about. In this case, EdFestVoice supports finding a location by name. If the user does not know the exact name, EdFestVoice uses the flexible grammar from GrammarBuilder to give the user a list of all locations that match the entered phrase. For users that are completely unsure about the name of the location, another feature allows to enter the name of a known landmark that is located near the desired location and then gives the name of all locations within a given radius. For the user studies, this feature has been hidden, a hint about this feature is given only when the user answers the question about the location with "don't know", "no idea" or similar phrases. The last feature is to find locations by category in a context sensitive way. All locations are listed that are within a certain radius from the current position. The radius starts with a default value and is automatically enlarged if no results are returned by the database within the given radius or if the user is not satisfied with the returned results and wants the radius to be enlarged.

Bandwidth and Online Costs

The communication between client and server use mobile networks like GPRS or UMTS to transmit data. Both systems are normally billed by used bandwidth, so EdFestVoice uses several levels of caching to save bandwidth. First, the EdFestVoice framework supports a local repository for static pages. For these pages, no bandwidth at all is used. But EdFestVoice does not use this feature to keep the system more flexible and allows to update both the data and the control logic on the fly. The second way is to mark relevant pages as cacheable. These pages are downloaded once and every time they are accessed, the CacheManager will check if the hash of the cached page and the server page are still the same. If the page has not changed, the local copy will be used, so the used bandwidth for a request is in case of a cache hit just some bytes for the URL and the hash

value of the local page. Non-cacheable pages are those that are unlikely to stay the same, e.g. they are the result of a context sensitive query. In this case the whole page is downloaded.

Social Information System

EdFestVoice was designed to be both an information and a communication system. A major problem of voice based systems is the total user attention that is required. In other words, users can either use the system or communicate with other persons and interact with their environment. EdFestVoice breaks this disadvantage by a special hardware. If the user wants to speak to the system, a little button on the headset wire has to be pressed, so the user can decide when and where he wants to use the system. Vocal communication with other people is possible when the button is not pressed, i.e. whenever the user decides so. On the other hand, EdFestVoice has a voice mail system to communicate with other people using the system. The implementation gives just a little impression of what is possible, in a more advanced version of the system it could be connected with a classic POP3 mail system to allow the user to also send and receive conventional emails or implement a VOIP (voice over IP) gateway to let users have telephone communication over the EdFestVoice system. This can help the user to stay in contact with his normal environment and communicate with friends at the holiday location using the same system.

Help

To give the user some guidance, in EdFestVoice a multi-level help system is implemented for almost all features. First, if a timeout occurs because the user has not entered a correct value, the system assumes that the user did not understand the prompt properly and reprompts in a slightly more concentrated way to point out what the system expects.

Privacy

While context sensitivity makes devices smarter, it makes it also easier to profile users and collect personal data. In the history of ubiquitous systems, a severe mistake that has led to user's decline was an insufficient clarification about what amount of private data is collected. EdFestVoice does not need server saved profiles and does not connect context data with personal data apart from temporary queries that are performed. All personal data is saved on the client side, voicemails and agenda entries are saved on server side but only under the anonymous handle of the user, not under the user's real name. So the user has a unique virtual identity that is not connected to the real life identity of the user. Data like the user's current position is only transmitted when a position sensitive query is performed. The system does not transmit the position periodically so the user can decide if this data should be used to get better query results or to use classic functionalities only and not provide the system with the current position.

Iterative Matching

Being tourist in an unknown city, an interesting problem arises: How can users find objects that they do not precisely know about ? To solve this, EdFestVoice uses a special mechanism, that can be called *iterative matching*. If a user enters an insignificant word, the grammar returns a token instead of an object ID. With this token, the system builds up a new grammar consisting only of those objects that have this token in their name and another iteration begins. For example by entering "theatre", in a first iteration all objects containing "theatre" are returned in a list and read to the user. Then the user can enter a more precise title that either matches directly or causes a new iteration.

3.2 Architecture

EdFestVoice is a classic client / server application. On server side, an OMS Pro server is running providing the object data and the prolog part of the application. As gateway, a java servlet is running on a tomcat server. Additionally to the java parts of the server side logic, it includes the GrammarBuilder, the CacheManager and the VoiceMailServer.

On the client side, a standalone java application named EdFestVoice Client communicates with the server and feeds the IBM VXML Browser with data, either by contacting the server, or from local repositories in case of cached or static content.

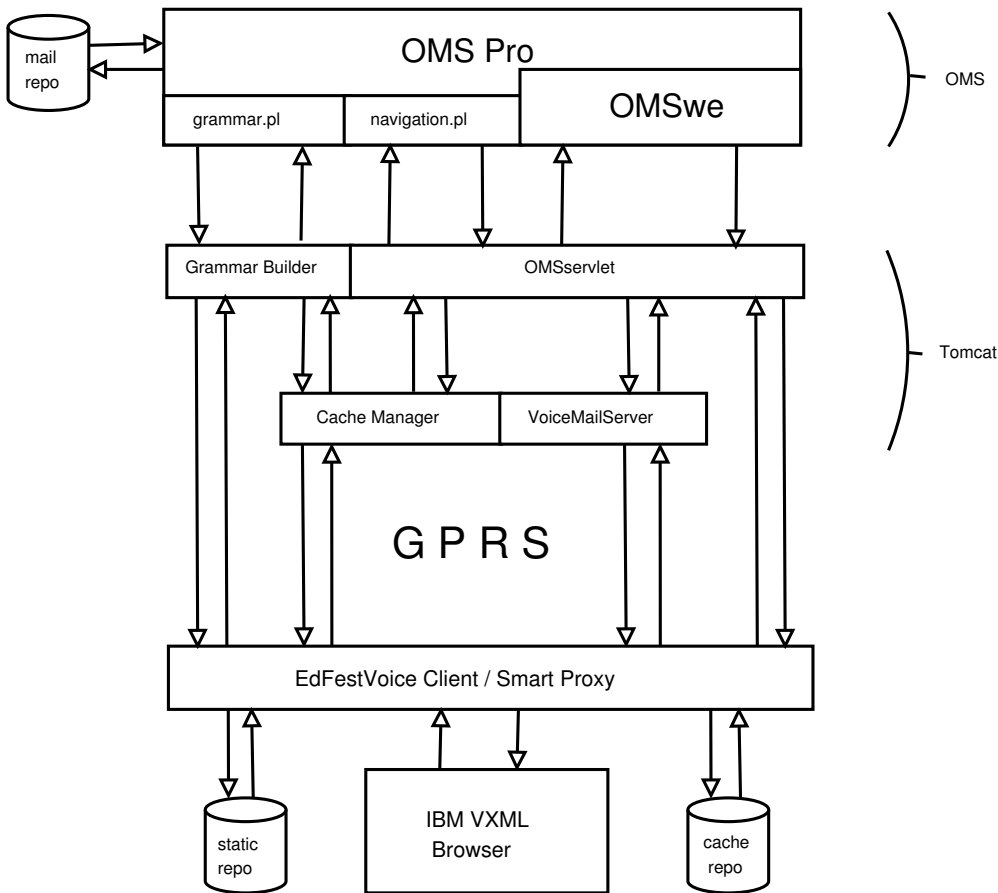


Figure 3.1: EdFestVoice Architektur

OMS Pro

The database backend is the 2.1.6 Version of OMSPRO/OMSwe. Each VXML page is modelled by one Webelement that can appear in different versions. Multi language support is integrated in the EdFestVoice framework but not implemented.

Prolog Modules

- **Grammar Builder Responder**
The responder for the grammar builder is a prolog extension embedded in OMS and the counterpart of the servlet-based Grammar Builder. The responder needs the name of a collection and the name of an attribute. The result is a set of OID's and the projection of the attribute. The Grammar Builder uses this information to build the on-the-fly grammar.
- **Navigation System**
The navigation system can guide the user from the current location to a venue or other location in the database. A full description of the used algorithm and the model can be found in appendix.

OMSServlet

OMSServlet is part of OMS and runs under Tomcat. It implements all functionality of an ordinary J2EE Servlet and has direct access to the OMS database gateway.

Grammar Builder

The grammar builder class inherits from OMSServlet and creates a grammar from a given property of objects in a collection in OMS. The result is a JSGF (Java Speech Grammar Format) compliant grammar. A detailed description of the theory behind the grammar builder can be found in appendix.

Voice Mail Server

EdFestVoice lets the users send and receive voice mails. The voice mail server implements HTML post and in this mode, it expects a multipart-encoded HTTP POST request that includes sender, receiver and the audio data. Voice Mail Server stores the audio file and creates the wrapping object in OMS. With a HTML GET request, a stored audio file can be retrieved.

Cache Manager

Cache Manager is the implementation of a concept to reduce bandwidth use. Instead of transmitting a page every time it is requested, a request can be redirected to the cache manager. Additionally, the voice client adds the hash value of the locally cached page and the cache manager can then decide if the local copy is still up to date or if the result it gets from the server has a different hash value and the page has to be retransmitted.

EdFestVoice Client

EdFestVoice Client's most important feature are the mappings that decouple VXML browser requests from server requests and VXML pages, similar to Tomcat's mapping between requests and servlet implementations. This does not

only allow to encapsulate all context aware information inside the client and makes this layer transparent to the browser, but also hides the different forms of storage and caching. The browser has a unified way of accessing features, even if they run over servlets or use the cache manager. A GPS module is embedded into the client as well as user information and language settings.

IBM VXML Browser

The IBM VXML Browser is part of the WebSphere Design Suite. The application was designed to provide a platform for prototyping and testing VoiceXML applications. In newer versions of WebSphere, the Browser is no longer a stand-alone program, so for EdFestVoice an old version is used that is purely based on Java (J2SE 1.3) and uses the JMF (Java Media Framework) and some basic functionalities of IBM ViaVoice. The need of J2SE 1.3 was a major drawback during implementing EdFestVoice as currently no complete implementation of J2SE 1.3 is available for small mobile devices. As a consequence, EdFestVoice could not be ported to PDA's etc., a broad reverse engineering would be necessary to use the IBM VoiceXML Browser with a J2ME profile.

An Open Source alternative to the IBM Browser is PublicVoiceXML, developed as part of the IST Project (Information Society Technology) funded by the EC commission, together with the Festival TTS engine, developed at the University of Edinburgh. But currently the PublicVXML platform heavily depends on the telephony hardware, e.g. it requires a CAPI (Common ISDN API) device. In the early stage of EdFestVoice, some attempts have been made to use PublicVXML together with a VoIP to CAPI bridge but the bridge was still in alpha stage and did not fulfil basic requirements of stability. So the easy to configure and well-tested IBM Browser has been used to interpret the voice markup.

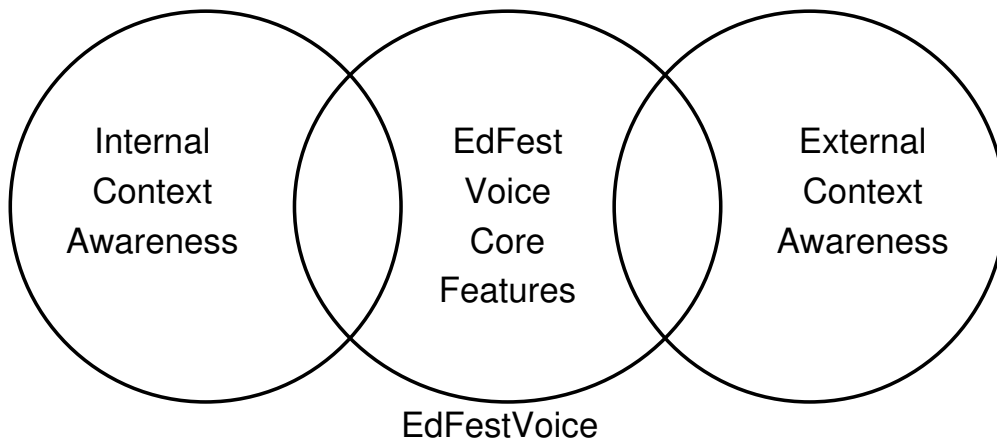


Figure 3.2: EdFestVoice Design

As depicted in Figure 3.2, EdFestVoice uses both external context like a GPS device and user profiles with preferences and also internal context provided by the GrammarBuilder. Context awareness is fully embedded into the application and appears transparent to the user. Only the results of context

dependencies are mentioned in the prompts, e.g. it is signalled to the user that a specific query finds the nearest location of a certain type relatively to the current position.

EdFestVoice has a main menu, as shown in Figure 3.3, that forms the central unit and is a point to where the user can return from almost every form and menu. From the main menu, four different areas can be accessed:

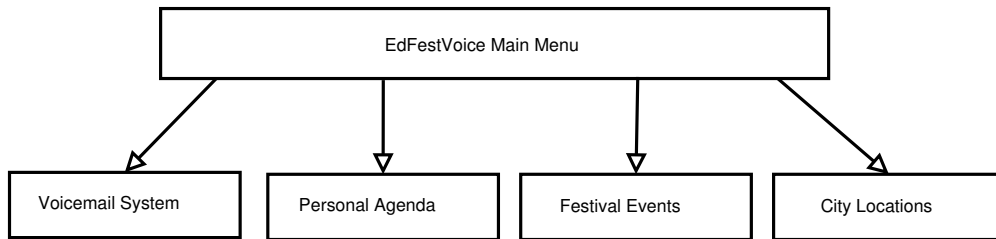


Figure 3.3: EdFestVoice Features

Voicemail

Voicemail is a social interaction system similar to email. Among users of EdFestVoice system, recorded voice messages can be exchanged. By this, users can interact with each other and can coordinate visiting events together or exchanging reviews about events. The embedded social component is meant to raise the users' acceptance and inject some dynamic to EdFestVoice. In the system framework, a notification in case of new incoming voicemails is intended but has not been implemented.

The two functionalities "List Unread" and "List Archived Messages" have the same forms and menus. It is an intended feature of the system that mails that have been opened once, are automatically moved from unread to archived messages. For testing reasons, this feature has been disabled. The recording uses iterative matching to define the receiver, that means, if the user specifies a receiver that does not match to a user's name in the system, the input can be stepwise refined until a single receiver remains. For recording the message, the VXML internal recording is used and the result is posted to the server as a multipart encoded document. The Voicemail Server then stores the audio file and creates the metadata in the OMS database. If an audio file is requested, the Voicemail Server fetches the file from its mail repository and delivers it to the EdFestVoice Client. Figure 3.4 shows the control flow in the voicemail section.

Personal Agenda

The personal agenda is a multifunctional unit designed to manage the user's appointments. It can contain appointments that were generated from events, in this case, the user has decided to visit an event and has placed an entry in the agenda. But it can also contain appointments in the form of recorded audio messages. The system tries to keep the agenda consistent, if the user wants

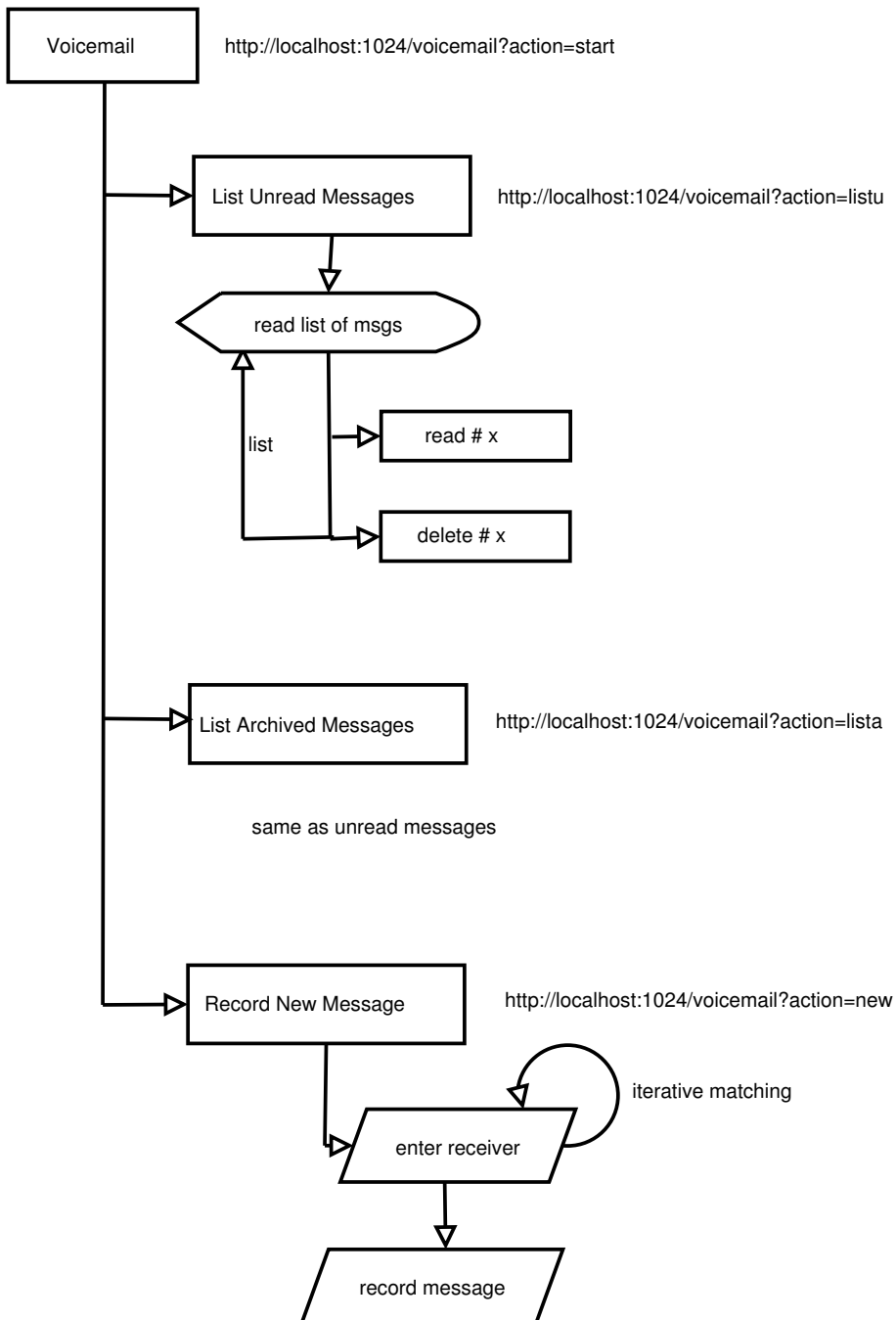


Figure 3.4: Voicemail

to enter an appointment that overlaps existing appointments, the system will inform the user about the collision.

The entry of the date supports the shortcuts "today" and "tomorrow" as well as all valid formats provided by the VXML date format. The user has a unified access to both recorded and generated messages. There is no difference as far as the user interface is concerned, as depicted in 3.5. All appointments

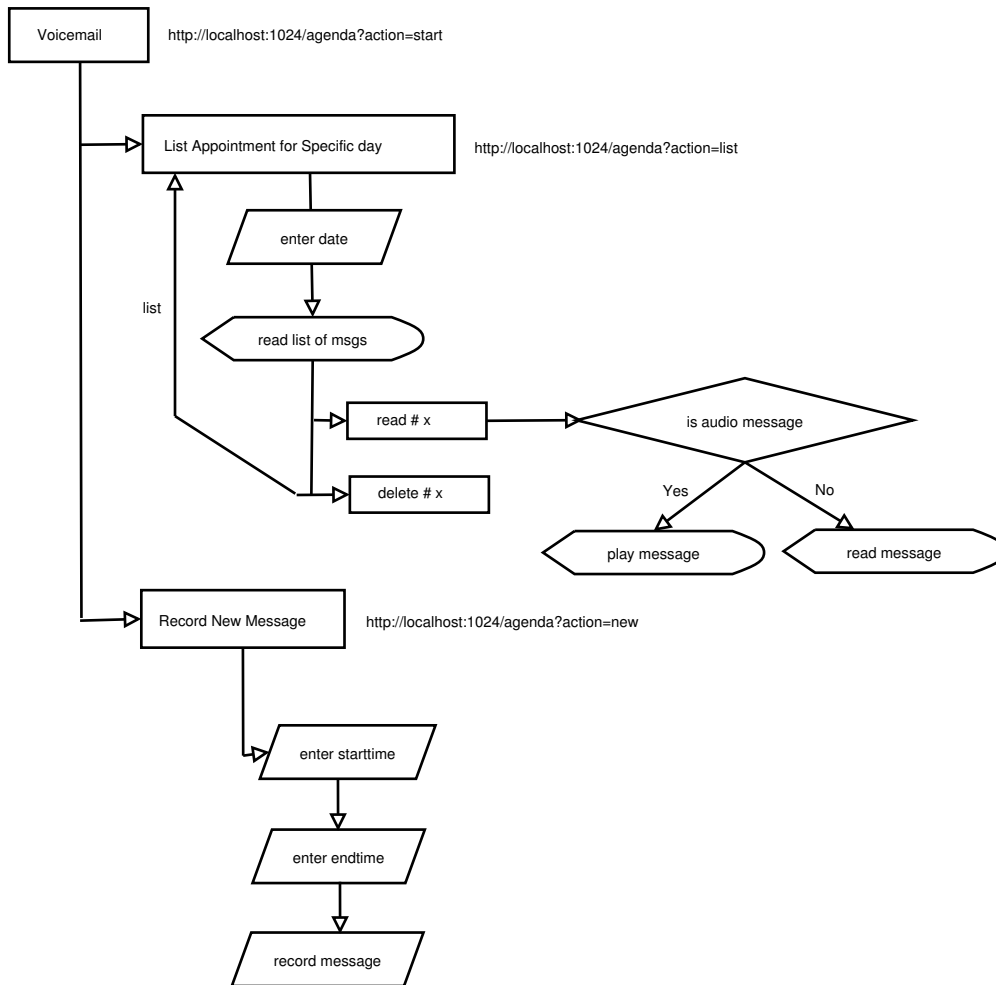


Figure 3.5: Personal Agenda

can be accessed by their number given the relative order for the specific day. Recorded audio messages use the same server-side logic as voicemails do while generated appointments are coming from the database as VXML documents.

Locations

Locations can be venues and other places of interest, like restaurants, bars or landmarks. They have attributes like the street where they are located, opening hours etc. Locations can be either found by name or by category. Figure 3.6 shows the different possibilities to get information about locations.

"Location by Name" requires the user to know at least parts of the name of a location. If a user does not know the name, a help function leads to the "place near" prompt. By this, the user can say a landmark in the city of Edinburgh that is located near the desired location and get a list of all places within a certain range. "Location by Category" is a context aware functionality that lists all locations of a certain type located near the user's current position. If there is no location of a certain type within range, the user can increase the

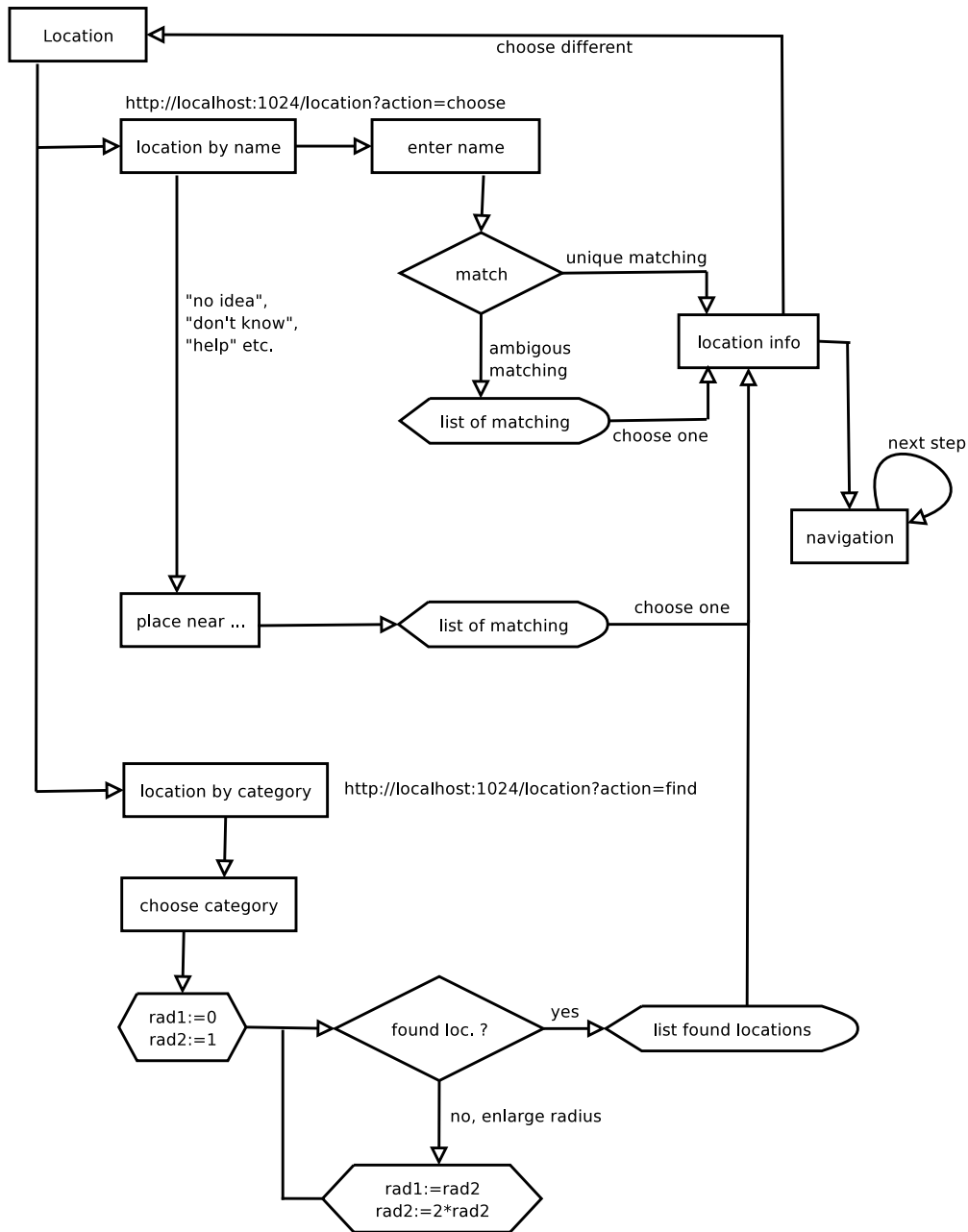


Figure 3.6: Location

radius and requery. By this, the user can find certain nearest locations like restaurants or bars. It would be possible to list only locations that are still opened, comparing the opening hours with the current time, but this is not yet implemented.

Events

Events are a set of events of all festivals. Users can browse events and hear descriptions and comments, e.g. newspaper reviews. Events can be added to the personal agenda if they do not overlap with existing appointments.

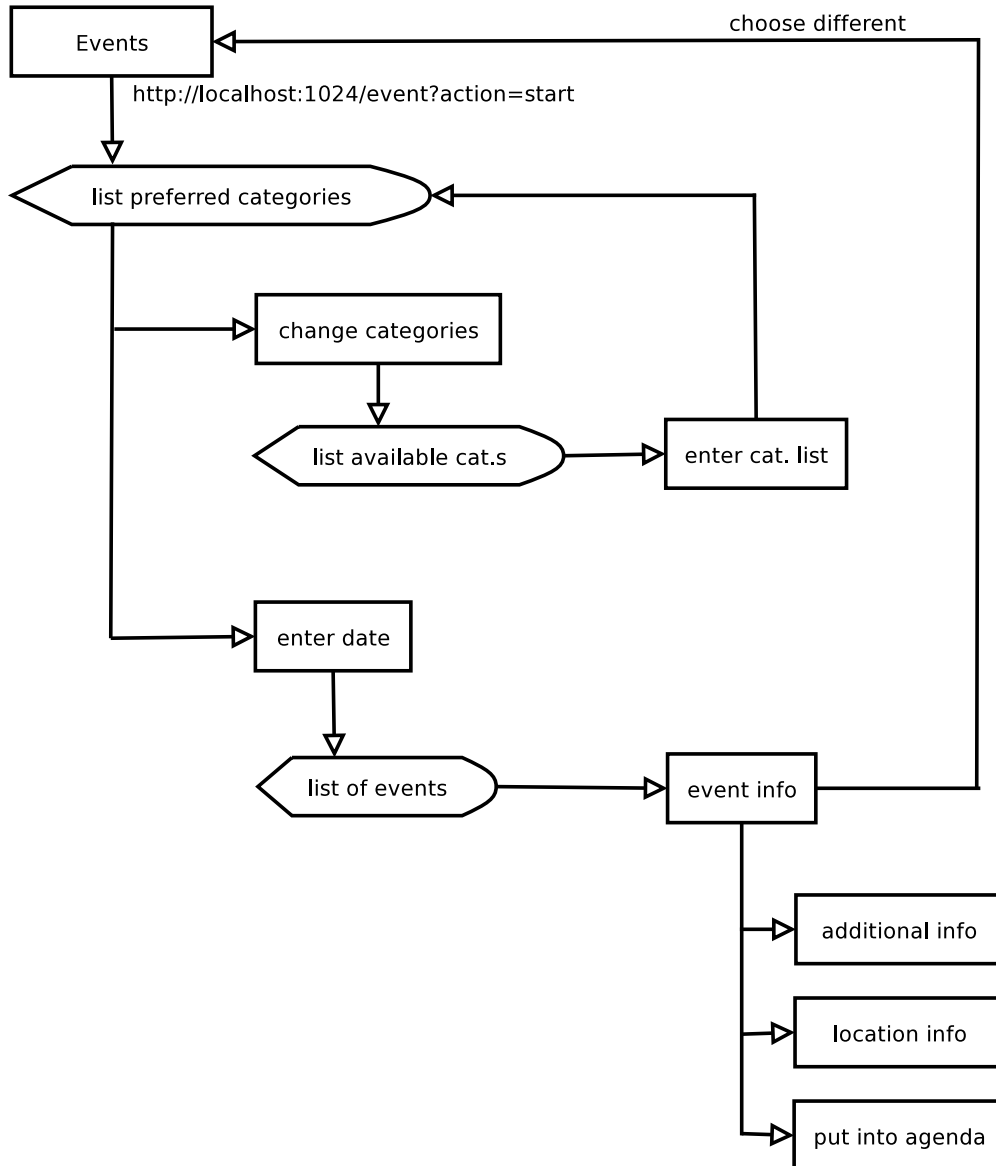


Figure 3.7: Events

Events are filtered by the user's preferred categories. If needed, the preferred categories can be changed. EdFestVoice supports java compliant property file entries defining the preferred categories, it has not been implemented but when a user first uses the system, an initial training could be combined with the definition of preferred categories and the registration of a user name etc. The dialogues for changing categories are dynamically built from the categories

that are available in the database. From a certain event, additional information like different performances can be accessed. There is also a link to the location where the event takes places. In a later implementation, handling of user comments could be implemented using the same mechanism that voicemail and personal agenda use, recording audio files and manage the files via database. 3.8 shows the OMS model of the underlying database.

OMS Database Design

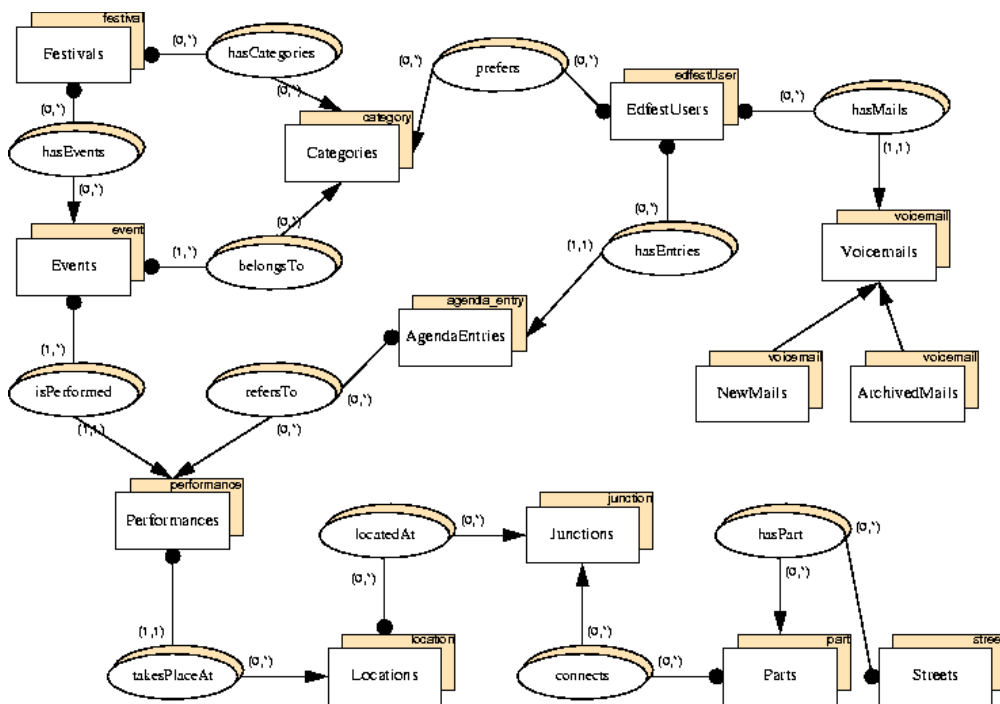


Figure 3.8: OMS Schema

Chapter 4

EdFestLight

4.1 Design Issues

EdFestLight is a lightweight implementation of the EdFestVoice System. All context aware parts have been omitted to keep nothing but the core functionality. These core parts are almost identical with those in EdFestVoice. To save development time, parts of EdFestVoice have been reused in EdFestLight. A purely voice based and context-unaware application like EdFestLight needs a sharp-edged design to provide maximum usability, within the possibilities of an application that is absolutely stateless and not even able to save information about the user. Certain ideas that came up during accomplishment of EdFestLight and dealing with these limitations have found their way back into EdFestVoice and improved the user interface. But, generally, EdFestLight is much more limited in terms of features and does not provide a comparable level of usability as EdFestVoice does.

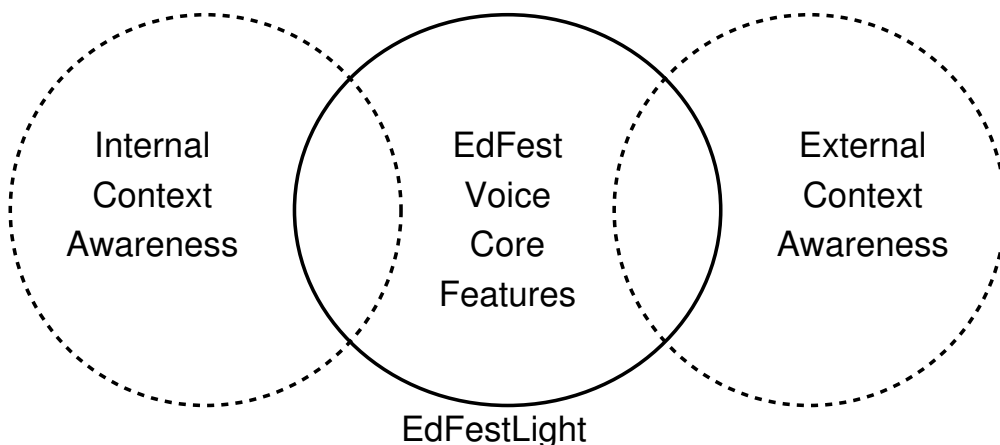


Figure 4.1: EdFestLight Architecture

4.2 Architecture

EdFestLight has only two sections, Events and Locations. Both sections are similar to those in EdFestVoice, only context aware parts have been left out.

On server side, EdFestLight requires an OMSPro server with Tomcat but does not add any server side logic. As client, the same IBM VoiceXML Browser as in EdFestVoice is used but it is connected directly to OMSPro. No caching or proxy is used, every VXML page is modelled by a web element in OMS.

OMS queries the data that can be the the result of a complex macro and transfers the result into an XML tree based representation in prolog. Figure 4.2 is an example of a macro that composes the data for the overview of events matching a certain group of categories and taking place at a certain day given by *Date*. The selected data is then transformed by an XSLT stylesheet, Figure

```
macro([Cat,Date]) :-
    token_to_tokens(Cat, Cats),
    format2intern('~dd~?~mm~?~yy', Date, OMSDate),
    get_coll(_, 'isPerformed', _, EventColl),
    get_coll(_, 'belongsTo', _, BelongsColl),
    findall((Event2, Perf2), (member((Event2, Perf2), EventColl),
    member((Event2, Cat2), BelongsColl),
    member(Cat, Cats),
    get_attrval_by_name(Perf2, 'Date', Date2),
    Date2 == OMSDate,
    we_browse(event, Event2),
    we_browse(performance, Perf2)),
    List).
```

Figure 4.2: Example macro

4.3 shows the appendant XSLT template. The result is a VXML compliant page built dynamically with the data from the database. The example page lists all events and lets the user choose one event and hear details about it. Used XSLT elements are selection and conditional control flow, the variable helps to enumerate the elements and compile the link URL, the OID has to be passed when an event is selected. The resulting VXML page can be requested by the URL `http://edfestServer:omsPort/oms?db_anchor=nocnt_event_list&nocnt_event_list1=ListOfCategories&nocnt_event_list2=Date`.

db_anchor is the name of the Webelement that is called inside OMS, the parameters are passed with *nocnt_event_list1* and *nocnt_event_list2* and are on call mapped to the corresponding parameters of the macro part of the Webelement.

```

<vxml version="1.0">
<menu id="list">
  <prompt>The following events match your requirements: </prompt>
  <xsl:for-each select="obj">
    <xsl:if test="@talias = 'event' ">
      <prompt>
        <xsl:value-of select=
          "attr[@name='Name']/value/atom"/>
      </prompt>
      <prompt>
        <xsl:value-of select=
"attr[@name='ShortDescr']/value/atom"/>
      </prompt>
    </xsl:if>
    <xsl:if test="@talias = 'performance' ">
      <prompt>
        <xsl:value-of select=
          "attr[@name='Starttime']/value/atom"/> till
        <xsl:value-of select="attr[@name='Endtime']/value/atom"/>
      </prompt>
      <prompt>
        <break msecs="1000"/>
      </prompt>
      <xsl:variable name="pos" select="position(.)-1"/>
      <choice>
<xsl:attribute name="next">
  oms?db_anchor=nocnt_event_info&amp;
  db_nocnt_event_info1=<xsl:value-of select="@oid"/>
</xsl:attribute>
<xsl:value-of select="//obj[$pos]/attr[@name='Name']/value/atom"/>
      </choice>
    </xsl:if>
  </xsl:for-each>
  <prompt>Please select an event</prompt>
  <choice next="oms?db_anchor=nocnt_main">
    go back to main menu
  </choice>
</menu>
</vxml>

```

Figure 4.3: Example XSLT stylesheet

Chapter 5

User Study

5.1 User 1

The first user study took place at Pleasance Courtyard in Edinburgh, Scotland during the Fringe Festival. People were sitting at tables, chatting, drinking and waiting for the next performances to start. The test user was female, in the late twenties and native english speaker, the test took place at a table with several other tables with people around. She did not want to be recorded on video but agreed that we recorded with a minidisk recorder. The full recording of the test and the interview can be found on the CD, a transcription of the test and the interview is in the appendix.

The user started with asking for information about events and changed the preferred category without special instruction and without being informed about this feature. After this, she used the context aware version of finding locations. Her overall impression of the system was good and she liked the idea of having small portable devices to guide tourists the way to locations and give them information about events. As improvement, she asked for a reminder for events that have been entered into the personal agenda.

The user's knowledge about computers and information devices appeared to be average. Her major interests during the interview were about possible use cases and what kind of features the system offers. The user felt comfortable when using the device and could easily operate the system although it was noisy during the test. Although the system crashed once, the user considered the general idea and the test implementation as practical and asked when the system will be commercially available for the public. It was not a problem for the user to work with an unknown system and finding out how to use it without special instruction or training. Recognition rate of the system was very high, despite the almost disturbing noise level from the people around. The attending person did not find the interaction with the device disturbing and would have no concerns against a cooperative usage, she would let the user use the device and talk about the results of information retrieval and propose to inquire information.

5.2 User 2

We found the second user outside a cafe in Princess Street in Edinburgh. The surrounding was filled with people passing by, artists performing and some handing out flyers. The test user was male, in the mid-twenties and native english speaker. The test took place at a table belonging to the cafe. Both video and audio recordings exist from this test.

In the beginning, the user had some problems with the microphone button, a short instructing about the usage helped. The first used feature was the voicemail system. The user was irritated about how to enter a receiver. As a consequence of this, an improvement of EdFestVoice could be a special mode for new users where the prompt contain more instructions and explanations and an expert mode designed for experienced users.

Then, the event information system was used. Again, the user did not understand how to enter a date for the query. Third feature was the context aware version of location finding.

Overall, the user's problems with the system were mainly because he expected the system to be more complicated than it is. It took some time until he realised that the system is designed to support multiple ways of entering or retrieving information and no strict way of usage is necessary. So the point of not knowing exactly how to use the device made the user feel uncomfortable, although the overall impression of the system and the idea behind it was very positive. Nevertheless, the user tried out the more complex features and could intuitively navigate through them. The system's recognition rate was very high, hardly any misinterpreted inputs. The user had no concerns against telling the system personal data as the situation being a tourist in a foreign city was not considered to be critical in terms of privacy. The system was taken as useful and helpful for potential touristic users.

5.3 User 3

The third session took place in Princess Street in the middle of a square at a memorial. The scenery was comparable to the last user study, the user was sitting on the floor and was in attendance of a colleague. She was female, also mid-twenties and native german speaker with average knowledge of english. She mentioned that she was not mainly in Edinburgh because of the festival. After some initial problems with the handling of the earpiece and the button, the user tried to find a location. She had problems listening to the system and communicating with us, probably because using a system in a foreign language required her entire concentration. An interesting point was her question how she can find things with the system that she does not really know about. The possibility of finding places via neighboured location seemed helpful to her.

In the interview, her main proposal was to have some kind of map of the features so users can see where they currently are in the control flow and what kind of input is expected. She did not seem to be very familiar with computer technology and appeared to be not a typical user of complicated technical equipment.

Maybe difficulties with the foreign language had some influence that she found the navigation not so easy to understand. The system's recognition rate was average in some cases, input was misinterpreted but rarely. Sometimes the user was unable to express what she wanted. The features were considered to be helpful, specially the navigation and customisation with personal preferences. She did not have special concerns about privacy, but during the discussion about the possibilities of creating user profiles and tracking movements by recording the locations where the context aware feature is used, she agreed that there should be some kind of policy to protect private data because commercial interests might lead to an abuse. A consequence of the experiences with a non native english speaker could be to implement the multi language feature that is intended in the architecture.

5.4 EdFestLight

Due to the limited time and technical difficulties, no user study could be made with EdFestLight. The intended hardware platform for the systems, the Xybernaut systems failed on the first day so a private notebook had to be configured to act as client.

But the results from EdFestVoice show that most of the extended features were accepted and conceived as useful by the users. However it would have been interesting to find out if the lightweight design has some advantages for users or if the extended features are more convincing

Chapter 6

Conclusions

During the work on EdFestVoice, many interesting aspects have arisen and many detail problems had to be figured out and solved until the final prototype was completed. On one hand, technical questions about communication and the data flow between the VXML client and the database with the different levels of server side logic in between was a challenge, on the other hand, creating a user interface that is usable for practical applications required much attention. An important issue was making the best within the limits of VoiceXML itself. So the GrammarBuilder can be seen as the key technology to improve the quality of the user interface and provide a smarter way of interaction. The IBM VoiceXML browser could be easily embedded into the client. Maybe the quality of the text-to-speech engine (TTS) is not the state of the art any more, but the user study showed that users were satisfied with the artificial voice.

OMS is a very suitable platform for a system like EdFestVoice, as modelling the data structure comes in a very natural way with the object oriented paradigm. Although OMS Web Elements are still in a prototype phase, all desired features could be finally be realised with OMSwe. Only the lack of up-to-date documentation lead to some greater initial problems.

The greatest problems appeared on hardware side, the Xybernauts failed very early in Edinburgh, the GPS device was always detected as USB mouse and made the mouse cursor jump chaotically over the screen every time it was attached. The notebook batteries halved their capacity every day during the tests and limited the possibilities of testing the system on the street.

Nevertheless, the testing in Edinburgh was very interesting and had some promising results. Generally, the acceptance of the system was even better than expected. Users liked the features and were surprised of the technical possibilities. All design aspects of the system came into use and were appreciated by the test users. The major concerns that users might be irritated by the robotic voice or might find it strange to talk to a little mobile device in public have not been pointed out as weaknesses and the general idea of building a device to help tourists with orientation in the city and overview of the various events got only positive feedback. Also an interesting point was that the users were much less concerned about privacy than expected.

Further development on EdFestVoice could include a peer review system for

events and more effort in direction of voice based communication platform. The connection to standards like VoIP or POP mail could improve the system. Reminders for chosen events would be helpful, as well as connection to e-commerce systems for purchasing tickets or booking hotels, hiring cars or making reservations for restaurants. The navigation system could be extended to use public transport facilities, information sources from outside the system could be integrated like the possibility of getting up-to-date flight information or weather forecasts via web services. All the logic could be integrated into the flexible and scalable architecture of EdFestVoice. The multi-language support that is prepared in the client could be enabled if the content is made available in different languages, this would surely be an important selling argument. Last but not least the intended platform for the client software has always been a small and mobile device like a PDA or a wearable computer. A real life implementation should surely run on a platform like this and more intensive user study with a prototype like EdFestVoice would indeed be very interesting. People should use the devices for a day or more to become familiar with the features and use them in real life situations. However, the tests in Edinburgh can be seen as a good preparation for a long term studies and have already given some impressions what could be improved and which more extensive features could be integrated.

Appendix A

Task

Voice Interface for EdFest

Semester project

The task of this semester project is to study purely voice based user interfaces for applications in tourism. The semester project consists of two parts: the first part consists of developing two user interfaces for the Edinburgh Festival. Both interfaces are purely voice based. One of them heavily uses context (location, history, user preferences) in generating dialogues, the second one doesn't. Both systems should be developed in consideration of main insights from human-computer interaction. The second part of the semester project consists of a comparison of the two systems, with a short user study which reveals the more usable system.

Appendix B

Installation of EdFestVoice

B.1 Server Installation

EdFestVoice Server comes with OMSPPro / OMSwe and Tomcat as Servlet Container. Copy the server directory of the distribution to a hard-disk and start the server with start.bat. This will start both the Tomcat server at port 8080 and OMSServer at the port defined in the preferences. At the moment, OMS cannot be called with a command line parameter that opens a database so this has to be made manually as well as starting the OMS server. If the server is a Windows 2000 or Windows 2003 Server or Advanced Server, autoExNT.exe from the resource kit can be used to launch start.bat as a service on start-up and as soon as command line parameters are supported by OMS, the startup process can run completely in background.

B.2 Client Installation

The EdFestVoice client is ready to use on the distribution CD. It can either be copied to a hard-disk or to a memory stick, as mentioned in the hardware recommendations in Appendix C. To start the client, execute start.bat in the distribution's home directory. A java SDK (Sun J2SE 1.4.3) is shipped with the distribution and will be used by EdFestVoice Client. The VXML browser that comes with EdFestVoice is the IBM VoiceSDK, part of the WebSphere Development Kit. The TTS Engine is IBM ViaVoice Outloud, also part of the VoiceSDK. Please note that the IBM components might be under copyright protection.

The client includes a file called client.properties, that has to be adapted. The following entries must be set according to the setup:

EdFestVoiceServer:

UserName:

UserPref:

The only installation step that might be required is the installation of the VoiceXML browser. Check the installation guide in the according directory.

Appendix C

Technical Documentations

C.1 Hardware

In the original design, EdFestVoice was planned to run on a Xybernaut mobile device. During the tests in Edinburgh, the Xybernauts were unstable and faulty, so the test implementation ran on a Dell Inspiron 2650 notebook. For real life application, smaller and more mobile devices would be more suitable, however, at the moment there are only a few devices on the market that can run a full (desktop) J2SE distribution and feature a fully working sound system. EdFestVoice itself does not require a J2SE 1.4 and could easily be ported to a J2ME profile, but the IBM VXML Browser would require some serious reverse engineering to run under J2ME. IBM ViaVoice Outloud is available for PDA's, but this has not been tested.

The server should be at least a standard SMB server with really a lot of memory, mainly because of Tomcat.

As network connection, wireless lan, Bluetooth or GPRS could be used, the implementation of EdFestVoice expects to have a network layer supported by the operating system and does not provide any extended support for a specific network technology so for tests, a wireless network will be the right choice. In a final version, the communication could be implemented using all network types in parallel and choosing the one with the best signal quality or the lowest costs. The headset is activated through a button between the hardware and the computer's input. The button consists of an ordinary plastic membrane button soldered as an interrupter of the headset wire. The more advanced version has a volume control taken from an old Walkman.

If no button is used, a wireless bluetooth headset can be used. The button only works with wired headsets, for the user study, a mono earpiece with integrated microphone has been used to make EdFestVoice an ambient system.

C.2 GrammarBuilder

The GrammarBuilder is used to form grammars of significant and insignificant words to provide better recognition rates with VoiceXML. It consists of two layers. As a gateway to OMS, a simple prolog module is used to retrieve the

requested data. It's a workaround for the problem that AQL does not deliver both a projected attribute and the OID at the same time. On java side of the

```
macro([CollName, AttrName]) :-
    write('<result>'),
    get_coll(_, CollName, _, Coll),
    \+ (member(Oid, Coll),
        get_attrval_by_name(Oid, AttrName, AttrVal),
        write('<object>'), nl,
        write('    <oid>'), write(Oid), write('</oid>'), nl,
        write('    <attr>'), write(AttrVal), write('</attr>'), nl,
        write('</object>'), nl,
        fail
    ),
    write('</result>').
```

Figure C.1: OMS prolog module

server, OMSGrammarBuilder is a servlet. It expects the first calling parameter to be an OMS collection and the second to be an attribute to where the objects are to be projected. These phrases are returned from the prolog side when OMS is called together with the OIDs. The phrases are then internally represented by *OMSElements* and split into tokens. If a token has not occurred yet, it is added to the element and marked significant, this is the optimistic approach in OMSGrammarBuilder. Figure C.2 shows the different classes of GrammarBuilder, figure C.3 the main loop where the tokens are processed and the decision on significance is made.

The token is added to the word list so the next time the same word occurs, it will be marked as insignificant and the ElementTable that contains all Elements processed so far is notified that the first occurrence of the word has to be remarked. The ElementTable has a Hashtable saving every entered word together with the Element in which it has been encapsulated so re-marking is in constant time. In OMSGrammarBuilder, the process of remarking is called a go-back-once strategy as every Element can only be remarked once.

C.3 Navigation

The navigation was implemented to show some every-day application for context aware information system. Relative to the current position, a route to the desired location is calculated and stepwise read to the user. The algorithm used in EdFestVoice has been developed for the ARA project (Autonomous Robot Architecture, <http://robot.wlab.ethz.ch>) and modified to work with OMS. Streets are modelled as junctions and parts, as Figure C.4 shows. Each part connects two junctions and a street consists of one or more parts. Junctions have (GPS) coordinates that are used to find the nearest junction and to check if a part leads closer to the goal by calculating the difference of junction coordinates. If a street has corners, it must be modelled by two parts and an intermediate

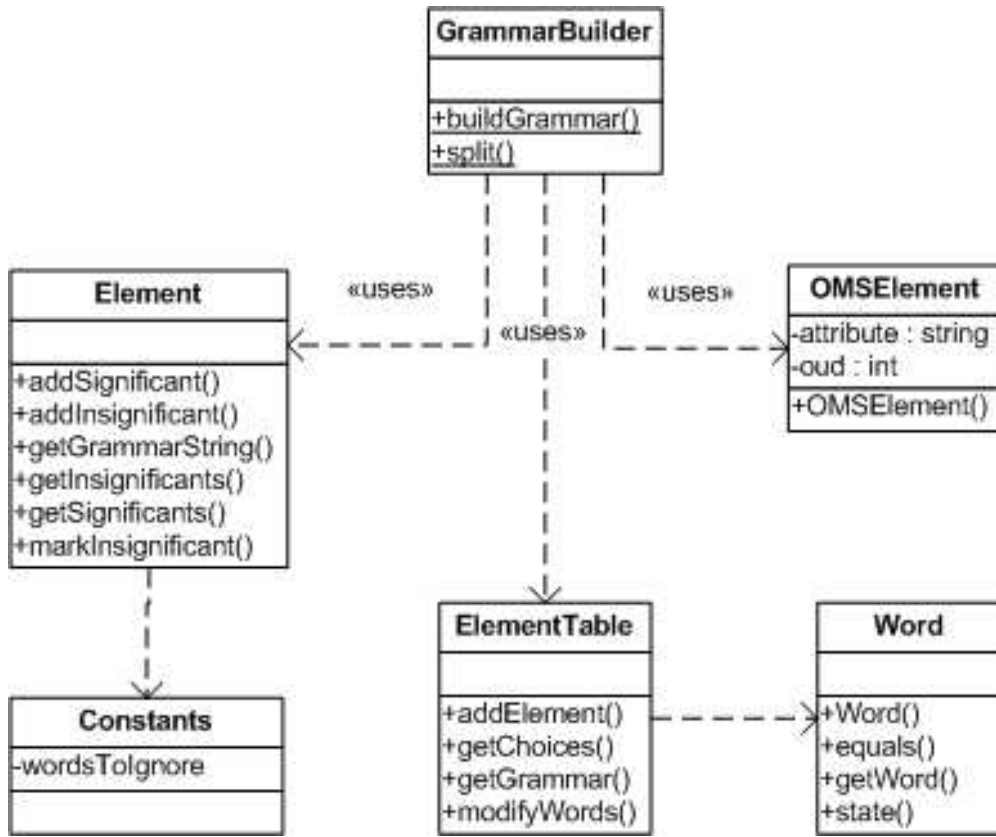


Figure C.2: GrammarBuilder Architecture

junction to keep the model consistent.

As shown in figure C.5, the navigation expects a starting point S and a goal G to be reached, together with a momentum MX, MY that is used to optimise the walk. Usually the momentum will be the direction from start to goal. J_s and P_s are the lists of junctions and the path list of the result. From the current position, all possible connections are tested. To reach better performance, the order of position is relative to the momentum, so connections that lead to a position closer to the goal will be tried out first. To avoid cycles, the algorithm works with assertions of visited positions. Within scenarios with dozens of connections and positions represented by integer values, the algorithm has worked fine. However, in the final system it has turned out to be buggy. This might be a result of the GPS coordinates that require high precision and some strange behaviour of the optimisation algorithm that is not stable for connections of junctions that are numerically close. But in general, the algorithm shows what might be possible in a further development.

```

public static ElementTable buildGrammar(Vector resultSet) {
    ElementTable choices = new ElementTable();
    Vector wordList = new Vector();

    // for all database entries
    for (Enumeration en = resultSet.elements(); en.hasMoreElements(); ) {
        OMSElement oms_obj = (OMSElement)en.nextElement();
        int oid = oms_obj.oid;
        String entry = oms_obj.attribute;
        String[] tokens = entry.split("\\s");
        Element element = new Element();

        // for all words in entry
        for (int i=0; i<tokens.length; i++) {
            String current = tokens[i];

            if (!wordList.contains(current)) {
                // word has not occurred before
                // so enter as significant
                element.addSignificant(current);
                // add to word list
                wordList.add(current);
            } else {
                // word has already appeared before
                // enter as insignificant
                element.addInsignificant(current);
                // tell the ElementTable to modify the word,
                // because it was added as significant the first time
                // and now changes state to insignificant
                choices.modifyWord(current);
            }
        }
        // put all significant words into Hashtable
        // so they can be easily be found in case
        // they become insignificant later
        choices.addElement(oid, entry, element);
    }
    return choices;
}

```

Figure C.3: Main loop of GrammarBuilder

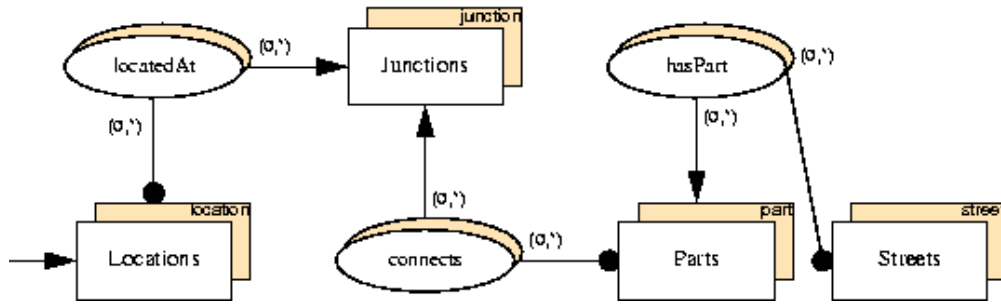


Figure C.4: Navigation schema

```

walk(S,G,MX,MY,Js,Ps) :-
    asserta(visited(G)),
    ((S = G) ->
    (
        asserta(visited(X)), Js = [], Ps = [], !
    )
    ;
    (
        tokens_to_token(['ran ((inv(connects) compose connects) dr
            (set[(',G,'):junction]))'],ConnQuery),
        aql(ConnQuery,_,ResColl),
        get_ext(ResColl, Connections),
        optimise(Connections, OptConns, MX, MY),
        if(member(Con, OptConns),
            (visited(Con) ->
                (
                    % write('already visited '),
                    % write(Con), nl,
                    !)
                ;
                (
                    walk(S,Con,MX,MY,Js1,Ps1),
                    append(Js1, [Con], Js),
                    tokens_to_token(['dom (connects rr
                        (set[(',G,'):junction])) minus
                        (connects rr
                            (set[(',Con,'):junction]))'],PartQuery),
                    aql(PartQuery,_, PartsColl),
                    get_ext(PartsColl, Parts),
                    member(Part, Parts),
                    append(Ps1, [Part], Ps),
                    !)
                ),
            fail)
    ).

```

Figure C.5: Navigation algorithm

Appendix D

Interviews

D.1 User 1

Protocol of the system test:

- information about events
- music events
- change categories
- film
- (system did not recognise) film
- user: "I think it's gone back to the start"
- tell the user that this might have happened, just go on
- find a location
- location by category
- find nearest theatre
- (system crashes)
- find a location
- location by category
- find nearest theatre
- (battery is low, finish testing)

Interview:

User: So this is a system that you guys have been developing ?

JR: Yes.

User: Wow, that's good. Where would you be using it, on the Internet ?

JR: No, we think everybody will have a little computer for the pocket and then you can carry it around and ask questions.

User: Oh, that's cool.

JR: You can even find locations with it, so you can find the next theatre and let the system tell you how to get there.

User: So will this be on own computers or on computers for the festival ?

JR: Well, we still don't know, maybe it's something that you can loan out for the festival or maybe on something that you can buy.

LV: Maybe something that you can lend for the festival and then give back.

User: Yeah, good for tourists as well.

LV: Yeah.

User: Will it be in different languages as well ?

JR: It already is.

User: So when do you hope to have this running, like next year or ... ?

JR, LV laughing.

LV: Good question.

JR: I don't think so.

LV: Not this year anyway.

LV: But do you think this would be helpful ?

User: Yeah, definitely. For instance if somebody is staying in a hotel, you could offer them the system to get around. Cause it is like when you go to museums, you can hire things that tell you in different languages where everything is and what they are. It's kind of similar, isn't it ?

LV: Yeah, exactly.

User: But can you store all your favourites ?

JR, LV: Yeah.

User: That's good.

JR: And we tried to make it a bit more smarter than the ordinary systems.

User: Yeah, definitely.

JR: We want the system to know your favourite categories.

User: Yeah, good. A reminder of: Oh, you are going to see this show, it is 6 o'clock this evening. That would be good.

JR: And we have this GPS, so the system knows where you are. If you want to find the next theatre, it knows that it must be somewhere around here.

User's colleague: Cleverer than the average human.

LV: (to the colleague) Was it annoying for you not to hear what she is hearing ?

User's colleague: No, I mean ...

User: It's always one person.

LV: So you would tell her afterwards, what you heard.

User: Yeah. It's always one who reads the map and find out where you are.

JR, LV: Okay, thank you.

D.2 User 2

Protocol of the system test:

- personal agenda
- tell the user to press the button while speaking
- personal agenda
- (quiet) compose a new message
- (louder) compose a new message
- user asks, what receiver can be entered, just one of his friends
- tell the user to use "jan" as this is currently the only user in the system
- (misunderstands) the only user
- tell the user to say "jan"
- (understands) jan
- Just seen a really good performance at the festival, it was a play I saw, (?), I think you should see it as well.
- tell the user to say "stop now"

- stop now
- yes
- main menu
- information about events
- music
- visual arts
- yes
- user does not know what to enter
- tell the user that the date is requested
- asks, how to enter the day, like Saturday or Sunday
- tell the user to say something like first of September or today or tomorrow.
- today
- tell the user that the system won't find any events because the database is still sparse
- user thinks that an example of how to enter the date would be helpful
- tell the user, that the second prompt is always more precise and the command "help" usually leads to an example
- user appreciates this
- find a location
- events near my current position
- user asks, if the witchery is the only in the database or why it went straight to this.
- tell the user, that in case of more than one matchings, a list of events is presented.

Interview:

User: So I don't know if this was any help to you.

LV: I think it was ...(noisy part)

User: Access help for any time. At the beginning, right at the beginning. Cause I didn't know the ? to do that.

JR: Okay, I didn't want to tell everything.

User: Oh no, that is fine.

JR: It's interesting to see how people react if they don't know anything about the system.

User: Yeah.

JR: If I would sell this, you would get some notes how to use it.

User: Okay, sure.

JR: But how did you like the context awareness ? We have the GPS device, and you can ask for the nearest restaurant. Does it make sense to you, would you use it ?

User: Yeah, it makes sense, I can really see, like, Americans using it.

JR, LV: laughing

User: No, yeah, to guide their way around the festival. It's very useful. Is it like, can I have the whole list of festival events ?

JR: Yeah.

User: Yeah, that's very handy.

JR: You could even ask the system to guide you the way to a special location. This isn't completely implemented, but with this GPS device, if you have the location that you want to go to, you can say, navigate me to this location.

User: Wow, okay.

JR: It will say something like: "turn left, turn right, go straight ahead" ...

User: That's really good. No, I think, this is a great idea.

JR: Wouldn't you mind that the system periodically sends your coordinates to a server, I mean, this has something to do with privacy.

User: No, no really, because, it's not something that you want to be particularly, you know, you are on holiday, it's not something that is personal, you know. Unless you are a criminal or something. You are not bothered if somebody knows where you are.

JR: So you would also tell the system your preferred categories ?

User: Yeah. Yeah, you can do that and you say and then it can suggest examples to you where you should go next.

JR, LV: Okay, thank you !

D.3 User 3

Protocol of the system test

- find information about events (user didn't press the button)
- tell the user to try it again
- user says that we can't talk to her while the system is talking to her otherwise she can't concentrate.
- find location
- tell the user to press the button all the time during talking to the system.
- find location
- the news theatre
- back to main menu
- (system crashes, restarting it)
- user asks, if she doesn't know a location, how can she ask for something that she doesn't know.
- little discussion between user and LV what the user is expecting. She wants a list because she doesn't know any location.
- tell the user that she can have this, explain the effects of iterative matching.
- location
- find a specific location
- find a specific location
- I don't know the location
- tell the user, that this was a good input because the system will react to "don't know"
- a theatre
- Princess garden
- user says that she is waiting for the list of locations
- tell the user that is maybe because there is not so much data in the database yet, but she can say "greater radius" to search in a larger periphery
- greater radius
- larger radius

Interview:

LV: (to the colleague of the test user) How was it actually for you, who could not hear what she was hearing, were you wondering, like ...

Colleague: I thought that something is going wrong, because she didn't speak much.

LV: Ah, okay. And the other question is, do you think, this kind of system will be of some use in Edinburgh, with all the festivals and all the events ?

User: Yes, sure, but it would be easier if you had something to look at as well.

JR: So it is hard for you to imagine what kind of actions you can do at a certain stage ?

User: Yeah. Specially if I don't know what's on.

JR: So, would you like to have a printed map with all the menus, if you don't know what you can do, you can look it up, like, okay, from this point, I can get information about locations.

User: Yes, that would be useful.

JR: Do you use a lot of technical equipment at home ?

User: What is a lot of technical ...

JR: Well, would you talk to a telephone answering machine of a person that you don't know ?

User: If it is something important that I have to say, then, yes.

JR: So you have no problem to talk to a computer ?

User: No, as long as I know what it wants from me.

JR: And, I think you didn't use that feature, but we have this feature that you can find the nearest restaurant, and, we have this GPS device, so the system knows where you are.

User: Oh, I see.

JR: Would you find this helpful, for example, you are here and you are hungry. So you ask the system where the next restaurant is ?

User: Yeah, why not. And it can maybe also tell me where the next Italian restaurant is and the next, I don't know, yeah, that would be great.

JR: And the system can also tell you the way to this location, when you say "navigate me to this", it will tell you "turn left", "second junction" and so on. Would you use it ?

User: Would I use it, if it was somewhere here, just standing around here, yeah, maybe, why not, yeah.

JR: Would you mind that the system somehow transmits your position to a server. I mean, this is a problem of privacy. It doesn't really know who you are but it does send your coordinates to a server. Would this be a problem for you ?

User: Never thought about that, but I mean, when I'm away, it doesn't know where I am going, I can go to that restaurant or I can go somewhere else.

JR: I mean, if we build this system up, it will be owned by somebody, maybe by the festival ... And they could use this data to create a profile of you, they don't know you but they could say, okay, you are the person that likes Italian restaurants and likes these venues.

User: And then I get advertising things.

JR: Yeah. We also have voicemails, maybe you would get voice spam then with advertising for Italian restaurants. I hope, this will never happen but would you mind about that ?

User: Yeah.

JR: Okay, so you would want the system to be closed and not public for commercial interests.

User: Yeah, that's right.

JR: Then we have this feature that you can tell the system, what your preferred event categories are, for example that you are mainly interested in theatre, books, something like this. And if you ask the system about events for today, it will use this information to find the events that you are interested in. Would you use something like this ?

User: The system tells me what I am interested in ?

JR, LV: No, no.

JR: You will have to tell the system.

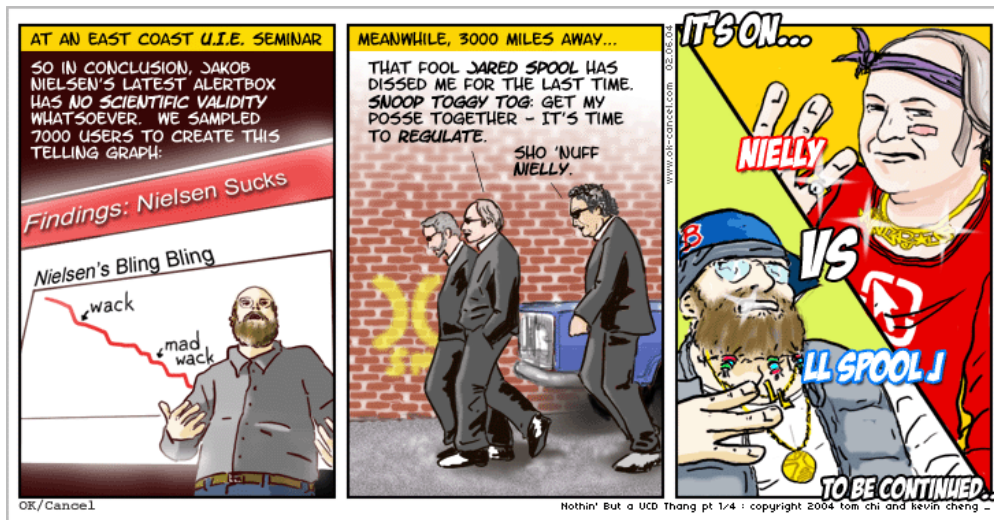
User: Of course, that's okay. As long as it stays there.

JR, LV: Okay, thank you.

Appendix E

The least

Usability guru clash:



(found on OK/Cancel).

Bibliography

- [1] R. Beasley, V. Bonnewell, M. Farley, J. O'Reilly, J. Quire: Design Differences: VUIs Versus GUIs In: informIT, Indiana, USA, May 2002
- [2] B. Duggan and M. Deegan: Consideration In The Usage Of Text To Speech (TTS) In The Creation Of Natural Sounding Voice Enabled Web Systems,
- [3] J. Eisenzopf: Top 10 Best Ptactices for Voice User Interface Design In: <http://www.developer.com>, April 2003
- [4] M. Farley: User-Centered Design for VoiceXML Applications In: VoiceXML Forum (<http://www.voicexmlforum.com>)
- [5] A. Hoffman: Speaking the Language of Voice XML In: Monster Technology.
- [6] T. Moran and P. Dourish: Context-Aware Computing is: Human-Computer Interaction, Volume 16, 2001
- [7] M.C. Norrie and A. Wuergler: OMS Pro, Introductory Tutorial, Technical Report OMS Pro Version 1.0.1, Zrich, Switzerland, March 2000
- [8] M.C. Norrie and A. Palinginis: Empowering Databases for Context-Dependent Information Delivery, Ubiquitous Mobile Information and Collaboration Systems (UMICS 2003), Klagenfurt/Velden, Austria, June 2003
- [9] C. Schmandt and E. A. Hulteen: The Intelligent Voice-Interactive Interface, Massachusetts Institute of Technology, Boston, USA, 1981
- [10] A. Truchard (editor) and R. Katz-Haas: Ten Guidelines for User-Centered Web Design In: Usability Interface, Vol 5, No. 1, July 1998