A Look Under the Hood - Design and Development of the First SmartWeb System Demonstrator

Norbert Reithinger, Simon Bergweiler, Ralf Engel, Gerd Herzog,
Norbert Pfleger, Massimo Romanelli, Daniel Sonntag
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
norbert.reithinger@dfki.de

ABSTRACT

Experience shows that decisions in the early phases of the development of a multimodal system prevail throughout the life-cycle of a project. The distributed architecture and the requirement for robust multimodal interaction in our project SmartWeb resulted in an approach that uses and extends W3C standards like EMMA and RDFS. These standards for the interface structure and content allowed us to integrate available tools and techniques. However, the requirements in our system called for various extensions, e. g., to introduce result feedback tags for an extended version of EMMA. The interconnection framework depends on a commercial telephone voice dialog system platform for the dialog-centric components while the information access processes are linked using web service technology. Also in the area of this underlying infrastructure, enhancements and extensions were necessary. The first demonstration system is operable now and will be presented at the Football World Cup 2006 in Germany.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Natural language; I.3.6 [**Methodology and Techniques**]: Interaction techniques

General Terms

Human Factors, Algorithms, Design

Keywords

Multimodality, Semantic Web, Interaction Design

1. INTRODUCTION

The process to realize a new project in the area of multimodal interaction systems should be straightforward: you write down your list of needed functionalities and technical requirements, read the relevant literature and collect all the technology you have on your shelves or that is available to you. Then you devise a good interaction metaphor, make some user studies and build your system.

In this contribution we describe the first steps in the development process for the project SmartWeb (`www.smartweb-project.de`, [21]). Together with a number of partners from industry and academia in Germany we started in 2004, and aim at the development of a context-aware, mobile, and multimodal user interface to the Semantic Web [8]. In our main scenario, the user carries a smartphone as interaction device – other scenarios are the integration in a motor-bike interaction device and in a car. The user poses open-domain multimodal questions in the context of a visit to a Football World Cup stadium in 2006. Using speech and gestures, she can ask for information about players, events, weather information and other services available through web services, Semantic Web pages, and the plain Web. The user can interact with the result by speech and gestures on the device, and browses multimodally through alternative results.

The multimodal input is interpreted and transmitted using UMTS or wireless LAN to a back-end server system. The multimodal recognizers, the dialog system, and the Semantic Web access subsystems are located on this server. They should be able to serve multiple end-users. The response to a question is presented on the mobile device and rendered on its screen.

The partners in the project share experience from earlier projects [20, 16, 17] where different sub-components were integrated to multimodal interaction systems. Like others [15] we learned some lessons which we use as guidelines in the development of our system [2]:

Multimodality More modalities allow for more natural communication, which normally employs multiple channels of expression. It is also the case that more modalities constrain interpretation and hence enhance robustness. However, the use of multiple modalities not only requires a lot of technical effort in the recognizers or generators, but also in the interaction design to get full leverage out of the employed technologies.

Representation Representation is important in two aspects. Firstly, in a complex interaction system a common ground of terms and structures is absolutely necessary. A shared representation and a common knowledge base ease the data flow within the system and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICMI'05, October 4–6, 2005, Trento, Italy.
Copyright 2005 ACM 1-59593-028-0/05/0010 ...$5.00.

avoid costly and error-prone transformation processes. Secondly, the representation is part of the global dialog history residing in the discourse module and can be accessed by any module, e. g., for multimodal reference resolution or generation at any time during the course of the dialog. (c.f. "No presentation without representation" [14])

Standards Standards ease scalability. For some representations and tools, we have previously developed custom-built software providing short-lived solutions. With the advent of standards like EMMA (see below), adopting these standards opens up the door for scalability since we can re-use ours as well as other's resources.

Interface In the case of a multi-module approach, use one well-defined representation for module communication. This combines our guidelines for representation and standards, which facilitate clean and well-defined interfaces structures. Since most public standard approaches are nowadays based on XML and the corresponding XML Schema language, you get tools for a lot of tasks for free.

Encapsulation Encapsulate the multimodal dialog interface proper from the application(s) as far as possible. This is similar to the Model-View-Controller design pattern [12], and as in traditional GUI-oriented interfaces, decouples the interaction logic from the application.

In this contribution, we will first present the interaction ideas and design rationale we derived from previous experience and talks with prospective users. We will then present the current architecture which consists of three major subsystems that are connected using clearly defined interfaces. Besides the overall architecture, we will present the dialog system and the interfaces we selected. They are based on standards like EMMA and RDFS which were extended for our use.

2. INTERACTION DESIGN

SmartWeb allows the mobile user to send requests to various services that are linked by a Semantic Web framework. The user can:

- Ask simple factoid and inspection questions or commands to search, explore, and inspect information.
- Ask for special services (action sequences) like online form filling. In this case, SmartWeb only forwards the query to an external service, instead of modeling the external service in the dialog.
- Control the system. He can ask for status information and cancel a running query.

We group the interaction modalities in our system into three major classes:

- Auditory: speech input and output – through a high-quality speech synthesizer – and earcons.
- Graphical: all modalities serving as input device on the screen, touch or stylus input, the keyboard – both the software one that comes with the mobile's operating system and the built-in sliding keyboard beneath the device –, and for output the graphical display itself.
- Keys: input buttons and the cursor joystick - to meet basic user interface principles, often referred to as quality dimensions such as simplicity, discoverability and efficiency.

Basic user interface principles [13, 4, 9] should still apply, most of which are prominent on a standard desktop PC and which were transported to the mobile area. This concerns the joystick use, for example. We do not plan to override the simple navigation metaphor. We allow for navigation through the input and output via the joystick buttons. This ensures system discoverability along an intuitive interaction mode. In detail we

1. Allow for feedback from user input.
2. Offer correction possibilities.
3. Provide interface simplicity by progressive disclosure. Since the screen display size is extremely limited on the mobile device, we do only display the most important information or functionality to the user. On the other hand, less frequently used functionality should also be covered, and can be discovered by, e. g., pull-down menus or speech input. In the latter case, the set of general commands, throughout available, is supposed to be very valuable.
4. Provide status information to the user.

The user should also be able to use the device in one hand: The most prominent functions are available by dedicated buttons on the device whose assigned function does not change. Accordingly, we define a set of general commands and a set of control words by speech input. Other programmable buttons are allocated in a dynamic fashion, depending on the context.

Figure 1: Display areas

The screen of the handheld device we currently use, a MDA III depicted in Figure 1, has a QVGA resolution of 240x320 pixels. However, we want a solution that scales up to advances in the handheld area. Therefore, the display design must be independent with respect to a certain resolution. We currently assume that it has a portrait orientation.

Based on a storyboard we asked users informally about the possible use of the device and developed a first version. We define several fixed regions in the user interface

(see Figure 1). In the top region 1 we display, besides some status icons, a direct feedback to the user's input. First, the recognized words are presented with the possibility to directly edit the input by manipulating the words with the stylus. This section also displays parts of the dialog history and the status information, e. g., whether the microphone is on/off, and a progress bar. In 2, we display the paraphrased question, e. g., in a template-based form before the request is sent to the Semantic Web. Again the user can edit the paraphrase. If a result from the Semantic Web access components is ready for display, it replaces this paraphrase.

The result media types are text, picture, video, audio, graphic, web page, link, and answer snippet (e.g., factoid answer). For pictures, audio and video, additional functionality such as length description, the source pointer, and play on demand are provided. In 3 we display, depending on the context, icons for result media types, interruption possibilities, a legend of symbols and colors and dynamic button allocations, and system control and system status.

Figure 2: The realized GUI of SmartWeb

Figure 2 shows the first realization of the interface based on this design considerations. The user asked for the year of the first win of the World Cup for Germany. The spoken request is visualized on the screen, together with the answer *1954* and a picture of the winning goal together with the caption *Rahn scores the 3:2*.

3. ARCHITECTURE

3.1 General Architecture

Within our main application scenario that focuses on a personal handheld device – which later will be integrated in vehicles – a central goal is to offer the mobile user a flexible multimodal interaction. This is realized by powerful processing components running on a remote server.

A flexible dialog system platform is required in order to allow for true multi-session operation with multiple concurrent users of the server-side system as well as to support audio transfer and other data connections between mobile device and dialog server.

Advanced intelligent multimodal interface systems usually comprise many sub-systems. Needed functionalities are, e.g. modality recognition for speech or gestures, modality interpretation and fusion, intention processing, modality fission and, finally, result rendering for graphics on a screen or synthesis of speech. For many of these sub-tasks software modules from academia or industry can be used off the shelf. Furthermore, in many projects integration frameworks for this types of systems have been developed and exist, like the Galaxy Communicator, Open Agent Architecture, Multiplatform, the General Architecture for Text Engineering or Psyclone [18, 6, 5, 11, 19].

In the commercial area we evaluated platforms from major vendors like VoiceGenie, Kirusa, IBM, and Microsoft. The platforms offer speech and browser based interaction using X+V[1], HTML+SALT[2], or derivatives. All can be used on mobile devices. However, for our purposes these platforms are too limited. To implement new interaction metaphors, we need an open platform that can be adapted to and extended for our requirements. For the basic infrastructure and the architecture we identified the following as central:

- Openness, especially for Semantic Web technology;
- Scalability to multi-user operation;
- Integration of multimodal server- and handheld based operations.

Instead of using an unsuitable off-the-shelf solution, one of our project partners, Sympalog Voice Solutions[3], contributed their commercial speech dialog system platform that is being adapted and extended for multimodal interaction. The platform runs on Linux and contains all necessary functionalities for call handling over phone lines, soundcards, or VoIP. It ships with client software for the PDA which runs Microsoft Windows Mobile 2003. It takes care of the basic infrastructure, providing IP connectivity between server and client. The server side call-manager also provides stubs for speech recognition, synthesis and the dialog manager, and manages multi-user calls and base-level barge-in processing.

On top of this platform we developed the basic architecture that is shown in Figure 3. It consists of three basic processing blocks PDA client, server system platform and dialog system.

The client running on the user's PDA. It consist of a local Java based control unit which takes care of all I/O. It is connected to the GUI-controller which is realized using Macromedia Flash with ActionScript. Since the graphic realization on a small device is challenging, a professional environment like Flash provides for a straightforward and graphically satisfying development environment. The use of non-device specific programming languages and environments like Java and Actionscript/Flash adds another positive aspect: For development and test purposes we can run these components on a standard desktop computer. At a later stage we will have a local VoiceXML based dialog system for interaction during link downtimes, camera and GPS based services connected to the controller.

On the server, the speech system platform instantiates one dialog server for each call, and connects the multimodal recognizer – currently speech recognition, later to be enhanced with head/gaze recognition –, speech synthesis and the dialog system.

[1] `http://www.w3.org/TR/xhtml+voice/`
[2] `http://www.saltforum.org/`
[3] `http://www.sympalog.de`

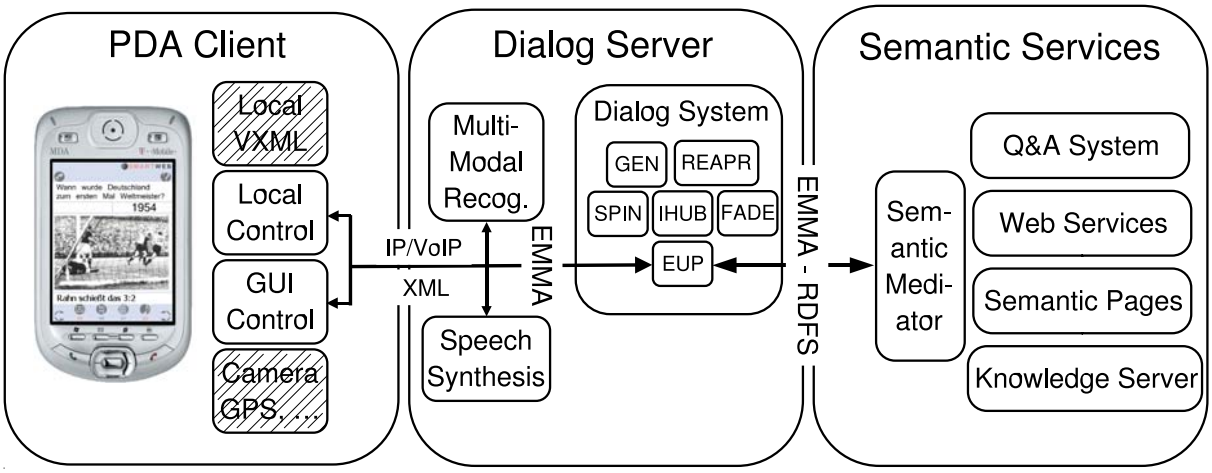

Figure 3: The basic architecture of SmartWeb

The dialog system instances send the requests to the Semantic Services, which provide the umbrella for all different access methods to the Semantic Web we will use. It consists of an open domain Q&A system, specialized web services, Semantic Web pages and a knowledge server. A component called Semantic Mediator encapsulates all these services.

3.2 The Dialog System

The dialog system itself consist of different, self-contained processing components which were partially used in our prior projects. To integrate them we developed a Java based hub-and-spoke architecture called iHUB. It is similar to the Galaxy Communicator and the Open Agent Architecture (OAA). In contrast to OAA the hub in our system does not reason about the content of a message. It only routes the messages between the components and controls their validity. Since all the components in the dialog system are written in Java, we don't have to accommodate different programming languages.

All messages are routed through the iHUB. It knows about the scheduling requirements between subcomponents and the addressee of a particular messages. The internal communication between components is based technically on the exchange of objects containing ontological information, using the standard Jena Semantic Web framework[4] for Java. The most important processing modules in the dialog system connected in the iHUB are:

- A speech interpretation component (SPIN) parses the N-best word chains from the speech recognizer. The output is an ontological description for the analyzable parts of the word chain (see next section for an example).
- A speech generation module (GEN) gets the internal representation of an utterance to be presented to the user and generates a Speech Synthesis Markup Language (SSML)[5] document to be realized by the speech synthesizer.
- A modality fusion and discourse component (FADE) keeps track of the ongoing discourse, completes different types of anaphora, and merges input from different modalities. Input is mainly a parsed word chain, and the gesture description on an ontological level with geometric position. The output is the enhanced word chain (e.g. resolved references) and the completed ontological description of the user's utterance.
- A system reaction and presentation component (REAPR) gets all input from SPIN (the processed best word sequence) and FADE (the query's words and completed ontological description). These intermediate results are presented on the PDA screen and can be edited by the user. If the user changes the paraphrase or original input, the message is sent back to the originating module for reinterpretation. Finally, REAPR sends the user's request to the Semantic Mediator. The results obtained from the Semantic Mediator are prepared for presentation and rendered on the mobile device.
- An EMMA Unpacker and Packer (EUP) component provides the communication with the external world and communicates with the other modules in the system, namely the Multimodal Recognizer, the Semantic Mediator which is the gatekeeper to the Semantic Web Access subsystem, and the Speech Synthesis.

3.3 Accessing the Web Services

For the realisation of the application core of SmartWeb, web-based communication protocols are being employed and combined. Between the dialog server and the Semantic Services, as well as within the services we use Java based web services as middleware technology. Again, this is a well defined interface using standardized data descriptions and procedures. In SmartWeb, web services are stateless application components, that support the internal interconnection of the different functional modules.

Web services achieve interoperability by means of XML-based standards, such as WSDL (Web Service Description Language)[6] and SOAP (Simple Object Access Protocol)[7]. This interoperability is the foundation for the exchange of

[4] http://jena.sourceforge.net
[5] http://www.w3.org/TR/speech-synthesis
[6] http://www.w3.org/TR/wsdl
[7] http://www.w3.org/TR/soap

application-specific data between individual processing components.

SOAP, an XML language, is used to describe messages and their parameters. It is independent of the message protocol, so the communication can be synchronous or asynchronous. The specification defines a structured framework for the transmission of XML messages over a transport protocol. WSDL is a type-safe and platform independent interface. The set of XML-based open standards and the web service architecture allow to build a homogeneous platform-neutral system environment with loosely coupled applications. For the specification of functional interfaces, the idea is not to refactor complex component-specific APIs as web services but instead, the main focus is on explicit representation structures for application-specific data, which can be transferred using XML-based documents.

The Dialog Server accesses the Semantic Web through a web service with a clearly defined interface. All application specific reasoning and processing is hidden behind the SOAP/WDSL API. The content of the interface structure is based on our common ontology as described in the next section.

The web services within SmartWeb are created with standard software, namely with Apache Axis (Apache Extensible Interaction System) in combination with the Apache Tomcat Server. This is one of the most popular Open Source web service toolkits and it provides a common set of tools, that are needed to define, test and deploy web services and the corresponding clients that interoperate with other web services and clients.

4. INTERFACES

As mentioned in the last section, SmartWeb partly consists of available components from the project partners, which are reused and extended in this project. To enable the components to communicate with each other, we use existing standards, which are extended where needed.

4.1 Using EMMA

The data exchange between the components of the system in the Dialog Sever is based on EMMA. As specified in the first section of the EMMA requirements document[8], EMMA was basically conceived for the representation of user input. The specification contains all necessary administrative information for a multimodal system, like mode type and time stamps.

The speech recognizer used in SmartWeb already generates word lattices, but in its own format. It was quite straightforward to adapt it to the format as required for EMMA. Figure 4 shows a (shortened) example for a recognizer output. We use the word lattice format, however we do not use alternatives in the lattice itself. Currently, we use the best 5 utterance interpretations which are represented in the lattice format. With the time stamps for each word, we have information needed for modality fusion, if the user uses a gesture together with speech.

For gestures and keystroke events, we do not use EMMA or InkML[9]. The presentation component REAPR is directly connected to the controller on the PDA. It uses a shortened, special XML format for data exchange. This is due to the limited processing power of the PDA, which does currently not allow for powerful XML parsers and generators. The limited bandwidth between PDA and Dialog Server must also be shared with the VoIP data transfer. However, for the future also this data should use a standardized exchange format.

```
<emma:emma version="1.0">
  <emma:interpretation emma:id="MMR-100"
      emma:process="MMR#15193" emma:turn-id="7"
      emma:lang="de" emma:start="1112271717920"
      emma:end="1112271722784" emma:mode="speech">
    <emma:one-of id="MMR-7">
      <emma:interpretation id="MMR-7-1"
          emma:confidence="1.00">
        <emma:lattice emma:initial="1" emma:final="5">
          <emma:arc emma:from="1" emma:to="2"
              emma:start="1112271718970"
              emma:end="1112271719300" emma:confidence="1.00">
            wer
          </emma:arc>
          <emma:arc emma:from="2" emma:to="3"
              emma:start="1112271719310"
              emma:end="1112271719630" emma:confidence="1.00">
            war
          </emma:arc>
          <emma:arc emma:from="3" emma:to="4"
              emma:start="1112271719640"
              emma:end="1112271721080" emma:confidence="1.00">
            1990
          </emma:arc>
          <emma:arc emma:from="4" emma:to="5"
              emma:start="1112271721090"
              emma:end="1112271721870" emma:confidence="1.00">
            Weltmeister
          </emma:arc>
        </emma:lattice>
      </emma:interpretation>
      <emma:interpretation id="MMR-7-2" emma:confidence="1.00">
        ...
      </emma:interpretation>
    </emma:one-of>
  </emma:interpretation>
</emma:emma>
```

Figure 4: EMMA encoded output of the speech recognizer

In order to properly represent also the output side and at the same time using a common interface language for communication among all system components, we decided to introduce an extention to the EMMA standard with specific output representation constructs. They are in the namespace `swemma`. In the following we present the two most important extentions:

The Result Tag The EMMA standard currently handles only interpretations. We could have use the `interpretation` tag also to represent, e. g., the input to the synthesis. However, we felt that an own tag for results is needed. This tag may occur as immediate child of the emma tag. In the result tag we permit also the use of Speech Synthesis Markup Language (SSML) tags to control the synthesis paramters. Figure 5 shows the use of the tag for synthesis input.

The Status Tag In the internal communication of SmartWeb we need to deliver a status of the progress and the result content at the different levels of the processing. This is information is provided in a `status` tag that together with internal turn-ids and timing information contains also instances of the discourse ontology.

[8] `http://www.w3.org/TR/EMMAreqs`

[9] `http://www.w3.org/2002/mmi/ink`

```
<emma:emma version="1.0">
 <swemma:result emma:id="DIA123"
                emma:process="DIA#42"
                emma:turn-id="42"
                emma:lang="de"
                emma:start="1087995961542"
                emma:end="1087995963542"
                emma:mode="speech">
  <swemma:result emma:confidence="1.0">
    <emma:derived-from emma:resource="#spin1"/>
    <speak version="1.0"
           xsi:schemaLocation="www.w3c.org/speech-synthesis.xsd"
           xml:lang="de">
      1990 war Deutschland Fussballweltmeister.
    </speak>
  </swemma:result>
 </swemma:result>
</emma:emma>
```

Figure 5: An (abbreviated) example of the use of the result tag.

4.2 The Common Ontology

User utterances can be categorized in four different classes:

- Open domain queries, i.e., the information is searched in the Internet
- Queries on information accessible through the common knowledge base
- Request for web services, e.g., a service providing pictures from web-cams
- Control commands, e.g., play a video found in the Semantic Web or repeat the last system utterance

All utterances except the open domain queries are represented in terms of a system-wide used knowledge source. As we are in the area of the Semantic Web, we made the obvious decision to use the standard formats and tools developed in that area. The Web Ontology Language OWL[10] is the current standard. In order to ease processing and to ensure a benign runtime behavior we currently use RDF schemata to represent the knowledge of the system. This representation formalism is immediately accessible to Jena. The use of the standard in ontological representations opened up the use of SUMO (Suggested Upper Merged Ontology) to jump start the development of an upper level ontology. Since the representation decisions in SUMO are sometimes not well founded, we integrate the methods as proposed in DOLCE [10] to clean the structure up a little bit.

The SUMO based knowledge provides an upper level of terminology. The web service requests and the control commands are represented using specific classes in the ontology which subsume SUMO classes. For the purpose of representing the human-computer interaction we developed a discourse ontology that gives specific account for the dialog acts possible in context of the application. The system-user dialog is modeled over two basic concepts originating in the question answering world: `question` and `answer`. These are modeled in a dialog act hierarchy (cf. [3]) as the most relevant concepts for the interpretation and representation as depicted in Figure 6:

[10] `http://www.w3.org/TR/owl-features`

Question This concept subsumes all questions posed by the user, except clarification requests which are only posed by the system. Questions are further subdivided in disjunctive polarization and general.

Answer An answer can both be an intermediate answer or the final result and may contain multiple answer type instances. Types like Location and Person are aggregations (and no subconcepts) to the answer concept. Answer types are distinct on their pragmatic value as *abstractAnswer*, *additionalAnswer*, *semanticAnswers* and *unknownAnswer*.

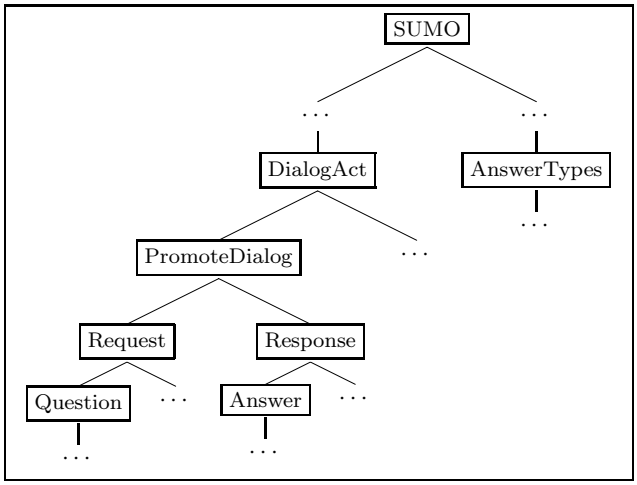

Figure 6: An excerpt from the discourse ontology hierarchy.

4.3 Representation of Queries

User queries are expressed as partially instantiated instances of the ontology which are used as search patterns in the Semantic Services. The instance representing the search pattern has to subsume an instance in the ontology, to select this instance as result. We prefer search patterns over queries formulated in a query language, like RQL (RDF Query Language) because search patterns allow to add additional contextual information relatively easy and straightforward: The information can be inserted as additional properties using standard techniques like overlay [1].

The contextual information can include information from former utterances, e.g., temporal information, or default values for certain properties, possibly determined using discourse or world knowledge. The additional information allows to formulate more powerful queries and is inspired by features of SQL. Possible additional information includes:

- The focus of the question, i.e. which part of the search pattern represents the instance or atomic value the user asks for.
- Constraints on properties containing atomic values, e.g., that the value has to be greater than a certain number or has to be the largest one of all found results.

```
<rdf:RDF
    xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://smartweb.org/ontology/emma#"
    xmlns:j.1="http://smartweb.org/ontology/discourse#"
    xmlns:j.2="http://smartweb.org/ontology/sportevent#">
  <j.0:Emma>
    <j.0:container>
      <j.0:Interpretation>
        <j.0:container
             rdf:resource="http://smartweb.org/ind#i2"/>
        <j.0:lang rdf:datatype=
          "http://www.w3.org/2001/XMLSchema#string"
        >de</j.0:lang>
        <j.0:end rdf:datatype=
           "http://www.w3.org/2001/XMLSchema#long"
        >1114605785</j.0:end>
        <j.0:turnId rdf:datatype=
          "http://www.w3.org/2001/XMLSchema#string"
        >42</j.0:turnId>
        <j.0:start rdf:datatype=
          "http://www.w3.org/2001/XMLSchema#long"
        >1114605781</j.0:start>
      </j.0:Interpretation>
    </j.0:container>
  </j.0:Emma>
  <j.0:OneOf rdf:about="http://smartweb.org/ind#i2">
    <j.0:container>
      <j.0:Interpretation">
        <j.0:container
          rdf:resource="http://smartweb.org/ind#i4"/>
      </j.0:Interpretation>
    </j.0:container>
  </j.0:OneOf>
  <j.1:Query rdf:about="http://smartweb.org/ind#i4">
    <j.1:text rdf:datatype=
      "http://www.w3.org/2001/2001/XMLSchema#string"
    >wer war 1990 Weltmeister</j.1:text>
    <j.1:dialogueAct>
      <j.1:Question />
    </j.1:dialogueAct>
    <j.1:focus>
      <j.2:DivisionNationalTeam
         rdf:about="http://smartweb.org/ind#i5"/>
    </j.1:focus>
    <j.1:content>
      <j.2:WorldCup>
        <j.2:heldOn rdf:datatype=
          "http://www.w3.org/2001/XMLSchema#string"
        >1990</j.2:heldOn>
        <j.2:winner
             rdf:resource="http://smartweb.org/ind#i5"/>
      </j.2:WorldCup>
    </j.1:content>
    <j.0:confidence rdf:datatype=
      "http://www.w3.org/2001/XMLSchema#float"
    >0.75</j.0:confidence>
  </j.1:Query>
</rdf:RDF>
```

Figure 7: Abbreviated RDF representation for the interpretation of *"wer war 1990 Weltmeister?"* (*who was world champion in 1990?*). The tag `j2:content` contains the query pattern, the tag `j1:focus` marks that the user asks for the winner of the World Cup.

- Functions to condense the results to a single value, e.g., the amount of all results or the sum of all values of a certain feature.

The reason not to include the described additional information in the search pattern, but to keep it in a separate structure is a technical one. Otherwise the top class of the SUMO ontology would have to be extended with query specific features, like a feature to store the focus of the query. Additionally, constraints on atomic values cannot be included directly as it is not possible in RDFS to define the range of a property to be either an atomic value or an individual representing a constraint.

The dialog system analyses the EMMA word lattice structure from the recognizer and returns as result a list of possible interpretations represented as instance structures using the classes and properties of the system-wide used ontology. The internal structure of Jena objects can be serialized as RDF document and embedded in a SOAP message to be sent to the Semantic Mediator. However, the information in the originating EMMA message like begin and end time or turn number is also of interest for the application services. They are also necessary to associate the corresponding answers to the originating queries. EMMA and our extensions are defined by an XML schema. It was straightforward to add classes and relations to our discourse ontology which represent the information contained in the EMMA schema.

4.4 A Detailed Example

Figure 7 shows an example RDFS structure that is used as query to the Semantic Web. Since the RDF structures can be hard to read, Figure 8 shows the basic instance structure of the query. The top query element contains, besides the string of the analyzed input, a substructure from the sport-related part of the ontology about world cup, and a focus pointer to the entity the user wants to know.

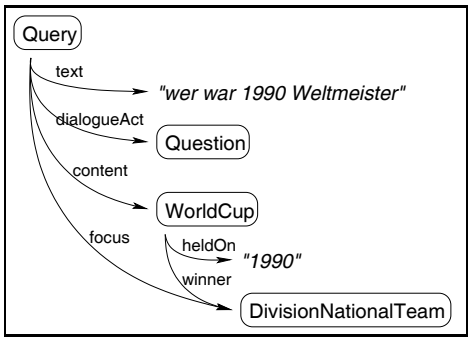

Figure 8: Graphic representation of the query

To build up the interpretation of the user utterance we use an existing semantic parser [7]. The parser is basically a rewriting system working on typed feature structures without detailed syntactic analysis. This suffices for the task and increases the robustness against speech recognition errors and disfluencies produced by the user. An outstanding feature of the parser is the availability of order-independent matching, i.e., the order of input elements does not always matter. This feature simplifies processing of free word order languages like German and enhances the robustness. Optimizations include a fixed rule application order and look-ahead tables to prune sub-optimal results.

The rules are partly manually and partly automatically created using linguistic information stored in the SmartWeb ontology and a set of meta rules. The linguistic information is associated with classes and is used for various tasks like the offline information extraction and text generation. Currently, the linguistic information contains only words or phrases in different languages. But we are working on an extension of the representation, e.g., to express the correspondence between features in the ontology and the roles of a verb. Meta rules generate new rules based on the

provided linguistic information using the same parser. This is possible as the parser is just a rewriting system for typed feature structures and the linguistic information is already available in RDF which can be easily transformed to a typed feature structure.

5. CONCLUSIONS

In this paper we presented the design and development decisions for the fist demonstrator of SmartWeb. Our research goal is the development of a mobile system for the open-domain access to the Semantic Web. Based on our previous experience we first designed an interaction scenario which uses multimodal input and output. The requirement analysis of the envisioned system and its basic building blocks resulted in an architecture that uses and extends an existing voice dialog system platform for the interaction functionality, and uses web services for the connection to the background application. The interface and content representations use W3C standards like EMMA and RDF Schema. This allows us to leverage tools and prior knowledge. For some functionalities, we had to extend and adapt the standards, e. g., for the representation of results and status information in EMMA. The world knowledge that is shared in all knowledge processing components also reuses and adapts existing ontologies, e. g., SUMO. For discourse processing purposes we added an additional layer of concepts and relations. It contains also an RDFS formulation of EMMA thus enabling us to use EMMA information in the Semantic Web centric interfaces.

The system's first version is operable right now and will be fully functional for the Football World Cup 2006 in Germany. It will be used in a multi-user environment.

6. ACKNOWLEDGMENTS

We would like to thank our partners in SmartWeb. This research was funded by the German Federal Ministry for Education and Research under grant number 01IMD01A. The responsibility for this article lies with the authors.

7. REFERENCES

[1] J. Alexandersson and T. Becker. The Formal Foundations Underlying Overlay. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, Tilburg, The Netherlands, 2003.

[2] J. Alexandersson, T. Becker, R. Engel, M. Löckelt, E. Pecourt, P. Poller, N. Pfleger, and N. Reithinger. Ends-based dialogue processing. In *Proceedings, Second International Workshop on Scalable Natural Language Understanding*, Boston, 2004.

[3] J. Alexandersson, R. Engel, M. Kipp, S. Koch, U. Küssner, N. Reithinger, and M. Stede. Modeling negotiation dialogs. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 441 – 451. Springer, 2000.

[4] R. Amant and C. Healey. Usability guidelines for interactive search in direct manipulation systems. In *Proceedings of The International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[5] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10, 2004.

[6] A. J. Cheyer and D. L. Martin. The Open Agent Architecture. *Autonomous Agents and Multi-Agent Systems*, 4(1–2):143–148, 2001.

[7] R. Engel. SPIN: Language understanding for spoken dialogue systems using a production system approach. In *Proceedings of 7th International Conference on Spoken Language Processing (ICSLP-2002)*, pages 2717–2720, Denver, Colorado, USA, 2002.

[8] D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.

[9] L. Gorlenko and R. Merrick. No wires attached: Usability challenges in the connected mobile world. *IBM Systems Journal*, 2003.

[10] N. Guarino and C. A. Welty. An overview of ontoclean. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 151–172. Springer, 2004.

[11] G. Herzog, A. Ndiaye, S. Merten, H. Kirchmann, T. Becker, and P. Poller. Large-scale Software Integration for Spoken Language and Multimodal Dialog Systems. *Natural Language Engineering*, 10(2/4):283–305, 2004.

[12] G. E. Krasner and S. T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, 1988.

[13] R. Longoria, editor. *Designing Software for the Mobile Context. A Practitioners Guide.* Springer, 2004.

[14] M. T. Maybury and W. Wahlster, editors. *Readings in Intelligent User Interfaces*. Morgan Kaufmann, San Francisco, 1998.

[15] S. Oviatt. Ten myths of multimodal interaction. *Communications of the ACM*, 42(11):74–81, 1999.

[16] N. Reithinger, J. Alexandersson, T. Becker, A. Blocher, R. Engel, M. Löckelt, J. Müller, N. Pfleger, P. Poller, M. Streit, and V. Tschernomas. SmartKom: Adaptive and Flexible Multimodal Access to Multiple Applications. In *Proc. of the 5th Int. Conf. on Multimodal Interfaces*, pages 101–108, Vancouver, Canada, 2003. ACM Press.

[17] N. Reithinger, D. Fedeler, A. Kumar, C. Lauer, E. Pecourt, and L. Romary. MIAMM - A Multimodal Dialogue System Using Haptics. In J. van Kuppevelt, L. Dybkjaer, and N. O. Bernsen, editors, *Advances in Natural Multimodal Dialogue Systems*. Springer, 2005.

[18] S. Seneff, R. Lau, and J. Polifroni. Organization, Communication, and Control in the Galaxy-II Conversational System. In *Proc. of Eurospeech'99*, pages 1271–1274, Budapest, Hungary, 1999.

[19] K. R. Thorisson, C. Pennock, T. List, and J. DiPirro. Artificial intelligence in computer graphics: A constructionist approach. *Computer Graphics*, pages 26–30, February 2004.

[20] W. Wahlster, editor. *VERBMOBIL: Foundations of Speech-to-Speech Translation*. Springer, 2000.

[21] W. Wahlster. Smartweb: Mobile applications of the semantic web. In P. Dadam and M. Reichert, editors, *GI Jahrestagung 2004*, pages 26–27. Springer, 2004.

View publication stats