TOWARDS A PROCESS OF BUILDING SEMANTIC MULTIMODAL DIALOGUE DEMONSTRATORS

Daniel Sonntag and Norbert Reithinger
German Research Center for AI (DFKI), Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany

Keywords: User/Machine dialogue, Semantic data model, Multimodal interaction, Prototype development, Usability planes.

Abstract: A generic integration framework should allow us to build practical dialogue systems for specific use case scenarios. While domain-specific dialogue systems are simpler to achieve than more general, open-domain conversational dialogue systems, the integration into use cases and demonstration scenarios requires a lot of difficult integration work, especially in multimodal settings where different user devices such as touchscreens and PDAs are used. The challenges for those systems include, apart from the dialogue modelling task, the integration modelling for specific use case and demonstration scenarios. This paper reports on dialogue system prototype development based on ontology communication structures and we draw special attention to the process of how to build demonstration systems that include a task-oriented, information-seeking, or advice-giving dialogue as an important fragment of practical dialogue system development.

1 INTRODUCTION

Over the last several years, the market for speech technology has seen significant developments (Pieraccini and Huerta, 2005) and powerful commercial off-the-shelf solutions for speech recognition (ASR) or speech synthesis (TTS). Even entire voice user interface platforms have become available. However, these discourse and dialogue infrastructures have only moderate success so far in the entertainment or industrial sectors. There are several reasons for this.

First, a dialogue system is a complex AI system that cannot be easily constructed since many natural language processing components have to be integrated into a common framework. Many of these components or pre-processing resources require additional language resources which have to be updated frequently. The components often have different software interfaces and cannot be easily run in a hub-and-spoke or agent-based architecture.

Second, the dialogue engineering task requires many adaptations to specific applications which often use a detailed subset of the functionality of specific components and also require high precision results. For example, a dialogue system for a specific domain must understand several variants for specific requests. The dialogue management process has to be customised for very specifc clarification requests or error recovery strategies, too. This poses a problem for most dialogue frameworks for industrial dissemination. They often work with canned dialogue fragments, hard-wired interaction sequences, and limited adaptation possibilities.

Third, the dialogue system prototype development process is often triggered by many auxiliary conditions of the demonstration environment. For example, in a project evaluation, you are mostly interested in a presentation where no unexpected questions and answers occur. Hence, the dialogue engineer has to rely on his intuition about the evaluation measurements in the demonstration scenario and optimise the dialogue management accordingly.

The first of these problems has been addressed by distributed dialogue system integration frameworks. We will discuss prominent frameworks in the related work section (section 2) and report on our own approach (section 3), developed in the last two years. This paper, however, focuses on new approaches for the prototype development process (section 4). Many customisations are different in task complexity, e.g., only speech-based long-distance dialing help differs much from multimodal business-to-business consultant applications in accomplishing a specific business task. Our impression is that we still have significant technical and methodological problems to solve before (multimodal) speech-driven interfaces become

Sonntag D. and Reithinger N..
TOWARDS A PROCESS OF BUILDING SEMANTIC MULTIMODAL DIALOGUE DEMONSTRATORS.
DOI: 10.5220/0003089503220331
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2010), pages 322-331
ISBN: 978-989-8425-30-0

truly conversational and/or accepted by the industry. Towards this goal, different demonstration settings and implementation requirements have to be considered (section 4.2) which do not offer the full range of linguistic analysis, dialogue structure processing, or backend reasoning functionality. Instead, we tried to make the interaction and demonstration most attractive and effective (dialogue and task performance) when considering the short implementation cycles in large-scale development and demonstration projects. We evaluated this in different scenarios while using our semantic dialogue framework which we linked to already existing standards supported by the W3C where for example SPARQL backend repositories and Web Services (WS) will become more and more important in knowledge-based systems. The results will be provided in a conclusion (section 5).

2 RELATED WORK

Prominent examples of integration platforms include OOA (Martin et al., 1999), TRIPS (Allen et al., 2000), and Galaxy Communicator (Seneff et al., 1999); these infrastructures mainly address the interconnection of heterogeneous software components. The W3C consortium also proposes inter-module communication standards like the Voice Extensible Markup Language VoiceXML[1] or the Extensible MultiModal Annotation markup language EMMA[2], with products from industry supporting these standards[3].

Some dialogue projects (Wahlster, 2003; Nyberg et al., 2004) integrated different sub-components into multimodal interaction systems. Thereby, hub-and-spoke dialogue frameworks play a major role (Reithinger and Sonntag, 2005). Over the last years, we have adhered strictly to the developed rule "No presentation without representation." The idea is to implement a generic, and semantic, dialogue shell that can be configured for and applied to domain-specific dialogue applications. All messages transferred between internal and external components are based on RDF data structures which are modelled in a discourse ontology (also cf. (Fensel et al., 2003; Hitzler et al., 2009; Sonntag, 2010)).

A lot of usability engineering methods are around and many guidelines or criteria for designing IT-systems presently exist, e.g., (Nielsen, 1994; Häkkilä and Mäntyjärvi, 2006; Shneiderman, 1998). Guidelines are based on theories, empirical data, and good practical experiences. In our experience, however, these guidelines lack a methodology for a prototype development process. (Muller et al., 1998) described a participatory heuristic evaluation of prototypes for use case scenarios, where the expected end users are involved in the usability testing process. We generalised this method so that the prototype development stage must be also done in partnership with representatives from the use cases for the evaluation/usability testing step before deployment. Iterative prototyping should be done with the help of generic, semantic interface elements (SIEs), that can be configured according to the end user's feedback.

3 DIALOGUE FRAMEWORK

The main architectural challenges we encountered in implementing a new dialogue application for a new domain can be summarised as follows:

- providing a common basis for task-specific processing;
- accessing the entire application backend via a layered approach.

In our experience, these challenges can be solved by implementing the core of a dialogue runtime environment, an *ontology dialogue platform* (ODP) framework and its platform API (the xxxxxxx spin-off company offers a commercial version), as well as providing configurable adaptor components. These translate between conventional answer data structures and ontology-based representations (in the case of, e.g., a SPARQL backend repository) or WS—ranging from simple HTTP-based REST services to Semantic Web Services, driven by declarative specifications. The dialogue framework essentially addresses the first problem of how to integrate and harmonise different natural language processing (NLP) components and third-party components for ASR and TTS.

Using a dialogue framework for the implementation of domain dialogue requires domain extensions and the adaptation of functional modules while implementing a new dialogue for a new domain or use case. Hence, an integrated workbench or toolbox is required, as depicted in figure 1 (right). The ODP workbench builds upon the industry standard Eclipse and also integrates other established open source software development tools to support dialogue application development, automated testing, and interactive debugging. A distinguishing feature of the toolbox is the built-in support for eTFS (extended Typed Feature Structures), the optimised ODP-internal data representation for knowledge structures. This enables

[1] http://www.w3.org/TR/voicexml20/
[2] http://www.w3.org/TR/emma/
[3] http://www.voicexml.org

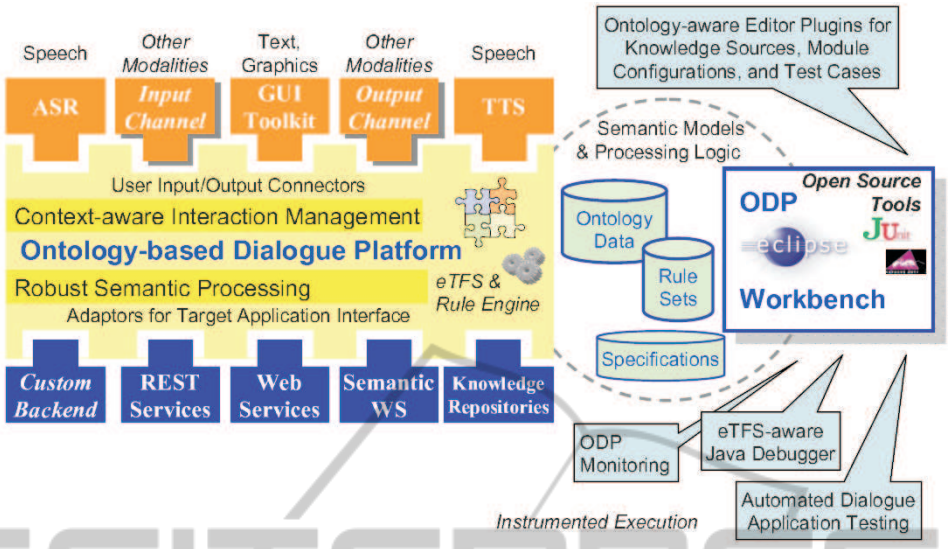

Figure 1: Conceptual architecture of the ontology-based dialogue platform (ODP) and our Eclipse workbench/toolbox. The workbench is used to specify the semantic dialogue management models and processing logic for specific demonstrator scenarios.

ontology-aware tools for the knowledge engineer and application developer to develop use case and demonstration specific prototypes. Hence, the automated dialogue application testing tools play a major role. In this paper, we will describe the top-down conceptual workflow with which the application evaluation/testing step is integrated, the prototype development cycle (section 4). More information on how we support a rapid dialogue engineering process by Eclipse plugins and ontology data structures can be found in [SelfRef].

It is important to point out that the architectural decisions are based on customisation and prototype development issues that arise when dealing with end-to-end dialogue-based interaction systems for industrial dissemination or demonstration scenarios. Our major concern stems from an observation over the last years in the development process for demonstrator scenarios: the incorporation of complex natural language understanding components (e.g., HPSG based language understanding) and open-domain question answering functionality can diminish a dialogue system's acceptability. Therefore, prototype development builds on robust systems and well-defined use case scenarios where the benefit of the (multimodal) speech based interaction comes to the fore. This means that the infrastructure needs to support a quick exchange of input and output modalities (via user input/output connectors) and must be easily adaptable to a specific target application via standardised technical application interfaces such as REST Services, Web Services, Semantic Web Services, or knowledge repositories. Custom backends must be supported as well, e.g., wrapped interfaces to YouTube or Google Maps (examples will be shown, cf. figure 7).

The technical semantic search architecture comprises of three tiers: the application layer (user interface with GUI elements (the SIEs) and the dialogue system/manager), the query model/semantic search layer (eTFS/SPARQL structures), and the dynamic knowledge bases layer for the application backend (figure 2).

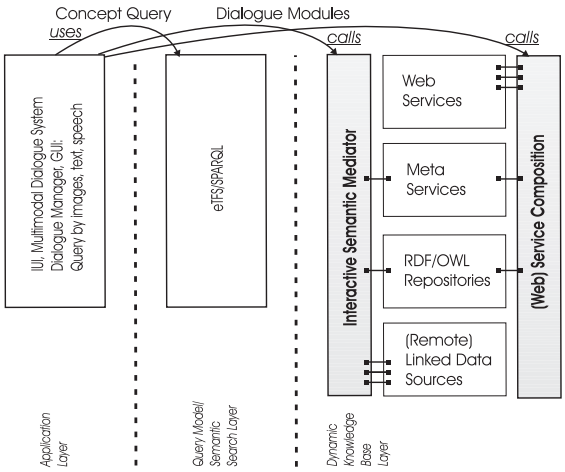

Figure 2: Three-tier semantic search architecture used in all demonstration scenarios.

We use an RDF store (Jena TDB[4]) with the RDF version of Yago (Suchanek et al., 2007) which is a

[4]http://jena.sourceforge.net/TDB/

structured representation of Wikipedia contents for answering domain-specific or domain-independent questions, respectively. As an intermediate query representation language, a syntax based on the SPARQL Inferencing Notation (SPIN[5]) is used. This is in effect a structured version of SPARQL, the RDF query language. SPARQL originally uses a custom plain-text/string query syntax similar to SQL. The SPIN format instead uses an RDFS vocabulary to represent the individual SPARQL terms. The RDF(S) structured syntax allows us to use rule engines and other RDF(S)-based tools to modify the actual eTFS query to build a backend-specific SPARQL query.

4 PROTOTYPE DEVELOPMENT PROCESS

Figure 3 shows a typical software development process which takes usability issues into account. It is a process with iterative steps, meaning the cycle is repeated but in a cumulative fashion. Please note that in our recommended form, the *analysis & design* and *implementation* steps, and the *evaluation/testing* and *requirements* steps timely overlap (work in each step is finished before work in the next step can finish instead of finishing the steps before the next step starts). The deployment step is often performed when the internal *evaluation/testing* cycle stops. A better approach is to involve representatives from the use cases for the evaluation/testing steps before the system is deployed at their organisation for testing. We will use this cycle as an abstract view of our prototype development process and focus on the *analysis & design*, *implementation*, and *deployment* stages.

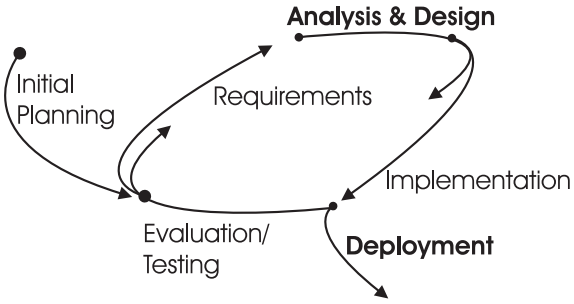

Figure 3: The prototype development process works best if done in partnership with representatives from the use cases for the evaluation step before deployment.

[5]http://spinrdf.org/

4.1 Analysis & Design

The analysis and design step of the prototype development process assumes the existence of several universal principles of design and a usability strategy which supports the iterative software development process.

Design Principles. We use design principles because demonstrator systems are mysteriously tied to the subconscious instincts and perceptions of the test user. We identified at least four such principles which we think apply to the prototype development process:

1. the 80/20 rule;
2. broad accessibility;
3. aesthetic effect;
4. alignment.

(1) The *80/20 rule* is a principal design rule which originally stated that eighty percent of the effects in large systems are caused by twenty percent of the variables, which suggest a link to normally distributed events (Juran, 1993). When applied to our demonstration scenarios, the rule means that 80 percent of a demonstrator's usage involves only 20 percent of the provided (dialogical) interaction competence or interface features. (2) The principle of *Broad accessibility* asserts that research and demonstrator systems should be accessible and usable by people of diverse abilities and backgrounds. Hence, a global user model can be used, and adaptations or modifications can be generally avoided. Of course, this principle cannot be applied generally without a certain controversy especially when designing business applications. Therefore, the accessibility concept has been extended to four standard characteristics of accessible designs: perceptibility, operability, simplicity, and forgiveness.[6] Interestingly, the perceptibility characteristic plays a major role when a trained presenter gives the presentation to a group of system evaluators; redundant presentation methods (e.g., textual, graphical and iconic) enhance the transparency of the system process. Likewise, forgiveness is implemented by ways to prevent usage errors (e.g., control button can only be used in the correct way) or by means to undo an action (e.g., an undo button or the speech command "Please go back to ..."). This characteristic is particularly welcome in situations where the prototype is presented to a greater audience, e.g., when exhibited at a fair. (3) The *aesthetic effect* describes the phenomenon that aesthetic designs are (subconsciously) more easily perceived and more effective at

[6]Also cf. the Web Content Accessibility Guidelines 1.0, available at http://www.w3.org/TR/WCAG10/.

fostering a positive attitude toward the demonstrator system (also cf. works on apparent usability, as, e.g., in (Kurosu and Kashimura, 1995)). (4) Finally, the *alignment* principal concerns the relative placement of visual affordances on a graphical screen. Broadly speaking, related elements in the design should be aligned with related elements to create a sense of unity, cohesion or semantic relatedness (as is, for example, the case with medical pictures of body regions which exhibit a spatial relationship). All principles should help to diminish the degree of misuse and misunderstanding based on the described factors.

Usability Strategy. Design principles can become manifest in a specific usability strategy. Broadly speaking, these strategies try to improve the user-friendliness and efficiency of a prototype. General usability guidelines further express that a usable product is easy to learn, efficient to use, provides quick recovery from errors, is easy to remember, enjoyable to use, and visually pleasing. While these guidelines provide a nice abstract list of optimisation parameters, experience shows they are only seldom used to go through the prototype development cycle, as (Cronholm, 2009) points out. The analysis and design principles described in the previous paragraph offer more technical advice when following the development cycle.

We used the analysis and design guidelines in combination with usability guidelines that consider five different planes (Garrett, 2002) in the development process. Every plane has its own issues that must be considered. From abstract to concrete, these are (1) the strategic plane, (2) the scope plane, (3) the structure plane, (4) the skeleton plane, and (5) the surface plane. In accordance with the software development process in figure 3, defining the users and their needs on the strategic plane is the first step in the design process. It is also useful to create personas that represent a special user group, e.g., the representatives of a specific business case. On the scope plane, then, you have to define the system's capacity (e.g., what a user should be able to say when using a multimodal Internet terminal for music download and exchange, cf. figure 7(3)) and then the requirements for the technical dialogue components.

4.2 Implementation

In the implementation stage, we rely on the specification of the strategic and scope planes according to the specific use case or demonstration workflow and the resulting dialogue shell requirements for the upper three planes, and we implement the functional components while taking the design principles into account (figure 4). The *broad accessibility* principle applies to the scope plane where the functional specifications and interface content requirements are met. Interestingly, the *alignment* principle often applies to the structure plane, i.e., the information architecture. According to the Semantic Web data specifications (RDF/OWL), we try to encode as many relationships for the concepts as possible. This means, e.g., in the context of medical images (figure 7, 3), that specific body regions have spatial relationships. These relationships can then be seen as an image alignment and used to influence/specify the layout on the skeleton plane. The *80/20 rule* principle on the skeleton plane essentially provides a recommendation for the implementation of the multimodal input possibilities. It specifies that 20 percent of the input possibilities should be available to the user at first sight. This way of modelling is particularly interesting in speech-based interaction systems. Although a 20 percent selection of graphical interface elements and functions is easily conceivable, the restriction of speech-based commands to a specific subset is counter-intuitive. Nonetheless, state-of-the-art ASR systems can be tuned to recognise speech utterances in context. Accordingly, we modelled our ASR and multimodal input integration components to work particularly robustly for approximately 20 percent of the input possibilities. This is particularly beneficial in demonstration scenarios since it can reduce the error rate while the presenter gains external control of the demonstration session.

The *aesthetic effect* principle applies to the surface plane and the visual design of the graphical interface. In many use case specific software environments, both corporate identity and native application side conditions hinder a proper implementation of this principle. For example, the mobile GUI in figure 7(2) is mostly driven by the other applications on the iPhone which a business expert uses for his daily work. In this situation, a good trade-off must be found and usability studies can deliver the necessary empirical indication. The situation is different for pure research prototypes that exhibit a new interaction form. Aesthetic designs are generally perceived as easier to use which has significant implications regarding acceptance; figure 7(3) shows a multimodal interface where the *aesthetic effect* principle has been obeyed with much care.

We implemented an abstract container concept called Semantic Interface Elements (SIEs) for the representation and activation of multimedia elements visible on the different interface devices. Semantic-based data integration frameworks for multimedia

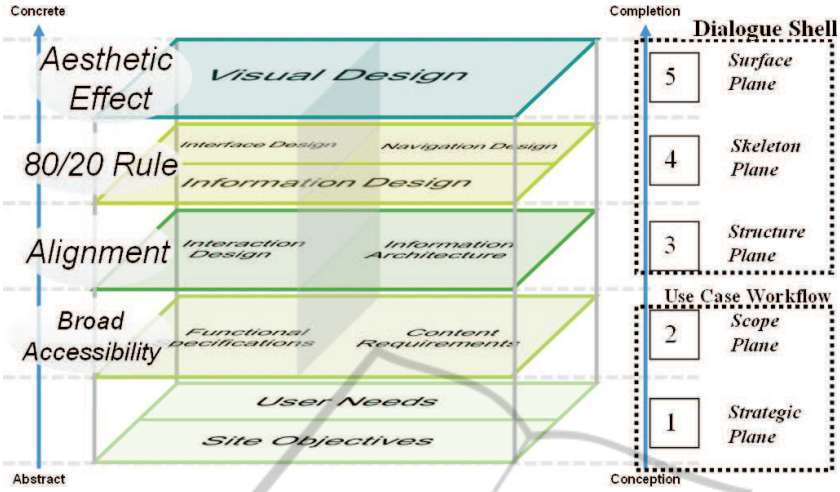

Figure 4: The five usability planes with a differentiation where a use case workflow and resulting requirements for the implementation of the dialogue shell remain (right) and the the applied design principles (left).

data allow authors to write interactive multimedia presentations. They are also very valuable in planning environments where the complete content displayed on screen is dynamically built up. A further objective is a description framework for the semantic output processing, which can be used on different end-user devices. The presentation ontology specifies interaction patterns relevant to the application. Currently, the graphical user interface driven by the SIE concept enables the following semantic presentation elements based on ontology descriptions (OWL): tables with selectable cells; lists with selectable elements and faceted browsing; graphs with nodes, which can be selected; containers for elements of lists, tables, graphs; and several different media types. Since the SIEs are a key driving concept of our iterative semantics-based GUI design, we will explain them in more detail.

Once the backend system has retrieved its answers from the backend, the ontology based multimodal dialogue system will use an internal representation for defining the presentation elements and contents to be applied. In our case the semantic interface elements are defined in the presentation ontology (in OWL or our own eTFS format) used in the dialogue engine to represent the current state of the GUI containing these visible SIE elements (figure 5, upper part). In addition, ontology instances of these SIE elements contain information about the their contents, e.g., which video to play (figure 5, middle part). In order to transfer the actual presentation, the SIE elements and their contents, to the user end device/client (figure 5, lower part), we serialise them into a special XML format we called PreML (Presentation Markup Language).

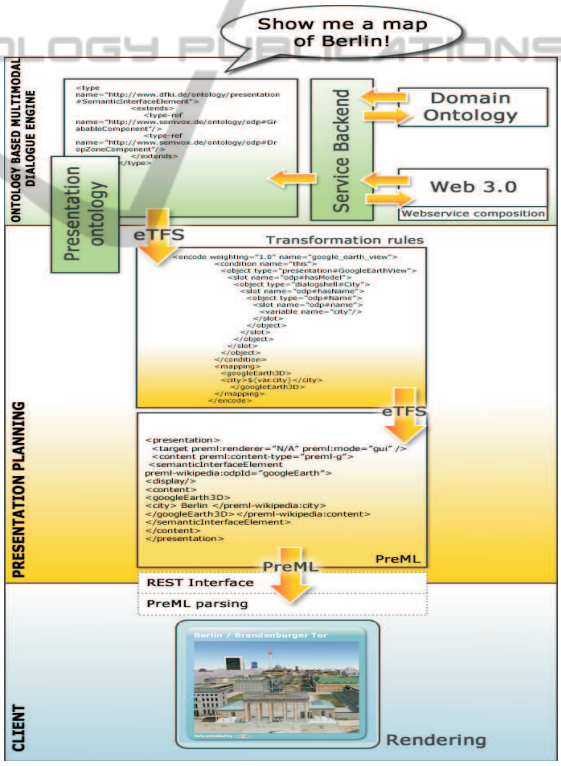

Figure 5: Processing workflow from backend retrieval to rendering stage with Semantic Interface Elements (SIEs).

With this approach only one generic PreML message is needed to target several client types (desktop, terminal, mobile) and different rendering technologies.

Figure 6 shows how a Semantic Interface Element for videos is represented in PreML (in our case a YouTube Player). The SIE also features with a list

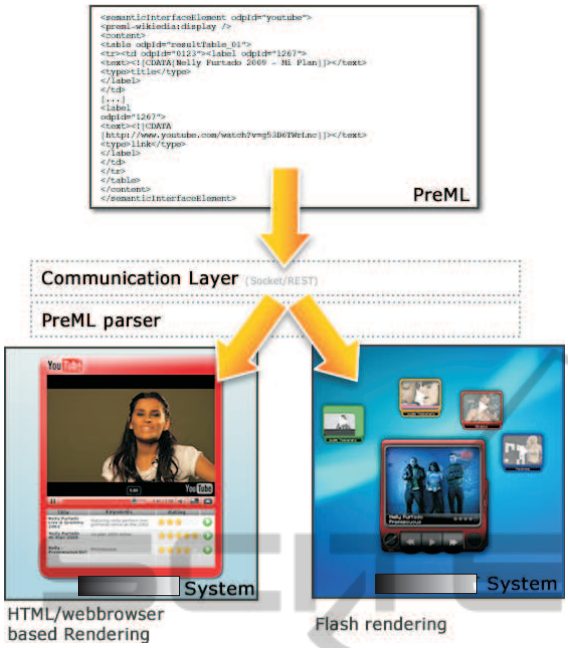

Figure 6: PreML conversion into instantiated GUI elements (SIEs).

of related videos corresponding to the answer to the spoken question “Show me videos by Nelly Furtado.” (Also cf. how we design and implemented combined mobile and touchscreen-based multimodal Web 3.0 interfaces [SelfRef].)

We will now focus on the implementation step of our multimodal dialogue demonstration systems where the speech-based dialogue with the user plays a major role. The discussion of the related natural language processing tasks is easily transferable to other sensor-based forms of multimodal input recognition tasks. According to the prototype implementation requirements, we distinguish seven dialogue components which need to be customised to the specific application domain or demonstration scenario: multimodal semantic processing, semantic navigation, interactive semantic mediation, user adaptation/personalisation, interactive service composition, semantic output representation, and query answering from unstructured data sources.

In many cases, the environmental dialogue setting (i.e., the use case or demonstration scenario) decides upon the complexity of the conversational interfaces. The use case or demonstration scenario drives the customisation effort. The positive side-effect is that it is often possible to build more robust ASR and NLU sub-components according to the environmental setting and a user model. However, the customisation effort during the deployment step includes the following issues:

- Handling of the recognition of speech input possibilities and the subsequent recognition of (domain-specific) intentions;
- Handling of the level of query/language complexity associated with a task while using a subset of available speech acts;
- Allowance of the user to control the dialogue at any time;
- Decision whether mixed initiative dialogue modelling is useful.

We implemented the first three issues in all our dialogue demonstrators. The fourth topic, mixed initiative dialogue modelling, is only available in the latest demo system since it requires a proper dialogue management component which is extremely time-consuming to create and therefore not available to many other demonstration systems. Figure 7 shows all demonstration systems.

The first system is an interactive multimodal dialogue system that enables the intuitive, multimodal operation of an iPod or similar device. Unique to the system is its direct access to the entire music collection and that the user is able to refer to all kinds of named entities during all stages of the interaction. Moreover, the system also supports the full range of multimodal interaction patterns like deictic or cross-modal references [SelfRef].

The second system is a a mobile business-to-business (B2B) interaction system. The mobile device supports users in accessing a service platform. A multimodal dialogue system allows a business expert to intuitively search and browse for services in a real-world production pipeline. We implemented a distributed client-server dialogue application for natural language speech input and speech output generation. On the mobile device, we implemented a multimodal client application which comprises of a GUI for touch gestures and a three-dimensional visualisation [SelfRef].

In the third interaction system, a mobile user scenario is combined with a touchscreen-based collaborative terminal. Multiple users should be able to easily organize their information/knowledge space (which is ontology-based) and share information with others. We implemented an MP3 and video player interface for the physical iPod Touch (or iPhone) and the corresponding virtual touchscreen workbench [SelfRef]. For this system, we would like to present an example dialogue. In the combined user scenario, users (U:) have stored their personal multimedia information, which they want to share with each other on their iPods or iPhones. In addition, the users wish to download additional video content. These (collaborative)

tasks can be fulfilled while using the touchscreen terminal system (S:) and natural dialogue, as exemplified in the following combined multimodal dialogue which we implemented:

1. **U:** Start client applications and register several iPods on the terminal. (WLAN IP connection)
2. **U:** Place, e.g., 2 registered iPods on the touchscreen (ID3 tagged)
3. **S:** Shows media circles for iPod contents.
4. **U:** Can select and play videos and search YouTube and Lastfm, etc. by saying: "Search for more videos and images of this artist."
5. **S:** Delivers additional pictures, video clips and replies: "I found 40 videos and selected 5. Do you want me to play them?"
6. **U:** "Yes, play the first two hits."
7. **S:** Initiates two video SIEs and plays the videos.
8. **U:** Drag image of Nelly Furtado onto second iPod; Remove second iPod (figure 7, 3) from the touchscreen. "Yes, play the first two hits."
9. **S:** The media is transferred via IP connection.

The multimodal query interface of the fourth system implements a situation-aware dialogue shell for semantic access to medical image media, their annotations, and additional textual material. It enhances user experience and usability by providing multimodal interaction scenarios, i.e., speech-based interaction with touchscreen installations for the health professional who is able to retrieve relevant patient information and images via speech and annotate image regions while using a controlled annotation speech vocabulary [SelfRef].

The fifth system is a demonstration system without a direct use case context. Its purpose was to demonstrate the competence of individual dialogue internal or external components such as video retrieval and semantic graph presentations.

4.3 Deployment

The deployment step is the activity that makes a demonstrator system available for use in the use cases or demonstration scenarios. Software deployment activities normally include the release, the modification of a software system that has been previously installed, etc. However, we focus on the deployment process that results in a new software distribution, demonstrator, or selected demo event during the project runtime. In large scale integration projects (i.e., projects with a runtime of three to five years) the distribution of these events is of particular interest. Figure 8 shows the respective timeline of our project.

The requirements analysis and revised analysis at the beginning of the development process is in a timeframe of fourteen months. This allows one or two

Figure 7: Examples of Multimodal Dialogue Demonstrators.

runs through the prototype development process before revised requirements are specified (also cf. figure 3).

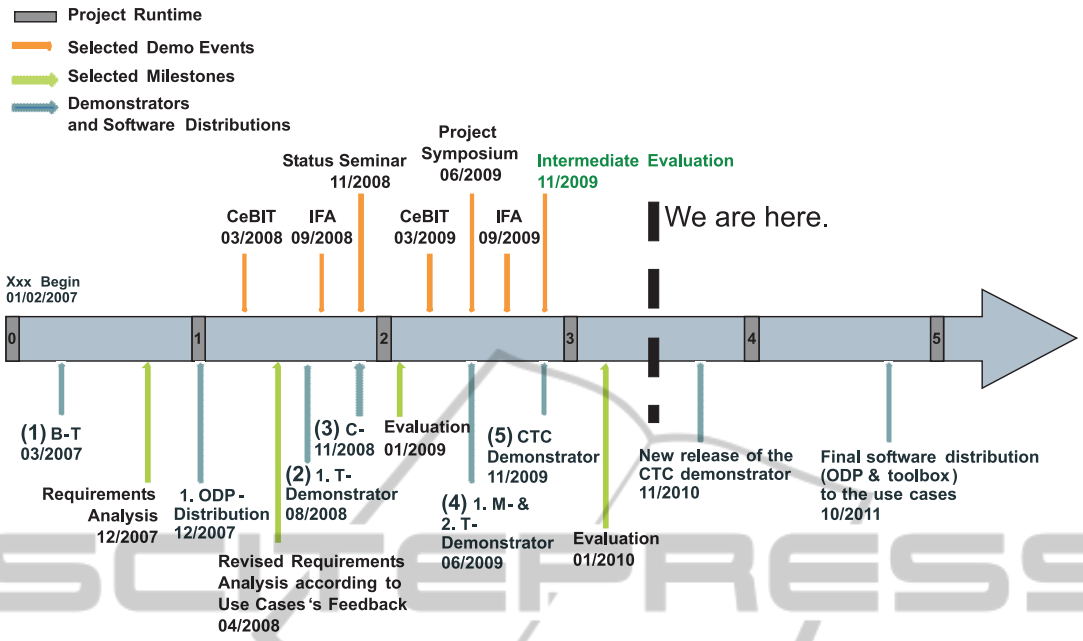

Figure 8: Project runtime, selected demo events, selected milestone and demonstrators and software distributions in a timeline.

Because the time-consuming steps in figure 3 overlapped, the first use case specific demo system could be deployed only four months after the revised specifications (figure 8, 2), followed by the third one after only three month of development time. Shortly after the dissemination of a new demonstrator system, a major demo event took place, e.g., CeBIT 2008, IFA 2008, CeBIT 2009, and IFA 2009. Figure 8 summarises the respective synchronisation activities according to the proposed prototype development process. All selected demo events were very successful; hence, we believe that our prototype development process is useful for large scale dialogue system integration projects.

5 CONCLUSIONS

We described a specific dialogue system prototype development process and drew special attention to the process of how to build demonstration systems that include a task-oriented, information-seeking, or advice-giving dialogue. All implementations follow our ontology-based dialogue system framework (ODP) in order to provide a common basis for task-specific semantic-based processing. Semantic-based processing includes the usage of abstract container concepts called Semantic Interface Elements (SIEs) for the representation and activation of multimedia elements visible on the different interface devices. Knowledge-based system aspects came into consideration while accessing the entire application backend via a layered approach—thereby address Web Services, RDF/OWL repositorties, and Linked Data Sources.

The prototype development process includes design principles and a usability strategy. We have shown a more complete picture of existing usability planes (which was not available before) by differentiating between a use case workflow and the resulting requirements for the implementation of the dialogue shell. We also applied design principles and then explained five dialogue system demonstrators that were built with the help of the new development process.

In our experience, obeying the *aesthetic effect* principle has a positive side effect on the implementation of the *broad accessibility* principle. This is particularly welcome in public demonstration scenarios. The demonstrator systems suggest that we are close to attaining a level of robust performance in the domain-specific realisation of multimodal dialogue systems. A special condition we prepared for when using the design principles and usability planes is the demonstration situation where an untrained user (i.e., the evaluator/test user) is given the opportunity to test the system and assess a judgement about how he perceives the system (figure 9).

Figure 9: Selected Demo Events.

REFERENCES

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. (2000). An Architecture for a Generic Dialogue Shell. *Natural Language Engineering*, 6(3):1–16.

Cronholm, S. (2009). The usability of usability guidelines. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction Conference*.

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

Garrett, J. J. (2002). *The Elements of User Experience*. American Institute of Graphic Arts, New York.

Häkkilä, J. and Mäntyjärvi, J. (2006). Developing design guidelines for context-aware mobile applications. In *Mobility '06: Proceedings of the 3rd international conference on Mobile technology, applications & systems*, page 24, New York, NY, USA. ACM.

Hitzler, P., Krtzsch, M., and Rudolph, S. (2009). *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.

Juran, J. M. & Gryna, F. M. (1993). *Quality planning and analysis : from product development through use*. McGraw-Hill, New York.

Kurosu, M. and Kashimura, K. (1995). Apparent usability vs. inherent usability: experimental analysis on the determinants of the apparent usability. In *CHI '95: Conference companion on Human factors in computing systems*, pages 292–293, New York, NY, USA. ACM.

Martin, D., Cheyer, A., and Moran, D. (1999). The Open Agent Architecture: a framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1/2):91–128.

Muller, M. J., Matheson, L., Page, C., and Gallup, R. (1998). Methods & tools: participatory heuristic evaluation. *interactions*, 5(5):13–18.

Nielsen, J. (1994). Heuristic evaluation. pages 25–62.

Nyberg, E., Burger, J. D., Mardis, S., and Ferrucci, D. A. (2004). Software Architectures for Advanced QA. In Maybury, M. T., editor, *New Directions in Question Answering*, pages 19–30. AAAI Press.

Pieraccini, R. and Huerta, J. (2005). Where do we go from here? research and commercial spoken dialog systems. In *Proceeedings of the 6th SIGDdial Worlshop on Discourse and Dialogue*, pages 1–10.

Reithinger, N. and Sonntag, D. (2005). An integration framework for a mobile multimodal dialogue system accessing the Semantic Web. In *Proceedings of INTERSPEECH*, pages 841–844, Lisbon, Portugal.

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

Shneiderman, B. (1998). *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, third edition.

Sonntag, D. (2010). *Ontologies and Adaptivity in Dialogue for Question Answering*. AKA and IOS Press, Heidelberg.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA. ACM Press.

Wahlster, W. (2003). SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In Krahl, R. and Günther, D., editors, *Proceedings of the Human Computer Interaction Status Conference 2003*, pages 47–62, Berlin, Germany. DLR.