Rapid Development of Multimodal Dialogue Applications with Semantic Models

Robert Neßelrath and **Daniel Porta**
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3
66123 Saarbrücken
{firstname}.{lastname}@dfki.de

Abstract

This paper presents a model-driven development approach to rapidly create multimodal dialogue applications for new domains. A reusable and consistent base model and generic processes inside a multimodal dialogue framework enable advanced dialogue phenomena and allow for a scenario- and domain-specific customization without the necessity to adapt the core framework. We introduce declarative adaptation and extension points within the discussed models for input interpretation, output presentation, and semantic content in order to easily integrate new modalities, domain-specific interactions, and service back-ends. Three multimodal dialogue applications for different use-cases prove the practicability of the presented approach.

1 Introduction

Speech-based applications gain more and more acceptance. On mobile devices, users can, e.g., search the Internet, dictate short messages, or even maintain shopping lists by uttering speech commands. This circumvents typing on small screen devices. In contrast, multimodal dialogue user interfaces still get less public attention, despite the benefits for a more natural human computer interaction.

Typical usage scenarios for applications with multimodal dialogue user interfaces comprise all kinds of mobile situations where users have to cope with an eyes-busy primary task, e.g., driving a car or walking through a shopping street, or intelligent environments that ease the user's daily life. Common to these scenarios on an abstract level is the support for several input and output modalities allowing for advanced dialogue phenomena, e.g., the use of deictic, elliptic, spatial, or temporal references.

On a closer look, however, every concrete application domain in one of the usage scenarios has its own mixture of interaction patterns. For example, in the ambient assisted living domain, as one incarnation of an intelligent environment, the focus lies on command and control of home appliances whereas multimodal dialogue infotainment applications implement searching or even question answering interaction patterns. While even different interaction patterns can be implemented to some extent in a generic reusable way, such that they can be applied in different domains, the actual application data and its formalism is totally domain-specific, ranging from, e.g., simple XML schema definitions to complex semantic models in the Web Ontology Language (OWL).

With their high complexity, multimodal dialogue user interfaces inherently require more development effort than traditional graphical user interfaces. We are investigating a generic framework for multimodal dialogue user interfaces which shall ease the development of such interfaces for various application domains in various usage scenarios, and hence reduce complexity by encapsulating recurring tasks and functionalities.

We already adopted our framework for implementing multimodal dialogue user interfaces. (Sonntag and Möller 2010) describe a medical system that supports a radiologist in finding a diagnose, asking for a second opinion, and deciding for an appropriate medical treatment. The system allows semantic image annotations and presents former patients with similar findings. The collaborative kiosk infotainment system described by (Bergweiler, Deru, and Porta 2010) can be deployed in museums or exhibitions. Visitors can ask for relevant information by interacting with a large tabletop surface and their mobile devices which can also be used for sharing media assets. The necessary data is retrieved by accessing a large heterogeneous service back-end which also comprises semantic Web services. (Porta, Sonntag, and Neßelrath 2009) describe a mobile business application that enables decision-makers on the go to still participate in important business processes. The mobile user can handle purchase order requisitions in an enterprise resource planning system and can search for alternative products. Found alternatives can be sorted according to different criteria and visualized in a 3-dimensional space.

The selected implemented systems show that different application domains have already been covered with our generic framework. But adaptations and extensions for new domains have been and will always be necessary. So far, such customizations needed a vast amount of time. In order to reduce this time, a reusable and consistent base modelling and generic processes are inevitable. At the same time, transparent scenario- and domain-specific customization points are required without affecting the base system. We tackle these requirements by applying a model-driven development approach. Multimodal dialogue user interfaces involve a large number of models that incorporate or depend on each other. Model-driven development is a software development methodology for automatically deriving running applications from formal representations of a domain and system. A formal description of system parts by models pays off in better readable documentation, easier reusability and adaptation, and hence in the reduction of development time.

This paper is outlined as follows. We begin with an overview of processing in multimodal dialogue systems (chapter 2). In chapter 3, we describe important models we use in multimodal dialogue applications and explain how we benefit from them in terms of a rapid development process. Chapter 4 highlights three applications that were recently developed with our generic framework. Finally, we conclude in chapter 5.

2 Processing in Multimodal Dialogue Systems

Our generic framework for building multimodal dialogue user interfaces is called the Ontology-based Dialogue Platform (ODP) (Schehl et al. 2008) and includes interfaces to relevant 3rd-party ASR/NLU (e.g., Nuance) and text-to-speech (TTS, e.g., SVOX) components. It also provides a runtime environment for multimodal dialogue applications supporting advanced dialogical interaction. The central component is a dialogue system which uses a production rule system (Pfleger 2004) for a context-aware processing of incoming requests (e.g., display and discourse context) and events. It is based on domain-specific models, e.g., the UI and discourse model. The models include the reaction to pointing gestures, the natural language understanding process, the representation of displayed graphics, and the speech output. Furthermore, the dialogue system provides a programming model for connecting multiple clients (session management) for presentation and interaction purposes. The external and application-specific components in the backend layer can also be accessed easily.

Additionally, the ODP supports the development process with a set of Eclipse-based integrated tools for editing, debugging and testing of semantic objects, rules and grammars (Sonntag et al. 2009). Experience from several research projects like SmartKom (Wahlster 2006) and SmartWeb (Sonntag et al. 2007) influenced the design of the framework. It is based on the abstract reference architecture for multimodal dialogue systems as introduced by (Bunt et al. 2005). Typically, an ODP application consists of one or more (thin) clients, the server-side ODP runtime environment, and the domain-specific application back-end. Hence, the ODP runtime environment acts as a middleware between the actual user interface and the back-end in order to hide the complexity from the user by presenting aggregated data. The internal workflow is divided into three processing phases: (i) understanding, (ii) dialogue management and action planning, and (iii) output generation. Figure 1 shows how these phases are realized within the ODP in groups of standard components for monomodal input interpretation, multimodal fusion and discourse resolution, dialogue management and action planning, service access, modality fission, and modality specific output generation.

The components for monomodal input interpretation are specialized recognizers for single modalities like a speech recognizer or a gesture classifier. References are resolved in the multimodal fusion and discourse resolution engine. The result of the understanding phase is a list of weighted hypotheses of user intentions. The dialogue manager selects the best fitting intention taking into account the computed weights and plans and executes the appropriate actions. Often, access to back-end services is necessary for task-driven interaction such as command and control or searching. Presentation planning computes a high level presentation of the result that is distributed to the output modalities by the fission component.

Figure 1 also emphasizes the adaptation work a developer has to perform. Green marked components execute generic processes that operate on domain-independent concepts. Actually, they do not require any adaptation for new applications. In detail, these are the fusion and discourse resolution engine and the multimodal fission component. Yellow marked components (input interpretation, interaction management, presentation planning) have to be adapted to a new domain. But this can be done in a purely declarative way. Although they already contain generic processes that operate on abstract concepts, concrete domain-specific concepts have to be derived. Optionally, the generic processes of these components might need to be refined. The red marked service access and semantic mapping component is rather domain-specific. Additional to the work required for the yellow components, implementation work for accessing domain-specific services is necessary.

3 Models in Multimodal Dialogue Systems

Throughout the framework, we use a semantic modelling approach to achieve a consistent description of content. (Araki and Funakura 2010) examine some possibilities to use ontologies in spoken dialogue systems. Benefits are found for language models, semantic analysis of utterances, frame-driven dialogue management and user modelling. In (Milward and Beveridge 2003), ontologies support dialogue management, generation of language models for speech recognition and generation and input interpretation. We adopt this idea and introduce semantic models that support the development process. Semantic models and appropriate abstraction layers enable reasoning algorithms to support generic processes throughout the system. In this paper, we focus on:

- The **Content Model** is the semantic representation of the domain-specific data. New content from services is integrated into the system after it is lifted onto a semantic level (section 3.1). In the figures of this paper, concepts, properties, and instances of this model are painted in yellow. Basic concepts are marked with the namespace prefix `base:`. Derived domain-specific concepts wear a different prefix indicating their originating source, e.g., `fb:` for Facebook or `dbpedia:` for DBPedia.
- The **Reference Model** provides means to describe references to content in a generic way throughout the whole framework (section 3.2). Concepts, properties, and in-

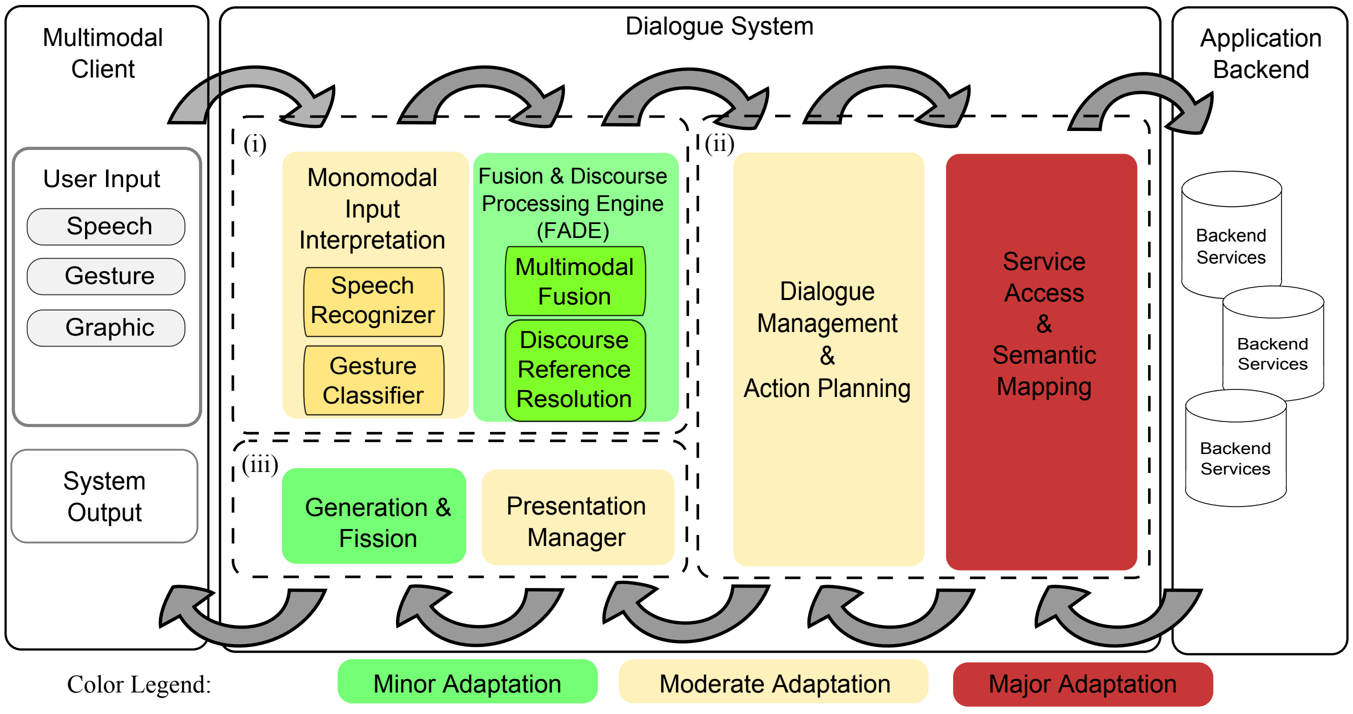


Figure 1: The dialogue processing cycle and development costs for building a new application

stances of this model are marked with the namespace prefix `ref:` and painted in red.

- The domain-independent **Input Interpretation Model** encapsulates the results of the monomodal input interpretation components and is processed in the fusion and discourse resolution engine and later in the dialogue management (section 3.3). Concepts, properties, and instances of this model are painted in green. Basic concepts are marked with the namespace prefix `base:`. Derived domain-specific concepts wear a different prefix indicating the application domain, e.g., `uch:` for the smart kitchen control application (section 4.3) or `isis:` for the ISIS information system (section 4.3).
- The **Graphical Presentation Model** describes the presentation, content and behaviour of graphical user interfaces (section 3.3). Concepts, properties, and instances of this model are marked with the namespace prefix `gui:` and painted in blue.

3.1 Semantic Content Model

With the years, the idea of Web content shifted from a technical domain only accessible by experts to an open community where anyone can contribute. Nowadays, more and more applications depend on Internet access in order to retrieve and store their data in the Web. The "Internet of Services" and "Cloud Computing" introduce lots of smart Web applications and service mashups exposing terabytes of heterogeneous and unstructured content. Hence, an important task is to improve consistency and availability of Web content in order to integrate it into applications (Allemang and Hendler 2008).

We attended to this task during the development of semantic-based multimodal dialogue applications that acquire their content from different heterogeneous Web services. Typically, it is worthwhile to hide the prevalent form of heterogeneity from end users in order to avoid complexity and raise their acceptance for the application. For example, it is incomprehensible for a user that presentation and interaction with an entity of type *Person* that is retrieved from Wikipedia is fundamentally different to the interaction with an entity of same type from a personal addressbook. Here, a consistent pattern for presentation and interaction is preferred.

Knowledge engineers envision the Semantic Web (Berners-Lee, Hendler, and Lassila 2001) which not only describes content but also the meaning and interrelationship of that content. It allows to combine common facts from different sources but also leaves room for perhaps contrary opinions. This follows the *AAA Slogan* "**A***nyone can say* **A***nything about* **A***ny topic.*" mentioned in (Allemang and Hendler 2008). Languages for knowledge and ontology representation like OWL support the merging of semantic content from different sources by offering mechanisms to define equivalent classes (`owl:equivalentClass`) and instances (`owl:sameAs`).

Unfortunately, a lot of relevant Web content is still not available in a semantic representation or full-fledged semantic processing at interactive speeds is hard to achieve. Work goes towards a more lightweight Semantic Annotated Web, e.g., RDFa is an extension for XHTML to embed RDF in Web documents (Adida et al. 2008). But until we reach a

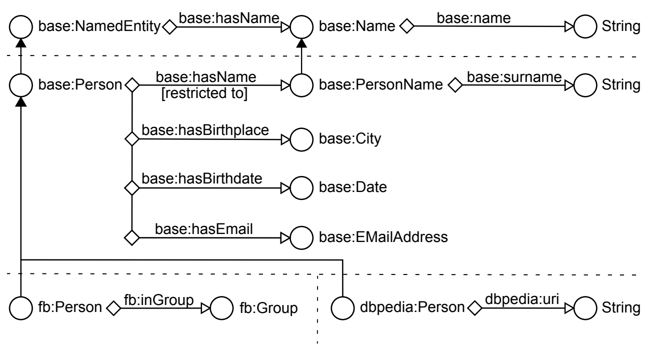


Figure 2: The domain-independent concept `base:Person` and the inherited concepts for DBpedia and Facebook contacts.

level at which most content is represented in a semantic form, more persuasion work and tooling support is needed to convince a larger community of this idea. Adopted from the Web 2.0, service mashups already found their way into the Semantic Web (Ankolekar et al. 2007). Usually, these (composed) services can be queried via standard interfaces like WSDL/SOAP or REST and return XML structures. If not offered by the service itself, these results must be lifted from a pure syntactic onto a semantic level in order to integrate them into a semantic-based application. Often, such service mashups provide valuable data for context-sensitive dialogue applications (Sonntag, Porta, and Setz 2010). Despite the advantages, e.g., making implicit knowledge accessible by inferencing, new problems arise in terms of semantic mediation of content from heterogeneous sources. Some semantic Web services like DBPedia (Bizer et al. 2009) are queried by SPARQL expressions and return their content in RDF bindings that conform to the underlying ontology, whereas not every semantic Web service is operating on the same (upper) ontology.

We have to cope with that in the ODP. Due to the lack of a sufficient semantic infrastructure in Web services, we still need a mapping process that integrates service answers into the running application in order to present them to the user and make them available for interaction. For this, we need well chosen concepts for knowledge representation within our base dialogue ontology which cover most aspects of the content we process (e.g., NamedEntity, Person, Location). Additional domain-specific data must be specified in inherited concepts. Content structures from heterogeneous Web services are mapped onto this extended ontology by applying a rule-based mediation process. Since content representation strongly depends on the domain and the set of accessed services, required mapping rules have to be written manually or in the best case semi-automatically.

As an example, we imagine an interaction system that handles person data from different back-end services. One is DBPedia, that makes Wikipedia content accessible in a semantic form. Second are contacts extracted from facebook. Most properties like the name or date of birth are common properties that are delivered from both information sources. Some others are very domain-specific. Figure 2 depicts the

```
<object type="ref#ReferenceModel">
  <slot name="ref#hasPattern">
    <object type="uch#Appliance">
      <slot name="uch#hasState">
        <object type="uch#PowerModeState" />
      </slot>
    </object>
  </slot>
  <slot name="ref#hasType">
    <object type="base#DeicticReference" />
  </slot>
</object>
```

Figure 3: Example of a reference model.

inheritance tree of the concept `base:Person` which bundles the intersection of the most common properties of a person. Since these properties are modelled in the base ontology, rules for interaction, dialogue management and presentation that handle abstract person instances can also handle instances from both sources in the same way. The person instances retrieved from DBPedia (`dbpedia:Person`) and facebook (`fb:Person`) contain additional mutual exclusive properties (Facebook groups or DBPedia resource URIs) which become relevant for very domain-specific interaction or back-end retrieval tasks.

3.2 Reference Model

Referring expressions are a key linguistic phenomenon of verbal interaction. One of the first systems that challenges multimodal fusion is the “Put-That-There“ system (Bolt 1980), where the speech command *“Put that there.”* gives information about the act itself but contains two placeholders, the first for an item on a screen, the second for a position where the item should be placed. These placeholders are filled with information given by pointing gestures.

In other cases, user intentions refer to a previous interaction or situational context. A dialogue system has to intelligently integrate this context information for the interpretation of the user's intention. A user that gives a speech command *“Turn on the lamp”* to turn on the lamp in the room in which he currently resides, gives an incomplete command insofar that the action but not its target is defined assuming that there are more rooms in the apartment. Nevertheless, the system retrieves information about the type of the target, in this case a lamp. Humans directly understand the actual intention of the command and switch on the lamp of the correct room. A system that takes a context model for the user's location and the type of controllable devices nearby into account can apply reasoning algorithms to infer the same conclusion.

In the ODP, we use a reference model for describing partially defined information. A semantic pattern restricts the type of the missing instances and their properties. Additionally, it comprises linguistic information like case, gender, part-of-speech, or number that is valuable information for the resolution of the reference. We build on the approach from (Pfleger 2007) that describes the rule-based fusion and

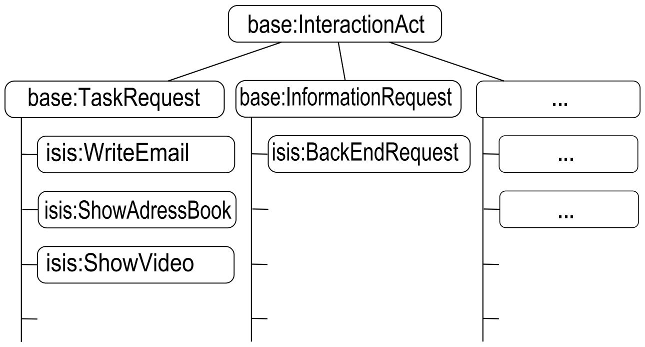


Figure 4: Upper level ontology of the interaction act model.

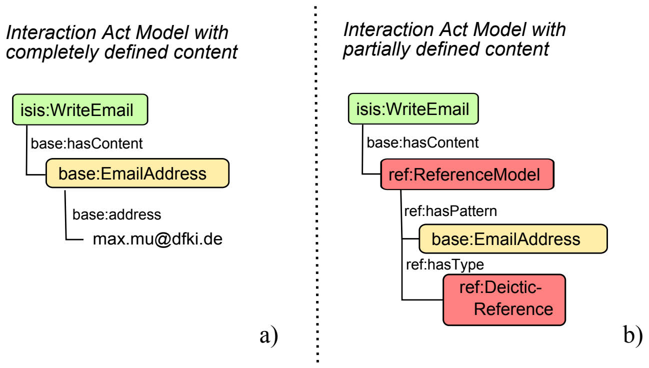


Figure 5: Instances of interaction act models with a) completely and b) partially defined semantic content.

discourse resolution engine FADE. Here, besides the pattern for semantic content, reference types define expectations about where to find the missing instances in the discourse context. These types are: spatial references, deictic references/multimodal fusion, elliptic expressions, temporal relations, and references to items in presented collections. Generic rules in the FADE component are responsible for reference resolution by applying unification and overlay algorithms (Alexandersson, Becker, and Pfleger 2006) on the references and the semantically represented discourse context.

Imagine a pointing gesture to a kitchen appliance combined with the command *"Turn this on!"* Here, the user's speech command is ambiguous. The missing and sufficient information is given with the pointing gesture. Figure 3 shows an instantiated reference model that comprises two details about the referred instance. First is a semantic pattern that only allows instances of type `uch:Appliance` with a power-mode functionality. Second is the type of the reference, in the example a deictically introduced instance.

For our model-driven development approach, we identified an additional purpose for referring expressions and their resolution strategies. By introducing the `ref:DataModelReference`, we enable a loose coupling in model definitions for content, interaction acts, and graphical presentation. Here, the role of a reference model is to describe the connections between these models by patterns (figure 7 provides an example). In the next two subsections, we will show how the reference model helps to accelerate the development process of semantic-based multimodal dialogue applications.

3.3 Input Interpretation Model

Depending on the situation, users prefer different input modalities to interact with an application. For every modality, a monomodal input interpreter hypothesizes user intentions. Their results are integrated into the ODP by means of the interaction act model.

Interaction Act Model (Bunt 2000) distinguishes between two content types in the meaning of an utterance. First is the information that is introduced into the dialogue. Second is the communicative function that describes the way, how the new information is introduced. Their combination is a dialogue act.

The base ontology comprises concepts for the representation of these content types. According to (Bunt et al. 2010) and (Alexandersson et al. 1998), these are communicative functions like grounding, informing, questioning, answering, task offering, turn taking, etc. They are enriched with semantic content that delivers more information about the user's intention and instances that are introduced with the act. Figure 4 shows an extract of the upper level ontology of this model. All interaction act concepts are derived from the base concept `base:InteractionAct`. Inherited concepts like `base:TaskRequest` or `base:InformationRequest` describe the communicative function of the act. The idea of domain-independent dialogue acts was already presented and applied in several related projects like TALK (Becker et al. 2006) and has proven of value for our development approach.

The interaction act example in figure 5 a) represents a task request to write an e-mail to a person like "I'd like to write an email to Max Mustermann". The communicative function is given by the domain-specific concept `isis:WriteEmail` that is derived from the abstract concept `base:TaskRequest`. The semantic content is delivered in the inherited property `base:hasContent`. It can be underspecified ("Write an email to this contact"). In this case, models for referring expressions give additional information about the missing content, e.g., linguistic or semantic information. In figure 5 b) the content is restricted to instances that unify with the concept `base:EmailAddress`. The content of the property `ref:hasType` describes the type of the reference, in this case a deictic one.

The agreement on an interaction act model enables the easy extension of applications with new input modalities. Input interpreters that define their results with interaction act models can be rapidly integrated into a multimodal dialogue application without adapting the internal framework components.

Fusion and Discourse Processing The fusion and discourse resolution engine resolves interaction acts with partially given content. In most cases, missing content must be filled with information from other modalities or the discourse context. In other cases, a new data instance is introduced and the missing user intention must be resolved. We observe the following sequence of utterances:

(1) "Turn on the lamp."
(2) "And the TV."
(3) "Increase the volume."

In this example utterance (2) introduces a new appliance, the TV, into the dialogue. It does not provide the user's intention what to do with the TV, i.e., turn it on. This must be reasoned from the previous turn (1) of the dialogue. Utterance (3) gives no information about the appliance of interest. Besides the TV, a radio could be running that also contains a volume control. This ambiguity can only be clarified in the discourse context.

Figure 6 shows three possible combinations for the completeness of the interpretation of a user's intention. Example a) is the complete interpretation for utterance (1). No further resolution work is necessary. Example b) shows an interaction model of an utterance with a deixis that refers to a pointing gesture or to an elliptic expression, like in utterance (4):

(4) "Turn this appliance on."

Here, the deixis is represented with a deictic reference model that restricts the referred instance to an `uch:Appliance` with a power-mode functionality. Example c) introduces a lamp entity to the discourse. This occurs when the user mentions the lamp by speech or points with his finger on a graphical object that represents it.

The fusion and discourse resolution engine follows several resolution strategies to fill the missing content for an incomplete interaction act. When receiving input from two different modalities it tests whether the semantic information of one modality unifies with the reference model pattern of the other one. So, in figure 6 the interaction act b) and c) would unify to the complete interaction act a). Notice that the reference pattern in example b) wouldn't unify with an object that is no appliance with power mode functionality, like a chair. Then the fusion and discourse resolution engine would try to find models from preceding turns for interaction act completion. When a television was introduced in a previous turn, the reference pattern would unify with it, based on the fact that this is an appliance with a power mode functionality.

3.4 Graphical Representation and Presentation

In most scenarios, multimodal dialogue applications support a graphical user interface. Hence, a model for graphical output representation is a central component in application development. We introduce a presentation model that describes the actual graphical output of an application.

Graphical Representation Model According to the principle *"No presentation without representation."* (Maybury and Wahlster 1998), we include a set of abstract concepts for view components in the base dialogue ontology. Every view component is derived from the concept `gui:ViewComponent`. Derived concepts `gui:InactiveViewComponent` and `gui:ActiveViewComponent` indicate the interaction capability of inherited view components. Active components are components the user can interact with, like a button or lever. Inactive components only present information, e.g., a label or an image. For creating hierarchical structures, the concept `gui:ViewComponent` owns a property `gui:hasViewComponent` that allows graphical components to contain others.

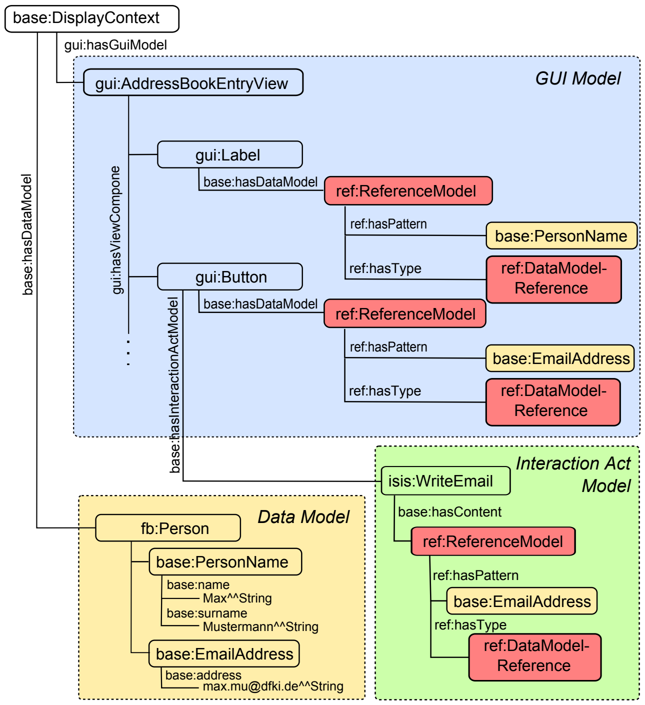


Figure 7: The display context as a combination of gui, data and interaction act models

Similar to the idea of RDFa, we enrich the view components with the semantic content they represent. Active view components can additionally embed an interaction act that defines the interpretation of the user's intention when interacting with the view component. We call the model for the graphical user interface together with the additional information about semantic content and interaction acts the *display context*.

Figure 7 shows an example of a concrete display context. The GUI displays an address book entry (represented by `gui:AddressBookEntryView`) that contains a label for the person's name. An interactive button displays the e-mail address. Pushing the button triggers the underlying interaction act (write an e-mail with the defined e-mail address). Models for user intentions are directly integrated into the definition of the GUI. So an interaction of the user with a view component can later directly be mapped onto an interaction act hypothesis.

```
<object type="base#TaskRequest">
  <slot name="base#hasContent">
    <object type="uch#ManipulateTargetTask">
      <slot name="uch#hasTarget">
        <object type="uch#Lamp">
          <slot name="uch#identifier">
            <value type="String">lamp01</value>
          </slot>
          <slot name="uch#hasState">
            <object type="uch#PowerModeState" />
          </slot>
        </object>
      </slot>
    </object>
  </slot>
  <slot name="base#hasContent">
    <object type="uch#BooleanStateCommand">
      <slot name="base#identifier">
        <value type="String">powerMode</value>
      </slot>
      <slot name="uch#commandType">
        <value type="String">set</value>
      </slot>
      <slot name="uch#hasValue">
        <object type="base#BooleanValue">
          <slot name="base#hasBooleanValue">
            <value type="Boolean">true</value>
          </slot>
        </object>
      </slot>
    </object>
  </slot>
</object>
```

(a)

```
<object type="base#TaskRequest">
  <slot name="base#hasContent">
    <object type="uch#ManipulateTargetTask">
      <slot name="uch#hasTarget">
        <object type="ref#ReferenceModel">
          <slot name="ref#hasPattern">
            <object type="uch#Appliance">
              <slot name="uch#hasState">
                <object type="uch#PowerModeState" />
              </slot>
            </object>
          </slot>
          <slot name="ref#hasType">
            <object type="DeicticReference" />
          </slot>
        </object>
      </slot>
      <slot name="base#hasContent">
        <object type="uch#BooleanStateCommand">
          <slot name="base#identifier">
            <value type="String">powerMode</value>
          </slot>
          <slot name="uch#commandType">
            <value type="String">set</value>
          </slot>
          <slot name="uch#hasValue">
            <object type="base#BooleanValue">
              <slot name="base#hasBooleanValue">
                <value type="Boolean">true</value>
              </slot>
            </object>
          </slot>
        </object>
      </slot>
    </object>
  </slot>
</object>
```

(b)

```
<object type="base#Inform">
  <slot name="base#hasContent">
    <object type="uch#Lamp">
      <slot name="uch#identifier">
        <value type="String">lamp01</value>
      </slot>
      <slot name="uch#hasState">
        <object type="uch#PowerModeState" />
      </slot>
    </object>
  </slot>
</object>
```

(c)

Figure 6: Instances of interaction act models with diverse complexities.

Display Context Generation In the first generation of our framework, display context representations were created manually and filled with semantic content by handwritten rules. This implied a lot of manual work for adapting, extending or creating new views. In a next step, we introduced templates for views that can be connected to arbitrary data models. Figure 7 shows an example for such a template. A view for an address book entry contains several view components for the visualization of contact information. A label shows the person's name and a button displays the e-mail address. The actual content that is presented to the user is connected to the view components by means of the reference model. It refers to the data model attached to the display context. For this, we extended the reference model with a concept `ref:DataModelReference`. The `ref:hasReferenceObject` restricts the possible content items for the view component to concepts that unify with a semantic pattern, here a `base:PersonName` for the label and an `base:EMailAddress` for the button. The references are resolved at run-time. The process applies a breadth first search on the data model structure to find an instance that matches the given restrictions in the reference model. Finally, the referred content for the label and the button are filled with the actual name and e-mail address of Max Mustermann from the attached data model.

Also, the description of the interaction act can be defined by a template. The e-mail button of the addressbook view contains an interaction act that describes the system's behaviour when the user pushes the button. Here, the data model is also defined by a reference. The resolution component retrieves and fills the missing content of the interaction act by searching an instance in the `base:hasDataModel` slot that unifies with the given pattern.

The loose coupling of the data model and the interaction act allows developers to rapidly change the content of the presented data and the behaviour of the system even during runtime by just exchanging the connected models. Even data of different types can be attached as long as it contains instances that match the patterns specified by the reference models.

4 Demonstrators

We already applied our described model-driven development approach and the ODP as semantic-based multimodal dialogue application framework for several demonstration systems and applications. In the following sections, we will briefly describe three of them, covering a wide range of usage scenarios and domains. Although the presented demonstrators do not retrieve missing or resolve ambiguous information by asking the user, the former Babble-Tunes system (Schehl et al. 2008), also developed with ODP, implements this functionality. A future work is to generalize the knowledge we gained from this demonstration system in order to derive concepts for the easy integration of clarification dialogues into new systems.

4.1 Smart Kitchen Control

The smart kitchen at the German Research Center for Artificial Intelligence in Saarbrücken is a completely equipped kitchen where all appliances are accessible via a network connection. The type of connection and protocols are a various set of different technologies. Hood, oven, hob, fridge and freezer are connected via powerline technology and can be controlled via a WSDL/SOAP interface. The light in the kitchen and additional sensor technology like movement, light, temperature or contact sensors for windows and doors are accessed via the battery and wireless radio technology EnOcean. A television, in form of the Windows Media Center, runs on a desktop PC. All appliances of the smart kitchen are integrated in the middleware technology UCH that implements the ISO/IEC 24752 Universal Remote Console standard (Zimmermann and Vanderheiden 2007) and provides the back-end service for a multimodal dialogue appliance control.

For this use-case, the dialogue ontology is extended by concepts for appliance and functionality description. That makes it possible to distinguish between discrete and continuous functionalities. E.g., a power-mode functionality is represented as a boolean value, so it only allows the commands on, off and toggle. A continuous value can be increased, decreased, and set to a certain value. This consistent modelling approach allows the dialogue system to connect user intentions to the correct appliance functionalities by unification.

Figure 8 shows the remote kitchen control, a mobile client running on the android platform. It enables multimodal dialogical interaction with the kitchen by (i) providing a graphical user interface that supports pointing gestures, (ii) streaming capabilities for sending and receiving audio data, and (iii) integrating a toolkit for gesture classification (Neßelrath and Alexandersson 2009). The gesture classifier exploits the accelerator sensor data of the device and enables control with hand gestures. Independent of the modality, all interpreters deliver interaction act models as described in section 3.3. This allows the system to resolve several dialogue phenomena discussed in the previous chapter:

- **Multimodal Fusion:** Speech commands are combined with pointing gestures. A user can point to an appliance symbol on the screen and give the speech command: "Turn this on."
- **Elliptic Reference:** The context for an incomplete command can be reasoned from previous turns. This allows the system to understand the following sequence of commands:
  "Turn on the Hood."
  "Increase the light setting."
  "Brighter."
- **Context resolution for hand movement gestures:** Movement gestures are interpreted as an appliance functionality manipulation without defining an appliance. Imagine that describing a circle is the meaning to turn on an appliance. The appliance of interest is reasoned from a previous turn or the actual display context of the screen.

4.2 Mobile Claims Notification

Car drivers that sustain a collision with, e.g., a game animal have to inform their car insurance of the incident if they

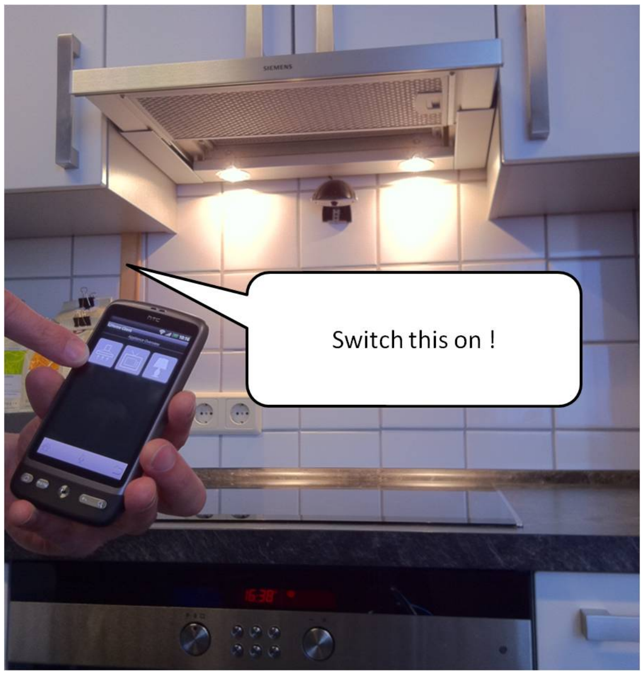


Figure 8: Multimodal dialogue remote control for the smart kitchen.

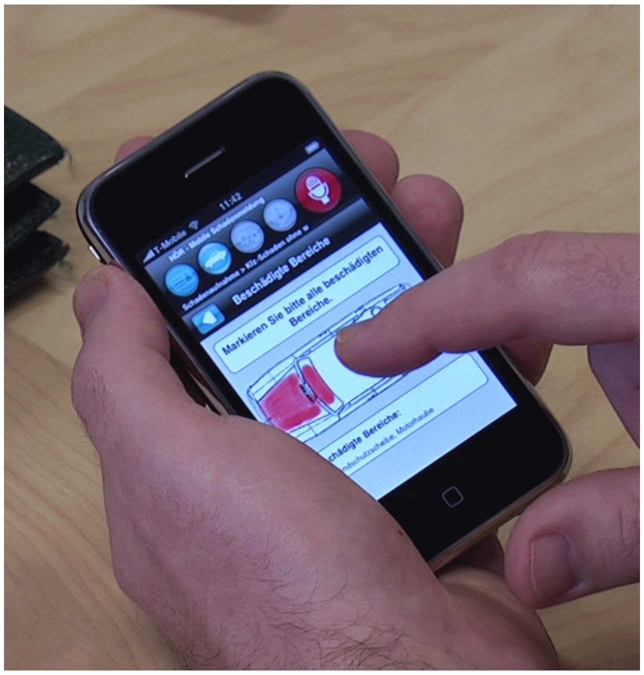


Figure 9: Multimodal dialogue user interface for mobile claims notification.

want to get the car repaired and a rental car in between without excessive costs. Usually, the initiation of such a rather complex and long-lasting business process takes some time, because various stakeholders participate in the process. Focussed on the claimant (the car owner), a mobile application allows an easy and fast initiation of the claims notification right from the location of the accident and further enables a user to participate and observe the subsequent claims adjustment. This is achieved by providing access to the back-end business process via a mobile multimodal dialogue user interface. Here, the GUI layout as shown in figure 9 reflects the relevant process structure from the claimant's perspective (identification, damage survey & claims notification, claims adjustment, and feedback) such that the current progress can be perceived immediately.

Technically, the mobile client runs on an iPhone and consists of a full-screen Web browser component for rendering the DHTML GUI and (analog to the remote kitchen control) a streaming component for sending and receiving audio data. The following dialogue phenomena are supported:

- **Multimodal Fusion:** Speech commands are combined with location information. A user can ask for the nearest repair shop and a route how to get there.
- **Elliptic Reference:** The context for an incomplete command can be reasoned from a previous turn. This allows the system to understand the following sequence of commands:

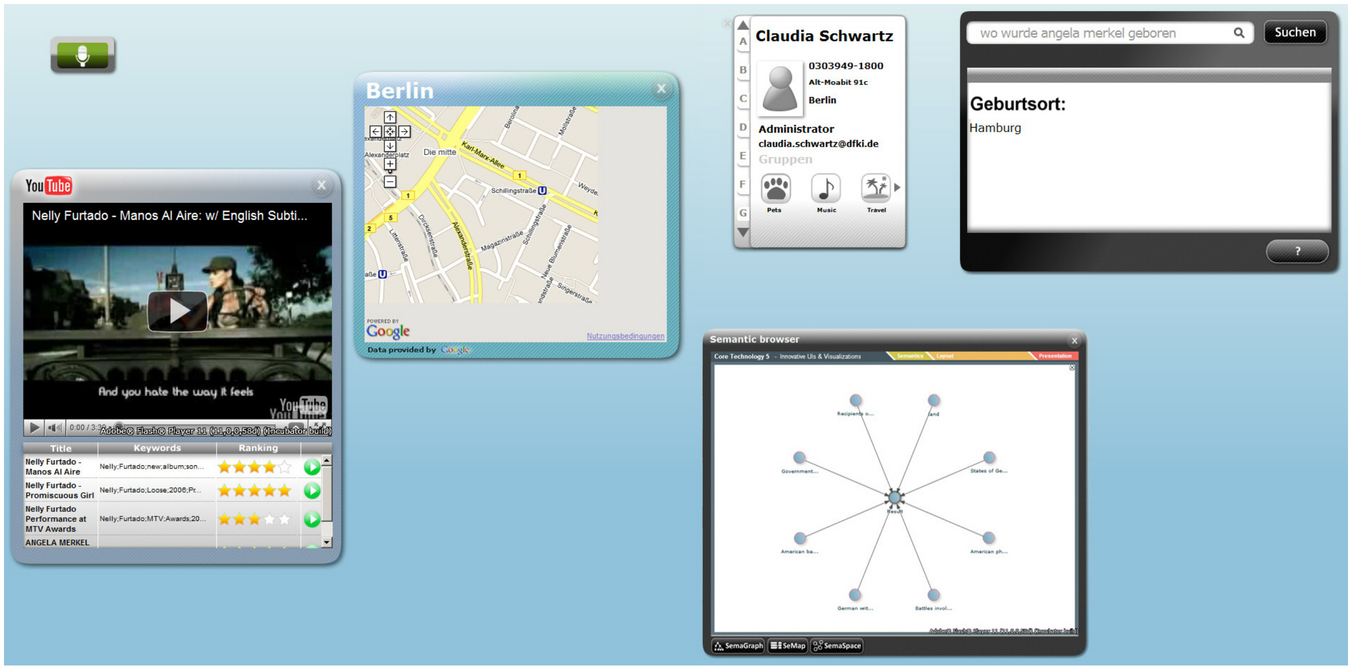
  "The front fender on the left is damaged."
  "And also the outside mirror on that side."
- **Mixed-initiative dialogue**: Since the application communicates with the back-end business process, it can receive notifications (*"Your car has been repaired."*) and information requests (*"Is your mobile phone number still valid?"*).

4.3 ISIS Information System

ISIS (Interacting with Semantic Information Services) is a multimodal dialogue information system that allows one or more users to interact with semantically annotated Web content from different sources (figure 10). In detail, it processes Wikipedia content via DBPedia, a music ontology and contact information from a personal addressbook. It supports natural language understanding of spoken language and typed input, pointing gestures, and graphical interactions. Additionally it allows to interact with multimedia content and web-based services like Google Maps and YouTube. The system interprets diverse input modalities and allows fast access to an ontological representation of extracted information. The client is running on a PC with touchscreen and renders its screen with HTML5. Again, several dialogue phenomena are supported:

- **Multimodal Fusion:** Speech commands are combined with pointing gestures. A user can point on an entry or a picture of a town and say: "Show this city on the map."
- **Elliptic Reference:** The context for an incomplete command can be reasoned from a previous turn. This allows the system to understand the following sequence of commands:

  "What is the name of Barack Obama's wife?"
  "Where is (s)he born?"
  "Show the city on the map."



Title	Keywords	Ranking
Nelly Furtado - Manos Al Aire	Nelly;Furtado;new;album;son...	★★★★☆
Nelly Furtado - Promiscuous Girl	Nelly;Furtado;Loose;2006;Pr...	★★★★★
Nelly Furtado Performance at MTV Awards	Nelly;Furtado;MTV;Awards;20...	★★★☆☆
ANGELA MERKEL		

Figure 10: Graphical user interface of the multimodal dialogue ISIS information system.

- **Identical interactions with content from different services**: Because the semantic objects for person instances from Wikipedia and the personal addressbook are derived from a common base ontology, rules for task processing of content from one service work also with content from the the other one.

5 Conclusion & Future Work

The paper presents a model-driven development approach for multimodal dialogue systems. Semantic models are introduced that represent semantic content, referring expressions, user intentions and graphical user interfaces. These models act as a contract for the integration of new modalities and domains into multimodal dialogue applications. The model-driven approach allows a loose coupling of content, interaction act and presentation models and enables the rapid adaptation of content and behaviour, even during runtime. Deriving concepts for domain-specific content from well-chosen basic concepts still allow generic processes to operate in new domains. Reasoning on semantic content supports advanced dialogue phenomena. Similar techniques are used to fill context-dependent placeholders in patterns for user intentions and graphical representations. The development approach was adopted by three multimodal dialogue applications for different domains. We plan to evaluate the system by giving the authoring tools to some groups outside our own research group to collect more information about usability and the learning curve.

In the paper, we do not deal with models for dialogue management. Future work will extend the resolution of referring expressions with the option of asking clarifying questions to the user. Semantic models can help the system to identify missing information and to select suitable situation-adapted callbacks to the user. Semantic models and the connection to lexical databases like WordNet can be exploited to generate language models for speech generation and recognition. Future investigations will also include more fine-grained coordination of multimodal input and output. A further important point is a user model that should be taken into account in all steps of the dialogue process to provide a user-centered application with personalized appearance and behaviour.

6 Acknowledgement

Parts of this work have been carried out in the scope of the Theseus programme funded by the German Federal Ministry of Economics and Technology (01MQ07016) and the i2home project funded by the European Commission under the grant 033502. The opinions herein are those of the authors and not necessarily those of the funding agency. Special thanks go to Matthieu Deru.

References

Adida, B.; Birbeck, M.; McCarron, S.; and Pemberton, S. 2008. RDFa in XHTML: Syntax and processing – a collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation. http://www.w3.org/TR/rdfa-syntax/.

Alexandersson, J.; Becker, T.; and Pfleger, N. 2006. Overlay: The basic operation for discourse processing. In Wahlster, W., ed., *SmartKom: Foundations of Multimodal Dialogue Systems*, Cognitive Technologies. Springer Berlin Heidelberg. 255–267. 10.1007/3-540-36678-4$_1$7.

Alexandersson, J.; Buschbeck-Wolf, B.; Fujinami, T.; Kipp, M.; Koch, S.; Maier, E.; Reithinger, N.; Schmitz, B.; and Siegel, M. 1998. Dialogue acts in verbmobil. Verbmobil report, DFKI, Saarbrücken.

Allemang, D., and Hendler, J. 2008. *Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL*. Morgan Kaufmann.

Ankolekar, A.; Krötzsch, M.; Tran, T.; and Vrandecic, D. 2007. The two cultures: mashing up web 2.0 and the semantic web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 825–834. New York, NY, USA: ACM.

Araki, M., and Funakura, Y. 2010. Impact of semantic web on the development of spoken dialogue systems. In *IWSDS*, 144–149.

Becker, T.; Blaylock, N.; Gerstenberger, C.; Kruijff-Korbayová, I.; Korthauer, A.; Pinkal, M.; Pitz, M.; Poller, P.; and Schehl, J. 2006. Natural and intuitive multimodal dialogue for in-car applications: The Sammie system. In *Proceedings of ECAI 2006, 17th European Conference on Artificial Intelligence, and Prestigious Applications of Intelligent Systems (PAIS 2006), Riva del Garda, Italy*, 612–616.

Bergweiler, S.; Deru, M.; and Porta, D. 2010. Integrating a multitouch kiosk system with mobile devices and multimodal interaction. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, 245–246. New York, NY, USA: ACM.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web: Scientific American. *Scientific American*.

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. Dbpedia - a crystallization point for the web of data. *Web Semant.* 7(3):154–165.

Bolt, R. A. 1980. Put-that-there: Voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '80, 262–270. New York, NY, USA: ACM.

Bunt, H.; Kipp, M.; Maybury, M.; and Wahlster, W. 2005. Fusion and coordination for multimodal interactive information presentation. In *Multimodal Intelligent Information Presentation*, volume 27 of *Text, Speech and Language Technology*. Springer Netherlands. 325–339. 10.1007/1-4020-3051-$7_1$5.

Bunt, H.; Alexandersson, J.; Carletta, J.; Choe, J.-W.; Fang, A. C.; Hasida, K.; Lee, K.; Petukhova, V.; Popescu, A.; Soria, C.; and Traum, D. 2010. Towards an iso standard for dialogue act annotation. In *Proceedings of 7th International Conference on Language Resources and Evaluation(LREC-2010), May 19-21, Malta*.

Bunt, H. 2000. *Abduction, Belief and Context in Dialogue*, volume 1 of *Natural Language Processing*. Amsterdam, The Netherlands: John Benjamins. chapter Dialogue Pragmatics and Context Specification, 81–150.

Maybury, M. T., and Wahlster, W., eds. 1998. *Readings in intelligent user interfaces*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Milward, D., and Beveridge, M. 2003. Ontology-based dialogue systems. In *Proceedings of the 3rd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Acapulco, Mexico*, 9–18.

Neßelrath, R., and Alexandersson, J. 2009. A 3d gesture recognition system for multimodal dialog systems. In *Proceedings of the 6th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems (KRPD-09)*, 46–51. Pasadena, California, United States: IJCAI 2009.

Pfleger, N. 2004. Context based multimodal fusion. In *Proceedings of the 6th international conference on Multimodal interfaces*, ICMI '04, 265–272. New York, NY, USA: ACM.

Pfleger, N. 2007. *Context-based Multimodal Interpretation: An Integrated Approach to Multimodal Fusion and Discourse Processing*. Ph.D. Dissertation, Universität des Saarlandes.

Porta, D.; Sonntag, D.; and Neßelrath, R. 2009. A Multimodal Mobile B2B Dialogue Interface on the iPhone. In *Proceedings of the 4th Workshop on Speech in Mobile and Pervasive Environments (SiMPE '09) in conjunction with MobileHCI '09*. ACM.

Schehl, J.; Pfalzgraf, A.; Pfleger, N.; and Steigner, J. 2008. The babbletunes system: talk to your ipod! In Digalakis, V.; Potamianos, A.; Turk, M.; Pieraccini, R.; and Ivanov, Y., eds., *ICMI*, 77–80. ACM.

Sonntag, D., and Möller, M. 2010. A multimodal dialogue mashup for medical image semantics. In *IUI '10: Proceeding of the 14th international conference on Intelligent user interfaces*, 381–384. New York, NY, USA: ACM.

Sonntag, D.; Engel, R.; Herzog, G.; Pfalzgraf, A.; Pfleger, N.; Romanelli, M.; and Reithinger, N. 2007. *SmartWeb Handheld - Multimodal Interaction with Ontological Knowledge Bases and Semantic Web Services (extended version)*. Number 4451 in LNAI. Springer. chapter 14, 272–295.

Sonntag, D.; Sonnenberg, G.; Neßelrath, R.; and Herzog, G. 2009. Supporting a rapid dialogue engineering process. In *Proceedings of the First International Workshop On Spoken Dialogue Systems Technology (IWSDS)*.

Sonntag, D.; Porta, D.; and Setz, J. 2010. Http/rest-based meta web services in mobile application frameworks. In *Proceedings of the 4nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. (UBICOMM-10)*. XPS.

Wahlster, W., ed. 2006. *SmartKom: Foundations of Multimodal Dialogue Systems*. Berlin, Heidelberg: Springer.

Zimmermann, G., and Vanderheiden, G. C. 2007. The universal control hub: An open platform for remote user interfaces in the digital home. In Jacko, J. A., ed., *HCI (2)*, volume 4551 of *Lecture Notes in Computer Science*, 1040–1049. Springer.