Natural Language Understanding

Ralf Engel

DFKI GmbH, Saarbrücken, Germany
ralf.engel@dfki.de

Summary. This chapter presents SPIN, a newly developed template-based semantic parser used for the task of natural language understanding in SMARTKOM. The most outstanding feature of the approach is a powerful template language to provide easy creation and maintenance of the templates and flexible processing. Nevertheless, to achieve fast processing, the templates are applied in a sequential order that is determined offline.

1 Introduction

The task of the natural language processing module is to transform the word lattice sent by the speech recognizer into a list of hypotheses representing in an abstract way possible user intentions. For this task a new template-based semantic parsing approach called SPIN (Engel, 2002) was developed within the SMARTKOM project. This chapter motivates and describes this approach.

1.1 Module Input and Output

First, a closer look at the module input and output should be given. The input consists of spoken utterances, which are intended to interact with a computer system. Therefore, they are usually syntactically less complicated than in other surroundings, e.g., human–human interactions, and they are also limited in length. But the utterances are often realized syntactically incorrectly by the user (because the user has to think and speak at the same time). Additionally, they contain many speech recognition errors due to a word error rate that ranges between 7.9% and 21.1% (dependent on the test set).

The output hypotheses are expressed as instances of the SMARTKOM ontology, which means that the output representation is already on a high semantic–pragmatic level and encodes direct instructions for the action planner. This entails that during the analysis of a spoken utterance many details can be ignored, as long as the representation is detailed enough to trigger the desired action in the action planner.

1.2 Approach

For the described scenario a semantic parsing approach is best suited, and semantic parsers are also widely used within dialogue systems. Characteristic of semantic parsers is that they do no syntactical analysis or only perform a very limited one but build up more or less directly the high-level output structure from word level. Examples for semantic parsers based on variations of finite state transducers (FSTs, see Roche and Schabes (1997)) are Miller and Stallard (1996), Potamianos and Kuo (2000) and Ward (1991). Examples for semantic parsers based on context-free grammars (CFGs) are Allen (1994), Gavald (2000), Kaiser et al. (1999), Lavie (1996) and Seneff (1992).

Instead of using one of the existing approaches for SMARTKOM, a new approach was developed. The main reason to develop a new approach and not to use one of the existing approaches was that writing and maintaining of the rules should be simplified compared to existing approaches. This includes especially the writing of rules that are robust against speech recognition errors and rules that are robust against word reordering in free word order languages like German. Since the rules have to be manually created[1] and the number of rules is quite large (435 templates are needed to cover all SMARTKOM applications), easy creation and maintenance of the rules is one of the most important issues for parsers in dialogue systems.

This is achieved mainly by allowing more powerful rules. This results in a more direct and natural encoding of rules. Most rules look like partial or complete utterances, where some of the words are exchanged with conditions on instances of the SMARTKOM ontology. Hence the rules are called templates in the following. The increased rule power also reduces the need for tricky constructions caused by the lack of expressive power. Such constructions are often time consuming to write and test, and later on are hard to understand and maintain for other template writers. The increased expressive power includes set-based operation, more than one expression on the left side, optional parts, additional constraints and the possibility of subsequent modifications of embedded structures. A more detailed presentation of the template language is given in Sect. 2.

1.3 Processing

The increased rule power has the effect that conventional parsing strategies and optimization techniques are either not sufficient or not applicable. Therefore, several measurements were developed to nevertheless achieve fast processing. Fast processing is an important issue since a dialogue system is an interactive system. Furthermore, the speech recognizer delivers not only one hypothesis but many hypotheses encoded in a word lattice. The lower the parsing time of one hypothesis, the more hy-

[1] Data-oriented rule creation is still a topic of research, e.g., Minker et al. (1996); Schwarz et al. (1997); Maedche et al. (2002) and is not fully solved. Additionally, the collection and annotation of the data is also very time consuming and error-prone.

potheses can be processed. Together with a clever selection mechanism[2] to choose one of these hypotheses, this can enhance the overall performance of the dialogue system.

The reduction of parsing time is mainly achieved by trying to avoid wasting time with the generation of results that are probably suboptimal. Results regarded as suboptimal are, e.g., results that are nonmaximal or results that are generated using general templates in cases where more special templates are available. The generation of such results is avoided by a fixed application order of the rules and by grouping rules that lead to alternative results using the same input. The application of rules is discussed in Sect. 3, the preprocessing of the rule set, like the computation of the template order, is presented in Sect. 4.

2 Template design

The template design is inspired by production systems where rules operate on a working memory (WM), i.e., a rule selects some elements of the WM, may delete some of them and adds new elements. In the presented approach the idea is similar: the templates operate on a WM that contains as possible elements words and instances of the SMARTKOM ontology. Each template consists of a condition part, a constraint part and an action part. The actions are only applied if all conditions and constraints can be satisfied.

2.1 Conditions

The condition part contains conditions on words and on instances of the ontology. Conditions on words can include tests concerning orthography, stem, semantic category and part of speech. A test on the stem instead of the orthography helps to be more robust against wrongly recognized inflections, a frequent problem in the German speech recognizer. A test on the semantic category is useful to match all words of a specific semantic type, e.g., all actors. A test on part of speech allows writing of some kind of common syntactic rules and partially compensates the lack of a syntactic analysis. Conditions on instances include checking of the type and tests on slots and their assigned instances. For the type check the type hierarchy of the SMARTKOM ontology is regarded. Tests on the embedded instances can again contain tests on their embedded content.

Each condition can be marked as optional, i.e., the condition does not need to match. Robustness against speech recognition errors can be increased if nonessential conditions are marked as optional. Also, each condition can have an associated variable that the matched element is bound to. That way, the actions of the same template can refer to elements matched by conditions. Sets of alternative conditions

[2] In SMARTKOM a first selection is done in the parser itself; further selections are done by the multimodal fusion and the intention analyzer considering more context.

```
or([road map],[car map],[vehicle map])
roadMap()

$S=word(semCat:SIGHT)
sight(location(locationName:$S))

[or(the,this) $O=physicalObject($R=refProp())]
$O($R(det:det))

there
to() sight(refProp(det:udet))

there
sight(refProp(det:udet))

[$M=streetMap() %with $S=sight()]
$M(object:$S)

tell %me %more [%about $I=sight()]
informationSearch(pieceOfInformation:$I)

%how %do %i or(go,get) %$R=route() %[from $L1=sight()]
    %[to $L2=sight()] !atLeastOne($L1,$L2)
motionDirectedTransliterated(source:$L1,target:$L2,path:$R,
    meansOfTransportation:onFoot())
```

Fig. 1. Excerpt from the SMARTKOM template set. The templates are separated by empty lines; the first line of each template contains the conditions and the constraints (marked with *!*); the second line, the actions. Optional conditions are marked with %. List-based matching is activated using square brackets

can be expressed to reduce the number of required templates. Negation of conditions is also available, i.e., a condition must not match any element in the WM.

By default, conditions can match elements at any positions in the WM (further referred to as set-based matching). Writing templates for free word order languages is greatly simplified by set-based matching. This can be changed by grouping conditions in so called sequence conditions (further referred as sequence-based matching). In this case the matched elements in the WM have to be exactly in the same order as the corresponding conditions in the sequence condition and other elements must not be in between. Set-based matching and sequence-based matching can be mixed in one template.

2.2 Constraints

Constraints allow us to express conditions over the content of several variables after the conditions have matched. Currently, three different constraints exist. *atLeastOne*

checks if at least one of the specified variables is bound. This is useful if a template has several optional conditions, but at least one has to be satisfied to produce a meaningful result. *checkOrder* checks the order of the elements bound by variables in the WM. This constraint can be used if the word order is relatively free but not completely negligible. In such a case the use of set-based matching together with *checkOrder* constraints is appropriate. *isOfType* can be used to check if the type of an instance was refined (see also Sect. 3.4).

2.3 Actions

Possible elements in the action part are constructors for words, constructors for instances of the SMARTKOM ontology and variables mentioned in the condition part. Constructors for words insert new words into the WM. This can be used to preprocess the WM so that a later applied template matches. This helps to reduce the complexity of templates. Constructors for instances insert new instances of the SMARTKOM ontology. Embedded content of the instances can be specified directly in the constructor. Within the embedded content it is possible to refer to variables of the condition part.

2.4 Robustness

Often the input utterance is not fully correct either due to speech recognition errors o. errors produced by the user. The templates have to be robust against three different types of recognition errors: deletion of words, insertion of words and exchange of words.

Errors occurring as the deletion of words are addressed by marking less essential conditions as optional. Less essential conditions may include tests on modal verbs, pronouns, prepositions etc. It is only important that an utterance is not unintentionally matched by the template. Of course, if essential words are missing, this does not help.

The insertion of words does not harm in the case of set-based processing. Additional words are simply ignored. If list-based processing has to be used, erroneous processing cannot be avoided. But often the input can be analyzed at least partially since list-based processing is used often for modifying instances.

Errors occurring as the exchange of words can be divided into two categories. The exchanged words are either wrong inflections of the original words (often a problem with the German speech recognizer) or completely different words. The first case is addressed by using the stem of a word as default for matching instead the orthography. The second case can be reduced to a combination of the insertion and the deletion of words.

3 Processing

3.1 Major Internal Data Structures: Working Memory and Agenda

The templates are applied on a working memory (WM). There may exist several WMs at the same time, each representing an alternative result. A WM contains words

and instances of the ontology. For each word and instance, the transitions of the word lattice that are used for creating and modifying the word or instance are stored. List-based matching uses this information to decide if two elements are directly in succession.

To support chart parsing the WM can also contain local alternatives. Local alternatives are structures that store words and instances that can be used only alternatively but not together. A sequence structure is available to store several words and instances within one alternative.

The second important internal data structure is the agenda. During processing at many points several possibilities exist how to proceed. In such cases one possibility is chosen, and the rest is stored in the agenda to be computed later.

3.2 Processing Overview

The WM is initially filled with the words of the best path through the word lattice. Before further processing, the words are enriched by syntactic and semantic information looked up in the lexical database. To also be able to process the lower scored paths in the word lattice, possible alternative branches in the word lattice are stored in the agenda.

The next step is the application of the templates in a predefined fixed order. The application order is not a sequential but a structured one. More details are given in Sect. 3.3.

In the case when a template can be applied in several ways, one way is chosen and all the necessary data to continue the other ways later on are stored in the agenda. More details on the application of a single template are given in subsection 3.4.

To build up a chart, the application of the next template is delayed in some cases, and first some of the agenda entries resulting from the last template application are continued. After the application the different results are merged, and local alternatives are created where necessary. The number of selected entries determines how fast the first result is found versus how effective the next results are generated.

After the application of all templates is finished, the first parsing solution is simply the current content of the WM, which is stored in a result list.

To produce the next solution one entry is selected from the agenda,[3] and the application continues where the processing was interrupted. This continues until the agenda is empty or a timeout (default 500 ms) is expired.

A scoring function (presented in Sect. 3.5) selects the n best results from the result list and sends them encoded in M3L to multimodal fusion (n defaults to 10).

Figure 2 gives an overview of the architecture; Fig. 3 shows the application of the templates in more detail.

[3] In SMARTKOM the agenda is only handled as stack (first in, last out). A more intelligent selection using a scoring function is the subject of current research.

3.3 Template Application Order

The application order is determined by grouping templates in so-called *blocks*, which can be nested. The blocks are somewhat similar to control structures in programming languages. Four different types of blocks are available:

- **Sequential block:** The embedded blocks and templates are applied in the specified order.
- **Repeat block:** The embedded blocks or templates are repeated until no block and no template can be applied anymore.
- **Alternative block:** The embedded blocks and templates are applied alternatively. If more than one block or template can be applied, the first one is applied and the alternatives are stored in the agenda.
- **Optional block:** The embedded block or template is applied optionally, i.e., in the case the block or template can be applied, the alternative of not applying this block or template is put on the agenda.

A simple sequential application of the templates is not sufficient since in such a case many relevant results could not be generated. Which block is used when and why is described in Sect. 4, in which the automatic offlinegrouping of the templates into blocks is described.

3.4 Application of One Template

A template is applied to the WM in four steps:

1. Each condition is tested to see if it is satisfiable with respect to the WM.
2. Alternative solutions omitting optional conditions are generated.
3. All constraints specified in the template are checked.
4. The tokens in the WM matched by the conditions are replaced by tokens generated by the corresponding actions in the action part of the template.

The steps cannot be presented in full detail here; only some features should be mentioned.

Refinement

Referring expressions are supported by allowing ontology instances to refine their types during matching. If a condition requires a more specialized type than the instance in the WM has, the instance can refine its type to match the condition. The instance retains the refined type during further processing.

This allows us to transform a referring expressing like *this one* first to an instance of the most common type for things (in SMARTKOM *physicalObject*). When a template later requires, e.g., an instance of type *avMedium*, the instance representing the referring expression can refine its type from *physicalObject* to *avMedium*. This refined type can be used later in the modality fusion for referential resolution, e.g., for querying the dialogue history.

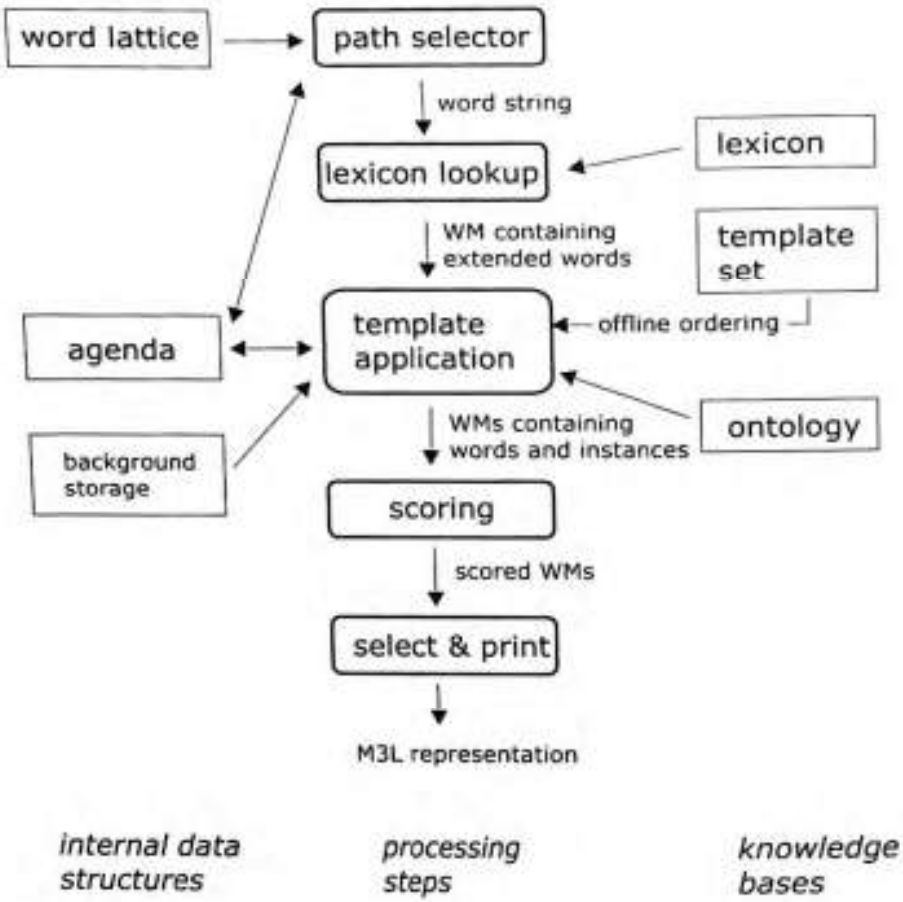


Fig. 2. The architecture of SPIN. The data flow is shown together with the knowledge bases and the most important internal data structures

Since it is possible that the user has meant an instance of a different subtype, alternative solutions with possible remaining types (original type without refined type) are also considered by adding a suitable entry to the agenda.

Optional Conditions

Templates may contain optional conditions. It is not obvious during the application of a template if any and which alternative solutions should be generated by omitting optional conditions. The generation of all possible solutions would lead to many that are suboptimal since nonmaximal solutions should be avoided to achieve faster processing. But it is also not always sufficient to generate only the maximum solution. For example, regard the case that several commands within one utterance are mentioned. Because of the sequential application order and the ability of set-based processing, an optional condition can unintentionally bind elements of the wrong command. To determine which solutions should be generated is checked using an offline-generated hash map if at least one succeeding template exists that can use this solution.

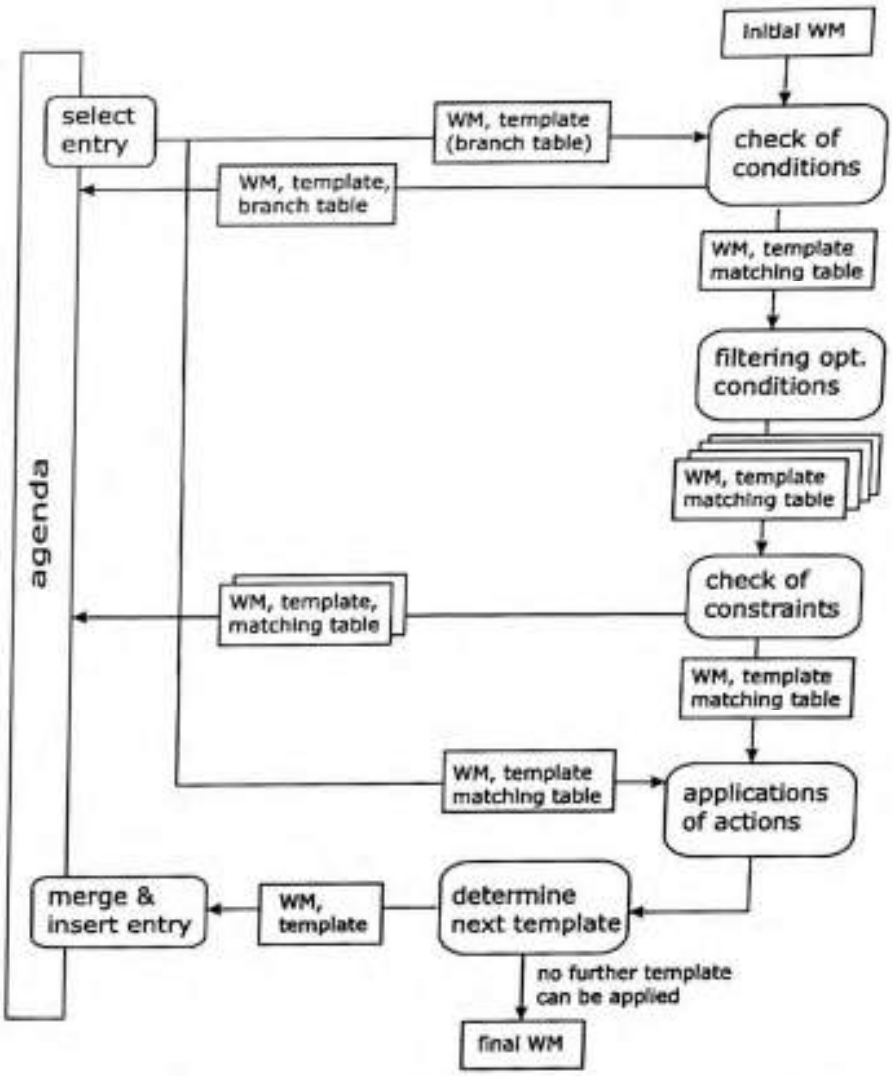


Fig. 3. The single steps within the application of the templates. additionally, the information passed between the single steps and the information that is stored in and retrieved from the agenda is shown

3.4.1 Considering Background Storage

The matching process uses instances in the background storage if the condition is not optional and no other element in the WM matches. The background storage is used for already recognized gestures to enhance robustness (see also Engel and Pfleger (2006)). If referring expressions are not recognized or are not uttered by the user, the problem can occur in which a template cannot be applied since an essential instance is missing. Putting the instances representing gestures in the background storage solves this problem.

3.5 Scoring Heuristics and Result Filtering

The scoring function has two tasks. First, the number of hypotheses sent to the modality fusion is limited. In the case that more results are generated than can be sent to modality fusion, the scoring function is used to determine which results are sent to the next module. Second, the value returned by the scoring function is used in the intention analyzer module together with scores from modality fusion, discourse modeling and situation modeling to select the best hypothesis to send to the action planner.

Two factors determine the score:

1. It is checked if either an event instance (an instance that triggers an action within the action planner) or an expected instance (e.g., a valid answer to a question) is part of the result. In this case, the score is in the interval $[0.5, 1]$. If not, the score is in the interval $[0, 0.5]$.
2. The portion of words that are used for the creation of event instances or expected instances in relation to all input words. If such instances are not present the remaining instances are regarded.

4 Offline Ordering and Grouping of the Templates

The order of the template application is determined offline in advance. Therefore, first a dependency graph is created describing which templates have to be applied before others. Subsequently, this dependency graph is transformed into the application blocks presented in Sect. 3.3.

Three different kinds of relations are possible between two templates. A relation between the templates T_1 and T_2 means that T_1 has to be applied after T_2 if all relevant results should be generated. It depends on the type of the relation if an additional application of T_1 before T_2 hinders the generation of all relevant results or leads only to additional suboptimal results. In the following, the relations are described informally minus some details; an exact definition is beyond the scope of this chapter.

- **require:** This relation expresses that template T_1 requires the application result of template T_2. For example, if T_1 modifies an instance of type X, it must be applied after a template T_2 that creates instances of type X. Another example is that T_1 deletes an instance of type X from the WM and T_2 modifies an instance of type X. T_1 must not be applied before T_2.
- **moreOrEqualGeneral:** The relation expresses that a template T_1 is more general than or equally general to a template T_2, i.e., for each condition in T_1 a corresponding condition in T_2 exists. The idea behind this relation is that it is assumed that the template writer wants that the more general template only be applied if the more specialized template cannot be applied. T_1 must not be applied before T_2.

- **modifyInstanceOfSameType:** The relation expresses that two templates modify the same instance and both templates use list-based processing. In this case, the order cannot be determined in advance, e.g., the input consists of a noun with two adjectives, and two templates are used to process the adjectives. The templates can be sorted, so that both can be applied, but as soon as the adjectives are swapped only one template matches. T_1 may be applied before T_2.

The dependency graph is built up by comparing the templates pairwise. Each node represents a template, and a transition between two nodes represents a relation. A direct linearization of the dependency graph is in most cases not possible due to cycles in the graph. The steps described in the following eliminate the cycles by combining nodes and assigning to the new node an application block containing the templates of the merged nodes. The steps are executed in the given order once.

1. All cycles according to *require* and *moreOrEqualGeneral* are detected. The nodes of each cycle are merged to one node, which is assigned a repeat block containing an alternative block. The alternative block in turn contains the templates of the merged nodes.
2. All cycles according to *require* and *moreOrEqualGeneral* and *modifyInstanceOfSameType* are detected. The nodes of each cycle are merged to one node, which is assigned a repeat block. The repeat block contains the blocks and templates in the order given by the relations *require* and *moreOrEqualGeneral*.
3. The final result is a sequential block containing the blocks and templates of all nodes. They are ordered according to the relations *require* and *moreOrEqualGeneral*. As a result of step 2 no cycles can exist.

The description of the steps lacks some introduced optimizations to avoid the generation of large alternative blocks that have a negative impact on performance. An example for an optimization is that some blocks or templates are removed from a cycle in step 1 by putting them in an optional block. An example of a dependency graph and the template application order derived from this dependency graph is shown in Fig. 4.

5 Conclusion

In this chapter SPIN, a template-based semantic parsing approach used for the task of natural language understanding in SMARTKOM, was presented. The design of the knowledge bases consisting of a lexicon, the SMARTKOM ontology and set of templates as well as the processing of the templates was motivated and described.

The approach was sufficient to produce the required output, while the templates remained clear and maintenance was relatively easy. Extensive system tests showed that the parser is fast (10 ms per path) and a timeout of 500 ms is used.

Further development will focus on the automatic optimization and extention of a given template set using machine learning methods. Optimization means here that conditions are detected automatically as nonessential and therefore can be marked as

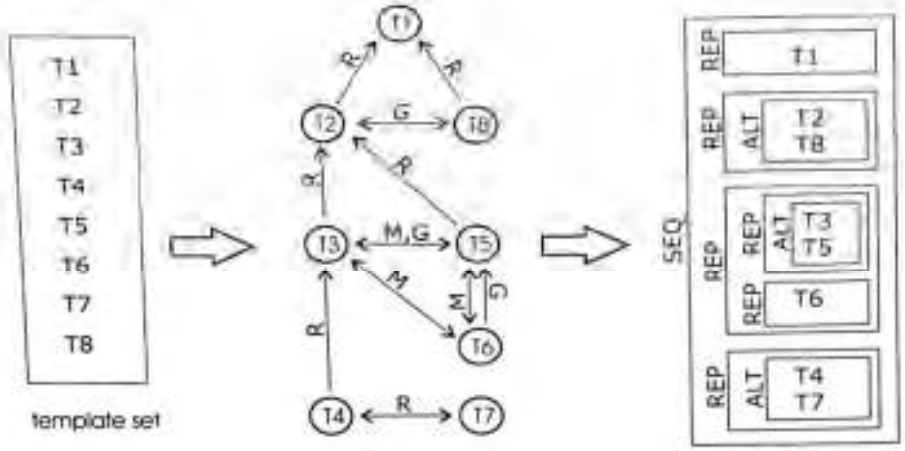


Fig. 4. Computation of the template application order starting from a set of templates and using a dependency graph as intermediate step. (*R* = Require, *G* = Moreorequalgeneral, *M* = Modifyinstanceofsametype, *Seq* = Sequential Block, *Rep* = Repeat Block, *Alt* = Alternative Block)

optional to increase robustness against speech recognition errors. Extension means here that the already existing templates are extended automatically when pairs of already working utterances and still nonworking utterances are provided.

References

J. Allen. *Natural Language Understanding*. Benjamin/Cummings, Menlo Park, CA, 2 edn., 1994.

R. Engel. SPIN: Language Understanding for Spoken Dialogue Systems Using a Production System Approach. In: *Proc. ICSLP-2002*, pp. 2717–2720, Denver, CO, 2002.

R. Engel and N. Pfleger. Modality Fusion, 2006. In this volume.

M. Gavald. SOUP: A Parser for Real-World Spontaneous Speech. In: *Proc. 6th Int. Workshop on Parsing Technologies (IWPT-2000)*, Trento, Italy, February 2000.

E.C. Kaiser, M. Johnston, and P.A. Heeman. PROFER: Predictive, Robust Finite-State Parsing for Spoken Language. In: *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-99)*, vol. 2, pp. 629–632, Phoenix, AZ, 1999.

A. Lavie. GLR*: A Robust Parser For Spontaneously Spoken Language. In: *Proc. ESSLLI-96 Workshop on Robust Parsing*, Prague, Czech Republic, 1996.

A. Maedche, G. Neumann, and S. Staab. Bootstrapping an Ontology-Based Information Extraction System. In: P.S. Szczepaniak, J. Segovia, J. Kacprzyk, and L.A. Zadeh (eds.), *Intelligent Exploration of the Web*, Heidelberg, Germany, 2002. Physica.

S. Miller and D. Stallard. A Fully Statistical Approach to Natural Language Interfaces. In: *Proc. 34th ACL*, Santa Cruz, CA, June 1996.

W. Minker, S. Bennacef, and J.L. Gauvin. A Stochastic Case Frame Approach for Natural Language Understanding. In: *Proc. ICSLP-96*, pp. 1013–1016, Philadelphia, PA, 1996.

A. Potamianos and H.K. Kuo. Statistical Recursive Finite State Machine Parsing For Speech Understanding. In: *Proc. ICSLP-2000*, Beijing, China, 2000.

E. Roche and Y. Schabes (eds.). *Finite State Language Processing*. MIT Press, Cambridge, MA, 1997.

R. Schwarz, S. Miller, D. Stallard, and J. Makhoul. Hidden Understanding Models for Statistical Sentence Understanding. In: *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-97)*, pp. 1479–1482, Munich, Germany, 1997.

S. Seneff. TINA: A Natural Language System for Spoken Language Application. *Computational Linguistics*, 18(1), 1992.

W. Ward. Understanding Spontaneous Speech: The Phoenix System. In: *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-91)*, Toronto, Canada, 1991.

The Gesture Interpretation Module

Rui Ping Shi, Johann Adelhardt, Anton Batliner, Carmen Frank, Elmar Nöth, Viktor Zeißler, Heinrich Niemann

Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
{shi,adelhardt,batliner,frank,noeth,zeissler,
niemann}@informatik.uni-erlangen.de

Summary. Humans make often conscious and unconscious gestures, which reflect their mind, thoughts and the way these are formulated. These inherently complex processes can in general not be substituted by a corresponding verbal utterance that has the same semantics (McNeill, 1992). Gesture, which is a kind of body language, contains important information on the intention and the state of the gesture producer. Therefore, it is an important communication channels in *human computer interaction*.

In the following we describe first the state of the art in gesture recognition. The next section describes the *gesture interpretation module*. After that we present the experiments and results for recognition of user states. We summarize our results in the last section.

1 State of the Art

1.1 Applications of Gesture

Gesture can be used in a wide range of applications: gesture in conventional human computer interaction (HCI), interaction through linguistic gesture and manipulation through physical contact. We cover each of these in the following.

Gesture in Conventional HCI

Under the window, icon, menu and pointing device (WIMP) paradigm, the use of mouse and pen of a graphic tablet such as that of the Wacom[1] Company are typical example applications of gesture. This kind of gestures with the help of pointing devices is intensively employed in *computer aided design* (CAD) (Sachs, 1990) and online handwriting recognition (Buxton et al., 1985). In the literature this category of gestures is called pen-based gesture. Rubine introduced the GRANDMA system (Rubine, 1991), in which the user is allowed to define arbitrary gestures interactively. These user-defined gestures can be input either through a mouse or with the help of a pen. The system is able to learn the static and dynamic properties of the gestures on the basis of some training data and subsequently analyzes them in real time.

[1] http://www.wacom.com

Interaction Through Linguistic Gesture

American Sign Language (ASL) and audio–video–speech recognition represent applications of linguistic gestures. In the ASL there exist, as a general rule, strict syntax, semantics and their mapping in the gesture configuration similar to their "acoustic" counterpart — speech. The understanding of ASL takes place in the space in which gesture and its grammatical structure are expressed through the hand movement and posture. Humans also use facial expression as well as head and body posture to support their expressions. In Attina et al. (2003) a system is described in which speech recognition is supported by the conventionalized gestures, similar to ASL. This kind of application often deals with hearing-impaired patients. Lip-reading is the only reliable way for these patients to communicate with other people in daily life, assumed that they have no hearing device and have not yet learned some strict sign language like ASL. The speech accompanied by such gestures is referred to as Cued Speech (Cornett, 1967). Moreover, linguistic gestures can be used in HCI as artificial commands, which a computer can interpret and execute. The Morpha system in Lütticke (2000) can be controlled through dynamic gestures, which are learnde by the system offline. This kind of gesture is more intuitive and flexible in comparison with the gestures in ASL.

Manipulation Through Gesture

Through the commitment of data glove and touch screen, the user can physimechanically interact with (virtually) presented objects in 2D/3D space. Virtual reality is by all means the direct application of such gestures, in which the user gesticulates with the virtual environment with relatively free gestures, as if he or she were also a part of that world. The use of the data glove can even improve the impression of authenticity by providing the user with feedback in response to the gesture input such as through pressure and temperature.

1.2 Approaches in Gesture Recognition

There are different methods in the field of gesture analysis. These methods are shown in the following with respect to sensor and recipient.

From the View of the Sensor

The very first attempts were data gloves, which were equipped with light sensors made of fibreglass (Zimmerman and Lanier, 1987; Marcus and Churchill, 1988; Eglowstein, 1990). The light sensors installed on the fingers convert each finger movement, like bending, rotation and outstretching, into analog signals, which are in turn used to calculate the angles of the joints of the fingers, their respective positions and the orientation of the hand. Different configurations and postures of the hand can be interpreted as commands for the computer. However, the data glove has a big disadvantage due to its unwieldiness: The user has to carry a lot of hardware with him- or herself which consequently makes this kind of gesture interaction unnatural.

Table 1. An overview of gesture analysis systems

Author (see references)	Hardware	Interaction	System
Oviatt (1999)	Pen	Gesture, speech	Service transaction
Rubine (1991)	Mouse	Mouse-based	GRANDMA
Buxton et al. (1985)	Mouse	Mouse-based	Editor for music note
Waibel and Yang	Graphic tablet, pen	OCR	INTERACT
Raab et al. (1979)	Magnetic field	–	Person tracking
Bolt (1980)	ROPAMS	gesture, speech	"Put-that-there"
Azuma (1993)	Ultrasonic	–	Person tracking
Fels and Hinton (1993)	VPL DataGlove	Speech synthesis	Glove-Talk
Kurtenbach and Baudel (1992)	VPL DataGlove	Presentation	HyperCard
Wu and Huang	Camera	Hand posture	Paper–Rock–Scissors
Quek (1995)	Camera	Hand gesture	Finger mouse
Kettebekov and Sharma	Front camera	Gesture, speech	iMap
Akyol et al. (2001)	Infrared camera	Gesture	Car infotainment

Electromagnetic fields (Raab et al., 1979) for gesture localization and recognition are also popular. However, the high acquisition costs, the sensitivty to noise and the short working range are the negative factors, which must be accounted for. In contrast, the video-based gesture analysis with the help of a CCD camera and the corresponding image processing technique seems more promising on account of its uncumbersome hardware. By using a camera, a set of modalities in addition to hand gestures can be integrated into the HCI, e.g., lip, gaze direction, head movement and interpretation of facial expressions. Several typical systems of gesture analysis are listed in Table 1, some of which operate simultaneously with speech. There are also other methods, which use special hardware to achieve high throughput and efficiency, e.g., the SiVit (Siemens Virtual Touchscreen) unit introduced in Maggioni (1995). SiVit is also integrated in SMARTKOM.

From the View of the Recipient

Video-based gesture analysis is advantageous according to the comparison above, thanks to the ever-improving efficiency and capacity of the computer hardware nowadays. The current major problem lies in the increasing demand for algorithms, which should be fast, robust, traceable, efficient, reliable, modularized and fault tolerant. There are mainly three different methods in the video-based gesture analysis: marker-based, hand model-based and view-based. Due to the nonconvex volume of the hand, many researchers attach markers to the hands, which are placed at certain positions of the hand. Normally, they have a special color or geometric form, with which the detection of hand and fingers becomes easier such as in case of occlusion of some part of the hand without markers. This is an indirect method and thus makes the gesture interaction unnatural. The hand model provides a full-fledged modeling of the respective finger joints and postures. Therefore, this method is able, theoretically, to analyze any gesture, given enough training data. In practice, however, the computing complexity and the lack of efficiency hinder its spread, although it can shed

Fig. 1. SMARTKOM demonstration system. **a** prototype with integrated SiVit; **b** overall view. with camera for facial expression analysis and microphones for speech analysis (**c**,**d**)

light on the solution of many practical problems. The view-based method utilizes the pixel values as its starting point, which can be either directly used as features or be converted to a suitable form through some transformations. It has a relatively small computing intensity and is therefore preferred in practice.

2 Module description

2.1 Gesture in SmartKom

Figure 1 shows the set up of an intended SMARTKOM system with an integrated SiVit unit at the top of the machine. A similar version of this system was used to collect the gesture data in the Wizard-of-Oz experiments. The SiVit unit consists of a video projector, an infrared camera and a virtual touch screen, which is not sensitive to vandalism. The system works in the following manner: The video projector projects all the graphical user interface (GUI) information onto the display, where the user can use his hand to select or search for objects. The infrared camera captures the trajectory of the users hand for the gesture analysis. Gestures are captured together with the recording of the face via video camera, and speech through a microphone array. The positions of these components are pointed out in Fig. 1.

In SMARTKOM the hand gesture is used in two categories: object manipulation and contribution to user state recognition. An introduction to the latter subject is

given separately in Streit et al. (2006), while experiments referring to user state recognition as well as object manipulation are shown in the following section.

2.2 Work Course of the Gesture Module

Apart from selecting virtual objects by gesture, the user state, which is expressed through gesture and describes the mood of the user, influences to some degree the way of gesturing: If the user gets annoyed, his gesture tends to be quick and iterating, while it becomes short and determined if the user is satisfied with the service and the information provided by the system. Both gesture and speech indicate the user state and both complement each other. Thus, we will base our experiment on a joint sample set of speech, gesture and facial expression. Since we deal with constantly changing user states, it is clear that the central point of this issue is concentrated on the dynamics of the gesture and its interpretation, instead of focusing on its segmentation from background. In SMARTKOM, all exchanged information packets are coded in an XML format.

2.3 Data flow in the gesture module

Figure 2 shows the data flow in the gesture module, which consists of two main input streams and two output streams, all coded in the XML format. Based on the two assignments of the gesture model, it reads from the data pool *generated.presentation* the geometric coordinates of the virtual objects on the GUI surface, and from *recognized.gesture* the trajectory of the gesture. After aligning the time stamps of these two packets and parsing these two XML packets, the module decides which object the user has chosen or manipulated with regard to the pointing gesture position and virtually depicted GUI objects. In this scenario, the hand gesture takes over the role of a mouse. Afterwards an object hypothesis will be generated in XML format and sent to the *gesture.analysis* pool, whose content can be evaluated further by other modules to respond to the user gesture input by calling the corresponding service such as cinema information or TV programs.

In the case of user state recognition a similar process happens, where the raw gesture data go through XML parsing, feature extraction and classification by Hidden Markov Models (HMMs). As a result, the recognized user state hypotheses will also be sent to the user state pool.

2.4 Hidden Markov Models and Gesture Analysis

HMMs are a suitable model to incorporate temporal continuity. Temporal continuity here means that a pixel of the gesture trajectory belongs to a certain category (state) for a period of time. If a pixel moves at a high speed at a given time, it is likely that this pixel will still keep moving fast at the next time step. HMMs are able to learn the observation distributions for different categories (hidden states) from the trajectory of the gesture. The training data are recorded in a system similar to the one depicted

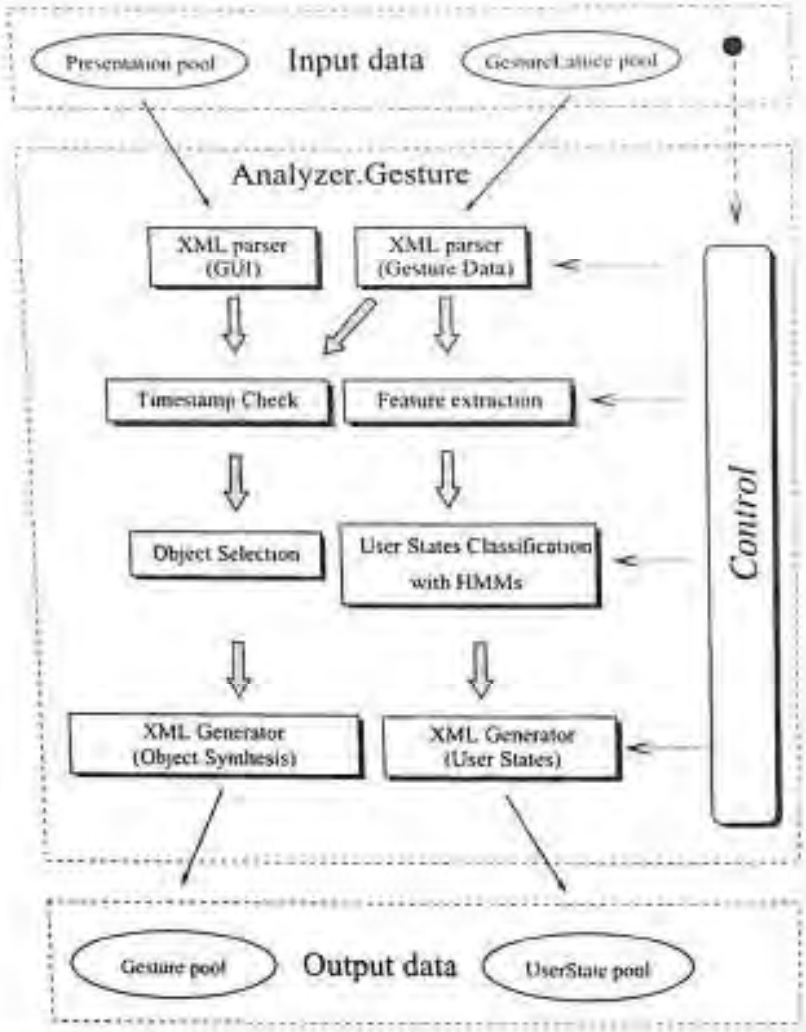


Fig. 2. Architecture of the gesture module

in Fig. 1. In this paper, each feature vector will be assigned to one of three or four hidden states of the HMM (see Sect. 2.7).

We use the standard *Baum–Welch* reestimation algorithm for the training, which is based on the expectation maximization (EM) algorithm (Rabiner and Juang, 1986), and the standard *Forward Algorithm* to solve the classification problem. A detailed description of these algorithms can be found in Rabiner and Juang (1986, 1993); an example of how to apply these algorithms can be found in Rabiner (1989). Here we use discrete HMMs because of their simplicity.

2.5 Feature Extraction

In order to incorporate the temporal continuity, we choose four features: trajectory variance, instantaneous speed, instantaneous acceleration and kinetic energy as the feature vector, which best represents the motion and the dynamics of the gesture. The

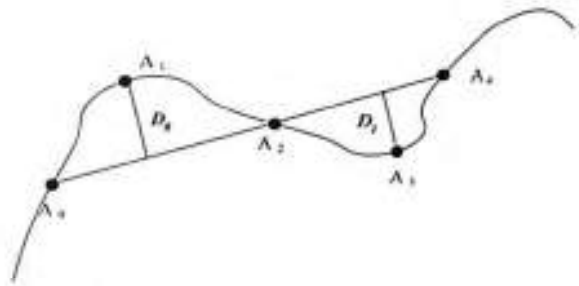


Fig. 3. Typical trajectory of a pointing gesture with sampling points $A_I, I = 0...N$ points and geometric variances $D_I, I = 0..N-1$, which show the intensity of fluctuation of the gesture

continuous two-dimensional coordinates (trajectories) and the time stamp, which are recorded by the SiVit unit, are the most important information on the dynamics of the gesture. The reason for computing the instantaneous velocity v over time is for the system to learn from the behavior of the user's gesture. That is, with simple data analysis, it would be possible to determine trends and anticipate future moves of the user. The next set of data points is the acceleration a of the gestures, which is easily computed by approximating the second derivative of the position coordinate. Kinetic energy K, which is just the square of the velocity while the mass is neglected, is also a significant factor.

In our feature set, the trajectory variance is also included. This is the geometric variation or oscillation of the gestures with respect to their moving direction. A large value of this variance can indicate that the user gesticulates hesitantly and moves his or her hand around on the display, while a determined gesture leads to a small variance. Figure 3 shows how the trajectory variance D is computed. So we have a feature vector

$$f = (v, a, K, D). \tag{1}$$

The vector D can be computed every N points along the gesture trajectory. Other possible features are, e.g., the number of pauses of a gesture, the transient time before and after a pause, the transient time of each pause relative to the beginning of the gesture, the average speed, and the average acceleration or change of moving direction. However, in this study we just consider the feature vector shown in Eq. (1).

2.6 Modification of User State Classes

As mentioned above, the goal of SMARTKOM is the combination of all three input modalities. Gesture, as one of the input channels, must define its own output to contribute to the fusion of the analysis of the three modalities. In contrast to facial and prosodic analysis, where four user states are defined, *neutral*, *angry*, *joyful* and *hesitant*, we define in gesture analysis only three user states: *determined*, *angry* and *hesitant*. The reason for making this mapping is the intuition that normally people cannot tell if the user is neutral or joyful by only observing his or her gesture. This

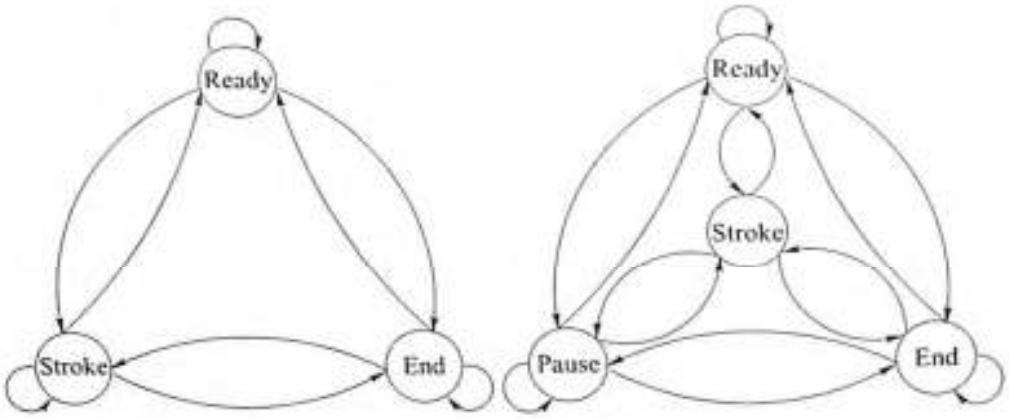


Fig. 4. Ergodic HMMs with different numbers of hidden states for gesture analysis

was confirmed by the preliminary experiments, where *neutral* and *joyful* had a high confusion. We decided thus in favor of the three states topology. The user state *determined* is given if the user knows what he wants from SMARTKOM, e.g., if he decides to zoom in a part of a city map on the GUI by pointing to it. If the user gets confused by SMARTKOM and does not know what to choose, his gesture will probably ponder around or zigzag among different objects presented on the SMARTKOM GUI. Finally, if he feels badly served by SMARTKOM or if the information presented is not correct, he can use gestures in such a way as to show a strong negative expression like a windshield wiper, which in our context corresponds to the user state *angry* in facial expression.

2.7 Choice of Different Topologies

For the HMMs, we evaluated different topologies; an HMM with three or four states gave the best results. We suppose that a gesture consists of some basic states such as *ready*, *stroke*, *end* and/or *pause*. This can be observed in the production of the gesture: The user moves her hand to a start position, then makes a gesture consisting of several strokes, probably with pauses in between, and finally ends her gesture. An alternative is to merge *pause* and *ready*. We also tried different connection schemata; the simplest one is an ergodic HMM, while a partially connected HMM better corresponds to the correct physical order of each state (Fig. 4). The conventional left–right HMM model is also an alternative that has been successfully used in speech recognition.

3 Experiments

Tables 2, 3 and 4 show the results of the gesture analysis. We can see that the user state *hesitant* is sometimes mismatched with *angry*. The reason is that some users, whose gestures are used in the training set, made similar gestures like those in *angry*

Table 2. Confusion matrix of user state recognition with gesture data using ergodic HMM (see Fig. 4, CL: classwise averaged recognition rate)

Reference user state	3 HMM states (%)			4 HMM states (%)		
	Determined	Hesitant	Angry	Determined	Hesitant	Angry
Determined	**61**	5	34	**80**	15	5
Hesitant	5	**72**	23	15	**77**	8
Angry	10	6	**84**	10	18	**72**
CL	72			76		

Table 3. Confusion matrix of user state recognition with gesture data using ergodic HMM (LOO, see Fig. 4, CL: classwise averaged recognition rate)

Reference user state	3 HMM states (%)			4 HMM states (%)		
	Determined	Hesitant	Angry	Determined	Hesitant	Angry
Determined	**62**	5	33	**75**	7	18
Hesitant	5	**74**	21	13	**74**	13
Angry	8	8	**84**	30	8	**62**
CL	73			70		

Table 4. Confusion matrix of user state recognition with gesture data using nonergodic HMM (CL: classwise averaged recognition rate)

Reference user state	3 HMM states (%)			4 HMM states (%)		
	Determined	Hesitant	Angry	Determined	Hesitant	Angry
Determined	**72**	16	12	**40**	49	11
Hesitant	32	**45**	23	2	**70**	28
Angry	60	12	**28**	2	24	**74**
CL	48			61		

states, in that the windshield wiper movement has the same zigzag only with different dynamics and speed. Probably, some persons gesticulate slowly while indicating anger, thus their corresponding gestures may have similar properties like those of a *hesitant* state.

Table 5. Confusion matrix of User state recognition using left–right HMM (CL: classwise averaged recognition rate)

Reference user state	3 HMM states (%)			4 HMM states (%)		
	Determined	Hesitant	Angry	Determined	Hesitant	Angry
Determined	**63**	4	33	**66**	6	28
Hesitant	6	**47**	47	13	**51**	36
Angry	20	4	**76**	30	4	**66**
CL	62			61		

Another reason for a wrong classification is that the training data for the user state *determined* consists of those from *joyful* and *neutral*; the latter makes the HMM for *determined* biased towards *hesitant* in Table 4 with four internal states. In general, the classification has a classwise (CL) averaged recognition rate of 72% for three internal states and 76.3% for four internal states, while the leave-one-out (LOO) test achieves 73% for three internal states and 67% for four internal states. Table 5 shows the recognition result when using a conventional left–right HMM model.

4 Conclusion

Gesture is an important communication channel in HCI, whose usage ranges from direct manipulation of object to indication of user states as shown above. These two models have been successfully integrated into the SMARTKOM demonstrator, which runs as an autonomous service agent between the user and different information sources through gesture, speech and facial expression. The user is, therefore, free to communicate with the system, similar to talking with another human. Furthermore, the gesture, speech and facial expression complement each other in a redundant way so that the demand of precise expression in each modality can be relaxed and thus extend the applicability with respect to prospective users.

References

S. Akyol, L. Libuda, and K.F. Kraiss. Multimodale Benutzung adaptiver Kfz-Bordsysteme. In: T. Jürgensohn and K.P. Timpe (eds.), *Kraftfahrzeugführung*, pp. 137–154, Berlin Heidelberg New York, 2001. Springer.

V. Attina, D. Beautemps, M.A. Cathiard, and M. Odisio. Toward an Audiovisual Synthesizer for Cued Speech: Rules for CV French Syllables. In: J.L. Schwartz, F. Berthommier, M.A. Cathiard, and D. Sodoyer (eds.), *Proc. AVSP 2003 Auditory-Visual Speech Processing*, pp. 227–232, St. Jorioz, France, September 2003. ISCA Tutorial and Research Workshop.

R. Azuma. Tracking Requirements for Augmented Reality. In: *ACM*, vol. 36, pp. 50–51, July 1993.

R. Bolt. "Put-That-There": Voice and Gesture. In: *Computer Graphics*, pp. 262–270, 1980.

W. Buxton, R. Sniderman, W. Reeves, S. Patel, and R. Baecker. An Introduction to the SSSP Digital Synthesizer. In: C. Roads and J. Strawn (eds.), *Foundations of Computer Music*, pp. 387–392, Cambridge, MA, 1985. MIT Press.

R.O. Cornett. Cued Speech. *American Annals of the Deaf*, 112:3–13, 1967.

H. Eglowstein. Reach Out and Touch Your Data. *Byte*, 7:283–290, 1990.

S. Fels and G.E. Hinton. Glove-Talk: A Neural Network Interface Between a Data-Glove and a Speech Synthesizer. In: *IEEE Transactions on Neural Networks*, vol. 4, pp. 2–8, 1993.

S. Kettebekov and R. Sharma. Multimodal Interfaces. http://www.cse.psu.edu/~rsharma/imap1.html. Cited 15 December 2003.

G. Kurtenbach and T. Baudel. Hypermarks: Issuing Commands by Drawing Marks in Hypercard. In: *ACM SIGCHI*, p. 64, Vancouver, Canada, 1992.

T. Lütticke. Gestenerkennung zur Anweisung eines mobilen Roboters. Master's thesis, Universität Karlsruhe (TH), 2000.

C. Maggioni. Gesture Computer — New Ways of Operating a Computer. In: *Proc. Int. Conf. on Automatic Face and Gesture Recognition*, pp. 166–171, 1995.

A. Marcus and J. Churchill. Sensing Human Hand Motions for Controlling Dexterous Robots. In: *The 2nd Annual Space Operations Automation and Robotics Workshop*, Dayton, OH, July 1988.

D. McNeill. *Hand and Mind: What Gestures Reveal About Thought*. University of Chicago Press, Chicago, IL, 1992.

S. Oviatt. Ten Myths of Multimodal Interaction. *Communications of the ACM*, 42 (11):74–81, 1999.

F. Quek. FingerMouse: A Freehand Pointing Interface. In: *Int. Workshop on Automatic Face- and Gesture-Recognition*, pp. 372–377, Zurich, Switzerland, June 1995.

F.H. Raab, E.B. Blood, T.O. Steiner, and H.R. Jones. Magnetic Position and Orientation Tracking System. In: *IEEE Transaction on Aerospace and Electronic Systems*, vol. 15, pp. 709–718, 1979.

L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proc. IEEE*, vol. 77, pp. 257–286, 1989.

L.R. Rabiner and B.H. Juang. An Introduction to Hidden Markov Models. *Acoustics, Speech and Signal Processin*, 3(1):4–16, 1986.

L.R. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

D. Rubine. Specifying Gestures by Example. In: *SIGGRAPH '91 Proceedings*, vol. 25, pp. 329–337, New York, 1991.

E. Sachs. Coming Soon to a CAD Lab Near You. *Byte*, 7:238–239, 1990.

M. Streit, A. Batliner, and T. Portele. Emotion Analysis and Emotion Handling Subdialogs, 2006. In this volume.

A. Waibel and J. Yang. INTERACT. http://www.is.cs.cmu.edu/js/gesture.html. Cited 15 December 2003.

Y. Wu and T.S. Huang. "Paper-Rock-Scissors". http://www.ece.northwestern.edu/~yingwu/research/HCI/hci.game.prs.html. Cited 15 December 2003.

T.G. Zimmerman and J. Lanier. A Hand Gesture Interface Device. In: *ACM SIGCHI/GI*, pp. 189–192, New York, 1987.

Part III

Multimodal Dialogue Processing

Modality Fusion

Ralf Engel and Norbert Pfleger

DFKI GmbH, Saarbrücken, Germany
{engel,pfleger}@dfki.de

Summary. In this chapter we give an general overview of the modality fusion component of SMARTKOM. Based on a selection of prominent multimodal interaction patterns, we present our solution for synchronizing the different modes. Finally, we give, on an abstract level, a summary of our approach to modality fusion.

1 Introduction

SMARTKOM provides access to a rich set of applications by giving the user the opportunity to employ various multimodal interaction patterns. This means that the user has the possibility not only to switch between the different modalities — depending on which modality is more suitable for a present situation — but also to use those modalities in an integrated and effective manner.

Starting with Bolt's "Put-that-there" system (Bolt, 1980), the history of multimodal dialogue systems has shown a wide variety of different approaches. However, nearly every system comprises a component that is capable of integrating the data streams of the different modalities into one combined representation (e.g., Cohen et al. (1997); Johnston et al. (2001); Wahlster (1991)). Basically, there are two major architectural approaches for the analysis part of multimodal dialogue systems: (i) early fusion — fusing modes already at signal or recognition level (Bregler et al., 1993; Pavlovic and Huang, 1998) — and (ii) late fusion – multimodal semantic processing (Johnston et al., 1997; Johnston and Bangalore, 2001). Even though the point within the processing pipeline and the type of processed data vary, it is generally accepted that fusing the separate data streams will improve system performance by reducing uncertainty and resolving ambiguous input hypotheses. Another interesting aspect of multimodal interaction is that users can adapt their strategies of how to use the available modalities according to their current needs. Users show, according to Oviatt et al. (1997), a strong preference for multimodal commands and requests during map tasks. For example, if a user wants to set the starting point for planning a route it, is easier and more precise to use a pointing gesture toward the desired starting point on a map and accompany it with a spoken utterance containing a deic-

tic reference (e.g., "I want to start here") than giving "exact" coordinates by speech (e.g., "I want to start about two and a half centimeters left from the church").

In SMARTKOM the integration and disambiguation of the different modalities is handled by the modality fusion component, called FUSION. This component is also responsible for initiating the resolution of those referring expressions that cannot be resolved by gestural input.

1.1 System Context

The above-mentioned tasks of the modality fusion component require communication with several other SMARTKOM modules. The main data communication channels (Fig. 1) are, on the input side, two channels bringing in data from the modality-specific analyzers and, on the output side, the channel transmitting the integrated intention hypotheses to the intention recognition module (te Vrugt and Portele, 2006).

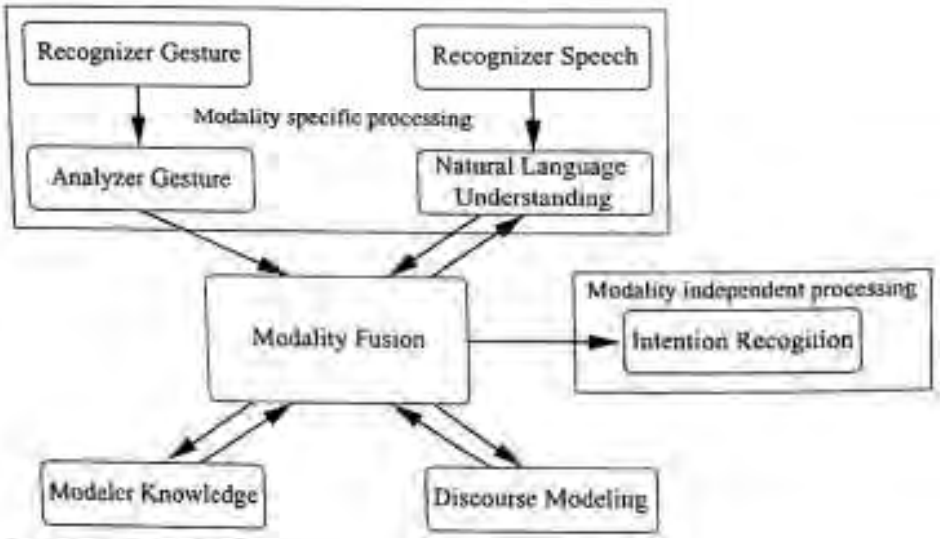


Fig. 1. Sketch of the dataflow

The gesture analysis component processes coordinates provided by a SiViT device (in case of SMARTKOM -Mobile by the pen interface) and a so-called *screen state* representing the currently displayed objects. The data FUSION receives from the gesture analysis component consist of a ranked list of probably referenced objects. The language understanding component provides a set of different hypotheses, each containing a semantic representation of a possible path within the word hypotheses graph provided by the speech recognizer (Berton et al., 2006). Important for FUSION is the fact that in SMARTKOM an "open microphone" approach is used and the gesture channel is always open, too. So, FUSION must be able to handle input at any time and provide meaningful ouput.

As we will see in the following section, FUSION needs access to information provided by two additional modules — modeler knowledge and modeler discourse.

The first one provides information about the current location of the system, which is needed for the resolution of expressions referring to the situational context, and the second component accomplishes the resolution of those referring expression that cannot be resolved without access to the discourse context.

As described in Gurevych et al. (2006), (nearly) all modules of SMARTKOM employ the same knowledge representation, and thus all meaningful data occur in the same format. For FUSION it is important that this data structure can be viewed in terms of *frames* (Minsky, 1975) or *typed feature structures* (Carpenter, 1992) since those data structures allow for recursive merging and matching of two data structures, thereby generating a fused or intergrated representation of them.

2 Multimodal Interaction Patterns: Phenomena and Processing Strategies

SMARTKOM enables the user to use speech and gestures in an integrated and efficient manner and provides wide coverage of multimodal utterances expressing the same intention. This freedom of interaction leads to a broad set of challenging interaction patterns that FUSION has to deal with. SMARTKOM incorporates three modalities (speech, gesture, and facial expressions). However, the task of FUSION is to combine only two of them, namely speech and gesture.

Besides the integration of the different modalities, FUSION has also the challenge to disambiguate the multiple hypotheses produced by the different recognizers. Therefore, FUSION computes for each integrated result a score based on a set of different parameters (Sect. 5). This score contributes to the final score for a hypothesis.

In the following section we will consider a smaller subset of prominent interaction patterns and show on an abstract level how FUSION handles them. For a detailed consideration of the underlying architecture and processing techniques, see Sect. 4.

2.1 Referring Expressions

An improtant feature of natural language is the possibility to refer to entities and events in the physical or mental environment verbally. These references occur often as deictic references, which means that the spoken referring expression (e.g., a demonstrative pronoun) is accompanied by a manual gesture selecting a particular object in the physical environment. Here, the task of FUSION is to combine the spoken and gestural expression into a single semantic representation of the communicative act (Johnston et al., 1997).

Consider, for example, the following sentence:

(1) User: *I want to go from here [↗] to there [↗]*

On an abstract level the processing strategy of FUSION for pointing gesture is dominated by the speech modality — it is assumed that each pointing gesture is accompanied by a spoken utterance (the only exception is called a direct gesture; see

Sect. 2.2).[1] The natural language understanding component (Engel, 2006) assigns to each referring expression a special marker (called *refProp*), thereby indicating that such an object is not yet fully specified and needs to be enriched with information of an appropriate referent. Then, FUSION tries to resolve those refProps by generating all possible combinations of refProps and recognized gestures that match the type of the expected referent (in this case a kind of LOCATION object).

Fig. 2. System presentation of the television program

Resolving those multimodal references as shown in example (1) is quite straightforward. However, things become more complicated when the type of the referent inferred from the intrasentential context is a substructure of the referenced screen-object. Consider, for example, a dialogue situation where the user is browsing the television program (Fig. 2) and wants to tape the currently running broadcast on a particular channel. In SMARTKOM each object presented on the screen is internally annotated with a representation of itself in terms of the SMARTKOM domain model (for more details, please see Poller and Tschernomas (2006)). For broadcasts this representation includes a slot that represents the channel on which the broadcast is running. Utterance (2a) can be treated in the same way as utterance (1). However, in order to be able to handle utterance (2b), FUSION needs access to the substructures of the referenced screenobjects. Therefore, FUSION needs at least some domain knowledge for extracting the correct information from the referenced objects.

(2a) User: *Please tape this broadcast [↗]*
(2b) User: *Please tape this channel [↗]*

[1] However, it should be noted that treating the speech modality as the primary input mode is not acceptable for all kinds of multimodal dialogue systems. For example, such systems processing speech and pen input will fail to handle many types of spontaneous multimodal construction (Oviatt, 1999).

2.1.1 Selecting Sets of Objects: Encircling Gestures

Encircling gestures are a specialization of pointing gestures. The user selects a set of objects displayed on the screen with one continuous gesture. In contrast to pointing gestures, encircling gestures require a somehow more elaborate processing to be able to handle interdependencies with the previous dialogue context correctly. Additionally, the integration of multiple objects requires world knowledge about the involved objects and processes to be able to integrate the encircled objects in a sensible way. Consider, for example, a situation where the system displays a seat reservation plan and asks the user to select the desired seats.

(3) User: *I want to reserve these three seats [encircling gesture]*

The gesture analysis (Shi et al., 2006) sends weighted lists of all objects captured by the encircling gesture to FUSION. However, the ranking reflects only salience factors detached from a specific dialogue situation. So, FUSION has to deal with two possible problem sources:

(i) The number of selected objects does not necessarily correlate with the desired number of objects provided by speech.
(ii) The most probable objects are not necessarily those the user intended to select, nor do they have to make sense when viewed with respect to the current dialogue situation.

Usually, the first problem source is easily removed when the user has directly specified the desired number of objects in an utterance accompanying the encircling gesture (like in the previous example). However, another possible course of interaction is when the user has already provided the desired number of objects in a previous turn and now performs an encircling gesture without saying anything. In that case FUSION requests the discourse modeling component (Alexandersson and Pfleger, 2006) in order to retrieve the size of the last collection object.

The second problem source becomes obvious when considering Fig. 3. It is a difficult task to encircle exactly the desired seats out of a seating plan comprising those small seats. So, it is possible that FUSION receives a set of encircled seats where the most probable ones are located in different tiers (Fig. 3). To avoid presenting the user with a somewhat odd seat selection, FUSION applies a set of plausibility tests. These tests depend on the type of selected objects and comprise, for example, a constraint stating that as many seats as possible should belong to the same tier.

2.2 Unimodal Interaction: Using only a Single Modality

Besides the integrated usage of both modalities, the user has the possibility to select one of the modalities with respect to the current task or dialogue situation, for example, a gesture referencing an object on the screen that answers a system question. On an abstract level those interaction patterns seem to be quite easily manageable for FUSION. However, viewed on a detailed level it becomes clear that we need a sophisticated synchronization mechanism in order to be able to decide whether a gesture

Fig. 3. Objects selected with an encircling gesture

event can be interpreted on its own or needs to be fused with a spoken utterance without adding any additional delays. Consider, for example, the following dialogue situation: a user is asked to select a point on a map to determine the starting point of a route plan and reacts with a pointing gesture selecting a particular object on the map. After a few seconds the user says something like: "Please zoom in".

Since the two processing chains are not synchronized, FUSION first receives the result of the gesture analysis thread and after a few seconds (plus the processing time needed by the speech recognizer and the natural language understanding component) the result of the speech thread. Interpreting the gesture on its own in this dialogue situation would lead to a hypothesis stating that the user wants to start at this location, but as the user provides additional information on the speech channel this interpretation is not feasible. However, to obtain a fast system response it is only sensible to wait for a speech event if there are any hints suggesting that the user actually uttered something. Additionally, FUSION will only pass on gesture results that already match the expectations of the action planner (Löckelt, 2006). The synchronization mechanism coupling the input modules is described in Sect. 3.

2.3 Referring to the Situational Context

The referring expression "here" requires a specical treatment since it can be used either as referring to the situational context (User: "*I want to park here*") or in combination with a pointing gesture like, for example, User: "*I want to start here,*" accompanied by a pointing gesture. If this referring expression is used in the first way FUSION needs to know the current location of the system in order to integrate it into the underspecified location object of the spoken request. This information is provided by the module called modeler knowledge (Porzel et al., 2006) and can be requested via a special data pool.

The challenge, however, is to identify when the adverb "here" occurs without any accompanying gesture whether it was used with respect to the situational context or with respect to the previous discourse context (e.g., when the user has requested during route selection dialogue information about a particular restaurant and utters in the next turn *"I want to park here."*

3 Synchronizing Different Modalities

In SMARTKOM the input channels are always open, i.e., the user can speak or perform gestures at any time. Additionally, the user can interact with the system using speech or gestures, alone and the user can also combine several gestures with one spoken utterance.[2] This implies the need for a mechanism deciding which gestures are combined with which speech input.

Since both speech and gesture messages contain time stamps indicating the start time, and the end time it is a simple task to decide which gestures and spoken utterances were performed at the same time. But often, the user does not speak and perform the gesture at the same time, instead, e.g., the user performs the gesture one second before starting speech or half a second after finishing speech. Therefore, gestures are also combined with spoken utterances if they occur at a certain time span before or after speech, further referred as t_{before} and t_{after}. Our own experiments have shown that 1.5 s for both time spans is a reasonable value.

To achieve the behavior described above, FUSION reacts depending on the type of input in a different way. The next two sections describe the respective procedure.

3.1 Messages Containing Speech Input

When a message containing speech input arrives, the module should let a time span $t_{after} - (t_{current} - t_{end\_of\_speech})$ pass waiting for new gestures until the combination of speech and gesture is started. But because of delay caused by the detection of the ending of speech and the processing time of the speech recognizer and natural language processing, $t_{current} - t_{end\_of\_speech}$ is in SMARTKOM always greater than t_{after}. Therefore, the computation of the combined speech and gesture hypotheses can start immediately. If no gestures are available for the time between $t_{start\_of\_speech} - t_{before}$ and $t_{end\_of\_speech} + t_{after}$ the speech hypotheses are passed to the intention analyzer without any changes.

3.2 Messages Containing Gesture Input

The treatment of a message containing a gesture is somewhat more complicated. As described above, it is not possible that speech input is already available for this

[2] Several spoken utterances need not be combined by FUSION since the silence detection within the audio module already decides the turn boundaries for speech.

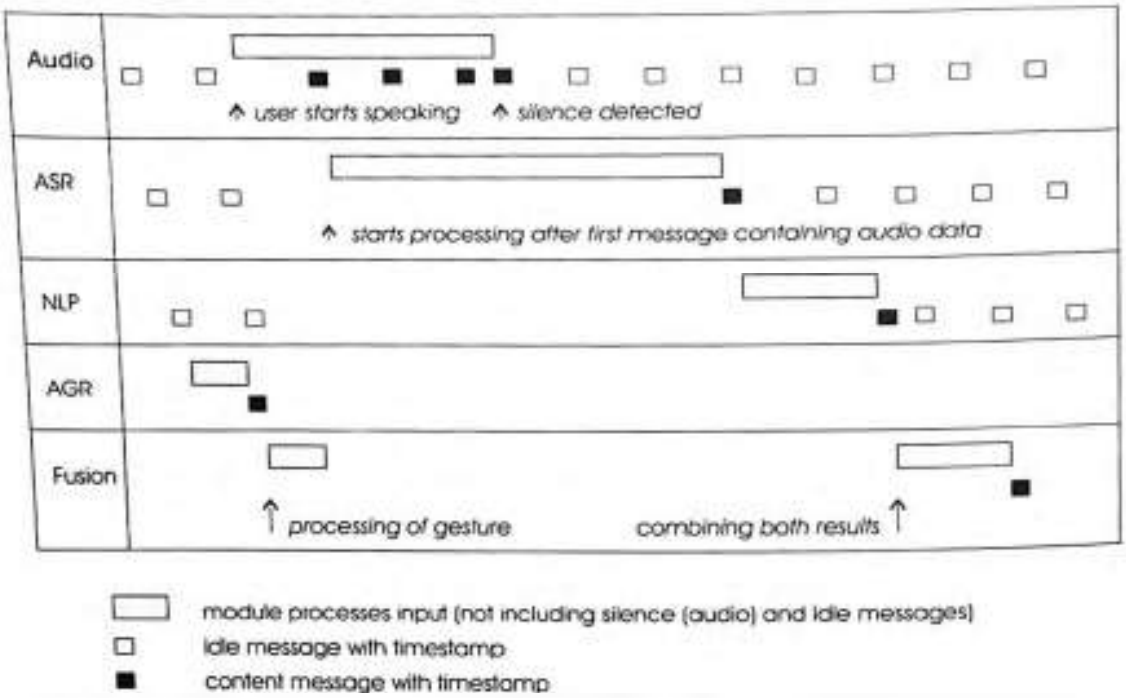


Fig. 4. Illustration of module activity and output messages

gesture due to the delays during the processing of speech.[3] Therefore, FUSION stores the gesture and waits for speech input.

Since it is possible for the user to perform gestures without any speech, it is not guaranteed that a corresponding speech message will arrive at all. A possible solution would be to wait for a timeout of $t_{before} - (t_{current} - t_{end\_of\_gesture})$ until the gesture is passed without any modifications to the intention analyzer. But as in most cases the processing time of the speech recognizer is not negligible, the timeout would be triggered falsely most of the time. At least two different modifications to the communication between the speech recognizer and FUSION are able to solve that problem:

1. The speech recognizer sends a message when it starts working. This message stops the timeout in FUSION. At the end the speech recognizer sends a result in any case, even if nothing meaningful was recognized. The drawback of this approach is that delays within the speech recognizer, e.g., the recognizer is currently updating its language model and does not process audio messages at the same time, as well as delays within the communication channels can nevertheless lead to erroneous timeouts.
2. FUSION does not use a timeout at all. Instead, when the speech recognizer does not process input, it sends idle messages to FUSION in regular intervals, e.g., ev-

[3] Otherwise it must be verified whether speech input is available for the time between $t_{begin\_of\_gesture} - t_{after}$ and $t_{end\_of\_gesture} + t_{before}$.

ery 500 ms. The idle messages contain time stamps so that they are independent from delays in the communication channels. As soon as FUSION receives an idle message it sends all gestures that satisfy $t_{end\_of\_gesture} + t_{before} < t_{idle}$ as gesture-only messages to the intention analyzer. The disadvantages of this approach are the increased communication overhead and the increased latency due to the time span between two idle messages.

In SMARTKOM the prevention of delay problems within the speech recognizer and the communication channels was considered to be more important than additional latencies and the increased communication overhead. Therefore, the second solution was chosen. An example for synchronization is shown in Fig. 4. Note that if the gesture recognition was more time consuming, such a synchronization mechanism would be required for gesture processing, too.

4 Integrating the Modality Hypotheses

A slot-filling approach was developed for the integration of speech and gesture. The term slot-filling approach means that the resulting hypotheses are produced by inserting the hypotheses of one modality into the hypotheses of the other modality. Such an approach is ideal in the context of SMARTKOM since it is powerful enough to handle all occurring tasks and requires no further knowledge bases besides the already existing SMARTKOM ontology.

4.1 Design Goals

Before the developed approach is presented in detail, the two most important issues that influenced the design should be mentioned: multimodal disambiguation and robust processing. Multimodal disambiguation means that ambiguities occur in one modality can be solved when the modalities are combined. Ambiguous input is represented as a sequence of several hypotheses in M3L. As a consequence the integration algorithm does not have to consider only the best hypothesis of each modality but also lower scored ones. Furthermore, something like a scoring function is required to detect within the generated results the ones in which disambiguation was successful.

Robust processing means that even in the case of input containing either recognition errors or errors produced by the user, FUSION still attempts to produce meaningful results. Robustness concerning speech input includes dealing with not-uttered or not-recognized referring expressions or additional referring expressions that were wrongly recognized. Robustness concerning gesture input includes imprecise pointing gestures that can be corrected using additional spoken information (e.g., the user points at a location close to a church and utters *information about this church*) and gestures that are performed unintentionally by the user. A good example for the latter case would be if the user rests too long with his finger on the same location when using SiVIT gesture recognition.

4.2 Approach

The basic idea of our developed approach is first to generate all meaningful combinations considering all hypotheses and then to select the n best results, which are passed to the intention analyzer. In SMARTKOM n is set to 3. The advantage of this approach is, besides its relatively easy realization, that during processing no decisions based on local incomplete context have to be made as all results are generated. Potentially, local decisions can lead to solutions that are not optimal if seen from a global point of view. The disadvantage of this approach is that under adverse circumstances (especially in the case of recognition errors or unintended gestures performed by the user) the generation of all meaningful combinations takes too much time. To address this issue processing is stopped after a timeout. The probability to find the correct solution is increased in the case of a timeout by optimizing the algorithms to find more probable solutions first, e.g., processing last gestures first.

4.3 Implementation

A detailed presentation of the used algorithms is out of the scope of this chapter, but a rough sketch should give an impression of the realization of the approach. Keep in mind the initial task that one utterance represented as a set of speech hypotheses and one or more gestures consisting also of several hypotheses have to be combined.

To generate all possible solutions nested loops iterate over the gestures, the speech hypotheses and the different hypotheses the gestures contain.[4] Within the innermost loop one gesture hypothesis is tried to be inserted into one speech hypothesis, which may already contain gestures inserted in previous iterations. To insert the gesture hypothesis five different strategies are applied in sequence, beginning with the strategy getting later the highest score.

1. The gesture hypothesis is compatible with an embedded instance representing a spoken referring expression and replaces it. Compatible means that the type of the gesture hypothesis is of the same type or of a subtype of the referring expression. Furthermore, all slots specified in the gesture hypothesis must have corresponding slots with a corresponding content within the referring expression. Examples covered by this strategy include *Show me information about that church* and *I want to know more about this*.
2. Same as 1, except that the compatible embedded instance is not marked as referring expression, e.g., an uttered determiner was not recognized, like in *Show me information about movie*.
3. The gesture hypothesis is inserted into the speech hypothesis by filling a suitable empty slot. Empty slots are detected with the help of the ontology. This is useful in cases where the referring expression was not recognized or uttered at all, e.g., *Show me information*.

[4] To increase performance, loops may be aborted earlier if further iterations can only produced lower scored solutions.

4. Same as 3, except that the slot is not empty, but is already filled with an incompatible instance. This increases the robustness against severe speech recognition errors.
5. The gesture is not inserted at all. This alternative is necessary if the case occurs that more gestures are recognized than can be inserted.

To find all possible solutions even if one strategy is successful, the subsequent strategies are also applied and may lead to additional solutions. It is also possible that within one strategy several solutions are generated, e.g., there is more than one matching referring expression. The additional solutions are added to the set of speech hypotheses, so usually the number of speech hypotheses increases during processing.

5 Scoring the Generated Results

The scoring function is used for two tasks. First, the function decides which generated results are sent to the intention analyzer. Because of the large number of results, only a reduced number can be sent. In SMARTKOM a maximum of three hypotheses is sent to the intention analyzer. Second, the scoring function determines the so-called fusion score that is attached to each hypothesis. In the intention analyzer this score is used in combination with other scores, e.g., the acoustic score assigned by the speech recognizer, to select the one hypothesis which is passed to the action planner.

Within the scoring function several criteria are considered, all providing their own scores. The scores are listed in order of their importance:

1. how often temporal crossing occurs between modalities, e.g., an earlier gesture replaces a later referring expression and a later gesture replaces an earlier referring expression
2. the fraction of replacements of an instance marked as referring expression
3. the fraction of replacements of an instance not marked as referring expression
4. the score of natural language processing
5. the difference in time between the gestures and their corresponding instances in the spoken utterance, i.e., the instances that the gestures replace

To determine the results that are sent to the intention analyzer, the single scores are compared pairwise in the given order. Only if two single scores are equal is the subsequent score pair considered. The fusion score sent to the intention analyzer is computed as a weighted sum of the above-mentioned single scores.

6 Conclusion

In this chapter we presented the modality fusion component of SMARTKOM. Starting with a general survey of some prominent multimodal interaction patterns, we emphasized important aspects for the integration of the different modes. Then we showed

on a detailed level how we solved the so-called synchronization problem and gave a general overview of the core integration algorithm.

The overall performance of the fusion component in SMARTKOM was verified in extensive system tests. Although the fusion component was not evaluated separately, the system tests showed that the fusion component performs very well.

Future research will aim for a more generalized application of the presented algorithms. Instead of representing the interaction patterns directly at the level of algorithms, we are amining for a knowledge-based approach where new interaction patterns can be added in a more flexible manner.

References

J. Alexandersson and N. Pfleger. Discourse Modeling, 2006. In this volume.

A. Berton, A. Kaltenmeier, U. Haiber, and O. Schreiner. Speech Recognition, 2006. In this volume.

R. Bolt. Put-That-There: Voice and Gesture at the Graphics Interface. *Computer Graphics*, 14(3):262–270, 1980.

C. Bregler, H. Hild, S. Manke, and A. Waibel. Improving Connected Letter Recognition by Lipreading. In: *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-93)*, Minneapolis, MN, 1993.

B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, UK, 1992.

P.R. Cohen, M. Johnston, D. McGee, S.L. Oviatt, J.A. Pittman, I. Smith, L. Chen, and J. Clow. QuickSet: Multimodal Interaction for Distributed Applications. In: *Proc. 5th Int. Multimedia Conference (ACM Multimedia '97)*, pp. 31–40, Seattle, WA, 1997. ACM.

R. Engel. Natural Language Understanding, 2006. In this volume.

I. Gurevych, R. Porzel, and R. Malaka. Modeling Domain Knowledge: Know-How and Know-What, 2006. In this volume.

M. Johnston and S. Bangalore. Finite-State Methods for Multimodal Parsing and Integration. In: *Finite-State Methods Workshop, ESSLLI Summer School on Logic Language and Information*, Helsinki, Finland, August 2001.

M. Johnston, S. Bangalore, and G. Vasireddy. MATCH: Multimodal Access to City Help. In: *Proc. ASRU 2001 Workshop*, Madonna di Campiglio, Italy, 2001.

M. Johnston, P.R. Cohen, D. McGee, S.L. Oviatt, J.A. Pittman, and I. Smith. Unification Based Multimodal Integration. In: *Proc. 35th ACL*, pp. 281–288, Madrid, Spain, 1997.

M. Löckelt. Plan-Based Dialogue Management for Multiple Cooperating Applications, 2006. In this volume.

M. Minsky. A Framework for Representing Knowledge. In: P. Winston (ed.), *The Psychology of Computer Vision*, pp. 211–277, New York, 1975. McGraw-Hill.

S. Oviatt. Ten Myths of Multimodal Interaction. *Communications of the ACM*, 42 (11):74–81, 1999.

S.L. Oviatt, A. DeAngeli, and K. Kuhn. Integration and Synchronization of Input Modes During Multimodal Human-Computer Interaction. In: *Proc. CHI-97*, pp. 415–422, 1997.

V. Pavlovic and T.S. Huang. Multimodal Tracking and Classification of Audio-Visual Features. In: *AAAI Workshop on Representations for Multi-Modal Human-Computer Interaction*, July 1998.

P. Poller and V. Tschernomas. Multimodal Fission and Media Design, 2006. In this volume.

R. Porzel, I. Gurevych, and R. Malaka. In Context: Integrating Domain- and Situation-Specific Knowledge, 2006. In this volume.

R.P. Shi, J. Adelhardt, A. Batliner, C. Frank, E. Nöth, V. Zeißler, and H. Niemann. The Gesture Interpretation Module, 2006. In this volume.

J. te Vrugt and T. Portele. Intention Recognition, 2006. In this volume.

W. Wahlster. User and Discourse Models for Multimodal Communication. In: J.W. Sullivan and S.W. Tyler (eds.), *Intelligent User Interfaces*, pp. 45–67, New York, 1991. ACM.

Discourse Modeling

Jan Alexandersson and Norbert Pfleger

DFKI GmbH, Saarbrücken, Germany
{janal,pfleger}@dfki.de

Summary. We provide a discription of the robust and generic discourse module that is the central repository of contextual information in SMARTKOM. We tackle discourse modeling by using a three-tiered discourse structure enriched by partitions together with a local and global focus structure. For the manipulation of the discourse structures we use unification and a default unification operation enriched with a metric mirroring the similarity of competing structures called OVERLAY. We show how a wide variety of naturally occuring multimodal phenomena, in particular, short utterances including elliptical and referring expressions, can be processed in a generic and robust way. As all other modules of the SMARTKOM backbone, DIM relies on the a domain ontology for the representation of user intentions. Finally, we show that our approach is robust against phenomena caused by imperfect recognition and analysis of user actions.

1 Introduction

This chapter provides a comprehensive description of the central repository for contextual information in SMARTKOM: the *discourse modeling module* (DIM [1]). For man–machine dialogue systems, the functionality of DIM is often viewed and implemented as a part of the dialogue manager (Jönsson, 1993, 1997; Larsson, 2002). In SMARTKOM, DIM manages contextual information by recording or exchanging information with almost every backbone module in such a way that the analysis modules and the generation modules can interpret or generate with respect to the discourse state. Here, we will focus on analysis support but, as we will indicate, our approach has been designed in such a way that the generation of system reactions can be supported too.

In SMARTKOM, we have further developed the approach for discourse modeling taken in the speech-to-speech translation system Verbmobil (Alexandersson, 2003). In Verbmobil, there is no dialogue manager in the traditional sense; since

[1] Most of the material in this chapter is found in Alexandersson and Becker (2001, 2003); Pfleger (2002); Pfleger et al. (2003a,b, 2002), see also Alexandersson et al. (2006)

the Verbmobil system does not actively participate in the dialogue, dialogue management boils down to discourse processing whose task it is to passively track the dialogue. In SMARTKOM, we have added a three-tiered context representation based on the ideas developed by Luperfoy (Luperfoy, 1992) and some ideas from Salmon-Alt and Pfleger (Salmon-Alt, 2000; Pfleger, 2002). In SMARTKOM, DIM is closely connected to the action planner (Löckelt, 2006) and utilizes its predictions for the context-dependent interpretation for, in particular, short answers and partial utterances (Sect. 4).

1.1 The Challenge

DIM is the main repository for dynamic contextual information in SMARTKOM. During the course of a dialogue, each new user contribution has to be interpreted in the light of the previous discourse context. Compatible information needs to be incorporated into the representation of the new contribution through a process we call *enrichment*. DIM not only has to maintain a representation of the previous discourse context but also has to provide appropriate functionality for enrichment. Furthermore, since both the recognizers and the analysis components of the different modalities produce multiple hypotheses, DIM has to contribute to the selection of the most probable one on the basis of the previous discourse context — a process we call *validation*. The following example highlights the need for the enrichment and validation of intention hypotheses.[2]

U1: *What's on at the movies tonight?*

S1: (Displays a list of movies) *Here [↗][3] you can see tonight's cinema program.*

U2: *And what's on TV?*

S2: (Displays a list of broadcasts) *Here [↗] are tonight's broadcasts.*

U3: *Is there a movie with Arnold Schwarzenegger?*

Turn U2 is a good example of where we need access to the previous discourse context to obtain the correct interpretation—the user wants to see tonight's television program and not the one that is currently running. Turn U3 exemplifies the need for a validation of intention hypotheses against the discourse context as the language understanding module will produce at least two hypotheses for this user contribution: one stating that the user wants to see a movie with Arnold Schwarzenegger at the movies and a second stating that the user wants to see a such a movie on TV. By considering the discourse context it becomes clear that the second hypothesis most probably reflects the intended meaning.

Besides enrichment and validation, DIM also has the task to integrate partial utterances into their previous discourse. Consider, for example, an elliptical expression, as in the following:

[2] The validation process is described in Alexandersson et al. (2006)

[3] [↗] stands for a pointing gesture.

U7: *What's on TV tonight?*

S7: *Here are tonight's broadcasts.*

U8: *And tomorrow?*

S9: *Here are tomorrow's broadcasts.*

Here, the *correct* interpretation is a request for showing the TV program for tomorrow night. In this case, DIM has to integrate the time expression of turn U8 into the representation of the previous turn, U7 thereby recovering the general request ("show me the TV program") and also the time interval 8:00 PM – 11:59 PM.

Another challenging task of DIM is the resolution of referring expressions like , e.g., anaphora or cross-modal references. Consider the continuation of the discourse above:

U10: *Tape the first one!*

To be able to resolve such a referring expression, DIM not only has to maintain a representation of the previous discourse context, but it also has to maintain a representation of the objects previously displayed on the screen together with some representation of their conceptual layout (e.g., their ordering).

To conclude, we pose the following requirements on discourse processing in SMARTKOM:

- We need a general and robust method for the enrichment and validation of user hypotheses.
- The context model must differentiate between modality-specific information and modality-unspecific information of a concept introduced into the discourse.
- The context model must reflect not only the temporal structure of the dialogue but also the topical structure.
- We need a robust and flexible method for resolving different kinds of referring expressions.

1.2 Our Approach

We use the following approach for these tasks:

Representation. Our approach for representing the discourse state is a compilation of three items:

- A unified modality-independent generic representation based on the three-tiered context representation presented in Luperfoy (1992). The three tiers are the *modality* layer, the *discourse object* layer and the *domain object* layer.
- The context structuring ideas of Salmon-Alt (2000). In particular, we deploy compositional information of collections.
- A simple and robust thematic-based generic focus mechanism, e.g., Carter (2000). Our focus mechanism contains a *local* and *global* structure organized around different *focus spaces* and accessible discourse objects of the focus space, respectively.

Ontology. The common ontology-based representation used in the SMARTKOM backbone (Gurevych et al., 2006). In DIM, the ontology is viewed and represented as typed feature structures (TFSs) which allows for the use of unification-based manipulation operations (see below).

Manipulation. Two fundamental operations for the manipulation of the ontology-based representation: unification and overlay (Alexandersson et al., 2006; Alexandersson and Becker, 2003).

The approach chosen enables resolution of different referential expression, e.g., anaphora, cross-modal referring expressions and a wide range of elliptical expression (see Sect. 1.1 for examples). Still, the deployed functionality is efficient and does not slow down the system. We have carefully designed the algorithms in such a way that we only use algorithms linear to the size of its arguments (unification and overlay). Additionally, the search space for resolving referential and elliptical expressions is drastically reduced by the topic structure.

The chapter is organized as follows: Sect. 1.1 provides a more thorough presentation of the challenges we are facing. Section 3 describes how we represent the discourse. In Sect. 4 the algorithms used are described. The description of overlay, however, is found in Alexandersson et al. (2006) Before we conclude the chapter in Sect. 6 we discuss our experiences gained in SMARTKOM in Sect. 5.

2 DIM in Context

As already indicated in the previous section, DIM needs to interact with many modules of the dialogue backbone. Figure 1 depicts the main communication from the viewpoint of DIM. In addition to the main data flow (solid lines), there are several auxiliary interfaces (dashed lines), which are used either to store contextual information being produced by action planner and output modules or to provide contextual information to the analysis modules.

The first module in the processing chain that interacts with DIM is the fusion component (FUSION, see Engel and Pfleger (2006)). In cases where FUSION encounters a referring expression that could not be resolved using gestural input, it requests DIM to resolve those referring expressions (the actual reference resolution is described in Sect. 4.2). In cases where DIM was able to determine the referent for those referring expressions, it sends the resolved object back to modality fusion, which in turn integrates it into the corresponding intention hypothesis. The next module that communicates with DIM is the intention recognition module sending the set of analyzed hypotheses to DIM. DIM in turn interprets them with respect to the current discourse context (the process of enrichment and validation; see Sect. 4.1). However, before DIM sends the enriched hypotheses back to the intention recognition modules, it requests the situation modeler to enhance the hypotheses with situational information (Porzel et al., 2006).

The analysis of a user contribution is finished when the intention recognition component eventually has selected an intention hypothesis. This intention hypothesis

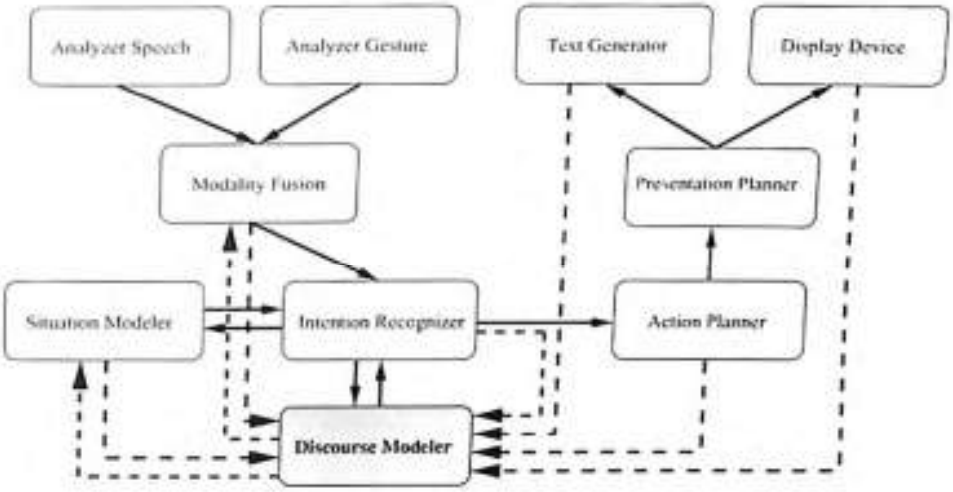


Fig. 1. The interfaces of DIM. *solid lines* depict the main data flow; *dashed lines* indicate auxiliary interfaces

is incorporated into the discourse history representing a user turn. The system turn is constructed by the expectations of the action planner (describing the anticipated user reactions to the next system turn), the screen state published by the display device (describing the objects presented on the screen) and concepts presented by the generation module (describing the system utterance).

2.1 Some Terminology

Before we continue with describing our approach to discourse processing, we have to introduce some terminology. The top level of the SMARTKOM ontology contains several *application objects*. The number of application objects is determined by the set of applications modeled by the action plans used in the system (Löckelt, 2006). Application objects describe events, e.g., a movie ticket reservation request, and can contain *subobjects* that are recursively composed of other subobjects. An application object describing a movie ticket reservation event includes, for example, subobjects describing the movie theater, the show time and movie information. Additionally, the type hierarchy is used to express relations between application objects. Subobjects that can appear in different application objects are introduced as early as possible in the type hierarchy. Additionally, some subobjects are called *slots*. Slots are characterized by their paths from the top level of an application object and represent objects in the real world that could be focused on during the conversation. Moreover, slots are predicted by the action planner via its expectations; a feature we rely on especially when it comes to the interpretation of short utterances (see below).

3 Discourse Representation

In this section we outline the concepts and realization of the structure of our discourse memory. Next, our extension to the three-tiered context representation of Luperfoy (1992) is described. We close this section by a brief description of our focus structure and the use of initiative response units.

3.1 A Three-Tiered Context Representation

Our approach to discourse modeling is based on a generalization of Luperfoy (1992) together with some ideas from Salmon-Alt (2000) and the Verbmobil dialogue module (Alexandersson, 2003; Wahlster, 2000). Following the ideas of Luperfoy (1992), we use a three-tiered context representation where we have extended her linguistic layer to a *modality* layer (Fig. 2). Additionally, we have adopted some ideas from Salmon-Alt (2000) by incorporating directly perceived objects and compositional information of collections. The basic discourse operations described in Alexandersson (2003) and Wahlster (2000) have been further developed (Alexandersson and Becker, 2003).

The advantage of our approach to discourse representation lies in the *unified* representation of discourse objects introduced by the different modalities. As we show in the next sections, this not only supports the resolution of elliptical expressions but allows for cross-modal reference resolution. The context representation of the discourse modeler consists of three levels: the discourse object layer, the modality layer and the domain object layer.

3.1.1 Discourse Object Layer

The discourse object layer is the central layer of the discourse representation. This layer comprises the concepts introduced into the discourse. An object on this layer is called a discourse object (DO, see Luperfoy (1992)). For each concept introduced during discourse there exists only one DO independent of how many modality objects (MOs, see below) mention this concept. Each DO is hence unique. A DO is created every time a concept is newly introduced into the discourse, independent of which modality was used. This holds not only for spoken concepts but also for visually perceivable concepts, e.g., graphically presented objects (Salmon-Alt, 2000).

DOs provide access to three classes of information:

1. modality-specific information via MOs
2. domain information by accessing substructures of the domain object layer
3. compositional information

Compositional information is available when a DO represents a collection or composition of objects. This information is provided by *partitions* (Salmon-Alt, 2000).

We use partitions to represent collections of objects. Collections are either based on perceptive information, e.g., the list of broadcasts visible on the screen, or discourse information stemming from grouping discourse objects. The elements of a

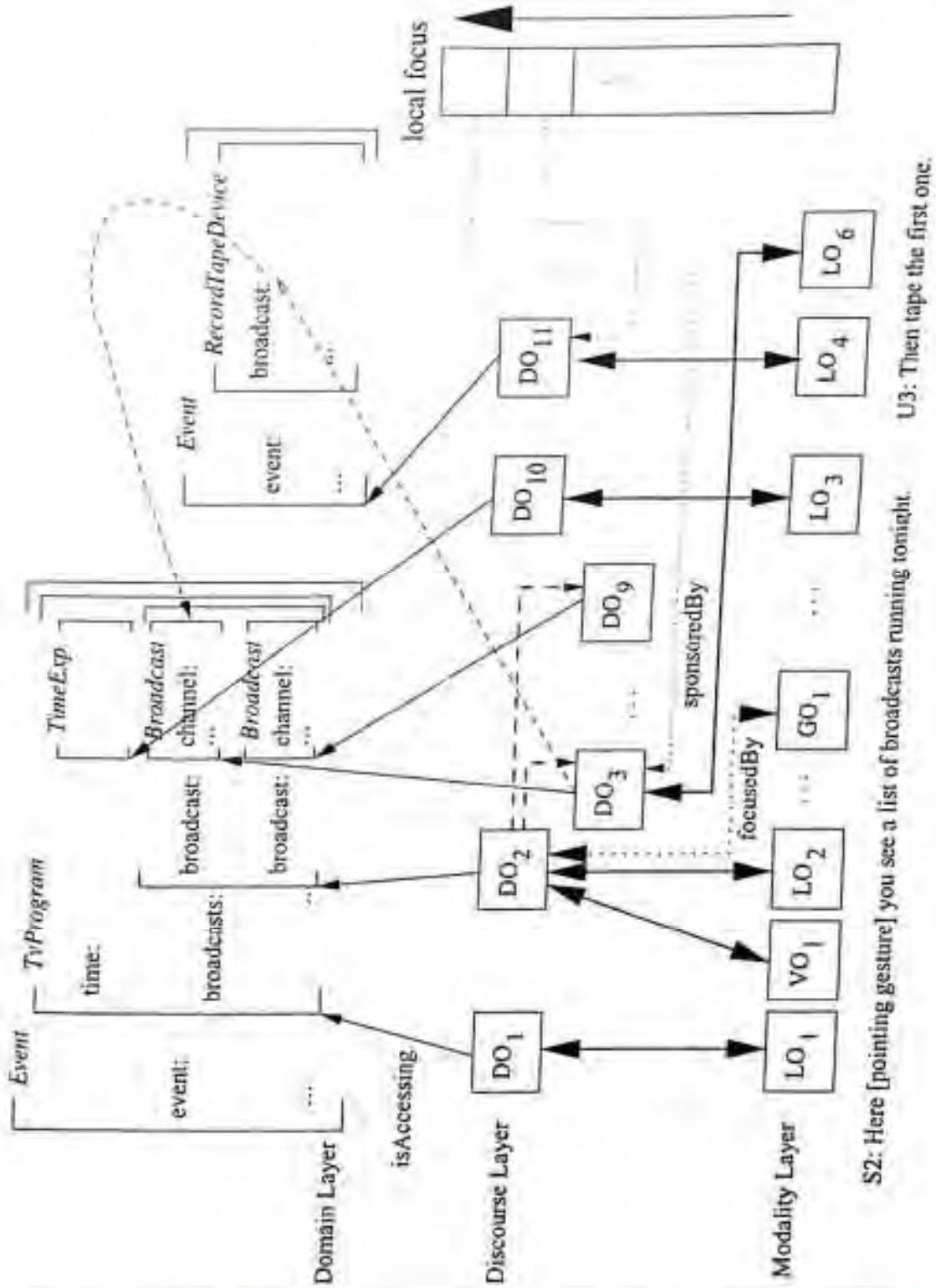


Fig. 2. Multimodal context representation. *Dashed arrow* indicates that the value of the broadcast in the (new) structure (to the *right*) is shared with that of the old one (to the *left*)

partition are distinguishable from one another by at least one *differentiation criterion*. Yet, one element alone of a partition may be in focus, according to gestural or linguistic salience. Figure 3 depicts a sample configuration of a discourse object (DO2) with a partition.

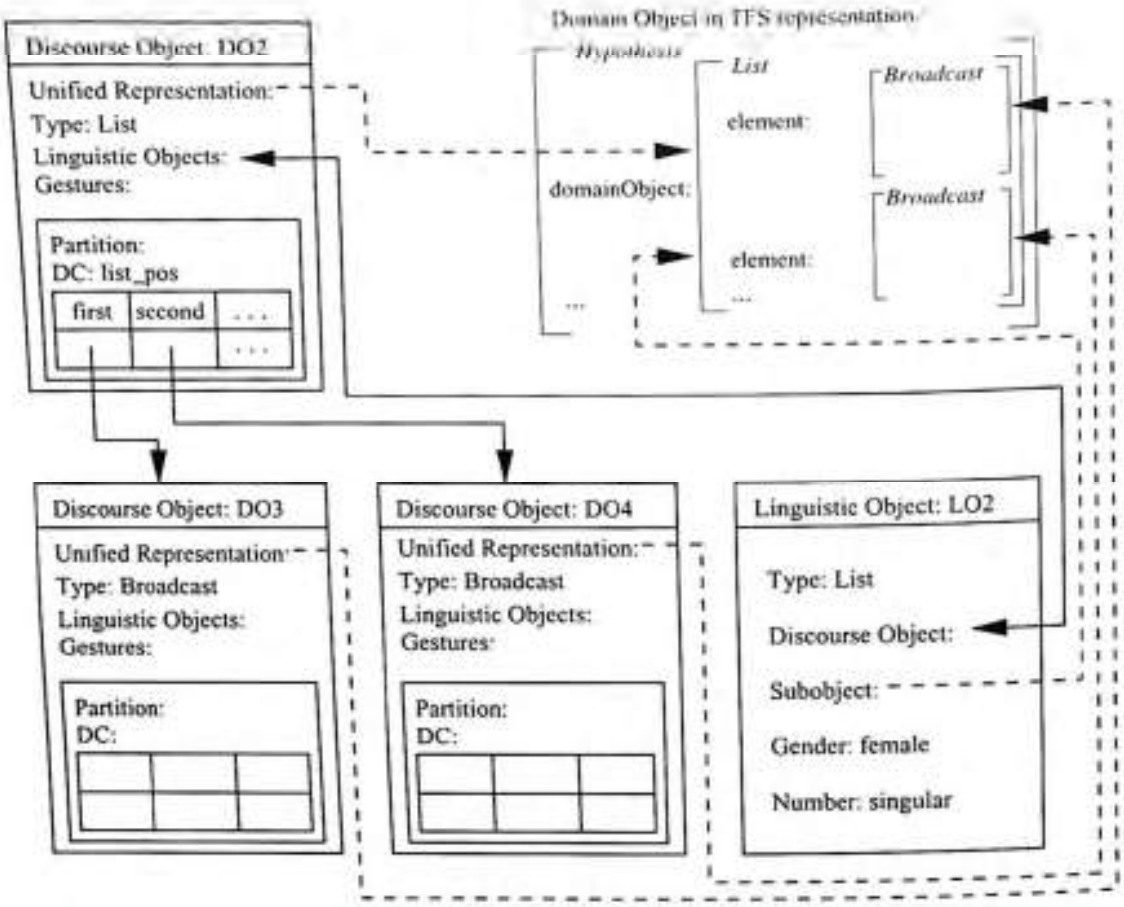


Fig. 3. Discourse objects

3.1.2 Modality Layer

Modality objects (MOs) are introduced to the modality layer every time a discourse object is mentioned in discourse (either by the user or by the system). Every MO is linked to its corresponding DO and encapsulates information about the surface realization of this particular mention of that DO. With respect to the three modalities that are stored by DIM, we differentiate three types of modality objects:

1. linguistic objects (LOs)
2. visual objects (VOs)

3. gesture objects (GOs)

Each type of modality object provides access to modality-specific information. A LO, for example, provides information about linguistic features (like number or gender) that were used during realization, whereas a VO provides information about the position the corresponding DO had on the screen.

Each MO is linked to a corresponding DO and shares its information about its concrete realization with the discourse object (Fig. 3). Note that a MO provides information about its original position within the event structure, i.e., its path in the corresponding application object in which it is embedded.

3.1.3 Domain Object Layer

The domain object layer encapsulates the instances of the domain model and provides access to the semantic information of objects, processes and actions. Initially, the semantic information of a DO is defined by a subobject (possibly embedded in an application object) representing the object, process or action it corresponds to. This information is only accessed by the DO. However, the semantic information of a DO might be extended as soon as the object is accessed again. Consider, for example, DO42 initially representing a movie with the title *The Matrix* (created by the user request *"When will the movie* The Matrix *be shown on television?"*). The system will respond by presenting a list of broadcasts of the movie *The Matrix* (accompanied with information specific to the different broadcasts, like time, database key, channel, etc). Now, if the user selects one of the movies — *"tape this [/] one"* — the initial information of DO42 will be extended with this additional information.

3.2 Focus and Attentional State

We differentiate between a *global* focus structure and a *local* focus structure restricting access to objects stored in the discourse model. The former represents the topical structure of discourse and resembles a list of focused items—*focus spaces*. The items there are ordered by salience. In SMARTKOM, the global focus is imposed by the action planner providing a flat structure of discourse in terms of discourse topics. A focus space covers all turns belonging to the same topic and enables access to a corresponding local focus structure (see also Carter (2000)).

A local focus structure provides and restricts access to all discourse objects that are antecedent candidates for later reference. Also on this level, the content of the structure, i.e., discourse objects, are ordered by salience. For each user or system turn, the local focus structure for this topic is extended with all presented concepts (see also Fig. 2).

4 Discourse Processing

We now turn to the task of processing, e.g., referring expressions. In this section we describe how the structures presented in the last section are utilized for interpretation

and validation based, on the one hand, on the discourse state and, on the other hand, on the state of the action planner (Löckelt, 2006).

4.1 Context-Dependent Interpretation

Our main operations for the manipulation on instances of our domain model is unification and a default unification operation, which together with a scoring functionality we call OVERLAY (Alexandersson, 2003; Alexandersson and Becker, 2003; Pfleger et al., 2003a; Alexandersson et al., 2006). The starting point for the development of the latter operation was twofold: First we viewed our domain model as typed feature structures and employed closed world reasoning on their instances. Second, we saw that adding information to the discourse state could be done using unification, at least as long as the new information was consistent with the context. This is, however, not always the case, as the user often changes his or her mind and specifies competing information, which causes standard unification to fail. There was thus a need for a nonmonotonic operation capable of overwriting parts of the old structure with the new information *and* at the same time keeping the information still consistent with the new information. The solution is default unification (Alexandersson and Becker, 2003; Carpenter, 1993; Grover et al., 1994), which has proven to be a powerful and elegant tool for doing exactly this: overwriting old, contextual information—*background*—with new information—*covering*—thereby keeping as much consistent information as possible.

U4: *What is on TV tonight?*

S4: [Displays a list of broadcasts.] *Here* [/] *you see a list of the broadcasts running tonight.*

U5: *What is running on MTV?*

S5: (Displays a list of broadcasts for MTV tonight.) *Here* [/] *you see a list of the broadcasts running tonight on MTV.*

U6: *And CNN?*

S6: ...

Fig. 4. Dialogue excerpt 2

We distinguish between full or partial utterances. An example of the former is a complete description of a user action, e.g., U4 in Fig. 4, whereas partial utterances often but not necessarily are elliptical responses to a system request. We handle these cases differently, as described below.

4.1.1 Full Utterances

For task-oriented dialogues, there are many situations where information can and should be inherited from the discourse history, as shown in the dialogue excerpt in

Fig. 4. Due to spatial restrictions on the screen it may be impossible for the system to display every broadcast for all channels, e.g., in S4. SMARTKOM therefore chooses some broadcasts playing on some channels. Clearly, the intention in U5 is to ask for the program on MTV *tonight*, thus requiring the system to inherit the time expression from U4. Full utterances are processed by traversing the global focus structure and picking the focused application object in each focus space (if any). In Fig. 4, default unifying U5 (covering) with U4 (background) results in *What is running on TV on channel MTV tonight*.

4.1.2 Partial Utterances

For the interpretation of partial utterances (henceforth partials) we gave a detailed description in Löckelt et al. (2002). For elliptical utterances (U6 in Fig. 4), the general idea is to extend the partial to an application object of the same type as the nearest focused application object—a process referred to as bridging—and then use this application object as covering and the focused application object as background.

There is one more challenge we have to face: resolving referring expressions. Given resolved referring expressions *and* the correct bridging, we can use the basic processing technique as described in Sect. 4.1.1. Next, we concentrate on the latter, whereas the former task is described in Sect. 4.3.

4.2 Resolving Referring Expressions

There have been many proposals in the literature for finding antecedents for referring expressions, e.g., Grosz et al. (1995). These approaches typically advocate a search over lists containing the potential antecedents, where information like number, gender agreement, etc., are utilized to narrow down the possible candidates. Our approach to reference resolution is a bit different. Additionally, in a multimodal scenario, the modality fusion first has to check for accompanying pointing gestures before accessing the discourse memory. In case of a missing gesture, DIM receives a request from modality fusion containing a subobject as specific as possible inferred by the intra-sentential context. This goes together, if possible, with the linguistic features and partition information. DIM searches the local focus structure and returns the first object that complies with the linguistic constraints in the respective LO and unifies with the object of the corresponding DO. We resolve three different referring expressions in the follwoing sections.

4.2.1 Total Referring Expressions

By a *total referring expression* (TRE) we mean that a referring expression corefers with the object denoting its referent (LuperFoy, 1991). Those referring expressions are resolved through the currently active local focus. The first discourse object that satisfies the type restriction and that was mentioned by a linguistic object with the same linguistic features is taken to be the intended referent. In this case, there is a

linguistic sponsorship relation established between the referring expression and its referent.

If no linguistic sponsorship relation can be established, DIM tries to establish a discourse sponsorship relation. This condition is characterized by a mismatch between the linguistic features, but the objects themselves are compatible (i.e., in our case unifiable). However, if both conditions are not fulfilled, the focused discourse objects (but only the one most focused on) of the other global focus spaces are tested, thereby searching for a discourse object that allows for a linguistic sponsorship relation.

4.2.2 Partial Referring Expressions

By a *partial referring expression* (PRE), the user refers to a part of a discourse object, e.g., selecting a movie from a list of movies. In this case, the processing is the same as for total referring expressions with one exception: the local focus is scanned for a compositional discourse object that satisfies (i) the differentiation criterion specified in the partition feature of the request (i.e., *"the first"*), and (ii) has a discourse object in its value feature that is compatible (i.e., in our case unifiable) with the requested discourse object (see Fig. 3). The first such discourse object is returned.

4.2.3 Discourse Deictic Expressions

A *discourse deictic expressions* (DDE) occurs when the user refers to a complete system action, for example, *"that is good,"* where "that" refers to the entire last system action. The complete functionality for a sophisticated treatment of DDEs is not included in the SMARTKOM system. However, our approach to discourse processing allows for resolving these expressions, too (Pfleger, 2002). The DDE case is processed as follows: If the type of the referring expression cannot be identified, then the focused discourse object of the currently active local focus is tested as to whether it shares the linguistic features with the request. If it does, that discourse object is taken to be the intended referent, otherwise the referring expression is interpreted as being a discourse deictic one, in which case the discourse object representing the last system turn is returned.

4.3 Partial Utterances Revisited

We return to the interpretation of partials again. After being processed by FUSION, an intention is now guaranteed to be either an application object or, as we will focus on now, a subobject representing a partial. Processing partials consists of two steps (Löckelt et al., 2002):

1. Find an anchor for the partial. If present, the anchor is searched for and possibly found under (i) the list of expected slots, (ii) in the local focus stack or, finally, (iii) in the list of possible slots.

2. Compute the bridge by using the *path* in either expected or possible slots, or in the MOs in the local focus stack.

Conceptually, the process of the interpretation of partials is summarized in Table 1. Depending on who is having the initiative, conceptually different processing is performed. Although we *always* use overlay for the combination of new and old information, it suffices in some cases to use unification.

Table 1. Possible combinations of partial utterances and different expectations given user or system initiative

	Procesing	
	User initiative	System initiative
Expected slot	NA	Expected Unify Plan continues
Possible slot	Plausible Unify Replanning	Plausible Unify Replanning
Filled slot	Plausible Overlay Replanning	Plausible Overlay Replanning
Other	implausible recover strategies	

There are, however, some exceptions to this general scheme, of which we provide two examples in the following.

4.3.1 User Provides Too Much Information

If the system has the initiative, thus requesting for information, the user might provide more, still compatible information than asked for. A good example is the case where, during seat reservation for a performance, the system asks the user to specify where she would like to sit. The expected slot is in this case pointing to a seat. In our domain model the seat part of a set construction contains not just one seat but, e.g., a set of seats and cardinality. If the user contribution specifies something like "*Two seats here [↖]*"[4] then the user contribution will not fit the expectation.

A representation of the reservation is schematically depicted in Fig. 5. The system asks for a specification of a seat, i.e., a pice of information at the end of the path *f1:f2:...:seats:seat* that is a partial of type SEAT. The answer *contains* the expectation which, however, is embedded in an object consistent with the expectation. The correct processing of such an answer is to walk along the expectation until the answer is found. If this happens before the end of the expected path, the rest of the path (*seat*) has to be part of the subobject.

[4] ... where [↖] stands for an encircling gesture

$$\begin{bmatrix} & & & & \text{CINEMARESERVATION} \\ & & & & \text{T1} \\ & & & & \text{T2} \\ f1 & f2\ldots & seats & \begin{bmatrix} & \text{SEATS} \\ & \textit{Cardinality } 2 \\ seat & \begin{bmatrix} \text{SEAT} \\ \ldots \end{bmatrix} \\ seat & \begin{bmatrix} \text{SEAT} \\ \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 5. An application object representing the reservation of two seats

4.3.2 Manipulations of Sets

For the correct processing of the example above, we had to extend OVERLAY with operations on sets, like union. Again consider a seat reservation scenario where we have the case where the user is not satisfied with the reservation and replies to the system request "*Is the reservation OK?*" with a modification containing manipulations of, in this case, a set of seats by uttering "*two additional seats here [↖]*". The processing consists of two steps: (i) SPIN marks the seats in the intention hypothesis with a set modification flag. (ii) FUSION requests the focused set of seats from DIM and computes the allowed set of, in this case, additional seats. (iii) The intention is now containing two seats (pointed at with the set modification flag), which are then processed by DIM in the same way as described above: the path from the root of the focused application object corresponding to the set of seats in the discourse memory. These are found in the local focus stack. After computing the covering, OVERLAY is performed where the sets of seats in the background and covering are unified with union.

5 Discussion

In the previous section we presented our approach to discourse modeling for a multimodal dialogue system. We showed how we achieve an enrichment and validation of user intention hypotheses with respect to their previous dialogue context using a single operation called *overlay*. It is important for the discussion here that we use overlay without any control mechanism to accumulate information stemming from previous dialogue states into a current user intention hypothesis. In many dialogue situations this accumulation of discourse objects is doing what we expect. However, by considering the example below, it becomes clear that there are situations where old information should be—or must be—ignored.

We showed that a least some of these overaccumulations can be avoided by inhibiting accumulation of previous information. In SMARTKOM, the speech interpretation component blocks parts of an event object if there are any explicit clues in the spoken utterance that suggest to do so (Engel, 2006).

The following dialogue excerpt is an example where the presented functionality of DIM will accumulate too much contextual information:

U1: What is running on TV tonight? (Context C1)

U2: ... and what is running on ARD? (Context C2)

U3: Show me information about this broadcast [↗]. (Context C3)

U4a: And is there something with Schwarzenegger?

U4b: Show me broadcasts with Schwarzenegger.

When considering U4a and U4b the question is which context to use. Possible contexts are C2 (channel: "ARD," evening) or context C1 (evening). An intuitive answer for U4a would be C2, as the syntactical structure of C2 and U4a both start with a conjunction. Utterance U4b, however, is a rather general request and should not be enriched with too much—if any—contextual information.

In general, enriching discourse contributions with respect to their previous discourse context is a complex venture. The approach used for SMARTKOM has—besides its robustness and generality—one important disadvantage caused by the fact that we deploy a straightforward use of overlay. For most dialogue situations this is feasable. As shown above, our current approach might lead to *overaccumulation*. Hence, there is a need for "forgetting" or "discarding" contextual information.

One can argue that this functionality should be placed in the action planner: in case the interpretation provided by DIM does not result in—in the example above—any broadcasts, the action planner could relax some of the contraints. This relaxation could, however, be provided by the discourse module by proving an ordered list of interpretations instead of a single contextual interpretation. In the example above, one could imagine the interpretations {OVERLAY(U4a,C2), OVERLAY(U4a,C1), OVERLAY(U4a,C3)...}. Now if the action planner fails to retrieve any broadcast using the first interpretation, the next—lower scored one—can be tested. This approach is only defeasable if the database is fast enough.

6 Conclusion

We provided a detailed description of the *discourse modeling module* of SMARTKOM. The resulting module is based on ideas and experience collected in several projects including the speech-to-speech translation system Verbmobil (Wahlster, 2000; Alexandersson, 2003), a typed front end to the ARIS database—NaRaTo—, and, finally, the multimodal dialogue system SMARTKOM. Hence our approach has proved flexible and powerful enough to meet the requirements of a wide range of dialogue systems. Current and future activities include the following topics:

Discourse segment recognition. As we have shown, there is a need for more detailed modeling of the topical focus structure. Several approaches based on a hierarchical structure have been suggested in the literature. The danger with such approaches is, however, that the intended referents might be shadowed and thus not found. Still, in many situations they provide a more precise access.

Granularity. In some cases OVERLAY should overwrite complete substructures of the background rather than include all information from the background. The keys for when this should be done are often found in the surface of the user actions.

Overlay. For setlike structures, i.e., seats in a movie-theater, it is unclear what operations should be used (see also Alexandersson et al. (2006)). The current version of DIM provides correct treatment for some examples. Ongoing work investigates how and when different set operations (i.e., adjoin, set difference, etc.) are used for the combination of new and contextual information.

Evaluation. The success of the overall system shows that our approach to discourse modeling is valid. Still, neither the overall system nor DIM has recieved a proper evaluation. Although there have been some attempts to create evaluation standards for dialogue systems, there exists—to our knowledge—no standard for how an intrinsic evaluation of DIM can be performed.

Forgetting. For longer dialogues (more than half an hour of discourse), the discourse memory runs out of memory. It is—even for most humans—necessary to forget information. How and when this can be done is still an open and challenging issue we will have to address in the future.

References

J. Alexandersson. *Hybrid Discourse Modelling and Summarization for a Speech-to-Speech Translation System*. PhD thesis, Universität des Saarlandes, 2003.

J. Alexandersson and T. Becker. Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System. In: *Workshop Notes of the IJCAI-01 Workshop on "Knowledge and Reasoning in Practical Dialogue Systems"*, Seattle, WA, August 2001.

J. Alexandersson and T. Becker. The Formal Foundations Underlying Overlay. In: *Proc. 5th Int. Workshop on Computational Semantics (IWCS-5)*, pp. 22–36, Tilburg, The Netherlands, February 2003.

J. Alexandersson, T. Becker, and N. Pfleger. Overlay: The Basic Operation for Discourse Processing, 2006. In this volume.

B. Carpenter. Skeptical and Credulous Default Unification With Applications to Templates and Inheritance. In: T. Briscoe, V. de Paiva, and A. Copestake (eds.), *Inheritance, Defaults, and the Lexicon*, pp. 13–37, Cambridge, CA, 1993. Cambridge University Press.

D. Carter. Discourse Focus Tracking. In: H. Bunt and W. Black (eds.), *Abduction, Belief and Context in Dialogue*, vol. 1 of *Studies in Computational Pragmatics*, pp. 241–289, Amsterdam, The Netherlands, 2000. John Benjamins.

R. Engel. Natural Language Understanding, 2006. In this volume.

R. Engel and N. Pfleger. Modality Fusion, 2006. In this volume.

B.J. Grosz, A.K. Joshi, and S. Weinstein. Centering: A Framework for Modelling the Local Coherence of Discourse. Technical Report IRCS-95-01, The Institute For Research In Cognitive Science (IRCS), Pennsylvania, PA, 1995.

C. Grover, C. Brew, S. Manandhar, and M. Moens. Priority Union and Generalization in Discourse Grammars. In: *Proc. 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 17–24, Las Cruces, NM, 1994. Association for Computational Linguistics.

I. Gurevych, R. Porzel, and R. Malaka. Modeling Domain Knowledge: Know-How and Know-What, 2006. In this volume.

A. Jönsson. *Dialogue Management for Natural Language Interfaces – an Empirical Approach*. PhD thesis, Linköping Studies in Science and Technology, 1993.

A. Jönsson. A Model for Habitable and Efficient Dialogue Management for Natural Language Interaction. *Natural Language Engineering*, 3(2/3):103–122, 1997.

S. Larsson. *Issue-Based Dialogue Management*. PhD thesis, Göteborg University, Sweden, 2002.

M. Löckelt. Plan-Based Dialogue Management for Multiple Cooperating Applications, 2006. In this volume.

M. Löckelt, T. Becker, N. Pfleger, and J. Alexandersson. Making Sense of Partial. In: C.M. Johan Bos, Mary Ellen Foster (ed.), *Proc. 6th Workshop on the Semantics and Pragmatics of Dialogue (EDILOG 2002)*, pp. 101–107, Edinburgh, UK, September 2002.

S. LuperFoy. *Discourse Pegs: A Computational Analysis of Context-Dependent Referring Expressions*. PhD thesis, University of Texas at Austin, December 1991.

S. Luperfoy. The Representation of Multimodal User Interface Dialogues Using Discourse Pegs. In: *Proc. 30th ACL*, pp. 22–31, Newark, DE, 1992.

N. Pfleger. Discourse Processing for Multimodal Dialogues and Its Application in SmartKom. Master's thesis, Universität des Saarlandes, 2002.

N. Pfleger, J. Alexandersson, and T. Becker. Scoring Functions for Overlay and Their Application in Discourse Processing. In: *Proc. KONVENS 2002*, pp. 139–146, Saarbruecken, Germany, September–October 2002.

N. Pfleger, J. Alexandersson, and T. Becker. A Robust and Generic Discourse Model for Multimodal Dialogue. In: *Workshop Notes of the IJCAI-03 Workshop on "Knowledge and Reasoning in Practical Dialogue Systems"*, Acapulco, Mexico, August 2003a.

N. Pfleger, R. Engel, and J. Alexandersson. Robust Multimodal Discourse Processing. In: *Proc. Diabruck: 7th Workshop on the Semantics and Pragmatics of Dialogue*, Wallerfangen, Germany, 4–6 September 2003b.

R. Porzel, I. Gurevych, and R. Malaka. In Context: Integrating Domain- and Situation-Specific Knowledge, 2006. In this volume.

S. Salmon-Alt. Interpreting Referring Expressions by Restructuring Context. In: *Proc. ESSLLI 2000*, Birmingham, UK, 2000. Student Session.

W. Wahlster (ed.). *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer, Berlin Heidelberg New York, 2000.

Overlay: The Basic Operation for Discourse Processing

Jan Alexandersson, Tilman Becker, and Norbert Pfleger

DFKI GmbH, Saarbrücken, Germany
{janal,becker,pfleger}@dfki.de

Summary. We provide a formal description of the fundamental nonmonotonic operation used for discourse modeling. Our algorithm—overlay—consists of a default unification algorithm together with an elaborate scoring functionality. In addition to motivation and highlighting examples from the running system, we give some future directions.

1 Introduction

This chapter introduces the overlay operation—a default unificationlike operation enabling the assignment of consistent background information into a new structure. In order to assess how good the new structure fits to the older one, overlay generates a score reflecting the structural consistency of its arguments. Within the SMARTKOM project we used this operation for the task of discourse processing (Alexandersson and Pfleger, 2006), where intention hypotheses are enriched and validated with respect to their preceding discourse context.

1.1 Adding, Refining, Revising, and Changing

User utterances in dialogue systems usually continue a discourse by supplying new information that adds to the current context. In simple slot-filling applications, e.g., VoiceXML, the new information typically fills the next slot, often as an answer to a direct system question (case (i), see below). In more elaborate systems, a user utterance can also refine prior information, e.g., specify "tonight at 7 PM" as a refinement of a previous, less precise "today" (case (ii)). In both cases, the new information is compatible with the current context and—given an appropriate representation—*unification* can be used as the formal operation to combine a new utterance with the current context.

In systems like SMARTKOM, which allow for more natural, conversational dialogue that is not system-driven, users can revise their input freely and often do so

unexpectedly (cases (iii) and (iv)). In the following we will describe a formal operation to capture such revisions. Before presenting the details of our formal representation for utterances and current context, we give a set of examples with progressive complexity:

(i) A simple case of filling a previously unfilled slot:

U: *I want to see a movie.*

S: *When do you want to see it?*

U: *Tonight.*

(ii) Refinement of a previous, less precise slot-filler:

U: *What's today's TV program?*

S: *Here you see a list of movies running today.*

U: *What's on TV tonight at 7 PM?*

(iii) A case of changing the value of a slot:

U: *First, I'd like to make a reservation in this movie theater.*

S: *Sorry, this theater does not take reservations.*

U: *Ok, then I take [/] this one.*

(iv) Finally, a more complex case of modifying the entire task:

U: *Which movies are showing tomorrow night on HBO?*

S: *Here's tomorrow night's program for HBO.*

U: *And how about the movie theaters?*

S: *Here's a list of movies in cinemas tomorrow night.*

Note how the system has kept the time information "tomorrow night," even though the task was changed from TV to movie theater information. This final case is picked up again in Sect. 3.

1.2 Representing the Current Context

In SMARTKOM we have chosen an ontology-based representation of the tasks that the user can solve through the system. Gurevych et al. (2006) give an in-depth description of the methodology. For the discussion in this chapter, we will assume that instances of the ontology can be represented as typed feature structures (TFS) with a type hierarchy that mirrors the ontology's class hierarchy.

1.3 Combining Current Context and User Utterance

The TFS-based representation allows us to discuss the four cases of *state update* introduced in the previous section in terms of *unification*, (Carpenter, 1992). In particular, (i) the user adding a value for an empty slot can be seen as unification in (untyped) feature structures. (ii) Refinement is a case of unification in typed feature structures, and (iii) revising and (iv) changing are cases of default unification or overlay (Carpenter, 1993). In Sects. 2 and 3, we formalize this approach. Sect. 3.2 describes an enrichment of overlay with a scoring operation that expresses how much information from the current context is retained.

2 Preliminaries: Typed Feature Structures and Unification

Almost 30 years frames or framelike formalisms have been used to represent knowledge (Minsky, 1975). Different requirements, such as processing speed, expressiveness, etc., have guided the development of the representation as such, as well as the algorithms for the manipulation of the representation into several directions. For representing linguistic knowledge and more abstract common-sense knowledge, typed frame-based formalisms are used, such as TFS (Carpenter, 1992) or description logics (Horrocks, 1998b). In parallel different inference operations meeting different needs have been developed. For linguistic analysis, different *unification* operations (Carpenter, 1992) and for description logics, the *classifier* (Horrocks, 1998a) are used.

In this work we are concerned with TFSs and, in particular, operations that ease the interpretation of discourse phenomena such as different elliptical expressions. The naïve standard unification is enough as long as consistent and not contradictory (conflicting) information is added to the discourse. However, as shown in Alexandersson and Pfleger (2006), there are a lot of cases where the user changes his/her mind and provides information that conflicts with the context. Still, it is desirable to obtain a complete description of the user intention that consists of the accumulated contextual information but where new overwrites the old competing information.

In what follows, we summarize our view on TFSs[1] in SMARTKOM (Gurevych et al., 2006) for a description of the actual ontology). We assume a type hierarchy where each pair of types has a least upper bound (LUB). A common characterization for type hierarchies is given in the following definition:

Definition 1. (Partial Order)
A relation $\sqsubseteq$ *on a set of types,* Type, *is a* partial order *where it is:*

- Reflexive, *i.e.,* $\forall x \in \text{Type}$ *we have* $x \sqsubseteq x$.
- Antisymmetric, *i.e.,* $\forall x, y \in \text{Type}$ *if* $x \sqsubseteq y \wedge y \sqsubseteq x$ *then* $x = y$.
- Transitive, *i.e.,* $\forall x, y, z \in \text{Type}$ *if* $x \sqsubseteq y \wedge y \sqsubseteq z$ *then* $x \sqsubseteq z$.

[1] For the complete description, a curious reader is referred to Carpenter (1992) or Krieger (1995)

In our case, we restrict ourselves to bounded complete partial orders (BCPO), which are partial orders where each pair has exactly one LUB.

TFSs are defined based on a finite set of features Feat and an BCPO$\langle$Type, $\sqsubseteq\rangle$:

Definition 2. (Typed Feature Structures)

A typed feature structure (TFS) is a tuple $F = \langle Q, \overline{q}, \theta, \delta\rangle$ *where:*

- *Q is a finite set of nodes with the root $\overline{q}$.*
- *$\overline{q}$ is the (unique) root node.*
- $\theta : Q \rightarrow \mathsf{Type}$ *is a total node typing function.*
- $\delta : \mathsf{Feat} \times Q \rightarrow Q$ *is a partial feature value function.*

We adopt the appropriate condition from Carpenter (1992), which states that a feature is appropriate for a type and all its subtypes.

Definition 3. (Appropriateness)

In an inheritance hierarchy $\langle \mathsf{Type}, \sqsubseteq\rangle$, *the partial function Approp:* $\mathsf{Feat} \times \mathsf{Type} \rightarrow \mathsf{Type}$ *meets the following conditions:*

- **Feature Introduction.** *For every feature $f \in \mathsf{Feat}$, there is a most general type—$Intro(f) \in \mathsf{Type}$—such that $Approp(f,(Intro(f))$ is defined.*
- **Downward Closure.** *If $Approp(f,\sigma)$ is defined and $\sigma \sqsubseteq \tau$, then $Approp(f,\tau)$ is also defined and $Approp(f,\sigma) \sqsubseteq Approp(f,\tau)$.*

In order to define unification and default unification, TFSs are ordered by the following subsumption relation which spans a lattice:

Definition 4. (Subsumption)

$F = \langle Q, \overline{q}, \theta, \delta\rangle$ is said to subsume $F' = \langle Q', \overline{q}', \theta', \delta'\rangle$, $F \sqsubseteq F'$, iff there is a total function $h : Q \rightarrow Q'$ such that:

- *$h(\overline{q}) = \overline{q}'$.*
- *$\theta(q) \sqsubseteq \theta'(h(q))$ for every $q \in Q$.*
- *$h(\delta(f,q)) = \delta'(f,h(q))$ for every $q \in Q$ and feature f such that $\delta(f,q)$ is defined.*

Definition 5. (Unification)

In an inheritance hierarchy as described above, the unification of two feature structures is their greatest lower bound (GLB) $\sqcup$.

It is worth noting that for treelike type hierarchies, two TFSs unify only in cases where they are type compatible. Finally, our default unification is characterized by the following elegant and powerful definition)taken from Carpenter (1993)):

Definition 6. (Credulous Default Unification)

$F \overset{<}{\sqcup}_r G = \{F \sqcup G' | G' \sqsubseteq G$ *is maximal such that* $F \sqcup G'$ *is defined* $\}$.

3 Overlay = Default Unification + Score

In our main application, we are concerned with two challenges. We provided a motivation for the first one—enrichment—in Sect. 1. The second one—validation—is concerned with the task of assigning the most probable hypothesis among several a higher number than the competing ones. In SMARTKOM, i.e., systems processing user actions run though nonperfect recognizers and/or analyzers, the possible interpretations of a user action will almost never be unambiguous. The task of comparing the representation of the user action with context must involve some kind of scoring telling how well the representation fits the context. Our solution is to combine default unification with a scoring functionality so that the actual score is computed during the combination of new and old information. To this end we developed an efficient implementation of default unification and equipped it with a scoring mechanism. Below we provide a formal operational semantics of this default unification operation (Sect. 3.1), whereas the heuristic scoring function is described in Sect. 3.2 (see also Sect. 3.3 for a highlighting example).

3.1 Default Unification

Although the theoretical machinery presented above provides a precise characterization of unification and default unification, it says nothing about how to compute unification and default unification. Several researchers have provided efficient algorithms for unification for use in parsing, e.g., Tomabechi (1991). However, in the case of default unification there exists little about efficient implementations. The few exceptions are found in the area of robust parsing of NL texts, e.g., Imaichi and Matsumoto (1995); Fouvry (2003). This is due to the fact that for more complex inheritance hierarchies, the credulous default unification is, in the general case, regarded to be exponential, e.g., Copestake (1993); Grover et al. (1994). The main reason is that credulous default unification is ambiguous and the fact that in type systems providing multiple inheritance and reentrant structures (coreferences), the search for the least upper bound, or more precisely, the minimal upper bounds, is exponential. In our case, there are no reentrant structures, and in the type hierarchy we use unary inheritance only. Therefore even a naïve algorithm as presented below computes default unification with a complexity linear to the size of its arguments.

For this chapter we adapt the definition of overlay given in Alexandersson and Becker (2001). The basic schema for the computation of overlay is based on two steps. In the first—called assimilation—the general case is that the background is transformed to the type of the LUB of the covering and background. An important consequence of the transformation is that only the features appropriate for the LUB are kept. There are cases when this viewpoint does not hold, since it could be that the covering is more general than the background, in which case the two structures are kept. In the second step, an algorithm close to naïve unification is deployed with the small change, that in case of conflicts, the information of the covering overwrites the one in background (atomic values), or overlay is recursively called (complex values).

The type of the resulting structure is the type of the covering in case of conflicting types and the type of their GLB in case of compatible types.

Definition 7. (Assimilation)

Let co and bg be two TFSs (covering and background) such that $co = \langle Q_{co}, \overline{q}_{co}, \theta_{co}, \delta_{co} \rangle$ *and* $bg = \langle Q_{bg}, \overline{q}_{bg}, \theta_{bg}, \delta_{bg} \rangle$. *Then, the assimilation—assim(co, bg)—is defined as*

$$\begin{aligned} &\textit{if } (co \sqsubseteq bg \textit{ or } bg \sqsubseteq co), \\ &\textit{then } assim(co, bg) := \langle co, bg \rangle, \\ &\textit{else } assim(co, bg) := \langle co, \langle Q_a, \overline{q}_a, \theta_a, \delta_a \rangle \rangle, \end{aligned}$$

where

- $t_a := LUB(co, bg)$,
- $\overline{q}_a = \overline{q}_{bg}$ *with* $\theta(\overline{q}_a) = t_a$,
- δ_a *is the same as* δ_{bg} *for* f *reachable from* $\overline{q}_a$ *(but for* $\delta_a(f, \overline{q}_a)$ *we require* $Approp(f, t_a)$*)*,
- Q_a *is the set of nodes reachable from* $\overline{q}_a$,
- θ_a *is the same as* θ_{bg} *for the nodes reachable from* $\overline{q}_a$ *with* $\theta_a(\overline{q}_a) = t_a$.

In what follows, we assume a function α that returns the resulting cover and background, i.e., $\alpha_{co}(assim(co, bg)) = co$. Next, we define overlay:

Definition 8. (Overlay)

Let co and bg be two TFSs (covering and background) such that $assim(co, bg) = \langle co', bg' \rangle = \langle \langle Q_{co'}, \overline{q}_{co'}, \theta_{co'}, \delta_{co'} \rangle, \langle Q_{bg'}, \overline{q}_{bg'}, \theta_{bg'}, \delta_{bg'} \rangle \rangle$. *Then,* $Overlay(co, bg)$ *is defined as:*

$$Overlay(co, bg) := Overlay'(co', bg') := \langle Q_o, \overline{q}_o, \theta_o, \delta_o \rangle, \tag{1}$$

where

$$\overline{q}_o := \overline{q}_{co'}, \quad \textit{and} \quad \theta_o(\overline{q}_o) := GLB(\theta_{co'}(\overline{q}_{co'}), \theta_{bg'}(\overline{q}_{bg'})), \quad \textit{and}$$

$$\delta_o(f, \overline{q}_o) :=$$

$$Overlay(\delta_{co'}(f, \overline{q}_{co'}), \delta(f, \overline{q}_{bg'})) \textit{ if } f \textit{ exists in } co' \textit{ and } bg', \tag{2}$$

$$\delta_{co'}(f, \overline{q}_{co'}) \textit{ if } f \textit{ exists only in } co', \tag{3}$$

$$\delta_{bg'}(f, \overline{q}_{bg'}) \textit{ if } f \textit{ exists only in } bg'. \tag{4}$$

3.1.1 Related Work

Within AI there are different approaches to default reasoning, i.e., how to draw inferences despite inconsistencies in the knowledge base. Within these approaches, the distinction between credulous and skeptical inferences is made. The latter type of inferences means that inferences are not drawn in case of conflicts, whereas inferences of the former type mean that as many consistent conjectures as possible are

drawn (Antoniou, 1999; Touretzky, 1986). A famous example related to this distinction is how in inheritance hierarchies one has to allow for overwriting in the case of describing a penguin. Since birds can fly, and penguins are birds but cannot fly, the information stating that birds can fly has to be overwritten. In our case we want to overwrite old contextual information with new conflicting information and at the same time be able to draw inferences—we are thus aiming for credulous default reasoning.

Within computation linguistics, the early work starts with Karttunen, e.g., Karttunen (1986), who used in his DPATR workbench a default unification algorithm called *clobber*. Kaplan (1987) introduces *priority union* as a technique for an elegant way of processing elliptical or morphological nonmonotonic constructions. These approaches are, however, based on nontyped feature structures. The processing of elliptical constructions is picked up by Grover et al. (1994), but because of the expense for computing priority union—it is exponential—the authors state that their implementation cannot be used in real applications.

In the beginning of the 1990s, techniques for the processing of typed feature structures were developed (Bouma, 1990; Calder, 1993; Carpenter, 1993). The application for default unification ranges from lexicon management (Villavicencio, 1999; Lascarides et al., 1996; Carpenter, 1993) over robust parsing (Imaichi and Matsumoto, 1995; Fouvry, 2003) to discourse modeling where, for instance, elliptical constructions have to be processed (Grover et al., 1994; Alexandersson and Becker, 2004).

3.2 Scoring

So far, we have seen that overlay can be used to transfer nonconflicting information from one structure to another—even in the case of type clashes or conflicting atomic values. What is missing is a way to assess the structural resemblance of the involved structures to be able to judge how well a background structure fits the covering. This scoring of the overlay result can be very useful, e.g., for tasks like disambiguation in discourse processing, where several ambiguous hypotheses have to be interpreted with respect to the discourse context. As a consequence, we extended the basic overlay operation with a scoring mechanism (Pfleger et al., 2002). This scoring mechanism is based on the collection of different parameters during the application of overlay. The actual scoring mechanism consists of two parts: (*i*) the collection of the scoring parameters, which is incorporated into the algorithm, and (*ii*) the actual scoring function, which compiles the collected parameters into one resulting score after the application of overlay is finished.

3.2.1 A Heuristic Scoring Function

For SMARTKOM we developed a heuristic scoring function that basically expresses the contrast between the amount of nonconflicting features and the amount of conflicting features (including type clashes). We identified four scoring parameters that need to be collected during the application of overlay:

co: a feature or a (atomic) value stemming from the covering is added to the result. **co** is incremented for each feature in the covering.

bg: a feature or a (atomic) value in the result occurs in the background; **bg** is incremented.

tc: type clash, i.e., the type of the covering and background was not identical. This is identified during the computation of the assimilation.

cv: conflicting values. This occurs when the value of a feature from the background is overwritten.

All these parameters are collected during the application of overlay, and in the end a function compiles that information into a single number reflecting the structural consistency of the two structures. The sum of **co** and **bg** minus the sum of **tc** and **cv** will be weighted against the sum of **co, bg, tc** and **cv**. This leads to a function (shown in the formula below) whose codomain is $[-1,1]$.

$$score(co,bg,tc,cv) = \frac{co+bg-(tc+cv)}{co+bg+(tc+cv)}.$$

The positive extremal ($score(co,gb,tc,cv) = 1$) indicates that the feature structures are unifiable. The negative extremal ($score(co,gb,tc,cv) = -1$) indicates that all information from the background was overwritten by information from the cover. A score within this interval indicates that the cover more or less fits the background: The higher the score, the better the cover fits the background. Negative values signal that conflicting and thus overlayed values outweigh unifiable values (and positive values vice versa).

3.2.2 The Distance Factor

The overall scoring of overlay should not only reflect the structural resemblance of the structures for particular tasks, but it is also necessary to incorporate the temporal distance of the overlayed structures into the scoring. In other words, it is necessary to weigh the overall scoring of the result against the amount of time that has passed since the occurrence of the background structure and the occurrence of the covering.

What kind of distance factor is best suited for weighting the score of overlay depends on the application in which overlay is employed. In an application like discourse processing, for example, the score described above has to be weighted by the distance between the two hypotheses in the discourse (Alexandersson and Pfleger, 2006).

3.3 An Example

The following example provides insight into the disambiguation as well as the enrichment functionality provided by overlay. As indicated above, one potential benefit of the scoring heuristic is the disambiguation of ambiguous user input or ambiguous recognition results. Consider, for example, a dialogue situation (see dialogue excerpt

in Fig. 1) where the user has requested information about tonight's television program (see Fig. 2 for a TFS representation), resulting in a system presentation of tonight's television program. Now the user wants to reduce the number of displayed movies by uttering *Is there a movie with Nicole Kidman?*. This user request, if interpreted separately, leads to at least two intention hypotheses — one where the user requests information about the current TV program (see Fig. 3) and one where the user requests information about the current cinema program (see Fig. 4).

U: *What's on TV tonight?*

S: (Displays the TV program listings) *Here you see a list of movies running tonight.*

U: *Is there a movie with Nicole Kidman?*

Fig. 1. Dialogue excerpt

If we now overlay these two interpretation hypotheses with the latest user contribution we get for both a suitable result. However, one of the results is more likely than the other, and this is expressed by the scoring of overlay. Column 2 of table 1 shows the collected parameters and the overall scoring for overlaying the first hypothesis (TFS in Fig. 3) and the previous user contribution (TFS in Fig. 2), whereas column 3 shows the same for overlaying the second hypothesis (TFS in Fig. 4) and the last user contribution. As expected, the result of overlaying the second hypothesis over the last user contribution receives the higher score, namely 1.0, which is the highest possible score and indicates that the two structures are unifiable.

Table 1. Parameters collected during the application of overlay

	TFS 2 and TFS 3	TFS 2 and TFS 4
co	10	10
bg	5	7
tc	2	0
cv	0	0
score	0.765	1.0

4 Conclusion

In this chapter we presented a formal description of overlay—the fundamental operation for discourse processing in SMARTKOM. In particular we described the two most important aspects of this operation—namely default unification and scoring—in detail. Furthermore, we outlined the application of overlay for the task of discourse processing and gave an example that highlights the value of the scoring mechansim.

```
[ IS
  pOI: [ TV
         bc: [ BC
               beginTime: [ TIME
                            analysis: [ INSTANT
                                        day: today
                                        daytime: evening ]
                            function: [ BETWEEN
                                        from: 2003-06-03T18:00:00
                                        to: 2003-06-03T23:59:59 ] ]
               avMedium: [ AVMEDIUM ] ] ] ]
```

Fig. 2. TFS representation for the utterance *What's on TV tonight?* (*IS* = InformationSearch, *pOI* = pieceOfInformation, *Tv* = TvProgram, *bc* = broadcast)

```
[ IS
  pOI: [ CP
         p: [ PERF.
              avM: [ AM
                     cast: [ CAST
                             member: [ ACTOR
                                       name: Nicole Kidman ] ] ] ] ] ]
```

Fig. 3. One possible interpretation of the utterance *Is there a movie with Nicole Kidman?* (*IS* = InformationSearch, *pOI* = pieceOfInformation, *CP* = CinemaProgram, *p* = performance, *P* = Performance, *bc* = Broadcast, *Bc* = Broadcast)

```
[ IS
  pOI: [ TP
         bc: [ BC
               aM: [ AM
                     cast: [ CAST
                             member: [ ACTOR
                                       name: Nicole Kidman ] ] ] ] ] ]
```

Fig. 4. A second possible interpretation of the utterance *Is there a movie with Nicole Kidman?* (*IS* = InformationSearch, *pOI* = pieceOfInformation, *CP* = CinemaProgram, *p* = performance, *P* = Performance, *aM* = avMedium, *AM* = AvMedium)

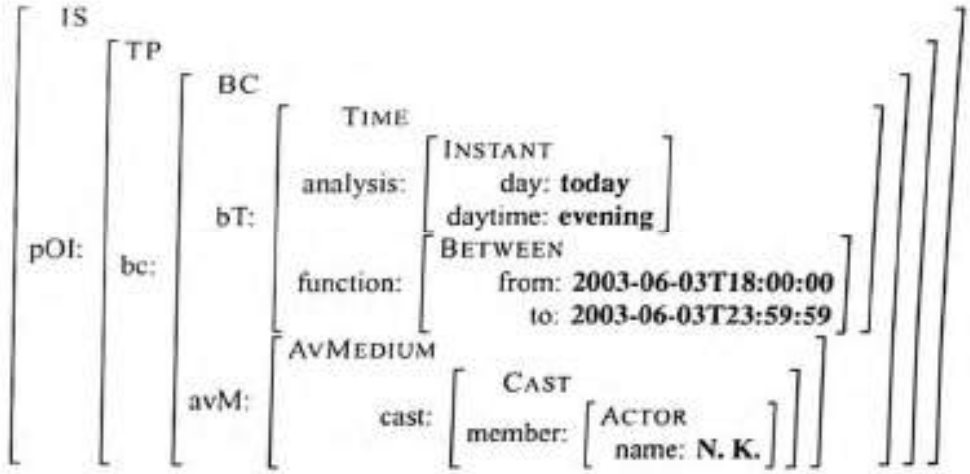


Fig. 5. Interpretation of the utterance *Is there a movie with Nicole Kidman?* with respect to the previous discourse context

For more information about the application of overlay in discourse processing, we refer to Alexandersson and Pfleger (2006).

Future work will cover the areas described below.

- This chapter provided a set-theoretical characterization of the behavior of overlay. The complexity of default unification for reentrant structures in complex inheritance hierarchies is regarded as exponential (Copestake, 1993). In Alexandersson and Becker (2003) we provide an efficient algorithm for the computation of default unification based on a multiple-inheritance type hierarchy.
- For SMARTKOM we decided to use typed feature structures (TFS) as the internal data representation for overlay. As we have shown above, TFS can be used to represent instances of an ontology, and thus we had the idea of an implementation of overlay which is directly based on ontological structures. To this end we are currently preparing an implementation of overlay for the Protégé ontology editor (`http://protege.stanford.edu/`).
- In some cases, the structures contain set-like objects, e.g., seats in a movie theater, which require a special treatment. For an ordered list it might be suitable to overlay the first element of the first list with the first element of the second one, and so on. However, in cases where the elements of a set are not ordered, it depends also on the objects themselves how overlay should be applied to the sets. Our current work outlines a framework for how and when what kind of operations should be used to combine such sets.
- Our present scoring function does not account for all available information, as it just counts the type clashes. There is, however, more information available that can contribute to the overall score of overlay. An example for this is the *informational distance*—the distance between the background type and the LUB

in case of a type clash (Alexandersson et al., 2004)—which assess how distinct conflicting covering and background are.

Acknowledgments

The authors would like to thank Massimo Romanelli for discussions and proofreading.

References

J. Alexandersson and T. Becker. Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System. In: *Workshop Notes of the IJCAI-01 Workshop on "Knowledge and Reasoning in Practical Dialogue Systems"*, Seattle, WA, August 2001.

J. Alexandersson and T. Becker. The Formal Foundations Underlying Overlay. In: *Proc. 5th Int. Workshop on Computational Semantics (IWCS-5)*, pp. 22–36, Tilburg, The Netherlands, February 2003.

J. Alexandersson and T. Becker. Default Unification for Discourse Modelling. In: H. Bunt and R. Muskens (eds.), *Computing Meaning*, vol. 3, Dordrecht, The Netherlands, 2004. Kluwer Academic.

J. Alexandersson, T. Becker, and N. Pfleger. Scoring for Overlay Based on Informational Distance. In: *7. Konferenz zur Verarbeitung natürlicher Sprache KONVENS-04*, Vienna, Austria, September – October 2004.

J. Alexandersson and N. Pfleger. Discourse Modeling, 2006. In this volume.

G. Antoniou. A Tutorial on Default Logics. *ACM Computing Surveys (CSUR) Archive*, 31(4):337–359, December 1999.

G. Bouma. Defaults in Unification Grammar. In: *Proc. 28nd Annual Meeting of the Association for Computational Linguistics*, pp. 165–172, University of Pittsburgh, PA, 6–9 June 1990.

J. Calder. Typed Unification for Natural Language Processing. In: T. Briscoe, V. de Paiva, and A. Copestake (eds.), *Inheritance, Defaults, and the Lexicon*, pp. 13–37, Cambridge, UK, 1993. Cambridge University Press.

B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, UK, 1992.

B. Carpenter. Skeptical and Credulous Default Unification With Applications to Templates and Inheritance. In: T. Briscoe, V. de Paiva, and A. Copestake (eds.), *Inheritance, Defaults, and the Lexicon*, pp. 13–37, Cambridge, CA, 1993. Cambridge University Press.

A. Copestake. *The Representation of Lexical Semantic Information*. PhD thesis, University of Sussex, 1993.

F. Fouvry. Constraint Relaxation with Weighted Feature Structures. In: *Int. Workshop on Parsing Technologies - IWPT'03*, pp. 103–114, Nancy, France, April 2003.

C. Grover, C. Brew, S. Manandhar, and M. Moens. Priority Union and Generalization in Discourse Grammars. In: *Proc. 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 17–24, Las Cruces, NM, 1994. Association for Computational Linguistics.

I. Gurevych, R. Porzel, and R. Malaka. Modeling Domain Knowledge: Know-How and Know-What, 2006. In this volume.

I. Horrocks. The FaCT System. In: *Automated Reasoning With Analytic Tableaux and Related Methods: Int. Conf. Tableaux'98*, pp. 307–312, Oisterwijk, The Netherlands, 1998a.

I. Horrocks. Using an Expressive Description Logic: FaCT Or Fiction? In: *Principles of Knowledge Representation and Reasoning: Proc. 6th Int. Conf. (KR'98)*, pp. 636–647, San Francisco, California, 1998b.

O. Imaichi and Y. Matsumoto. Integration of Syntactic, Semantic and Contextual Information in Processing Grammatically Ill-Formed Inputs. In: *Proc. 14th IJCAI*, pp. 1435–1440, Montréal, Canada, 1995.

R.M. Kaplan. Three Seductions of Computational Psycholinguistics. In: P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood (eds.), *Linguistic Theory and Computer Applications*, pp. 149–188, London, UK, 1987. Academic Press.

L. Karttunen. D-PATR: A Development Environment for Unification-Based Grammars. In: *Proc. 11th COLING*, pp. 25–29, Bonn, Germany, August 1986. Institut für angewandte Kommunikations- und Sprachforschung e.V. (IKS).

H.U. Krieger. *TDL — A Type Description Language for Constraint-Based Grammars. Foundations, Implementation, and Applications.* PhD thesis, Universität des Saarlandes, Department of Computer Science, september 1995.

A. Lascarides, T. Briscoe, N. Asher, and A. Copestake. Order Independent and Persistent Typed Default Unification. *Linguistics and Philosophy*, 19(1):1–89, 1996. Revised version of ACQUILEX II WP NO. 40.

M. Minsky. A Framework for Representing Knowledge. In: P. Winston (ed.), *The Psychology of Computer Vision*, pp. 211–277, New York, 1975. McGraw-Hill.

N. Pfleger, J. Alexandersson, and T. Becker. Scoring Functions for Overlay and Their Application in Discourse Processing. In: *Proc. KONVENS 2002*, pp. 139–146, Saarbruecken, Germany, September–October 2002.

H. Tomabechi. Quasi-Destructive Graph Unifications. In: *Proc. 29th ACL*, pp. 315–322, Berkeley, CA, 1991.

D. Touretzky. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, San Francisco, CA, 1986.

A. Villavicencio. Representing a System of Lexical Types Using Default Unication. In: *Proc. 9th Conf. of the European Chapter of the Association for Computational Linguistics (EACL'95)*, Bergen, Norway, 1999.

In Context: Integrating Domain- and Situation-Specific Knowledge

Robert Porzel, Iryna Gurevych*, and Rainer Malaka

European Media Laboratory GmbH, Heidelberg, Germany
robert.porzel,rainer.malaka@eml-d.villa-bosch.de
iryna.gurevych@eml-r.villa-bosch.de
* Current affiliation EML Research gGmbH.

Summary. We describe the role of context models in natural language processing systems and their implementation and evaluation in the SMARTKOM system. We show that contextual knowledge is needed for an ensemble of tasks, such as lexical and pragmatic disambiguation, decontextualizion of domain and common-sense knowledge that was left implicit by the user and for estimating an overall coherence score that is used in intention recognition. As the successful evaluations show, the implemented context model enables a multicontext system, such as SMARTKOM, to respond felicitously to contextually underspecified questions. This ability constitutes an important step toward making dialogue systems more intuitively usable and conversational without losing their reliability and robustness.

1 Introduction

The human enterprise of answering or responding to conversational speech input in a suitable and felicitous manner is not imaginable without three essential features:

- the ability to recognize what was said by the questioner
- the ability to infer information that is left implicit by the questioner
- the ability to infer what constitutes a useful and felicitous answer

The realization of such abilities poses a formidable challenge in the development of conversational intuitive dialogue systems with more than one domain, modality or situational context. The SMARTKOM system has to deal with contextual dependencies as well as cross-modal references based on the system's symmetric multimodality (Wahlster, 2003). It can handle multiple requests in different domain contexts and features special scenario-specific situational contexts. Thus, *decontextualization* is needed to resolve the arising contextual ambiguities (McCarthy, 1986, 1990). In the case of restricted and controlled single-domain systems, the problem of contextually implicit information can be solved by generating full paraphrases out of the underspecified user utterances (Ebert et al., 2001). In systems with multiple contexts

additional knowledge sources and dynamic context modeling are needed. Herein we describe the central contextual processing unit which combines ontological and situative knowledge. In the SMARTKOM system, discourse contextual influences are handled by unification-based operations, for example OVERLAY(Alexandersson and Becker, 2003; Löckelt et al., 2002), which operate on the schemas automatically created from the SMARTKOM ontology (Gurevych et al., 2003b; Porzel et al., 2003b). They also interact closely with the contextual processing module described herein, for the resolution of deictic expressions.

This integration of basic common-sense domain knowledge with situative context knowledge constitutes a necessary building block for scalable natural language understanding systems that facilitate felicitous cooperation and intuitive access to web-based, location-based and personal information. In single-context systems, such as train schedules or help desk systems (Aust et al., 1995; Gorin et al., 1997), this does not constitute a problem, since conversational phenomena such as pragmatic and semantic ambiguities or indirect speech acts do not occur (Porzel and Gurevych, 2002). A multidomain, multiscenario and multimodal system faces diverse usage contexts (e.g., at home or in mobile scenarios), conversational phenomena (e.g., indirect speech acts and pragmatic ambiguities) and multiple cross-modal references (e.g., gestural and linguistic discourse objects). A comprehensive understanding of naturally occurring discourse and of the often implicit questions embedded therein still has many unsolved issues in computational linguistics. In this work, we describe research on context and knowledge modeling components that enable dialogue systems with multiple contexts to realize the needed capabilities outlined above.

For example, in most conversational settings a passerby's response to a question such as:

(1) Excuse me, how do I get to the castle?

will most likely not be followed by asking where and when the spatial instructions should start. More likely, directions will be given. But, as the collected field data (see Sect. 5.1) shows, the felicity of spatial instructions is also dependent on contextual factors such as distance, mobility of the questioner or weather. Information concerning time or place, for example, is rarely explicated when given *default* settings, based on *common ground* (Krauss, 1987) hold. If not, however, such information is very likely to be expressed explicitely. In some cases, which are commonly labeled as *indirect speech acts* or *pragmatic ambiguities*, we are not only faced with implicit information, but also with implicit intentions.

It is, however, possible to resolve the ensuing ambiguities and determine appropriate default settings using additional context, dialogue and system knowledge. We will show how such knowledge can be based on collected data relevant to the domains and situations at hand. Next to the *Wizard-of-Oz* data collections and data collected in evaluations, based on the PROMISE framework (Beringer et al., 2002), we included existing lexicographic and ontological analyses, e.g., a model of frame semantics (Baker et al., 1998) as well as an ontological top-level (Russell and Norvig, 1995), and conducted additional experiments and data collections in *Hidden-Operator* experiments (Rapp and Strube, 2002) and the newly developed

Wizard-and-Operator paradigm (Gelbart et al., 2006; Porzel and Baudis, 2004). This collected, analyzed and modeled information, then became part of the ontological domain knowledge, i.e., the hierarchies, relations and cardinalities modeled therein.

Ontologies have traditionally been used to represent domain knowledge and are employed for various linguistic tasks, e.g., semantic interpretation, anaphora or metonymy resolution. In our case, the aggregate model of situative and domain knowledge contains the SMARTKOM ontology (Gurevych et al., 2003b; Porzel et al., 2003b; Gurevych et al., 2006). As follows from interfacing with automatic speech, gesture and emotional recognition systems, a significant amount of uncertainty is involved, which is probably best reflected in the ensuing intention lattices and their confidence scores. Whether one looks at intention, word lattices or *n*-best lists of hypotheses, the problem of facing several different representations of a single utterance arises. This remains, even though multimodal systems can use the individual modalities to disambiguate each other. Different hypotheses of what the user actually might have said, of course, lead to a different understanding and, in consequence, to potentially different requests to the background system. The role of the context model in this light is to assist in evaluating the competing intention hypotheses against each other to find out what was said. Then, such contextual domain and situational knowledge can be used for augmenting such intention hypotheses with implicit information, to spell out their underlying intentions and, finally, to define a common background representation for the processed content, i.e., intention lattices in the case of the SMARTKOM system. Summarizing, a context model, therefore, can be employed in the following tasks:

- The explication of situationally implicit information. This task can be further differentiated into two subtasks:
  - provision of information that is indexical — such as time and place — based on common ground and common sense defaults and their dynamic instances, e.g., the current position of the user
  - provision of information that is pragmatic, such as speech acts and intentions and their dynamic instances, e.g., the actual open or closed state (accessibility) of the goal object

- The scoring of individual interpretations in terms of their contextual coherence. Again, this task can be further differentiated into two subtasks:
  - using the ontological domain context to measure the semantic coherence of the individual interpretations, e.g., the ranking *n*-best lists or semantic interpretations thereof
  - using dynamic situational and discourse information, e.g., previous ontological contexts of prior turns

Additionally, we can use the ontological knowledge modeled therein as the basis for defining the semantics and content of the information exchanged between the modules of multimodal technology systems, as described by Gurevych et al. (2003b) and Gurevych et al. (2006)

After an overview of the state of the art of dialogue systems in light of their domain and context specificity, we discuss the nature of domain and situation models and their role in multidomain, multiscenario and multimodal dialogue systems. Finally, we describe the architecture and processing of the context-model in the SMARTKOM system (Wahlster, 2003).

2 Contextual Processing in Dialogue Systems

Earlier approaches to handle conversational natural language input produced only "toy" systems. Their respective aims were to cope with special linguistic problems and/or to model particular cognitive capacities of language users. Broad coverage of constructions, lexical information sources and semantic/pragmatic behaviors was not the primary concern and was also far outside the scope and capabilities of these systems. Today's linguistic development environments, representations and methodologies have shown that approximately complete coverage may be achieved in the areas of morphology and computational grammar. Furthermore, large lexical resources have been made available for linguistic applications in the area of parsing and also for natural language interfaces to application areas with restricted domains. Even though a broad coverage of frame semantic specification is still in the annotation progress (Baker et al., 1998), the handling of lexical semantics is still not set in stone (Allen, 1987; Gil and Ratnakar, 2002) and formal methods for dealing with pragmatic factors are in their beginning stages (Bunt, 2000; Porzel and Strube, 2002; Porzel and Gurevych, 2002). Systems are in development that can offer suitable natural language interfaces both on the reception and the production side. These systems can be (and are) employed in domain-specific applications or demonstrators where they are commonly linked to nonlinguistic applications (called the *background system*) such as databases (Gallwitz et al., 1998), geographic information systems (Malaka and Zipf, 2000; Johnston et al., 2002), task planning systems (Allen et al., 1995; Ferguson and Allen, 1998) or customer service systems (Gorin et al., 1997).

Some open issues in handling the multidomain problem are successfully beginning to be handled in the question-answering arena, by improved question parsing techniques coupled with more knowledge-based information understanding methods (Hovy et al., 2001; Prager et al., 2001; Moldovan et al., 2002). These information retrieval solutions assume more traditional desktop scenarios and more or less homogeneous content bases. While this is a reasonable thing to do for the type of information retrieval tasks with which the respective systems have to deal, conversational dialogue systems are faced with additional complications that are added to the general open domain problem. Next to the spontaneous speech recognition input, additional factors are the changing context/situation of a mobile user and system on the one hand, and the multitude of heterogeneous content bases that are needed to handle the topical informational need of a mobile user (e.g., a tourist) on the other. The content sources encompass, for example, rapidly changing online cinema information services, electronic program guides, or hotel reservation systems; slower

changing remote geographic information systems; or relatively stable historical and architectural databases.

Natural language understanding in the area of parsing and pragmatically understanding questions as well as in terms of extracting their underlying *intentions* and finding suitable and felicitous answers is far from being solved. Still, a variety of robust parsers can deliver valuable contributions beyond part of speech and treebank tags (Pieraccini et al., 1992; Collins, 1996; Gavaldá and Waibel, 1998; Engel, 2002).

The fact that multidomain, multiscenario and multimodal conversational dialogue systems have so far been nonexistent in the real word is in part due to the fact that in all areas of natural language processing (NLP) we face a mixture of context-variant and context-invariant factors that come into play at every level of NLP pipeline, e.g., speech recognition, semantic disambiguation, anaphoric resolution, parsing or generation. Ensembles of techniques and experiments are therefore needed to identify whether a factor is context-variant or not, and to identify specific types of contextual settings on which a given context-variant factor depends. Based on these findings and their application, decontextualization can be performed based on the contextual knowledge extracted, learned and modeled from the collected data. The results of such decontextualizations, e.g., for semantic disambiguation, then in turn can be evaluated using existing evaluation frameworks (Walker et al., 2000; Beringer et al., 2002).

(2) Where is the cinema Europa?

In real situations seemingly "simple" questions such as (2) are difficult to understand in the absence of context. A felicitous answer often depends on the situational context in which the dialogue takes place. That is, as the data collected and analyzed shows (Porzel and Gurevych, 2002), *where is* questions are either answered by localizations — if the reference object happens to be closed or with an instruction — if the reference object, e.g., a cinema or store, is open. In such cases of pragmatic ambiguity, the model resulting from the analyses of the corresponding data has to embed the utterance at hand into a greater situational context, e.g., by computing a contextual coherence score for the competing interpretations. The situation model consequently has to monitor the corresponding situational factors relevant to resolving such pragmatic ambiguities. Additionally, these situational observations are also needed for the resolution of indexicals, e.g., in the case of spatial or temporal deixis (Porzel and Gurevych, 2003).

The contextual coherence computations that are needed for decontextualization have to be able to deal with a variety of cases:

- For example, if decisions hinge on a number of contextual features, e.g., the situational accessibility of referenced objects (Porzel and Gurevych, 2003), or domain-specific and pragmatic factors based on relations between referenced objects, as found in metonomyzation (Nunberg, 1987). Here both ontological factors as well as situational factors come into play, e.g., semantic roles, weather, and discourse factors, including referential status, as well as user-related factors, e.g., tourists or business travelers as questioners and their time constraints.

- Additionally, if decisions hinge the contextual coherence of sets of concepts and their relations by applying both dialogical as well as semantic coherence measurements (Gurevych et al., 2003a; Porzel et al., 2003a), e.g., for ranking speech recognition hypotheses or semantic ambiguities.

3 Context Modeling

Utterances in dialogues, whether in human–human interaction or human–computer interaction, occur in a specific situation that is composed of different types of contexts. A broad categorization of the types of context relevant to spoken dialogue systems, their content and respective knowledge stores is given in Table 1.

Table 1. Contexts, content and knowledge sources

Types of context	Content	Knowledge store
Dialogical context	What has been said by whom	Dialogue model
Ontological Context	World/conceptual knowledge	Domain model
Situational context	Time, place, etc.	Situation model
Interlocutionary context	Properties of the interlocutors	User model

Following the common distinction between linguistic and extralinguistic context[1] our first category, i.e., the dialogical context, constitutes the linguistic context, encompassing both cotext as well as intertext (Bunt, 2000). In linguistics the study of the relations between linguistic phenomena and aspects of the context of language use is called *pragmatics*. Any theoretical or computational model dealing with reference resolution, e.g., anaphora or bridging resolution, spatial or temporal deixis, or nonliteral meanings, requires taking the properties of the context into account.

As knowledge sources in dialogue systems, domain models are regarded to "hold knowledge of the world that is talked about" (Flycht-Erriksson, 1999). Following this general definition comes the observation that:

> Information from the domain model is primarily used to guide the semantic interpretation of user's utterances; to find the relevant items and relations that are discussed, to supply default values, etc. The knowledge represented in a domain model is often coupled to the background system, e.g., a database system ... the domain knowledge is used to map information in the user's utterance to concepts suitable for database search.

We propose a different definition of the role of domain models in NLP systems such as SMARTKOM. In our minds the knowledge contained in a domain model is to be modeled as an ontology proper, i.e., independent from the way an utterance or

[1] All extralinguistic contexts are also often referred to as the *situational context* (Connolly, 2001), however, we adopt a finer categorization thereof.

query is processed by the background system, that is, the knowledge about (going to) cinemas, (seeing) movies and (getting) tickets. The representation of this knowledge is the same whether the background system is a specific database, a set of Web-spidering agents or a combination thereof.

Statistical models based on specific corpora can serve to define *context groups* (Widdows, 2003) and allow us to differentiate between sets of distinct domain contexts that feature respective sense- and co-occurrence distributions. In our terminology, this formal context group function outputs a domain, i.e., the real-world utterance-based linguistic target of our definition. It is important to note that despite the multitude of domains that are to be encompassed by the SMARTKOM system, the central aim is to create a kernel NLP system capable of dealing with multiple and extensible domains, which ultimately can be added to the system during runtime (Rapp et al., 2000).

One of the central ideas embedded within the SMARTKOM research framework is to develop a kernel NLP system that can be used in a variety of situations, i.e., scenarios, domains and modalities, see Wahlster (2002), whereby:

- *Scenarios* refer to different manifestations of the system, i.e., a home, office and public (booth) manifestation as well as a mobile one.
- *Modality* refers to speech, gesture, mimics, affectives and biometry.
- *Domain* refers to the general backdrop against which dialogues can be pitted, i.e., areas such as train schedules, movie information or hotel reservations.

These additional scenario-specific contexts feature:

- dynamic mobility of the user - where traditional input modalities, such as keyboard- and mouse-based input, are highly unsuitable
- prolonged dialogues throughout sometimes hour-long spatial navigation tasks
- context-dependent intentions

Therefore dynamic, e.g., situational, context information has to be integrated together with the domain knowledge.[2]

Speakers may not always be aware of the potential ambiguities inherent in their utterances. They leave it to the context to disambiguate and specify the message. Furthermore, they trust in the addressee's ability to extract that meaning from the utterance that they wanted to convey. In order to interpret the utterance correctly, the addressee must employ several context-dependent resources. Speakers in turn anticipate the employment of these interpretative resources by the hearer and construct the utterance knowing that certain underspecifications are possible since the hearer can infer the missing information. In the same way, certain ambiguities become permissible due to shared common ground (Krauss, 1987). The role of the interlocutionary context is therefore also of importance in this process. Since it is assumed in the

[2] As noted in Porzel and Gurevych (2002), current natural language understanding systems need a systematic way of including situational factors, e.g., the actual accessibility of goal objects has been shown to be a deciding contextual factor determining whether a given `Where interrogative` at hand is *construed* as an instructional or a descriptive request.

SMARTKOM context, that general user model information is supplied via external sources, e.g., via a user's *SmartCard*, only the interaction preferences of the users are monitored actively by the system.

4 Decontextualization in SmartKom

In line with our proposal stated above to separate domain and application knowledge, the implementation within the SMARTKOM system exhibits a clear distinction between domain-specific knowledge and application-specific knowledge. This is consequently mirrored by respective modules: the domain and situation model (each can be addressed via separate blackboards/communication pools) implemented in a module called *modeler.knowledge*, and the function model, implemented in a module called *modeler.function*. This module can be described as the module that contains the knowledge of how specific plans are realized (given the actual software agents, databases and hardware devices). It therefore can be regarded as a translator between representations coming from the NLP and knowledge system and those of the background system.

4.1 The Modeler Knowledge Module

The running module for situational and ontological knowledge receives dynamic spatio-temporal information, e.g., global positioning system (GPS) coordinates and local times as well as (multiple) representations of user utterances in the form of *intention hypothesis* as input. It converts the incoming documents into document object models, on which it operates.[3] After processing, a decontextualized *intention hypothesis* document is returned as output.[4]

The context-dependent tasks performed by the context module implemented in SMARTKOM are:

- to know which information is ontologically required and provide the adequate situational and ontological information
- to detect situationally appropriate readings
- to compute contextual coherence scores for alternative intention hypotheses

4.2 Modeler Knowledge at Work

The first and foremost function is to add situation-specific discourse and dialogue knowledge. For example, no agent can check room vacancies without knowing the

[3] See http://www.w3c.org/DOM for the specification.

[4] The modeler.knowledge module features additional task- and domain-independent functionalities to probe and manipulate and compute on the ontology (http://www.w3c.org/OWL) as well as on the schema hierarchy (http://www.w3c.org/XMLS), the dynamic respective situational data and the static database information.

arrival date and duration of the intended stay, neither can a theater agent reserve tickets without knowing the seat numbers, etc. In human–human dialogues this knowledge is responsible for determining relevant answers to given questions. Consider the following exchange given in examples (3) and (4), where additional turns, asking the user to specify time and place, are avoided by decontextualizing the question and providing corresponding answers.

(3) *Was läuft im Kino ?*
What runs in the cinema ?

(4) *Hier sehen Sie was heute in den Heidelberger Kinos läuft .*
Here see you what today in the Heidelberg Cinema runs .

Table 2. Context-specific insertions into a sample intention hypothesis resulting from the intepretation of a speech recognition hypothesis

```
<informationSearchProcess>
  <entertainment>
    <performance>
      <cinema>
        <contact>
          <address>
            <town>
              here
            </town>
          </address>
        </contact>
      </cinema>
      <time>
        <beginTime>
          <at>
            now
          </at>
        </beginTime>
      </time>
    </performance>
  </entertainment>
</informationSearchProcess>
```

```
<contact>
  <x> 70.345 < /x>
  <y> 49.822 < /y>
  <town>
    Heidelberg
  </town>
</contact>
<time>
  <at> 19:00:00T26:08:03 </at>
</time>
```

```
<scores>
  <contextualCoherence>
    0.46
  </contextualCoherence>
</scores>
```

The SMARTKOM context model enables the system to act analogously, i.e., to provide hitherto implicit knowledge concerning what is talked about. The simplified structures given in Table 2 show insertions (in boldface) into an intention hypothesis made by the model in the case of a question such as that given in example (3). In this case, the insertions made via contextual knowledge are threefold:

- For the cardinally required time and place slot in the performance object respective defaults are inserted.

- These indexical defaults are contextually resolved[5] by means of accessing a GPS. This information can be used to resolve here with an appropriate level of granularity, e.g., town or spatial points, by means of a geographic information system in much the same way as today will also be replaced with granularity-specific temporal information, e.g., year, date or time.
- A contextual score for each hypothesis is computed indicating the contextually most adequate reading, as SMARTKOM processes *intention lattices* consisting of several intention hypotheses.

By means of explicating such information and providing topical and contextually adequate values, the system can retrieve appropriate information from Web sites or databases on what is currently playing in town, produce maps featuring cinema locations and then offer further assistance in navigation or reservation, for example.

We have therefore linked the context model to interfaces providing contextual information. For example, within both the SMARTKOM and the DEEP MAP framework (Malaka and Zipf, 2000), a database called the *Tourist-Heidelberg-Content Base* supplies information about individual objects including their opening and closing times.[6] By default, objects with no opening times, e.g., streets, can be considered always to be open. A GPS built into the mobile device supplies the current location of the user, which is handed to the geographic information system that computes, among other things, the respective distances and routes to the specific objects. It is important to note that this type of context monitoring is a necessary prerequisite for context-dependent analysis.

5 Application and Evaluation in SmartKom

5.1 Data and Annotations

We collected two types of data for demonstrating and realizing context-dependent effects in the SMARTKOM scenarios. First, we collected field data, by asking SMARTKOM-specific questions of pedestrians on the street and tracking the situational context factors and responses. The logged and classified field data was then used to train classifiers for recognizing specific intentions based on contextual factors. In a previous study another corpus of questions was collected and annotated in terms of their underlying intentions and was turned into a "gold standard" (Porzel and Gurevych, 2002). Second, we collected laboratory data, i.e., dialogues in *Hidden-Operator* and *Wizard-and-Operator* experiments (Rapp and Strube, 2002; Gelbart et al., 2006; Porzel and Baudis, 2004). All utterances were transcribed. Then specific

[5] For example, the topical resolution of *here* and *now* - enable the system to produce a suitable response, such as retrieving a map of the cinemas of Heidelberg and the specific performances. Therefore here and today constitute *placeholders* for defaults that are replaced almost immediately with actual values by the situation model or discourse model.

[6] Additional information extraction agents are able to gather data and information from the web, using ontological translators and updating the local database.

sets of the audio files were sent to the speech recognizer. We logged the speech recognition hypothesis (SRH), *n*-best lists of SRHs and the module's input and output for all utterances.

Using the laboratory data we created specific corpora for annotation experiments. In a first set of annotation experiments on a corpus of 1300 SRHs, the SRHs were annotated within the discourse context, i.e., the SRHs were presented in their original dialogue order. For each SRH, a decision had to be made whether it was semantically coherent or incoherent with respect to the best SRH representing the previous user utterance. In a second experiment the annotators saw the SRHs together with the transcribed user utterances. The task of annotators was to determine the best SRH from the *n*-best list of SRHs corresponding to a single user utterance. The decision had to be made on how well the SRH expressed the intentional content of the utterance (Porzel et al., 2003a). In the first experiment the interannotator agreement was 80%, and in the second 90%. Last, the annotators had to create corresponding gold standards by means of discussing the cases of disagreement until an optimal solution was found.

5.2 Evaluation in SmartKom

For evaluating the performance of the model described above we computed the task-specific accuracies as compared to the gold standards described above. The situational models trained on the field data of 366 subjects using a c4.5 machine learning algorithm (Winston, 1992) achieved an intention recognition accuracy of 88% as compared to baseline achieved by a context-insensitve model of 59% evaluated against the annotated gold standard of a corpus of dialogues with 50 subjects featuring various kinds of spatial interrogatives (Porzel and Gurevych, 2002).

For evaluating the contextual coherence scores of the model we logged the scores of all scoring modules (speech recognizer, parser and discourse model) that rank *n*-best lists of speech recognition hypotheses produced out of word graphs (Oerder and Ney, 1993) and those that rank the representations produced by the parser (Engel, 2002). As described above, these speech recognition hypotheses were annotated in terms of their coherence, correctness and *bestness* and turned into corresponding gold standards (Gurevych et al., 2002; Porzel et al., 2003a).

For computing contextual coherence in the evaluation, the module employed three knowledge sources, an ontology of about 730 concepts and 200 relations and a lexicon (3600 words) with word-to-concept mappings, covering the respective domains of the system and a conceptual dialogue history, including the concepts and relations of the previous best hypothesis. The final evaluation was carried out on a set of 95 dialogues. The resulting dataset contained 552 utterances resulting in 1.375 SRHs, corresponding to an average of 2.49 SRHs per user utterance.

The task of hypothesis verification, i.e., finding out what was said, in our multimodal dialogue system is to determine the best SRH from the *n*-best list of SRHs corresponding to a given user utterance. The baseline for this evaluation was the overall chance likelihood of guessing the best one, i.e., 63.91%.

The context- and knowledge-based system (Gurevych et al., 2003a; Porzel et al., 2003a) achieves an accurracy of 88%.[7] The knowledge-based system without the dialogical context features already exceeds that of the acoustic and language model scores produced by the automatic speech recognizer, reaching 84.06% on the same task.

The evaluation of the contextual coherence scoring in terms of its disambiguation performance meant calculating how often contextual coherence picks the appropriate reading, given an ambiguous lexicon entry such as *kommen* associated in the lexicon with both WatchPerceptualProcess and MotionDirectedTransliterated. For this evaluation we tagged 323 lemma with their contextually appropriate concept mappings, and achieved an accuracy of 85% given an aggregate majority class baseline averaged over the majority class baselines of each individual lemma of 42%.

6 Conclusion

The basic intuition behind explicating contextual dependencies originally proposed by McCarthy (1986) was that any given axiomatization of a state of affairs, meanings or relations presupposes an implicit context. Any explicit context model employed in processing information, therefore, needs to provide the information why specific meaning can be assigned to the underspecified information and, thus be, applied to its processing. This has often been called *fleshing out* and was considered impossible in its maximal form, e.g., Akman and Surav (1996) state that:

> It is seen that for natural languages a fleshing-out strategy — converting everything into decontextualized eternal sentences — cannot be employed since we do not always have full and precise information about the relevant circumstances.

Herein, we have presented a context model that performs a set of *fleshing out* tasks, which, as the successful evaluations show, suffice to enable a multicontext system, such as SMARTKOM, to respond felicitously to contextually underspecified questions. We have developed a corresponding system that integrates domain, dialogue and situative context in a multidomain, multiscenario and multimodal dialogue system. We have shown how:

- this knowledge can be used for improving the speech recognition reliability in the case of hypothesis verification, i.e., for finding out what was said
- this knowledge can be used to explicate contextually implicit information, i.e., for resolving indexical expressions
- this knowledge can be used to resolve context-dependent ambiguities, i.e., for lexical and pragmatic disambiguation

[7] This means that in 88% of all cases the best SRH defined by the human *gold standard* is among the best scored by the module.

We have therefore demonstrated that the inclusion of such contextual interpretation in natural language processing can enable natural language understanding systems to become more conversational without losing the reliability of restricted dialogue systems. Given the challenge to extract the underlying intentions from conversational utterances such as "is there a bakery close by" or "I don't see any bus stops", we presented the necessary knowledge stores and inferential capabilites necessary for their decontextualization, which is a prerequisite for understanding utterances and responding felicitously. This enables us to restate McCarthy's original claim to say that for natural languages a fleshing out strategy can be employed if we have sufficient and precise knowledge about the relevant contextual circumstances.

Acknowledgments

This work was partially funded by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the SMARTKOM project under Grant 01 IL 905 K7 and by the Klaus Tschira Foundation. The authors would like to thank Ralf Panse, Christof Müller and Hans-Peter Zorn for their valuable implementation and evaluation work.

References

V. Akman and M. Surav. Steps Toward Formalizing Context. *AI Magazine*, 17(3): 55–72, 1996.

J. Alexandersson and T. Becker. The Formal Foundations Underlying Overlay. In: *Proc. 5th Int. Workshop on Computational Semantics (IWCS-5)*, pp. 22–36, Tilburg, The Netherlands, February 2003.

J.F. Allen. *Natural Language Understanding*. Benjamin/Cummings, Menlo Park, CA, 1987.

J.F. Allen, L.K. Schubert, G. Ferguson, P. Heeman, C.H. Hwang, T. Kato, M. Light, N. Martin, B. Miller, M. Poesio, and D. Traum. The TRAINS Project: A Case Study in Building a Conversational Agent. *Journal of Experimental and Theoretical AI*, 7:7–48, 1995.

H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The Philips Automatic Train Timetable Information System. *Speech Communication*, 17:249–262, 1995.

C.F. Baker, C.J. Fillmore, and J.B. Lowe. The Berkeley FrameNet Project. In: *Proc. COLING-ACL'98*, pp. 86–90, Montreal, Canada, 1998.

N. Beringer, U. Kartal, K. Louka, F. Schiel, and U. Türk. PROMISE: A Procedure for Multimodal Interactive System Evaluation. In: *Proc. Workshop on Multimodal Resources and Multimodal Systems Evaluation*, pp. 77–80, Las Palmas, Spain, 2002.

H. Bunt. Dialogue Pragmatics and Context Specification. In: H.C. Bunt and W.J. Black (eds.), *Computational Pragmatics, Abduction, Belief and Context; Studies in Computational Pragmatics*, pp. 81–150. John Benjamins, Amsterdam, 2000.

M.J. Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In: *Proc. 34th ACL*, pp. 184–191, Santa Cruz, CA, 1996.

J.H. Connolly. Context in the Study of Human Languages and Computer Programming Languages: A Comparison. *Modeling and Using Context*, 2116:116–128, 2001.

C. Ebert, S. Lappin, H. Gregory, and N. Nicolov. Generating Full Paraphrases of Fragments in a Dialogue Interpretation System. In: *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*, pp. 58–67, Aalborg, Denmark, September, 1-2 2001.

R. Engel. SPIN: Language Understanding for Spoken Dialogue Systems Using a Production System Approach. In: *Proc. ICSLP-2002*, pp. 2717–2720, Denver, CO, 2002.

G. Ferguson and J.F. Allen. TRIPS: An Intelligent Integrated Problem-Solving Assistant. In: *Proc. 15th Nat. Conf. on Artificial Intelligence & 10th Conf. on Innovative Applications of Artificial Intelligence*, pp. 567–573, Madison, WI, 1998.

A. Flycht-Erriksson. A Survey of Knowledge Sources in Dialogue Systems. In: *Proc. IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pp. 41–48, 1999.

F. Gallwitz, M. Aretoulaki, M. Boros, J. Haas, S. Harbeck, R. Huber, H. Niemann, and E. Nöth. The Erlangen Spoken Dialogue System EVAR: A State–of–the–Art Information Retrieval System. In: *Proc. 1998 Int. Symposium on Spoken Dialogue (ISSD 98)*, pp. 19–26, Sydney, Australia, 30 November 1998.

M. Gavaldá and A. Waibel. Growing Semantic Grammars. In: *Proc. COLING-ACL'98*, pp. 451–456, Montreal, Canada, 1998.

D. Gelbart, J. Bryant, A. Stolcke, R. Porzel, M. Baudis, and N. Morgan. SmartKom English: From Robust Recognition to Felicitous Interaction, 2006. In this volume.

Y. Gil and V. Ratnakar. A Comparison of (Semantic) Markup Languages. In: *Proc. 15th Int. FLAIRS Conf.*, pp. 408–412, Pensacola, FL, 2002.

A.L. Gorin, G. Riccardi, and J.H. Wright. How May I Help You? *Speech Communication*, 23(1/2):113–127, 1997.

I. Gurevych, R. Malaka, R. Porzel, and H.P. Zorn. Semantic Coherence Scoring Using an Ontology. In: *Proc. Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting 2003*, pp. 88–95, Edmonton, Canada, 2003a.

I. Gurevych, R. Porzel, and R. Malaka. Modeling Domain Knowledge: Know-How and Know-What, 2006. In this volume.

I. Gurevych, R. Porzel, E. Slinko, N. Pfleger, J. Alexandersson, and S. Merten. Less Is More: Using a Single Knowledge Representation in Dialogue Systems. In: *Proc. HLT-NAACL'03 Workshop on Text Meaning*, pp. 14–21, Edmonton, Canada, May 2003b.

I. Gurevych, R. Porzel, and M. Strube. Annotating the Semantic Consistency of Speech Recognition Hypotheses. In: *Proc. 3rd SIGdial Workshop on Discourse and Dialogue*, pp. 46–49, Philadelphia, PA, July 2002.

E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. Toward Semantics-Based Answer Pinpointing. In: *Proc. 1st Int. Conf. on Human Language Technology*, 2001.

M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An Architecture for Multimodal Dialogue Systems. In: *Proc. 10th ACM Int. Symposium on Advances in Geographic Information Systems*, pp. 376–383, Washington, DC, 2002.

R. Krauss. The Role of the Listener: Addressee Influences on Message Formulation. *Journal of Language and Social Psychology*, 6:91–98, 1987.

M. Löckelt, T. Becker, N. Pfleger, and J. Alexandersson. Making Sense of Partial. In: C.M. Johan Bos, Mary Ellen Foster (ed.), *Proc. 6th Workshop on the Semantics and Pragmatics of Dialogue (EDILOG 2002)*, pp. 101–107, Edinburgh, UK, September 2002.

R. Malaka and A. Zipf. Deep Map: Challenging IT Research in the Framework of a Tourist Information System. In: D.R. Fesenmaier, S. Klein, and D. Buhalis (eds.), *Proc. 7th. Int. Conf. on Information and Communication Technologies in Tourism (ENTER 2000)*, pp. 15–27, Berlin Heidelberg New York, 26–28 April 2000. Springer.

J. McCarthy. Notes on Formalizing Contexts. In: T. Kehler and S. Rosenschein (eds.), *Proc. 5th Nat. Conf. on Artificial Intelligence*, pp. 555–560, Los Altos, CA, 1986. Morgan Kaufmann.

J. McCarthy. Generality in Artificial Intelligence. In: V. Lifschitz (ed.), *Formalizing Common Sense: Papers by John McCarthy*, pp. 226–236, Norwood, NJ, 1990. Ablex Publishing Corporation.

D. Moldovan, M. Pasca, S. Harabagiu, and M. Surdeanu. Performance Issues and Error Analysis in an Open-Domain Question Answering System. In: *Proc. 40th ACL*, pp. 33–40, Philadelphia, PA, 2002.

G. Nunberg. *The Pragmatics of Reference*. Indiana University Linguistics Club, 1987.

M. Oerder and H. Ney. Word Graphs: An Efficient Interface Between Continuous-Speech Recognition and Language Understanding. In: *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-93)*, vol. 2, pp. 119–122, Minneapolis, MN, 1993.

R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.G.E. Levin, C. Lee, and J. Wilpon. A Speech Understanding System Based on Statistical Representation of Semantics. In: *Proc. Int. Conf. on Acoustics Speech Signal Processing*, San Francisco, CA, 1992.

R. Porzel and M. Baudis. The Tao of CHI: Towards Felicitous Human-Computer Interaction. In: *Proc. Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting 2004*, Boston, MA, 2004.

R. Porzel and I. Gurevych. Towards Context-Adaptive Utterance Interpretation. In: *Proc. the 3rd SIGdial Workshop on Discourse and Dialogue*, pp. 90–95, Philadelphia, PA, 2002.

R. Porzel and I. Gurevych. Contextual Coherence in Natural Language Processing. In: P. Blackburn, C. Ghidini, R. Turner, and F. Giunchiglia (eds.), *Modeling and Using Context*, LNAI 2680, Berlin Heidelberg New York, 2003. Springer.

R. Porzel, I. Gurevych, and C. Müller. Ontology-Based Contextual Coherence Scoring. In: *Proc. 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, July 2003a.

R. Porzel, N. Pfleger, S. Merten, M. Loeckelt, I. Gurevych, R. Engel, and J. Alexandersson. More on Less: Further Applications of Ontologies in Multi-Modal Dialogue Systems. In: *Proc. 3rd IJCAI 2003 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, 2003b.

R. Porzel and M. Strube. Towards Context Adaptive Natural Language Processing Systems. In: M. Klenner and H. Visser (eds.), *Proc. Int. Symposium: Computional Linguistics for the New Millenium*, pp. 141–156. Peter Lang, Berlin, 2002.

J. Prager, D. Radev, and K. Czuba. Answering What-Is Questions by Virtual Annotation. In: *Proc. 1st Int. Conf. on Human Language Technology*, 2001.

S. Rapp and M. Strube. An Iterative Data Collection Approach for Multimodal Dialogue Systems. In: *Proc. 3rd Int. Conf. on Language Resources and Evaluation (LREC 2002)*, pp. 661–665, Las Palmas, Spain, 2002.

S. Rapp, S. Torge, S. Goronzy, and R. Kompe. Dynamic Speech Interfaces. In: *Proc. Workshop Artificial Intelligence in Mobile Systems, ECAI 2000*, Berlin, Germany, 2000.

S.J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

W. Wahlster. SmartKom: Fusion and Fission of Speech, Gestures, and Facial Expressions. In: *Proc. 1st Int. Workshop on Man-Machine Symbiotic Systems*, pp. 213–225, Kyoto, Japan, 2002.

W. Wahlster. SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: R. Krahl and D. Günther (eds.), *Proc. Human Computer Interaction Status Conference 2003*, pp. 47–62, Berlin, Germany, June 2003. DLR.

M.A. Walker, C.A. Kamm, and D.J. Litman. Towards Developing General Model of Usability with PARADISE. *Natural Language Engineering*, 6, 2000.

D. Widdows. A Mathematical Model of Context. In: P. Blackburn, C. Ghidini, R. Turner, and F. Giunchiglia (eds.), *Modeling and Using Context*, LNAI 2680, Berlin Heidelberg New York, 2003. Springer.

P.H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1992.

Intention Recognition

Jürgen te Vrugt and Thomas Portele

Philips Research Laboratories GmbH, Aachen, Germany
{Juergen.te.Vrugt,Thomas.Portele}@philips.com

Summary. The intention recognition module identifies the analyzed representation of the user input to the SMARTKOM system that best represents this input in a collection of possible representations. The alternative representations are generated by recognition and analysis components, being enriched with knowledge, e.g., from the discourse and context. A probabilistic model combines various scores, based on features in the representations and computed by the SMARTKOM modules during processing to support the selection. The parameters inside the model are optimized on annotated data that has been collected using the SMARTKOM system using both a parameter study and a rank based estimation algorithm.

1 Introduction

The intention recognition in SMARTKOM (Wahlster et al., 2001) is part of the multimodal interaction subproject. In this working area the results of the analysis components for the various modalities are brought together, the user input is embedded into discourse and world context and the most likely interpreted reading of the user input has to be chosen. SMARTKOM has to react in an appropriate way. Reactions presented to the user are initiated by components located in this subproject, and the interaction of SMARTKOM with applications is also organized here. Due to the heterogeneous orientation of SMARTKOM into the scenarios "Home", "Mobile" and "Public", the components of the subproject must be easy to configure, independent of certain applications, and robust.

This chapter deals with the identification of the user's intention. This is the computation of the most likely interpreted reading of the input provided by the user. The first part of this chapter describes how the intention recognition fits into SMARTKOM. The next part deals with the basic model for the combination of ratings from various sources. The following part is concerned with how data were collected and prepared in order to compute the parameters for the combination of the ratings, and finally the computation of the parameters is addressed.

2 Intention Recognition in the SmartKom Context

The decisions of the intention recognition are strongly depending on the preceding components in the information processing chain. On the other hand, the decision taken by the intention recognition has a strong influence on the actions executed and reactions toward the user.

2.1 Intention Recognizers' environment

Figure 1 shows the context of the intention recognition module in the SMARTKOM system as it is used in the three scenarios.

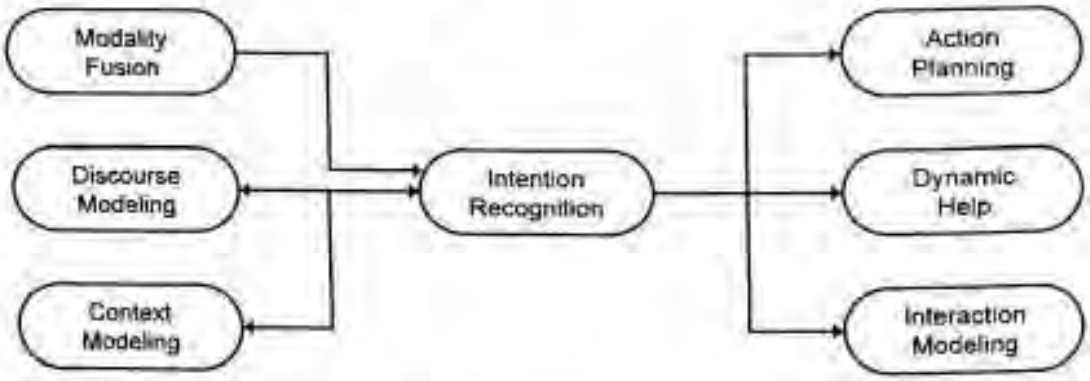


Fig. 1. Section of the multimodal interaction from the intention recognition point of view

2.1.1 Input to the Intention Recognition Module

The input of a user is initially processed by recognizers like a speech recognition or a gesture recognition component. Analyzers for the recognizers produce a semantic interpretation of the monomodal recognition results. For example, such a result is the interpretation of a grammar applied to the speech recognition result, or the representation of an element presented on the screen which is likely to be addressed by the user. Inside some of these components, stochastic methods are applied in order to compute readings of the respective input. Instead of computing one single "best" result, a number of hypotheses with a rating of the quality of each hypothesis might be computed.

The interpretations as produced by the monomodal recognition and analysis components are combined in the multimodal fusion. This results in a number of hypotheses for the multimodal input based on the monomodal hypotheses generated by the single recognition engines. The result of the multimodal fusion serves as the basis for the computation of the user intention. More details on the fusion can be found in Engel and Pfleger (2006).

The context model in SMARTKOM is divided into a model for the discourse (Pfleger et al., 2003) and a model for parts of the surrounding world (Gurevych et al., 2003). Before the intention recognition takes a decision on the set of hypotheses for the user input, the context models are requested to enrich the information in the hypotheses with context knowledge. These enrichments are evaluated by the discourse model and the context model to rate the quality of the augmentation of a hypothesis with knowledge from the discourse and the surrounding world, and to support the decisions to be taken by the intention recognition.

2.1.2 Processing of the Selected Hypothesis

After selecting the interpretation of the user input which is considered to be the best matching one, this interpretation is provided for further processing mainly to three components:

- The action planning takes decisions on the applications to contact and the next interaction with the user based on the hypothesis provided by the intention recognition.
- The dynamic help analyzes the hypothesis from the intention recognition to support the user in the interaction with the SMARTKOM system, e.g., the dynamic help identifies turns where the user asks for guidance by the system.
- The interaction model computes information on the user from the hypothesis reflecting the user input. This could for example be used to adapt the interaction style of the system to the interaction style of the user.

A more detailed description of these components can be found in the work of Löckelt (2006), Streit (2006) and Streit et al. (2006)

2.2 Communication with Surrounding Components

The communication of the modules in the "Modality Analysis And Fusion" part of SMARTKOM is based on intention lattices.[1] Initially, the intention lattice format coded the information transfered by the modules completely. This structure was separated into a hypothesis-related part and the content-related part. This separation was inspired by the introduction of an ontology in the context-modeling module. From this ontology, the event structure is derived. More details can be found in Porzel et al. (2006)

Basically, the intention lattice format contained the possibility to integrate a number of hypotheses (or — to be be more precise — hypothesis sequences) into the data exchanged by the modules. Each of these hypothesis sequences itself consists of one or more atomic hypotheses. The input of a user might be segmented into independent units, each unit represented by an atomic hypothesis. An example for such input with a possible reading using two atomic hypotheses in the "Home" scenario is a device control command followed by a database request like *"Turn on the TV and show*

[1] The specification of the interface is realized in M3L, see Herzog and Reithinger (2006).

me the movies of this evening!". While a collection of atomic hypotheses forms a hypothesis sequence, the hypothesis sequences stand for alternative readings of one and the same user input.
While the atomic hypotheses carry information on the content of the user input, the hypothesis themselves are equipped with technical information. Prominent examples are the ratings of the modules.

3 Internal Model

In the early stages of the SMARTKOM project, the development of the various models[2] had just started, or existing models had to be adapted for the use in SMARTKOM. The selection of the hypothesis considered to be the best matching one was mainly influenced by rule-based decisions. After more and more ratings showed up in the various modules, the rule-based decision was replaced by a rule-based preprocessing[3] and a stochastic model.

To illustrate the stochastic model motivated by Young (2000), the following nomenclature is used: The set of hypotheses generated by the modules preceding the intention recognition is denoted by $\mathcal{H} = \{h_1, \ldots, h_n\}$. The set $\mathcal{H}$ contains possible interpretations of one user input. Capital letters are used for random variables, where H is used for the hypothesis under consideration, and I_{ASR}, I_G, I_P and I_M are the input to the speech recognition, gesture recognition, prosody recognition and mimic recognition, respectively, provided by a user.

The primary role of the intention recognition is to identify the hypothesis $\hat{h}$ that is the best match of the given user input u:

$$\hat{h} = \text{argmax}_{h \in \mathcal{H}} P(H = h | U = u), \tag{1}$$

with $U = (I_{ASR}, I_G, I_P, I_M)$. If D, C and F denote the preliminary hypothesis created by the discourse model, context model and multimedia fusion, respectively, the distribution $P(H|U)$ can be rewritten to

$$\begin{aligned} P(H = h | U = u) = \sum_c & P(H = h | U = u, C = c) \\ & \sum_d P(C = c | U = u, D = d) \\ & \sum_f P(D = d | U = u, F = f) P(F = f | U = u). \end{aligned}$$

Assuming that the result of the intention recognition, discourse model and context model depends only on the preceding models and not directly on the user input U, this simplifies to

[2] Therefore, the realization of the models in SMARTKOM modules was also in an embryonic stage.

[3] Requirements for such a rule-based preprocessing have been formulated. The development of a proprietary language was discarded in favor of the upcoming XSL Transformations, which could easily be applied to the M3L documents (World Wide Web Consortium (W3C) / XSL Working Group).

$$P(H = h|U = u) = \sum_c P(H = h|C = c) \sum_d P(C = c|D = d) \sum_f P(D = d|F = f)P(F = f|U = u).$$

Since hypotheses in SMARTKOM are not recombined,[4] for a given hypothesis h for a given user input u there exists exactly one processing result of context model c, discourse model d and media fusion f with positive probability for all distributions. This leads to

$$P(H = h|U = u) = P(H = h|C = c)P(C = c|D = d) P(D = d|F = f)P(F = f|U = u)$$

for an outstanding sequence (c, d, f), depending on the hypothesis under consideration h and the user input u. If W, S, G, P and M denote the result of the speech recognition, speech understanding, gesture recognition, prosody recognition and mimic recognition component, respectively, it can be derived, that:

$$\begin{aligned}
P(F = f|U = u) &= \sum_{s',g',p',m'} P(F = f|S = s', G = g', P = p', M = m', U = u) \\
&\quad P(S = s', G = g', P = p', M = m'|U = u) \\
&= \sum_{s',g',p',m'} P(F = f|S = s', G = g', P = p', M = m') \\
&\quad \sum_{w'} P(S = s'|W = w')P(W = w'|I_{ASR} = i_{ASR}) \\
&\quad P(G = g'|I_G = i_G)P(P = p'|I_P = i_P) \\
&\quad P(M = m'|I_M = i_M) \\
&= P(F = f|S = s, G = g, P = p, M = m) \\
&\quad P(S = s|W = w)P(W = w|I_{ASR} = i_{ASR}) \\
&\quad P(G = g|I_G = i_G)P(P = p|I_P = i_P)P(M = m|I_M = i_M)
\end{aligned}$$

by using the assumptions:

- The result of the multimodal fusion depends only on the output of the recognition and analysis components (and not on the acoustic input, for example)
- The computations of the results of the recognition components rely only on the input for these components and are independent of the input for other components (e.g., the recognition of speech is independent of mimics)
- The result of the speech understanding unit depends only on the words recognized by the speech recognition engine and not on the acoustic input
- Each result of the fusion is created with positive probability by exactly one input constellation (s, g, p, m) and recognized word sequence w.

[4] That is, if one of these models decides to create more than one different hypothesis out of one hypotheses, these will be regarded as different hypothesis for the rest of the processing chain up to the intention recognition.

Putting it all together, this results in a model of the form:

$$P(H|U) = \prod_{\text{model } M} P_M(\cdot), \tag{2}$$

P_M being the distribution of model M that might depend on other models. The problem of finding the hypothesis that maximizes $P(H|U)$ can be transferred into the equivalent problem of maximizing scores, for example by applying the logarithm. That is,

$$S(H|U) = \sum_{\text{model } M} S_M(\cdot). \tag{3}$$

The measures used in the SMARTKOM modules differ a great deal, e.g., some modules use stochastic models while other modules create a rating based on counts of features. Some models might not fulfill the requirements of a mathematical measure at all, which is a prerequisite of a probability distribution (Billingsley, 1995). The ratings used in the intention recognition are finite, so these ratings could at least be normalized.

Taking this into consideration and denoting the rating of module M by R_M, the intention recognition combines the various models in SMARTKOM by a weighted sum of scaled ratings:

$$R_{\text{total}}(\cdot) = \sum_{\text{model } M} w_M f_M(R_M(\cdot)), \tag{4}$$

where w_M is the weight of model M, $f_M(\cdot)$ scales the ratings $R_M(\cdot)$ of model M. If necessary, $f_M(\cdot)$ renormalizes the rating of model M and transfers the ratings into the score space.[5] The introduction of weight parameters becomes necessary because of the approximations made in this model and since the ratings of the models might not be probability distributions at all.

While the model introduced in this section tries to identify the most likely hypothesis based on the user input and the preceding processing stages, the probabilistic model in Fleming and Cohen (2001) formulates benefits and costs to decide on actions to take in the interaction with the user. Another example of probabilistic dialogue management can be found in Roy et al. (2000), where the decisions are based on (partially observable) Markov decision processes.

4 Data Acquisition for Optimal Model Combinations

The computation of the hypothesis that describes the user input optimally is coupled to the model in Eq. (4). Initial setups and adjustments[6] of the parameters $(w_M)_{\text{model } M}$ have been done manually, but in order to reliable adjust the parameters, a data collection has been carried out.

Due to speed constraints, in the standard setup of SMARTKOM the intention recognition gets less than five hypotheses, usually two to three. The configurations

[5] In case of a probability distribution, f_M is the logarithm.

[6] For example, when new measures have been introduced.

of the speech recognition module and the speech understanding module have been modified for the data collection in order to obtain a large number of alternative hypotheses per user input.

The data collection was carried out in three steps that will be outlined in more detail below. The recordings focused on speech recognition as input modality. Speech was the first modality available for interaction with the SMARTKOM system.

4.1 Collection

In order to ensure sufficient coverage of the user input by the system (especially by the speech analysis grammar), a small data collection was carried out at Philips Research, Germany.Because of the modified settings for the speech recognition engine and the speech understanding component, it was not possible to collect run-time data from real interactions with the system. The resulting number of hypotheses led to a significant decrease of system performance in terms of processing speed. Therefore a number of exemplary interactions of a user with the system have been designed. These interactions are based on the examples provided by the SMARTKOM "Home" scenario managers and online interactions with the system (Portele et al., 2003). Two generic and seven "Home"-specific interactions that can be combined in different ways have been formulated. These interactions contain around 80 user-turns. Table 1 summarizes the data collection.

The decision to restrict the user utterances to a fixed set was triggered by the need of a basic grammatical coverage for these utterances, i.e., utterances the system cannot understand are worthless for the computation of optimal combination parameters. Utterances of five different persons have been recorded; four of them resulted in transcribed intentional structures. A close-talk microphone in two different positions was used. For the recordings, a simple SMARTKOM system running only the audio input module was used.

Table 1. Recording and transcription statistics

Observation in corpus	Occurrences
Recorded and orthographically transcribed user utterances	931
Recordings containing speech	919
Nonspeech recordings	12
Different formulations total	150
Formulations used once	66
Formulations used 2–10 times	39
Formulations used 11–20 times	42
Formulations used 21–26 times	3
Utterances with transcribed intentional structures	433

4.2 Preparation

After recording data from a user, each user input was fed into a SMARTKOM system containing the processing modules needed to create the input data for the intention recognition module in the context of the discourse so far, and the intention recognition module itself. This includes the recognition and analysis modules and the modules containing the domain and context models. The result of this operation is a set of intentional structures for each user input.

For each user input, the data preparation took place in two steps:

1. For each hypothesis produced, the semantic units created by the speech analysis component are extracted. These semantic units have been manually aligned to parts of the spoken input as annotated during the data collection.
2. For each hypothesis, the alignment of semantic units to text phrases can be used to create possible semantic interpretations from the annotated user input. These artificial interpretations are compared to the real hypothesis created by the system. The result of the comparison was used to resort the hypothesis.
   The sorted list of hypotheses was then presented to a human annotator (together with information on the "real" history in the discourse) who had to choose the best matching hypothesis created by the system given the transcription of the spoken user input. Due to the sorting, the hypothesis the annotator rated the best was, in most cases, contained in one of the top positions.

From this annotated data, equivalent hypotheses[7] are eliminated. The outcome of this procedure is an n-best list of hypotheses for each user input with one marked hypothesis being the best representation of the user input, see also Table 2.

Table 2. Statistics on the 433 transcribed intentional structures

Observation in corpus	Occurrences
Hypotheses per user input	8.9
Intentional structures with two or more hypotheses	221
Hypotheses per user input	16.4
Intentional structures with outstanding hypothesis marked by annotator and at least one alternative hypothesis	158
Hypotheses per user input (equivalent variants removed)	3.8

5 Computation of Combination Parameters

The 433 transcribed intentional structures have been split into a development and test set by taking every kth dataset out of the collected corpus and moving this dataset into the test set. To compute a setup for the parameter set $(w_M)_{\text{model } M}$ for the three

[7] Hypotheses that differ in parts not relevant for the computation of combination weights.

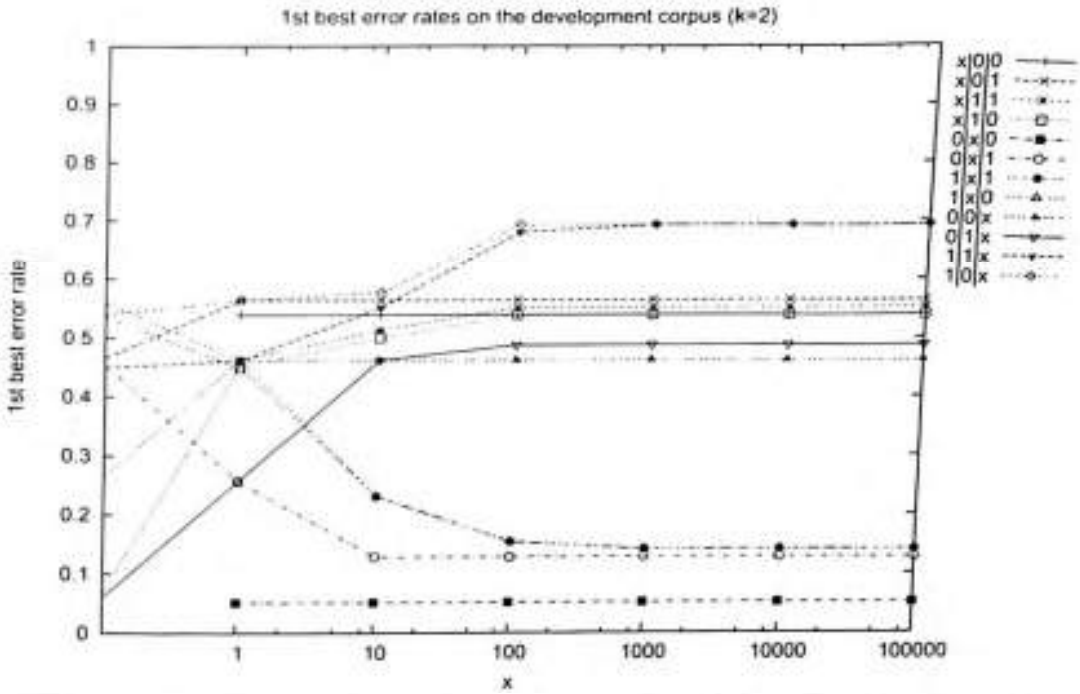


Fig. 2. First-best error rates on the development corpus ($K = 2$; $M_1|M_2|M_3$ denotes the shares of the models in the combination, i.e., model I gets the weight $M_I/(M_1 + M_2 + M_3)$)

models providing a score (speech recognition, speech understanding and discourse model), a parameter study was carried out. To allow an easier computation of optimal combination weights once data has been prepared, an algorithm to automatically estimate the parameters was also developed.

5.1 Parameter Study

Figure 2 allows for a glimpse on the parameter study. It gives a hint in which direction to search for a good parameter setup. From the studies, it turns out that the optimal configuration is a model based purely on the rating from the speech analyzer.[8] The speech analyzer is heavily dependent on the result of the speech recognition, therefore the score of the speech analysis already contains the relevant knowledge from the speech recognition engine. The measure provided by the discourse model was refined during SMARTKOM and might become more relevant. The intention recognition combines knowledge of modules based on different approaches. For example, models based on artificial intelligence compute a rating separately, while modules using a stochastic model have an inherent rating.

From the parameter study it turned out that the best weight combination was stable with respect to the different setups of development and test corpus. Table 3

[8] In the real SMARTKOM system, the number of hypotheses is strongly restricted (mostly 2, maximal 3 hypotheses per user input). Furthermore, these hypotheses are very similar.

Table 3. First-best error rates on development and test corpus for varying splits of the collected corpus

Corpus split parameter k	*Development* Entries in corpus	*Development* Error rate (%)	*Test* Entries in corpus	*Test* Error rate (%)
2	78	5.1	80	7.5
3	100	5.0	58	8.6
4	114	7.0	44	4.5
5	128	7.0	30	3.3
6	126	5.6	32	9.4
7	137	7.3	21	0.0

lists the first-best error rates on these corpora. For comparison, a small parameter study was also carried out on the test corpora. As far as this study can prove, the optimal setup computed on the development corpus was also optimal on the test corpus.

5.2 Automatic Estimation of Parameters

In order to automatically compute an optimal parameter setup $(w_M)_{\text{model } M}$ for the combination model formulated in Eq. (4), an algorithm motivated by rank-based statistics was developed.

5.2.1 Basic Idea

The initial situation for the algorithm is a processing chain that creates a list of n hypotheses describing the processing result for some input that has been fed into this chain (Fig. 3).

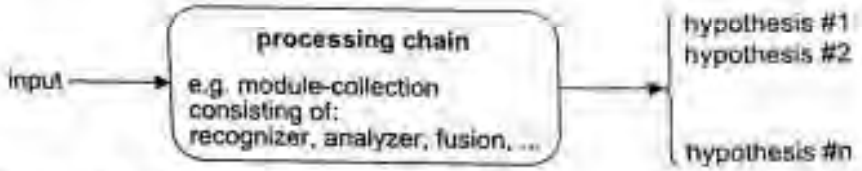


Fig. 3. Prerequisite situation for the automatic computation of the parameters $(W_M)_{\text{Model } M}$ in Eq. (4)

It is now assumed that one of these hypotheses can be regarded to be the "best" result of the processing for the input by the given chain. This hypothesis will be denoted as the true hypothesis in the remaining part of this section. For a fixed set of parameters $(w_M)_{\text{model } M}$ the true hypothesis is placed at a certain position in the n-best list of all hypotheses produced from the input by the processing chain according to the total rating as formulated in Eq. (4). For different sets of parameters $(w_M)_{\text{model } M}$, this

rank of the true hypothesis relative to the competing hypotheses will differ. Given a collection of inputs to the processing chain, this results in a collection of n-best lists of hypotheses.[9] For each n-best list in this collection, one true hypothesis should be identifiable.

The goal of the algorithm is to compute parameter setups $(w_M)_{\text{model } M}$ such that the true hypothesis is positioned at the best possible rank simultaneously for all n-best lists in the collection. Instead of defining a setup $(w_M)_{\text{model } M}$ and computing the resulting quality measure for the simultaneous optimization, the algorithm places the true hypotheses on certain ranks in the n-best lists of the collection and tries to compute parameter setups that place the true hypothesis on exactly these predefined ranks. Such a parameter setup[10] for a given positioning of the true hypothesis may not exist. Considering all configurations of ranks for which a parameter setup exists, the ones with the best quality measure lead to the wanted parameter setups.

A canonical quality measure for the simultaneous optimization of all ranks of the true hypothesis in the collections is the mean rank of the true hypotheses:

$$\text{rank}_{\text{mean}} = \frac{\sum_{n-\text{best lists in collection } l} \text{rank}_{\text{pos of true hypo in list}}(l)}{\#\, n-\text{best lists in collection}}. \tag{5}$$

Of course, the $\text{rank}_{\text{mean}}$ quality measure could be replaced by another quality measure. For example, consider some application where it is necessary to place the true hypothesis within the first k hypotheses while a rank larger than k is neglected during further processing. The exact rank of the true hypothesis does not matter. Then the function $\text{rank}_{\text{pos of true hypo in list}}(\cdot)$ in Eq. (5) might be replaced by the function

$$f_{\text{rank of true hypo} \leq k}(l) = \begin{cases} 1, & \text{if } \text{rank}_{\text{pos of true hypo in list}}(l) \leq k, \\ \textit{penalty}, & \text{else}, \end{cases}$$

with l being a n-best list containing a true hypothesis and *penalty* being some constant, usually larger than 0.

5.2.2 Outline of the Algorithm

Positioning all hypotheses of all n-best lists of a collection at certain ranks in these lists results in a certain arrangement of the true hypothesis. We will call this the configuration of ranks. Figure 4 sketches the basic flow of the search for the optimal parameter setup.

The computation of the quality measure for a given configuration c has been discussed before. It remains to compute a parameter setup for such a configuration c. Let

$$\text{rank}(l, h) \stackrel{\text{def}}{=} \#\left\{h' \in l; \text{rating}(h') > \text{rating}(h)\right\} + 1$$

[9] The value of n might depend on the input.

[10] Depending on the exact formulation of the algorithm, the trivial solution $(w_M)_{\text{model } M} = (0)_{\text{model } M}$ might be a possible solution for all configurations. Therefore only nontrivial solutions are searched for.

```
compute all possible configurations of ranks C

for all configurations c in C:
{
  if { exists a parameter setup for configuration c }
  {
    compute quality measure qm for configuration c

    if { qm is better than qm' of best configuration(s) so far }
    {
      empty the list of best configurations

      put c into list of best configurations
    }
    else if { qm is equal to qm' of best configuration(s) so far }
    { append c to list of best configurations }
    else
    { ignore configuration c }
  }
  else
  { ignore configuration c }
}

return the list of best configurations
```

Fig. 4. Algorithmic framework of search for the optimal parameter setup

denote the rank of hypothesis h in the n-best list l. The rating $R_{\text{total}}(h)$ for hypothesis h is defined by Eq. (4), $R_{\text{total}}(h)$ is a linear polynomial in the variables $(w_M)_{\text{model } M}$. Therefore rank (l,h) depends on $(w_M)_{\text{model } M}$ if rating$(\cdot) = R_{\text{total}}(\cdot)$.

For a certain n-best list l, let h_k $(1 \leq k \leq n)$ be the true hypothesis. Consider the problem of computing a parameter setup such that the n-best list l has the ordering $(h_1, \ldots, h_k, \ldots, h_n)$ (without restrictions: rank $(l,h_k) = k$), i.e.,

$$rank(l,h_i) \begin{matrix} < \\ \leq \end{matrix} rank(l,h_{i+1}), \text{ if } \begin{matrix} i = k-1, \\ 1 \leq i < k-1, \text{ and } k \leq i \leq n. \end{matrix}$$

This can be reformulated by the set of inequalities

$$R_{\text{total}}(h_i) \begin{matrix} > \\ \geq \end{matrix} R_{\text{total}}(h_{i+1}), \text{ if } \begin{matrix} i = k-,1 \\ 1 \leq i < k-1, \text{ and } k \leq i \leq n, \end{matrix}$$

in the variables $(w_M)_{\text{model } M}$. A parameter setup for a configuration c can now be computed by the algorithm described in Fig. 5.

The main disadvantage of the algorithm presented is its complexity: n hypotheses of an n-best list can be ordered in $n!$ ways, therefore the complete collection of n-best lists contains

$$\prod_{n-\text{best lists } l} (\#\text{ hypos in } l)!$$

```
let c be a configuration
let I be an empty set of inequalities in w

for each single n-best list l in c:
{
  create set of inequalites in w from l (see text)

  add inequalities to I
}

if ( exists nontrivial parameter setup w' that solves I )
{ return all nontrivial parameter setups that solve I }
else
{ return NULL }
```

Fig. 5. Computation of nontrivial parameter setups for a given configuration c ($W = (W_M)_{\text{Model } M}$)

possible configurations. For example, in a collection with p lists, each list containing n hypotheses, there are $(n!)^p$ collections.
To reduce the complexity of the algorithm, first improvements of the algorithm have been realized. Among these are

- a grouping of different collections
- the reduction to the problem of computing a solution, where the true solution has *at least* a certain rank (and not exactly such a rank), and
- a repeated application of the algorithm on the probabilistic preprocessed data taking into account the result of the previous application

5.2.3 Results on the In-House Corpus

The algorithm was applied on the collected data presented in Sect. 4. The results on various splits of the corpus conform completely with the results of the parameter study presented in Sect. 5.1. The improved version of the algorithm allowing up to 100 repetitions with preprocessed input data was applied. The algorithm converged to the solution after four repetions, at most. In addition to the parameter study, approaches based on ideas of the downhill simplex methods in multidimensions and a combination of simulated annealing together with the downhill simplex method (Press et al., 2002) verified the outcome of the presented algorithm.

6 Conclusion

This chapter described the role of intention recognition in the SMARTKOM project. It collects information from preceding sources and tries to identify the reading of the analyzed user input that fits best to the user input. The internal model for the

combination of ratings was motivated and derived. To obtain reliable parameters, a data collection was carried out and tools have been provided to the annotator to ease the annotation of the data at the various annotation stages. A parameter study on development corpora led to optimal parameter setups for the combination model. The optimality of these setups was proven on the test corpus by a second parameter study on the test data. The setups were stable with respect to the segmentation of the collected corpus into development and test corpora. The application of the algorithm for the automatic computation of an optimal parameter setup confirmed the results of the parameter study. The complexity of the improved versions of the algorithm needs to be compared to the original algorithm. The application of the algorithm on other data, e.g., to combine acoustic and language model score in a speech recognizer, also looks promising.

Acknowledgments

The authors would like to thank Eva Lasarcyk for supporting us during the data collection and annotation.

References

P. Billingsley. *Probability and Measure*. John Wiley, New York, 3 edn., 1995.

R. Engel and N. Pfleger. Modality Fusion, 2006. In this volume.

M. Fleming and R. Cohen. Dialogue as Decision Making Under Uncertainty: The Case of Mixed-Initiative AI Systems. In: *Proc. NAACL-2001 Adaptation in Dialogue Systems Workshop*, Pittsburgh, PA, 2001.

I. Gurevych, R. Malaka, R. Porzel, and H.P. Zorn. Semantic Coherence Scoring Using an Ontology. In: *Proc. Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting 2003*, pp. 88–95, Edmonton, Canada, 2003.

G. Herzog and N. Reithinger. The SmartKom Architecture: A Framework for Multimodal Dialogue Systems, 2006. In this volume.

M. Löckelt. Plan-Based Dialogue Management for Multiple Cooperating Applications, 2006. In this volume.

N. Pfleger, R. Engel, and J. Alexandersson. Robust Multimodal Discourse Processing. In: *Proc. Diabruck: 7th Workshop on the Semantics and Pragmatics of Dialogue*, Wallerfangen, Germany, 4–6 September 2003.

T. Portele, S. Goronzy, M. Emele, A. Kellner, S. Torge, and J. te Vrugt. SmartKom-Home — An Advanced Multi-Modal Interface to Home Entertainment. In: *Proc. EUROSPEECH-03*, Geneva, Switzerland, 2003.

R. Porzel, I. Gurevych, and R. Malaka. In Context: Integrating Domain- and Situation-Specific Knowledge, 2006. In this volume.

W.H. Press, W.T. Vetterling, S.A. Teukolsky, and B.P. Flannery. *Numerical Recipes in C++ — The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2 edn., 2002.

N. Roy, J. Pineau, and S. Thrun. Spoken Dialogue Management Using Probabilistic Reasoning. In: *Proc. 38th ACL*, Hong Kong, 2000.

M. Streit. Problematic, Indirect, Affective, and Other Non-Standard Input Processing, 2006. In this volume.

M. Streit, A. Batliner, and T. Portele. Emotion Analysis and Emotion Handling Subdialogs, 2006. In this volume.

W. Wahlster, N. Reithinger, and A. Blocher. SmartKom: Multimodal Communication with a Life-like Character. In: *Proc. EUROSPEECH-01*, vol. 3, pp. 1547–1550, Aalborg, Denmark, September 2001.

World Wide Web Consortium (W3C) / XSL Working Group. The Extensible Stylesheet Language Family. http://www.w3.org/Style/XSL. Cited 6 April 2006.

S. Young. Probabilistic Methods in Spoken Dialogue Systems. *Philosophical Transactions of the Royal Society (Series A)*, 358(1769):1389–1402, 2000.

Plan-Based Dialogue Management for Multiple Cooperating Applications

Markus Löckelt

DFKI GmbH, Saarbrücken, Germany
loeckelt@dfki.de

Summary. The SMARTKOM dialogue manager implements the personality of the system and its behaviour. It plans and manages the task-oriented dialogue with the user and coordinates the operations of the applications to reach his goals. It also helps the analysis modules of the system by providing hints about the expected future dialogue.

1 Introduction

Users access the functionality of SMARTKOM by delegating their wishes to Smartakus, the system agent. Smartakus abstracts away the underlying structure of multiple interoperating applications by providing a unified and personalized interface. The dialogue manager module implements the behavior of Smartakus in two aspects: First, it plans and performs the dialogical interaction of Smartakus with the user. Second, it models the applications the system provides and triggers the interactions with the application and device modules needed to give access to the functionality offered by the system. The overall objective is to provide a flexible and convenient way for users to achieve their goals. The knowledge regarding the current state of the dialogue is also used to give the analysis modules feedback that aids in interpreting expected utterances.

Other important aspects for the SMARTKOM dialogue manager are scalability, incrementality, and reusability for other systems. The number of 14 integrated applications is quite large for a multimodal system, and many tasks of SMARTKOM can only be accomplished by concerted interoperation of several applications. The set of applications also was not integrated all at once, but was built up during the course of the project. Thus, to avoid impeding the parallel development of all modules of the system, it was essential to have a working prototype most of the time. As the system cannot be used sensibly without a dialogue manager, it was vital that the addition of new features could be done with minimal disruption. Two main design decisions helped in addressing these issues: taking advantage of the common ontological knowledge representation, and a flexible and extendable communication paradigm treating all communication partners uniformly.

Related Systems

The MATCH system (Johnston et al., 2002) implements one application similar to the route planner in SMARTKOM. It uses several modalities (speech, graphics, and pen input) that can be used alternatively or in combination. Its dialogue manager uses a toolkit similar to TrindiKit. MATCH features expectation rankings and a common representation across modalities.

Other systems derived from TrindiKit (e.g., IBIS (Larsson, 2002), TRIPS (Ferguson and Allen, 1998) and COLLAGEN (Rich and Sidner, 1998) use related approaches. However, most of the time a single application domain is covered at one time, which keeps the number of interacting modules low, and scalability requirements do not have that great an impact. TrindiKit uses a dialogue move engine and a rule-based approach to drive it and to update the *information state* of the engine. Information state distinguishes PRIVATE and SHARED information, but keeps both parts in one module. We chose to split the responsibility for these two parts in SMARTKOM, where the SHARED information is managed by the discourse modeler.

Variants of the dialogue manager were used in the MIAMM system (Reithinger et al., 2004), which allows multimodal access to information using an interface of speech, haptic interaction and graphics; and NaRATo (a system allowing natural-language database access).

2 Architecture

2.1 Communication with Other Modules

As a result of the complexity and concurrency of the system, it does not have a strict serial pipeline of input analysis, processing, and output generation. The dialogue manager is situated at the pivotal point between input analysis and output generation. It has to process input from three modules and four data pools, and it outputs messages going to six different modules (Fig. 1). The main data flow, along the thick arrows in the figure, consists of user input entering the module via the intention recognizer, and being answered by multimodal output for presentation planning. To use the functionality provided by the applications, additional communication is made via the function modeler module and various data pools. When an incoming message cannot be processed (e.g., an utterance was not understood), it is passed on to the dynamic help module, which will try to resolve the problem. Changes in the availability of modalities (e.g., inhibited graphical output in lean-back mode) are effected by communication with the interaction modeler module. Also, the dialogue manager publishes information about the state of the dialogue and its expectations about subsequent user inputs.

2.2 Internal Modular Structure

The dialogue manager module was developed for the SMARTKOM system, but was explicitly designed to be reusable for different dialogue systems. The module consists of two parts: the core dialogue move engine and a part that adapts it to the

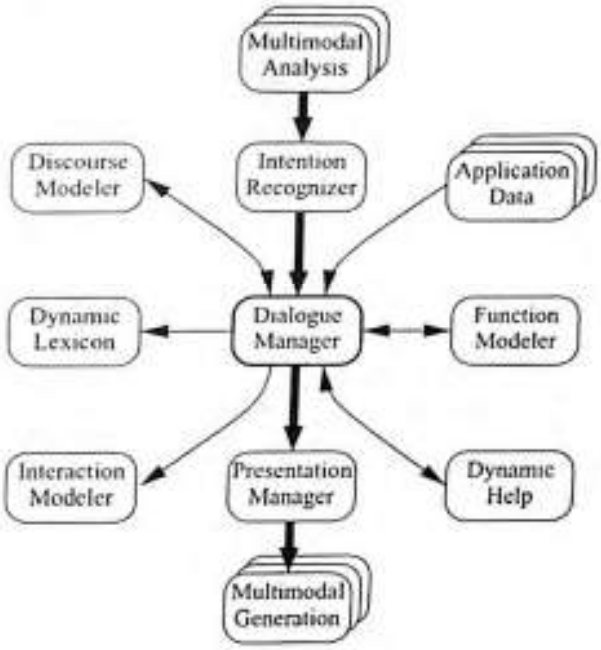


Fig. 1. Modules in direct communication with the dialogue manager

specific circumstances of the system, and also contains the plan knowledge bases for the applications. Their interrelation is shown in Fig. 2.

In the core engine, a planner module is responsible for devising the next actions, which are realized and monitored by the executor. The controller interprets incoming messages, maintains the goal stack, dispatches planning tasks to the planner, and computes the expectations (Sect. 5.3). It also initializes and shuts down the module.

The system-specific part contains data sources giving the plan operator specifications, which together with the process descriptions in the ontology define the behavior for the applications. To convert inbound and outbound messages from and to the data formats needed by the communication partners, modules implementing input and output channels are necessary. The dialogue manager is connected to the communication architecture of the system by a module interface.[1]

2.3 Channels

The various communication partners use different interfaces; however, the core engine of the dialogue manager represents all communications in terms of uniform intention structures, as will be described in Sect. 3.2. Therefore, the intention representation must be translated to and from the message format of the communication partner (except for module-internal communication). For example, when the user is

[1] In SMARTKOM, the Multiplatform blackboard architecture implements intermodule communication (Herzog et al., 2004); for MIAMM, a SOAP facilitator was used instead, while in NaRaTo, methods were called directly.

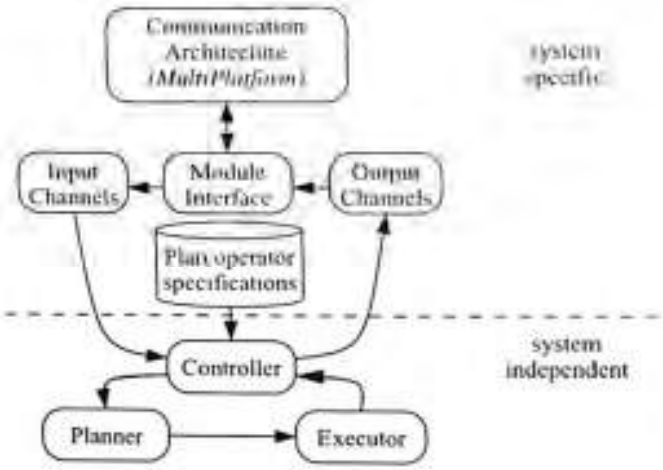


Fig. 2. The internal structure of the module

asked for a telephone number, the request will be represented by a presentation template requesting an ontological role, but the same request has to take the form of a database query if the address book application is addressed.

The communication channels are introduced to abstract away this mapping for the core engine. Each communication link is assigned a channel tailored to the specific message format that has to be dealt with. It then acts as a gateway for the messages, translating them to and from a particular outside representation as they cross the border of the module, inbound or outbound.

3 Action Model for Dialogue Planning

Following the dialogue metaphor underlying the system (Wahlster, 2002), the user delegates his wishes and goals to the animated agent Smartakus, which then collaborates with the user to accomplish the corresponding tasks. The dialogue planner, in determining its actions, realises the "personality" of the system agent. Smartakus was designed to be polite and helpful, but to never appear to be obtrusive. It guides the dialogue and cooperates with the user; however, if the user does not follow the leads strictly in its utterances, Smartakus attempts to accommodate the dialogue and integrate them. It will, however, not risk overinterpreting user utterances when they have not been clearly understood (a well-known cause of human irritation with software assistants). In these cases it will employ the help module instead to ask for clarification. Indicative for insufficient understanding are intentions that lack important information, or have been assigned low confidence scores by the analysis modules.

3.1 The Use of Ontological Processes

This section describes briefly how the dialogue is modeled in terms of processes provided by the SMARTKOM ontology. A more comprehensive description of the ontology representation and its use in our system is found in Porzel et al. (2003).

The domain the dialogue can be about is defined by the ontology underlying the backbone communication, which encompasses some entities to match general dialogue concepts, such as greetings, and others modeling the services offered by the applications of the system. The set of available applications varies across the different scenarios (public, home, and mobile). Some applications are not accessible directly by the user, but support other applications (e.g. the biometrics application). The utterances of the user, after being processed by the analysis modules, reach the dialogue manager in a data structure called *intention*, which contains an ontological description of the utterance content.

The development of the ontology has been closely tied to the development of the dialogue and application capabilities of SMARTKOM. The possible events that occur in the dialogue or the rest of the system are modeled as *processes* with associated *roles* in a frame-based approach. To meet the obligation of satisfying the user, the system has to determine which processes and roles will be involved, impose an ordering, conduct the dialogue with user and applications, and report the results back in presentations. For example, the wish to make a phone call will trigger a *PhoneTelecommunication* process involving (among others) the role of a *phoneNumber* to dial, and may possibly also incorporate subprocesses like an *AddressbookSearch* if needed, which might retrieve the *phoneNumber* based on a *callee* role, and report it back to the user for confirmation before making the call. It is also necessary that the system reacts gracefully to deviations from the planned course of action by the user or the system.

Each action, from the act of greeting the user to providing a complex service, is represented by a process instance. Requesting a service then translates to setting the goal to perform actions associated with that process. The actions are defined by plan operator specifications. They usually involve manipulating roles of the process, such as using them as parameters for some action, or finding instantiations for them. Also, frequently the results of other processes are required to achieve subgoals. To determine the appropriate processes and their execution ordering is the task of the planner component. An executor component then performs the subtasks and monitors the outcome.

3.2 Intentions

User utterances are represented in a data structure called an *intention lattice*, which may contain one or more *intention* segments that partition the utterance sequentially (in case of, for example, multisentence utterances). Each intention of the user can specify a set of dialogue acts, which in turn can be either *goal* or *slot manipulations*. These manipulations always occur in relation to a discourse object (or process in terms of the ontology). It also specifies whether an utterance was understood to be

explicit positive or negative feedback. The constituents of each intention segment that are important for the dialogue manager are therefore:

- A discourse object derived from the ontological framework, containing information about processes and roles the dialogue contribution is about.
- Goal manipulations:
  - setting a goal to adopt a new task, usually putting one or more processes on the agenda
  - retracting a goal to abort a task currently pursued
  - retaining a goal (this confirms that it is still actively being talked about, and an utterance occurs in the context of the goal)
- Slot manipulations:
  - setting slots to establish new information
  - retracting slots to take back / invalidate current information,
  - retaining slots to confirm information

  Slot manipulations always occur in the context of a specific goal.
- Annotation of the intention as positive and negative feedback, which might either be an answer to a simple question, or an expression of the emotional state of the user.
- Confidence scores from the analysis modules.

These types of constituents, except the last two, also apply to messages from the applications and devices. Additionally, the intentions contain additional information not directly relevant for dialogue planning, such as lexical items occurring in the input, which is passed on when triggering system output. This enables, e.g., the text generator component to adapt to the user's choice of words.

3.3 Communicative Games

Communicative games are derived from the notion of *dialogue games* (see, e.g., Carletta et al. (1996); Carlson (1983)). In addition to the contributions of the user and the system in the actual surface dialogue, they also capture the communications between the system and the application modules, as well as intrasystem communication. As many games do, they consist of consecutive moves, attributed to one participant each, and rules apply as to whether a move is allowed, depending on the situation. Composite games are construed from atomic games (like request–response) by concatenating or nesting them.

From the point of view of the dialogue manager, the dialogue does not only take part between the user and the animated agent. Smartakus does not know the current TV programs and cannot do a biometric authentication all by itself. Instead, to accomplish the tasks at hand, the devices and applications also need to enter the conversation (but this remains transparent to the user). So, in a typical interaction like the one depicted in Fig. 3, the dialogue manager will have subdialogues with the user, but also with applications: Smartakus acts as a delegate on the user's behalf. System-internal exchanges like the lexicon update, which informs other modules of

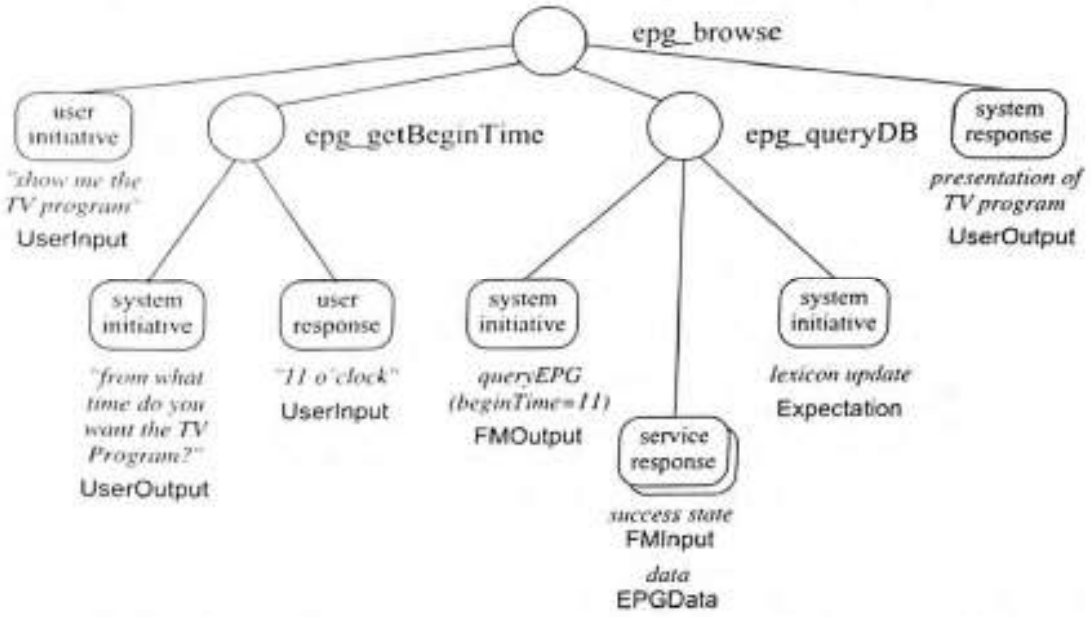


Fig. 3. Structure of a composite communicative game for television guide information search in SMARTKOM. The *circles* represent games; the *boxes*, game moves. The content of the messages and the channel used are shown below the boxes

new words likely to occur in future utterances (see Sect. 5.3.2), are also seen as moves of the communication game.

This model has the advantage that it captures the interactions with the user, the applications and devices, and other system modules in a uniform representation. Application protocols can be specified and treated as flexible as the user interaction. As an example, the database protocol prescribes that the results of voluminous database queries are returned to the dialogue manager in two parts, one indicating success or failure of the query, the other containing the actual data content. This interaction could be called request–{response$_1$,response$_2$}, wherer one request mandates two responses. The responses come from different sources and can arrive in arbitrary order. Also, the application requires that television program database requests should always entail a lexicon update (but not before the composite response has arrived). The *epg_queryDB* game in Fig. 3 bears this structure.[2]

4 Plan Operators and How Games are Played

Each application requires knowledge sources defining the set of plan operators to specify the services and the available dialogues of the application. The data for-

[2] As outlined in Sect. 5.3, an internal expectation message is sent (implicitly) every time the system releases the initiative. These moves were left out of the figure, except in the case where additional information — the lexicon update — is explicitly included by a plan operator.

mat of these specifications is given by XML schemata defining a plan operator language, which is based on the classic STRIPS language and extends it to effect actions through dialogical communication on several channels. Two types of definition files exist, one for each planning mechanism (see Sect. 5). A specification file consists of a header that relates a set of *slots* for the application to the ontological roles they are associated with, and a body defining a set of "states" similar to strips operators. The part corresponding to the body of the strips operator consists of a series of moves in communicative games. If an operator is applied, the moves will be placed on an execution stack. During the dialogue, the execution stack is reduced by the executor. This is done in two ways: If the topmost move is on an input channel and therefore an input is expected (e.g., an answer to a question, from the user or from an application), the system releases the initiative and waits for a contribution on the channel. If the move is on an output channel, a corresponding message is constructed and sent. For moves on output channels, the move specification also prescribes how the intention content of the corresponding communicative act is to be constructed.

An Example

Figure 4 shows a (slightly simplified) operator for the request–response action associated with the user browsing a TV program. The presence of a *goal* tag signifies that the result of this operator is a user-specifiable goal (i.e., "show me the TV program" or equivalent); the tag contains actions that will be done first upon adoption of the goal (the value of the slot *epg_broadcastSelection* is retracted). The *provides* tag specifies postconditions (the slot *epg_broadcastShown* will have a value when the goal is reached). This operator has no preconditions, which would otherwise be specified under a *needs* tag. The first *move* in the body says that this operator is to be triggered by an intention on the user input channel and should be answered on the user output channel. If the user sets the goal in an intention, both moves are put on the stack, and the first move is popped again immediately to "consume" the user utterance, leaving the "response" move on top of the stack. However, the "response" move has a precondition: It needs the slot *epg_broadcastSelection*.[3] Since this slot is retracted upon adopting the goal of *epg_browse*, it will not have a value after the first move, barring execution of the second. To proceed, this slot must be filled. The planner will be called to insert subgames (in this case, it will be another request-response game with the TV database) to satisfy the precondition. After the subgame has satisfied the precondition, the original "response" move can be made.

As can be seen from Fig. 4, the second move has two intention *templates* to cover different situations. If a move has intention templates, the first template whose preconditions are fulfilled is taken to realise the move intention. In this case, the templates define two alternative responses. The first template is selected if the slot *epg_broadcastSelection* has a value, but no textual content — which is the case if the TV database finds no results for the specific query. The answer of the system will then be an intention of type "noDataAvailable," generated as "I have not found

[3] Conditions test if a set of predicates holds. The condition *needs(slotname(X))* is really a convenience shorthand for *needs(slotIsSet(slotname(X)))*.

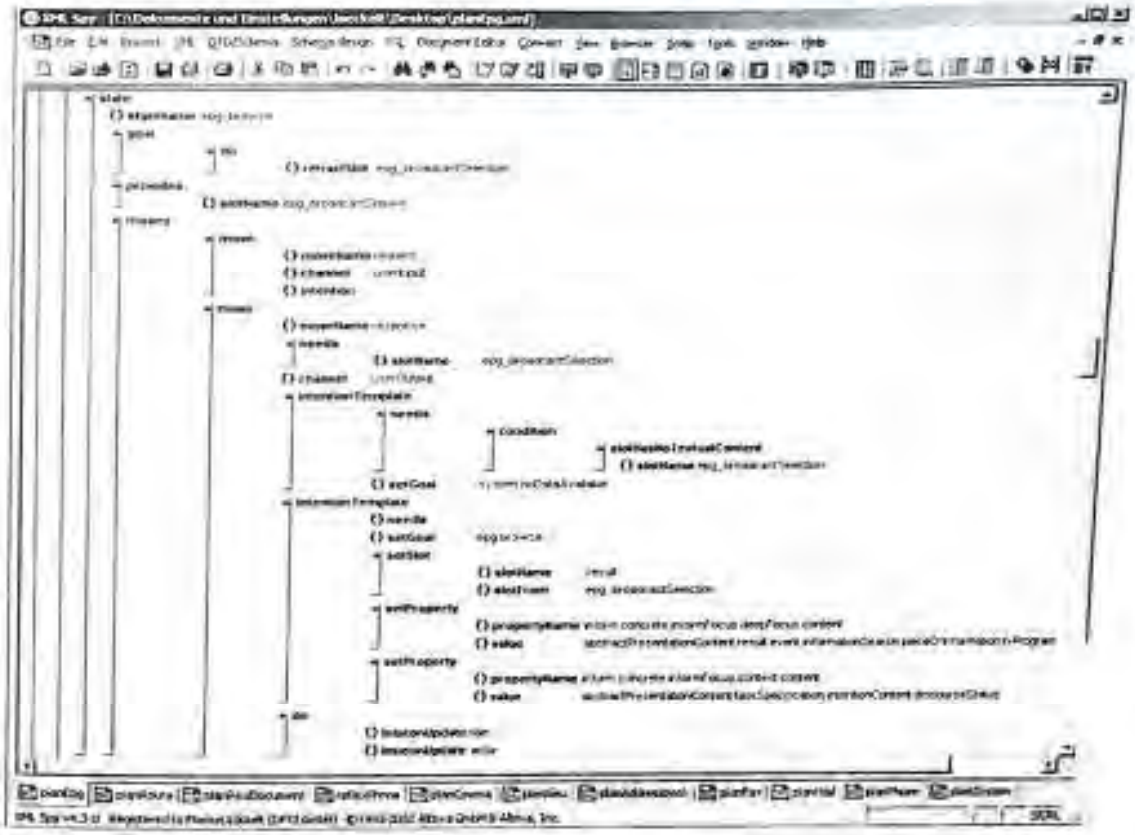


Fig. 4. Plan operator specification for EPG browsing

any data for your request." If, however, broadcasts matching the request are found, the second template (which has no preconditions) will be selected. The template results in an intention containing the results of the database query, taken from the *epg_broadcastSelection* slot, with some semantical markers for the graphical realization inserted (by the setProperty action). This intention is converted by the output channel to a TV program presentation, which is then passed on to the presentation manager. Finally, the *do* tag contains actions to be carried out during the move. In this case, entries are scheduled to be included in the next lexicon update after the system releases the initiative (Sect. 5.3.2).

Table 1 illustrates some of the keywords of the plan language, divided into structural keywords, predicates for conditions, and actions. In total, the language comprises 55 keywords; a complete descriptive listing is beyond the scope of this text.

5 Planning Mechanism

The dialogue manager uses two separate mechanisms to decide what to do next. For quick reactions to simple utterances or "stimuli," it employs *reflexes*, while inputs

Table 1. Descriptions of some of the keywords of the plan language

Structural keywords

slotDefinition	Assigns a slot name to a role of the ontology for a particular application
needs	Defines preconditions in terms of required slots and Boolean conditions
provides	Defines postconditions
do	Specifies a series of actions to be done
doTemplate	Like *do*, but with preconditions. Given a set of templates, only the first template with fulfilled preconditions is executed

boolean slot conditions

slotIsSet	True if a slot has some value
slotHasValue	True if a slot has a given value
predicate	True if a binary relation (e.g., "≥") holds between two slot values
slotEqualsSlot	True if the value of one slot equals the value of another

actions

setSlot	sets the value of a slot for an application
copySlot	Copies a slot value to another slot
genId	Generates a unique identifier
setProperty	Connects two parts of an intention by a link (as a semantical marker)
setGoal	Adopts a goal state. It is possible to specify whether a goal should be treated as a subgoal, or should replace the currently pursued active goal

that require a planned response are handled by a planning procedure. In this section, we first describe the method used for reflexes, before turning to planned actions.

5.1 Reflex Actions

nput events that do not need much planning, but instead require fast action from he system, are handled differently from normal input. We call this type of input and action *reflexive*, and they are treated somewhat analogously to animal reflexes in that the input is processed and reacted upon by a special, simple, and quick mechanism (the *spine*), and enters the planning process proper only afterwards. One example of reflexive input is entering a telephone number via a keypad, where the delay in displaying the typed numbers must be minimized to obtain adequate responsiveness, because the user will (reasonably) accept much less delay for this type of interaction than for more complex utterances. Reflex reactions are determined by a simple table look-up, and are therefore quite inflexible, but the advantage is that they can always be processed in small constant time. The operations that can be performed by the spine component are restricted: no planning is possible, and no parametrized intentions are generated. Of course it is also possible to pass on information, so that the planning component can deal with it (e.g., handing over the telephone number after dialing is complete).

Table 2. Top-level steps for sending a fax message

Step	Task
1	Clear screen
2	Obtain picture from realDocument application
3	Obtain fax number from phone application
4	Inform "fax being sent"
5	Convert picture to fax format
6	Send fax with picture attachment
7	Wait for confirmation of sending fax
8	Inform about completion of task

5.2 Planned Actions

In most cases, user utterances must be integrated into an overall plan of ongoing dialogue. The dialogue manager maintains a stack of goals that have been adopted during the dialogue. At any one time, one goal is focused for the planner, the *current goal*. A plan is created and then pursued that specifies a possible course of subsequent communicative acts to reach the goal. If the user fully cooperates with the system, the originally generated plan will be followed. However, frequently the user will make contributions that are not intended by the plan, or he will provide information in an order different from the planned sequence. In this case, the plan has to be revised and modified to account for changed circumstances, and to continue the dialogue.

An action plan specifys a series of plan operators that establish preconditions and ultimately lead to an operator representing the goal state. It is constructed by a backward search from the goal state. Where the plan needs to refer to another, subordinate application (e.g., the phone application needs a lookup in the address book application to fetch a telephone number), another hierarchical level is entered and the subplan is worked out in detail. The assumption is that the subordinate application guarantees that it can provide the service it offers (such as fetching a number), and therefore the planning can be deferred until the application is actually started. Thus, if the dialogue takes an alternative path (for example, the user might decide that she would like to enter the number via a keypad), the subplan will not need to be generated at all. After a plan has been generated, the series of its plan operators is followed, by playing the communicative games and executing the actions associated with the individual steps in order. During this process, the dialogue manager releases the dialogue initiative when it has to wait for an answer from either the user or one of the applications (i.e., the current game move is on an input channel).

Upon retaking the initiative, the planner checks if the assumed conditions still hold true at this point of execution. This might not be the case if something unexpected has happened, such as the user being uncooperative or changing his mind, or an error from an application. In this case, the plan is regenerated. In the SMARTKOM applications, the absolute plan lengths on each hierarchical level are not very long, so the overhead for full plan generation and regeneration instead of plan repair is small (the plan length is usually shorter than ten communication moves, and takes not more

that one or two seconds to generate). Therefore it does not hamper the dialogue flow by causing an excessive wait for a system answer.

5.2.1 An Example Plan: Sending a Fax Message

In this section, we give an example of the plan that is generated for sending a fax when the user makes an utterance "I want to send a document". Upon this request, a *sendTelecommunication* process is initiated in the fax application. Planning backwards from a goal state, the top-level plan will be the one shown in Table 2.

However, to complete the task, several applications and services have to work together. If the initial plan is rolled out in full, it would contain the steps shown in Table 3. The current application changes as the different subdialogues are delved into. The table also shows the channels on which the communicative games are played. The games are *inform*, *instruct*, *request-response*, and the special type *graphicalAction* (clearing the screen, presenting the document scanning area). The channels include the user input/output channel and the function modeling channels for the devices realDocument and telephony. Note that the request to remove the document in 2-7 is answered not by the user, but by the realDocument device detecting the removal in 2-8. Not shown are the internal moves for the expectations, sent each time on giving up initiative to the user or some device (after steps 2-3, 2-5, 2-7, 2-9, 3-1, 5-1, and 6; the expectation mechanism is explained in the next section), and the reflexive moves during keypad interaction (collecting the number and updating the keypad display). The user presentations are in graphical-only (steps 1, 2-11), speech-only (2-2, 2-7, 4) and multimodal (2-1, 3-1, 8) form.

Before the execution of each plan step, its preconditions are rechecked to confirm that they still hold. It is possible that, during the course of the dialogue, the user would throw in "I want to send it to Anselm." It would then not be necessary to require the user to type in the fax number, since the number can be retrieved using the address book application. This would be noticed upon executing the plan step designated as "get the fax number," since that step has a branching condition that prefers using the address book instead of the keypad subdialogue if an addressee name is known. So, upon reaching this step, the dialogue is replanned and an address book search is used to expand step 3 instead.

5.3 Expectation Feedback

The fact that future moves of the dialogue are predicted by the dialogue plan can be used to help the analysis modules to resolve ambiguities or to prime the lexicon for words that may occur in the input. After each dialogue turn, the dialogue manager publishes an updated data structure, called an *expectation*, to inform other modules about likely subsequent utterances. The expectation mechanism is also used to inform the discourse memory when the user is being asked to confirm a past utterance. A more thorough description of the use of expectations is given in Löckelt et al. (2002).

Table 3. Fully expanded plan for sending a fax message

Step	Task	Channel	Application
1	Present clear screen	→ user (g)	fax
2-1	Present scanning area	→ user (g,s)	realDocument
2-2	Instruct to place the document	→ user (s)	
2-3	Initialize document scanner	→ realDocument	
2-4	Response initialization complete	← realDocument	
2-5	Request start scanning	→ realDocument	
2-6	Response scanning complete	← realDocument	
2-7	Request to remove document	→ user (s)	
2-8	Response document removed	← realDocument	
2-9	Request scanned image	→ realDocument	
2-10	Response scanned image	← realDocument	
2-11	Present scanned image	→ user (g)	
3-1	Present keypad and request number	→ user(g,s)	phone
3-2	Collect number	← user (reactive)	
3-3	Response number	← internal	
4	Inform "fax being sent"	→ user (s)	fax
5-1	Request transcribe picture	→ realDocument	realDocument
5-2	Response transcribed image	← realDocument	
6	Request sending of fax	→ telephony	fax
7	Response sending complete	← telephony	
8	Inform about completion of task	→ user (g,s)	

5.3.1 Slot and Goal Expectations

To give information about the state of the dialogue, the expectation structure specifies the currently active goal and the application to which it belongs. Additionally, if the system has released the initiative to the user, e.g., to answer a question, the analysis modules, especially the discourse modeler, receive indications about expected discourse contributions from the slot expectations, which come in three categories:

- *Expected slot*: If the user is expected to provide specific information, the associated slot is named. The discourse modeler can resolve ambiguities by assigning inputs referring to ontological roles that correspond to an expected slot a higher score.
- *Filled slots*: The slots related to the current task that have already been assigned a value. These will get preference, e.g., if the utterance indicates that the user wants to change something already said, such as in "and what's on TV tomorrow?". Such values can be present if either they were provided during the previous dialogue steps, or they were preset values (such as taking the fastest route by default instead of the shortest in a route planning application).
- *Possible slots* are either slots that would — according to the plan — be addressed in later steps of the dialogue, or that would not necessarily be mentioned at all. An example would be a genre restriction for a movie database query.

An additional flag of the expectation structure is used to inform the discourse modeler of the special case of asking for confirmation of the last input step. An elliptical "yes" in answer to, e.g., "Are you sure you want *X*?" can be put into context this way and be interpreted as a combination of "retainGoal" and/or "retainSlot" operations.

Since the expectations themselves are not limited to what is the single most likely utterance in the dialogue, but offer some fine-grainedness for different possibilities and utterance types, the user has freedom to deviate from the direct trail offered by the dialogue and can, for example, forestall a question not yet asked or change her mind about what he said earlier. The expectation structure will likely be helpful in putting the utterance into context. Dialogue contributions that are not explicitly focused on but are related to the current goal will likely still be of an expected category, although they will possibly get a score penalty.

5.3.2 Lexicon Updates

Applications can specify that they require dynamic lexicon updates during certain situations. For example, the lexicon does not contain composite movie titles or geographical landmarks at startup, since the necessary word list would be inappropriate in size and for actuality requirements, and even if a comprehensive list was available, the recognition rate would suffer if lots of words clogged up the lexicon. This type of lexical item is most likely to appear only in certain specific situations. So instead of statically holding all words that may occur at some point in the dialogue, the lexicon is adapted dynamically, inserting, e.g., movie titles in the context of the cinema application. During a cinema dialogue, the dialogue manager scans the results of movie database queries, and lexical items of several predefined categories (movie titles, actors, directors) are extracted. The next published expectation then contains an update for this word category. The mechanism also for the removal of words when the context changes. Analogous lexicon updates are performed for the television guide and route planning applications.

6 Conclusion

6.1 Summary

We have described the dialogue manager of SMARTKOM, which plans and executes user and application interactions in a large multimodal system comprising multiple applications in different scenarios. It features a flexible architecture suited for integration in different systems.

The actions of the system comprise dialogues with the user, the devices and applications, and other modules of the system, treated uniformly via communication channels. The ontological framework of SMARTKOM is used to model the problem domain in terms of processes and roles. We talked about how the plan operator

specification language resting on this foundation is used to define the behaviour of SMARTKOM in a declarative XML-encoded knowledge source.

Two mechanisms for determining the actions of the system were described: the reflex mechanism used to generate fast responses to simple stimuli, and the planning mechanism to deal with inputs that require coordinated and planned actions. An added useful feature is the publication of expectations, which allow the analysis component of the system to take into account the current state of the dialogue in interpreting user utterances.

6.2 Further Work

The dialogue manager is currently being redesigned to extend its functionality. This will introduce the ability to conduct multiparty dialogues involving several virtual characters, and to use more sophisticated planning and plan accommodation mechanisms. A third focus of the ongoing work seeks to address the difficulties that were encountered in defining a nontrivial set of interconnected applications, where the dependencies can quickly become difficult to handle, and errors are easily overlooked. Separating dialogue management proper from the application functionalities as far as possible, we investigate possibilities to support the designer of the application in improving the overview of content and dependencies, as well as providing easier manipulating and testing facilities.

References

J. Carletta, A. Isard, S. Isard, J. Kowtko, and G. Doherty-Sneddon. HCRC Dialogue Structure Coding Manual. Technical Report HCRC/TR-82, HCRC, 1996.

L. Carlson. *Dialogue Games*. Synthese Language Library. Reidel, Dordrecht, The Netherlands, 1983.

G. Ferguson and J.F. Allen. TRIPS: An Intelligent Integrated Problem-Solving Assistant. In: *Proc. 15th Nat. Conf. on Artificial Intelligence & 10th Conf. on Innovative Applications of Artificial Intelligence*, pp. 567–573, Madison, WI, 1998.

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, 2004. Special issue on Software Architecture for Language Engineering.

M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An Architecture for Multimodal Dialogue Systems. In: *Proc. 10th ACM Int. Symposium on Advances in Geographic Information Systems*, pp. 376–383, Washington, DC, 2002.

S. Larsson. *Issue-Based Dialogue Management*. PhD thesis, Göteborg University, Sweden, 2002.

M. Löckelt, T. Becker, N. Pfleger, and J. Alexandersson. Making Sense of Partial. In: C.M. Johan Bos, Mary Ellen Foster (ed.), *Proc. 6th Workshop on the Semantics and Pragmatics of Dialogue (EDILOG 2002)*, pp. 101–107, Edinburgh, UK, September 2002.

R. Porzel, N. Pfleger, S. Merten, M. Loeckelt, I. Gurevych, R. Engel, and J. Alexandersson. More on Less: Further Applications of Ontologies in Multi-Modal Dialogue Systems. In: *Proc. 3rd IJCAI 2003 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, 2003.

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. Bersen (eds.), *Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, Dordrecht, The Netherlands, 2004. Kluwer Academic.

C. Rich and C.L. Sidner. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction*, 8(3-4):315–350, 1998.

W. Wahlster. SmartKom: Fusion and Fission of Speech, Gestures, and Facial Expressions. In: *Proc. 1st Int. Workshop on Man-Machine Symbiotic Systems*, pp. 213–225, Kyoto, Japan, 2002.

Emotion Analysis and Emotion-Handling Subdialogues

Michael Streit[1], Anton Batliner[2], and Thomas Portele[3]

[1] DFKI GmbH, Saarbrücken, Germany
streit@dfki.de
[2] Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
batliner@informatik.uni-erlangen.de
[3] Philips Research Laboratories GmbH, Aachen, Germany
Thomas.Portele@philips.com

Summary. The chapter presents the cognitive model–based approach of abductive interpretation of emotions that is used in the multimodal dialogue system SMARTKOM. The approach is based on Ortony, Clore and Collins' (OCC) model of emotions, which explains emotions by matches or mismatches of the attitudes of an agent with the state of affairs in the relevant situation. We explain how eliciting conditions, i.e., abstract schemata for the explanation of emotions, can be instantiated with general or abstract concepts for attitudes and actions, and further enhanced with conditions and operators for generating reactions, which allow for abductive inference of explanations of emotional states and determination of reactions. During this process concepts that are initially abstract are made concrete. Emotions may work as a self-contained dialogue move. They show a complex relation to explicit communication. Additionally, we present our approach of evaluating indicators of emotions and user states that come from different sources.

1 Introduction

For a long period, the concept of rational agents, which exchange rational arguments, was the predominant paradigm for research on dialogue systems. In the last decade the scientific community became aware of the fact that emotions, moods and other attitudes play an important role in natural communication. This insight in human behavior resulted in research on affective interfaces, development of believable agents, and in research that aims at supporting the recognition of problematic dialogue situations by analyzing the affective state of the user.

While there are considerable advancements in generating affective artificial agents that display believable emotions in appropriate situations (Picard, 1997), the recognition and interpretation of human emotions in dialogue systems is still in its infancy. In SMARTKOM we explore the handling of emotions by focusing on exemplary cases, which show how emotions work as dialogue moves or and how the emotional color of an utterance may change the literal meaning into the converse.

The term *emotion* normally aims at pronounced, clear forms of human states marked by strong feelings such as, e.g., anger, fear, sadness, joy, etc. — the so-called "full-blown, big" *n* (*n* typically ranging between 4 and some 20) — emotions. Under a close look, however, almost nothing is that clear-cut: the underlying (bodily and cognitive) processes are not yet fully understood, emotions do often occur in mixed, not in pure forms, their marking can be overtly suppressed due to social constraints and rules (Cornelius, 2000; Batliner et al., 2003a,c,b). Finally there is no full agreement as for a catalogue of emotions, and of pivotal characteristics, telling emotions apart from other states such as attitudes, mood, etc.

The OCC Model of Emotions

Research concerned with the generation of an affective and believable behavior of artificial agents is often based on the so-called OCC model of emotions (Ortony et al., 1988), which explains emotion by cognitive processes relating the user's goals, standards, likes and dislikes to the actions of other agents and the state of the world that results from these actions. This approach starts from clearly distinguished emotions, which result from a cognitively comprehensible *appraisal* of the situation by the agent. How intensive emotions are, or if they are displayed at all is determined by the intensity of the attitudes of the agent and by social display rules.

Though mixing or suppressing emotions is a problem for the recognition of emotions as well as for the fine-tuning of the artificial generation of emotional behavior, the OCC model provides a systematic account for relating a certain situation of an interaction to emotional states that fit to this situation. The logical structure of the situation that causes a certain emotion is not affected by the question of how intensive an emotion is or if it is displayed at all. This is not only attractive for the generation of affective behavior, but also for inferring appropriate reactions on emotions that are observed in a certain situation.

Emotions and Other Affective States

For research, which is concerned with the detection of problematic situations in communication by analyzing the user's behavior, not only emotions are relevant. This is the case, independently from the question of whether the catalogue of emotions is completely defined or not. For instance, if the user is hesitant, she may need help, or if she is tired this may affect the presentation of information, or the system may even request the user to stop some activity that needs high attention.

We use the term "(emotional) user states" to encompass all nonneutral, somehow marked behavior of the user within a human–machine communication. Thus, user states as bored, stressed, irritated, tired, etc., can and have to be addressed as well, irrespective of whether they belong to the one or the other psychological or physiological category. In contrast, the psychological or physiological category of a state is relevant for its interpretation. The SMARTKOM system will react to this match or mismatch, e.g., by trying to repair the mismatch or by promoting further matches.

The spirit of the approach, namely to consider what type of conditions elicit the affective state of the agent, extends to some nonemotional states, but not to all. For

instance, the user may be hesitant, because she has a goal, but does not know how she can achieve it, or she does not know how to decide between certain alternatives, etc. Such considerations add up to a very similar kind of cognitively comprehensible appraisel of a situation, as elicit emotions in the OCC model. In contrast, the state of tiredness (in the literal meaning) cannot be accounted for by such an approach that analyses the *cause* of the user state. Rather, the system should consider the possible consequences of the user's state.

Preconditions for a Working System

Some important conditions have to be met, however, if one wants to deal with user states in an automatic system:

- It must be possible to classify the state correctly up to a satisfying extent.
- Thus there has to be a sufficiently large training sample.
- The respective user state can be processed within the whole system, not only at the classification stage.

The first condition means that we should start with user states that are clearly marked. This rules out such states as "slightly irritated," even if they might occur quite often and have a strong impact on the felicity of communication. We are thus left with those pure emotions like anger or joy, which do not, alas, occur often in real or in Wizard-of-Oz human–machine communications (Batliner et al., 2003c,a,b; Ang et al., 2002). Thus we decided to concentrate on some few user states and to collect acted data for our training sample.

Overview

The focus of the paper is to present the cognitive model–based approach of abductive interpretation of emotions as it is used in the SMARTKOM system. Additionally, we present our approach on the collection and evaluation of indicatons of emotions, user states, problematic situations and other states of the user or the interaction. Recognition of emotions from prosody and facial expressions is described in Zeißler et al. (2006) and in Frank et al. (2006).

We start in the second section with a brief description of the architecture of emotion analysis in SMARTKOM. In the third section we introduce the type of interaction we want to focus on. In the fourth section we describe the method for evaluating indicators from different sources. At the time, we use only a little part of the information provided there, but much more would be useful The remaining sections are dedicated to the interpretation of emotions and user states and the generation of reactions to these states. First, we introduce the OCC model of emotions, then we say a little bit about abduction and the problems that occur if we use the OCC model for an abductive interpretation. In the next step we show how we generate explanatory hypotheses by introducing abstract concepts as fillers of eliciting conditions. We introduce conditions for explanation selection and recovering operators for generating initiate reactions of the system. Conditions and generation of reactions are interwoven: usually conditions provide parameters that are used by the recovering operators.

2 Emotion Processing in the SmartKom System

SMARTKOM (http://www.smartkom.org) is a multimodal dialogue system that provides access to multiple applications (Wahlster, 2003). In addition to input-modalities that are used for intentional communication, the system accounts for the emotional state of the user as it is displayed by facial expression or by prosody. The processing of emotions and user states consists of three stages.

- At the first stage the emotional state of the user is recognized from facial expression and prosody.
- At the second stage indications of problematic situations and the emotional state of the user are collected from several sources and collectively evaluated. The component also analyzes the dialogue in respect to the style of interaction and the task and paradigm knowledge of the user (Portele, 2003).
- The interpretation of emotions and user states, and the generation of reactions to these states build the third stage. It is realized by so-called dynamic help. This component is dedicated to manage subdialogues, and to provide presentation specification and intention analysis in problematic situations that are not handled by the standard dialogue component of SMARTKOM.

3 The Use Cases

To demonstrate the added value of user state classification and its subsequent processing in the SMARTKOM system, we designed so-called **use cases**. The first use case is intended to show how a merely emotional reaction, without explicit communication, can work as a self-contained dialogue move. In this case, joy or anger are interpreted as positive or negative feedback, respectively. In the second use case emotion works as a semantic operation that turns a positive feedback into a negative one, which is considered as a form of sarcasm. In both use cases, the system suspects that the emotional reaction may be caused by a like or a dislike concerning the properties of the presented objects. If reasonable candidates of such likes or dislikes can be identified that are not already known by the system, it starts a preference update dialogue.

If the system knows positive or negative preferences, it first presents objects that contain a preferred feature. Objects that show a disliked feature will be shown last. [1]

user: *What's on TV tomorrow?*
system: Shows talk show at the top of the display, in the middle popular music, and crime at the bottom.
user: *And what's in the evening, in the first program?*
system: Shows a science fiction movie.

[1] It is possible that an object has both liked and disliked attributes, e.g., there may be a movie with a preferred genre, in which a disliked actor plays.

First constellation: emotion only
user: Displays joy via facial gestures.
system: *Do you like science fiction? Shall I account for that in future presentations?*

Second constellation: emotionally marked verbal communication
user: *That's really a beautiful program!* (She produces this sentence with an angry prosody. The positive feedback is analyzed as being sarcastic.)
system: *You don't like science fiction? Shall I account for that in future presentations?*

user: *Yes./No.*
system: *OK. I'll take care of that!*

(Suppose the user's answer was *yes*: In the first constellation **science fiction** will be presented at the beginning of a presentation, in the second constellation at the end.)

user: *Please, again tomorrow's program!*
system: Shows **science fiction** at the beginning (at the end) of a presentation.

Instead of answering *no* or *yes*, the user may also correct the supposed like or dislike, e.g., by saying *No, I like crime movies*, or she may just ignore the question, by moving to a different topic. In such cases, the system will simply not rearrange the order of presentation.

4 Indications of User States and Problematic States of Interactions

We introduced in SMARTKOM a component, the *interaction module*, that collects and evaluates indications of emotions, problematic situations and other aspects of the interaction. This approach is used, on the one hand, to estimate recognition results by taking other indicators into account that may support or devaluate the result. On the other hand, the component introduces its own results concerning states of the user or charcteristics of the interaction. It operates by analyzing a set of possible *indicators*.

Indicators can have values between 0 and 1 and these values may change in time. For example, the mimic analysis module delivers information about the user's current emotional state as a set of four probabilities (likelihood of anger, likelihood of joy, likelihood of dilatoriness, neutral). The set of indicators is fairly fixed and is defined by the capabilities of the modules in the system (Table 1).

The interaction module provides a set of *models* as output. Each model value is also in the range between 0 and 1. Several models support the recognition of emotions and try to detect problematic situations during a dialogue.

4.1 Indicators

The indicator values are mapped to the models by means of a matrix multiplication. One element of the matrix denotes the influence of one indicator value to one model. This design is motivated by the observation that most indicators can contribute to different models, and that the combination of simple indicators to complex models may be optimized by machine-learning algorithms. Furthermore, new models on demand from other modules can be constructed easily by combining the indicator set with a different weighting scheme.

Table 1. List of indicators

Source	Description
Mimic recognizer	Mimically conveyed anger
Prosody recognizer	Prosodically conveyed anger
Mimic recognizer	Mimically conveyed joy
Prosody recognizer	Prosodically conveyed joy
Mimic recognizer	Mimically conveyed dilatoriness
Prosody recognizer	Prosodically conveyed dilatoriness
Speech recognition	Linguistically conveyed anger
Speech understanding	Ratio of unanalyzable words
Intention analysis	Overall score of the best hypothesis
Intention analysis	Difference in score between first and second best hypotheses
Intention analysis	Number of possible hypotheses (depth of lattice)
Speech recognition	Score of the speech recognizer
Gesture recognition	Score of the gesture analyzer
Speech understanding	Score of the language analyzer
Media integration	Score of multimodal integration
Discourse history	Score of the discourse module
Domain model	Score of the domain models
Intention analysis	Final score of the intention module
Intention analysis	Number of elements in the user input
Discourse history	Number of new (not previously mentioned) elements
Speech understanding	Number of elements addressed by speech
Gesture analysis	Number of elements addressed by gesture
Media integration	Number of elements addressed by speech and gesture
Intention analysis	Importance of speech recognition score for overall score
Intention analysis	Importance of gesture analysis score for overall score
Intention analysis	Importance of domain model score for overall score
Intention analysis	Importance of language understanding score for overall score
Intention analysis	Importance of discourse model score for overall score
Speech understanding	Relative number of sentence-like units in one turn
Speech understanding	Relative number of words in one turn
Speech understanding	Relative frequency of pronouns
Speech understanding	Relative frequency of verbs
Speech understanding	Relative frequency of adverbs
Speech understanding	Relative frequency of nouns
Speech understanding	Relative frequency of content words
Speech understanding, language generation	Relative frequency of content words appearing in the system output
Speech understanding, language generation	Relative frequency of content words not appearing in the system output

4.2 Models

The module delivers four sets of models. The distribution of the models to the sets is somewhat arbitrary and is mainly governed by the intended use within the SMARTKOM system. Therefore the models for task and paradigm knowledge use many in-

Table 2. List of models

Set	Description
Problem	Likelihood of a problem
Problem	Likelihood of an analysis problem
Problem	Discourse progress rate
Problem	Likelihood of the user being angry
Problem	Likelihood of the user being happy
UserKnowledge	Estimation of user familiarity with task
UserKnowledge	Estimation of user familiarity with system
Modality	Ratio of spoken input content
Modality	Ratio of gestural input content
Modality	Ratio of multimodal input content
ModalityContrastive	Ratio of contrastive usage of multimodal input
ModalityRedundant	Ratio of redundant usage of multimodal input
Linguistic	Adaptivity of user's lexical choices to former system output
Linguistic	Likelihood of long turns
Linguistic	Likelihood of long sentences
Linguistic	Ratio of pronoun usage
Linguistic	Ratio of verb usage
Linguistic	Ratio of adverb usage
Linguistic	Ratio of noun and verb usage

dicators that also contribute to the problem detection models. This overlap reflects the design principle of the module.

- One set of model values reflects the assumed task and paradigm knowledge of the user. The task knowledge describes the user's knowledge of the current task (e.g., programming a VCR), while the paradigm knowledge indicates how well the user is accustomed to dealing with multimodal dialogue systems, and, especially, with SMARTKOM. These models can be employed by the dynamic help and the presentation/language generation to deliver an appropriate amount of feedback and assistance.
- A second set of models describes the linguistic behavior of the user regarding the number of, e.g., referential expressions, usage of complete sentences and average length of input. Some of these features can, in principle, be used by language models or parsers (although this is not the case in the current SMARTKOM system). All of them may help to adapt the language generation in order to reflect the user's style—based on the assumption that this is beneficial. Furthermore, the adaptivity of a user's lexical choices to former system output is estimated, which can helping adapting dynamic language models used in SMARTKOM and language generation in order to maximize this value as a measure of the common vocabulary.
- The third set compares the use of different modalities by the user (for instance, a preference for gestures or spoken input). The presentation and the behavior of the animated character can be adapted toward the user's preferred distribution of the different modalities — users who prefer pointing gestures could be supplied with an interaction display with more possibilities and details, while users interacting mainly through speech should get the most important system feedback also by spoken output.
- Problematic situations and user state information are expressed by three models in the fourth set of the module:

- One model describes the likelihood that the user is angry by combining scores from mimic analysis, emotion extraction from prosody and use of certain words. These indicators are either computed directly from other modules or are obtained by counting appearances of specific words (a related model assesses the user's satisfaction by looking for joyous emotional expressions; this and the anger model are used for setting content-based user preferences).
- A second model combines confidence values from recognizers and similar scores from speech analysis, domain model, discourse history and intention recognition as well as differences in the distribution of these values among concurring hypotheses; this model is supposed to indicate problems in the analysis part of the system.
- A third model estimates the dialogue progress. Here, the ratio of new information items to total information items (after completion by overlay in the discourse history) is employed as one indicator, another one being the ratio of overlayed (i.e., changed, interpretable as corrected) information to unaltered information. A further indicator is the overall number of information items in the user input. Because of the system design, all other indicators can contribute to the model values as well (and the indicators used here are also used for other models, e.g., about user experience with the task or the interaction paradigm), but the indicators named above are assumed to be the most important ones for the respective models.

These model values are part of the input of the dynamic help module.

5 Cognitive Model–Based Interpretation of Emotions

Our approach to the analysis of emotions is based on the OCC model of emotions developed by Ortony, Clore and Collins. Following the OCC model, emotions are characterized by their eliciting conditions. These conditions consist of a certain combination of

- the goals of the agent in this situation
- her attitudes to certain events (mainly likes and dislikes)
- the standards that she uses to (morally) judge an event
- the facts that hold in a certain situation
- the actions (of other agents) that caused these facts

For triggering an emotion, it is important to know how facts are related to the goals and the likes and dislikes of the user. In particular, it is interesting if they coincide or not. Standards are important for emotions such as anger or gratitude that contain criticism or praise of another agent based on her actions. Eliciting conditions can be viewed as expressing the cause of an emotion by providing a cognitively comprehensible explanation of an emotion. The following eliciting condition for anger is taken from Prendinger et al. (2002):

```
anger(Agent1,Agent2,State,Sit) if
   holds(did(Agent2,Action),Sit),
   causes(Action,State,Sit0),
   wants(Agent1, non_State,Sit),
   blameworthy(Agent1,Action)) ,
   (Sit0 < Sit)
```

This condition means that the agent is angry if she believes that another agent caused some state of affairs that contradicts her goals by performing an action that is not acceptable according to the user's standards (expressed by the *blameworthy* predicate). By the situation variables Sit, Sit0, one can express how the elements of the conditions are connected with respect to the sequence of situations that occur (subsequently we will omit situation variables).

Recognizing the intensity of emotions could provide additional valuable information, e.g., slight anger may occur at the beginning of a problem, while strong anger may indicate an enduring problem. But the recognition of the situation that caused the emotion and the generation of appropriate reaction is basically the same whether emotions are displayed slightly or strongly.

5.1 Abductive Interpretation of Eliciting Conditions

The OCC model is mainly used for the generation of the behavior of an animated agent. In this case, one can deliberately define the agent's likes, dislikes and standards in advance. If we want to interpret emotions that are displayed by an agent, we have to find out which combination of facts, attitudes and standards may have caused the emotion. Our approach is to achieve this by analyzing eliciting conditions in an abductive manner. Abduction as a form of practical inference is introduced by Peirce (1995). Abduction is often characterized as inference to the best explanation: Suppose we observe some fact A, which is surprising for us. If we know the rule

$$B,C \rightarrow A$$

(i.e., A is true if B and C are true), then we may suspect that also B and C are true, because this would plausibly explain A. If we know that there is another rule

$$D \rightarrow A$$

then D is another candidate for explaining A. Hence we need a criterion to decide which explanation is better. The quality of an explanation depends on two factors: Do we know all relevant rules (i.e., explanations)? Do we possess criterions to choose from explanations? With eliciting conditions we have the advantage of possessing schemata that claim to characterize all possible explanations of an emotion.

5.2 Problems with Abductive Interpretation

Eliciting conditions are abstract schemata that cannot be used directly to infer possible causes of emotions. To perform abductive reasoning on eliciting conditions, we have to identify concepts that could be filled into the schemata. Seemingly, we are in a problematic situation. The system has no information about the user's standards,

likes and dislikes in advance. It can get information about her goals from the user's input. But, on the one hand, this information may be based on misunderstanding, and, on the other hand, the user may have goals which cannot be recognized from her utterances. Similar problems occur with the actions of the system. The concrete actions that are based on misunderstanding are not relevant for the analysis of the user's emotion.[2]

5.3 Abstract Goals and Actions for Emotion Interpretation

To overcome the problems mentioned in the last paragraph, we introduce metagoals concerning general principles of communication and abstract goals concerning user needs that (to some extent) depend on the application. For every metagoal or abstract goal we introduce an abstract action that satisfies the goal.

For instance, to account for misunderstandings, we introduce *understanding* as an action on the metalevel and *to be understood* as a goal on the metalevel. To account for user preferences, we introduce the concept that *a presentation accounts for the user's preferences* as an abstract action of the system — let it be called *presentByPreferences* — and accordingly the possible abstract fact or user goal *isPresentedByPreferences*.[3] This goal is abstract and underspecified because we do not know the concrete preferences of the user. Further, the relevant types of preferences depend on the type of the application.

Reasonable goals (facts, actions, likes, standards) have to be identified by careful analysis of general principles of communication and the needs of the user with respect to the type of applications with which she is working. This needs empirical validation, which could not be provided within the scope of the SMARTKOM project. Which set of concepts is chosen also depends on practical decisions: which goals will the system support at all, will the system possibly recognize goals that it is not able to handle, will the system react on any recognized emotion in some way (e.g., by regretting as a default in case of anger), or will it only react to emotions to which it can provide a repair or other meaningful cooperative reaction? We demonstrate the approach by the example of **anger**.

General Concepts

We first look for actions or facts that may contradict the user's wishes, likes, dislikes or standards on a general level. Important candidates for abstract actions that contradict the user's wishes are **misunderstanding**, **slow processing** and **requests with a negative or disliked outcome**. Accordingly, we stipulate abstract or general goals, e.g., the goal to be understood properly.

[2] Although the type of the action that the system wrongly performs may influence the intensity of the user's negative feelings.

[3] For convenience we often identify the name of the fact and the name of the goal to make this fact true.

Application-Dependent Concepts: Problematic Results of Database Queries

According to our use cases we concentrate on requests with liked or disliked outcome as a source of negative or positive emotions. We identified four types of disliked results:

- The result is empty.
- The majority of retrieved objects show features that are not liked by the user.
- The objects are presented in a way that is contrary to the preferences of the user, e.g., by presenting disliked objects first.
- The user query resulted in a recall, which is too large. The user may need help for further specification possibilities.

We assume for this list of topics that the disliked or problematic results are not due to misunderstanding. Misunderstanding is taken as evoking its own class of constellations. If misunderstanding is involved, the result is not relevant for the analysis.

User-Specified Goals and System-Initiated Actions

As far as no misunderstanding is involved, the SMARTKOM system will usually simply follow the user's specification. If this works, no anger should arise with respect to the fact that the system tries to achieve this goal (but perhaps instead *joy*). In specific situations the system may initiate actions that are necessary from the point of view of the system, but that may be disliked or even considered blameworthy by the user. For instance, the system may require a biometric verification, which the user dislikes. Such actions are relevant for explaining negative emotions but are not considerd in our implementation. Inappropriate or undesired reactions on emotions could also be a cause for anger (or for being bored). In fact, this is a subcase of disliked system-initiated actions.

6 Analyzing and Handling Pure Emotion

With the concepts introduced in the last section, we are able to build instantiations of eliciting conditions that allow us to infer combinations of goals, facts, actions, likes and dislikes that possibly explain the user's emotion. We call instantiations of eliciting condition schemata **eliciting constellations**. To get criteria for selecting the relevant constellation, we augment constellations with conditions and organize these conditions internally as a decision tree.[4]

Further, the system has to determine reactions that are appropriate for

- resolving the situation that caused the negative emotion
- avoiding negative emotions in future in similar situations
- promoting the occurrence of positive emotions in similar situations

[4] As mentioned in the conclusion, we could perform testing only in a limited way. Thus no training of the decision tree was possible.

It is also desirable to include methods that provide abstract underspecified goals and actions with presumable values. Such values are not only used for determining concrete system reactions, they serve as a part of the constellation conditions.

According to our use cases, we have to consider database queries that retrieve disliked objects. The system offers as repair that it will regard the likes and dislikes of the user in its presentations.

A constellation for handling anger according to our use cases is given below (leaving out some minor details) in a Prolog-style notation. It applies to browsing television programs or cinema programs. For these applications preferences are actually taken into account for the presentation. These rules are basically processed in the following manner: First the conditions are tested (internally the conditions are processed in a decision tree–like order). Then the cause of the emotion, which is represented by the clauses above the conditions, is considered as a reasonable explanation, whereby the variables are filled by the result of the condition processing. Then the system action is performed.

```
anger(thisConstellation,user,system) if
  holds(did(system, non_presentByPreference(dislike(user,X))),
  causes(non_presentByPreference(dislike(user,X)),
         non_isPresentedByPreference(dislike(user,X))),
  wants(user, isPresentedByPreferences(dislike(user,X))),
  blameworthy(user, non_presentByPreferences(dislike(user,X)))),
  conditions(thisConstellation,X),
  (proposed system action:) update(dislike(user,X)).
```

The constellation expresses that there is a concrete reading of the goal *presentByPreferences* that may be a goal of the user, that this goal is not satisfied and that ignoring the goal is against the standards of the user. The constellation contains facts and actions that are not concretely specified. For instance, we do not know whether the presentation contains some possibly disliked feature, and we do not know which feature it is.

We test the salience of the constellation by establishing the following conditions. The predicate *presentationEntriesContainCommonFeature(X)* also delivers a concrete presumable instance of the user's dislike.

```
conditions(thisConstellation,X)  if
      presentationEntriesContainCommonFeature(X)),     (1)
      non_specified(user,X),                           (2)
      non_knows(system,like(user,X)),                  (3)
      non_knows(system,dislike(user,X)).               (4)
```

Condition 1 verifies if the user perceives *too many* objects with the supposed disliked feature. (It also excludes the case that there is no result at all, which would support a different explanation for anger.) It is important for the other tests that the predicate delivers a hypothesis for the disliked feature. Condition 2 excludes that the user is angry about the occurrence of features that she has specified in her request

(there is a possibility of misunderstanding). Condition 3 excludes, that the user is angry about a feature that she has already declared to like. Condition 4 excludes that the system in fact tried to present the disliked feature appropriately but just did not find other objects.

For emotions displayed by facial expressions, we prove if the emotion emerges in a certain time interval after the presentation was displayed. With prosodically displayed emotion we prove if the verbally expressed content was compatible with the explanation of the emotion. It turned out that it is not sufficient to test if there are already stored preferences. It should additionally be proved, if a user has not agreed with storing a preference. This has to be remembered, otherwise the system may propose the same preference repeatedly.

The action *update(dislike,user,X)*, which is attached to the constellation, initiates a subdialogue that verifies if the user has the supposed dislike. It is not only a repair action, but takes part in the explanation process.

The conditions mentioned so far are not sufficient to **discriminate competing explanations**. Such competing explanations have to be modeled, even if no reaction is foreseen for these cases. We distinguished three main sources of anger: misunderstanding, slow processing and requests with a negative or disliked outcome. Evidence for problems in the analysis part is detected by the interaction module (Sect. 4). Slow processing is a possible explanation for anger, if anger occurs during the analysis. Also, the absolute duration of processing is a criterion. These dates are accessible via a module (the so-called watchdog) that monitors the processing state of the system.

Extending the Method — Beyond the Use Cases

We present the handling of an example not covered by the use cases in order to show how the method extends. By using the relaxation functionalities, which are provided by the dynamic help component, one could add constellations that handle anger concerning the factual result of a query. In this case, it is debatable if the observed anger must be directed to the system. It may also directed to the television program. We stipulated as a possible source of anger (or, even more likely, as a source of disappointment) the fact that the recall on a query contains no objects that are liked by the user.

```
anger(user, system)
    (The relevant facts that hold in this constellation and
    the actions that caused the facts are omitted)
    if
      wants(user, getObjectsAccordingToLikes)
      blameworthy(user, NONgetObjectsAccordingToLikes)

      (conditions(X):)
      (
       knows(system,dislike(user,X)) ,
       NONspecified(user,X),
```

```
      ResultContainsProminentCommonFeature (X))
     )
      or
     (
      knows(system,like(user,X)) ,
      NONResultContains (X))
    )

    (proposed system action:)
    relaxRequest
```

Case: Anger on Empty Result

This is a simple case:

```
anger(user, system)
   if
   wants(user, getNonEmptyResult)
   blameworthy(user, getNonEmptyResult)
   (conditions:) emptyResult
   (proposed system action:)
   relaxRequest
```

Emotions and Communicative Acts

Emotions that are signaled by facial expressions do not need to be accompanied by additional communication at all. Emotions expressed by voice are naturally related to some acoustic output. In the extreme, this output is only a container for the expressed emotion, but usually it contains a certain semantic content. The analysis of the relation between semantic content and underlying emotions is in its infancy, compared, e.g., with the relation between verbally communicated semantic content and pointing gestures. The latter is sufficiently known to build practical application. We distinguish in the following between communicative acts with semantic content, which are provided by speech and gestures, on the one hand, and emotions, on the other hand. The interpretation of pointing gestures and verbal utterances can be conceived as a fusion process that unifies pieces of information. Semantic contradictions between pointing gestures and verbally provided information are indications for errors. The relation between emotions and communicative acts is much more complicated. We give a presumably nonexhaustive classification of types of interaction between displayed emotion and communicated semantic content.

Redundancy

Semantic content redundantly expresses a simultaneously displayed emotion such as *that makes me angry* or *I'm glad about that*, or semantic content expresses an attitude that corresponds to the direction of the emotion (whether it is positive or negative) such as *great*, *bad*.

Contribution to the Explanation of the Emotion

Semantic content expresses a concrete attitude (like or dislike) that is involved in triggering the emotion such as *I don't like thrillers* or *great movies*, or semantic content addresses the facts and actions that caused the emotion such as *you didn't understand me* or *that takes too much time*, or simply by uttering *thrillers* accompanied by a positive or negative emotion.

The *thriller* example contributes the concrete feature that may fill the abstract goal of being presented accordingly preferences. But this example does not necessarily express a like or dislike as *great movies*. With a negative emotion, the example may also belong to the topic *Semantic Content as Repair Action*.

Semantic Content as Repair Action

The semantic information is provided to repair the state of affairs that has caused the emotional state of the user. The example *thriller* works also here: *thriller* could be a correction of a misunderstanding of genre. There is no direct relation between the content of the utterance and the displayed emotion.

This is very common and important in human–machine dialogue as well as in human–human dialogue: The dialogue partner repeats or reformulates her request and concurrently displays a negative emotion. With overt anger, it could also be expected that the user cancels the interaction as a final form of repair.

Change of Semantic Content

The user displays a negative emotion and communicates verbally a positive attitude such as *marvelous, great movies*. The direction of the valenced attitude that is communicated verbally is changed by the direction of the displayed emotion. This is a simple form of sarcasm.

8 Conclusion

A complete implementation of the whole processing chain was available at the end of the project. There was no opportunity for systematic tests, which require high effort. For instance, the recognition of facial expression needs careful preparation of the environment in respect to lighting conditions in order to work. Our limited testing shows that, provided recognition is correct, the emotion interpretation generates the reactions that are requested by the use case specification.

We successfully implemented a cognitive model-based approach for analyzing emotions and other affective states of a user who participates in a multimodal human–machine dialogue. This is a success, but it will still take considerable effort to make it practically useful. The approach is based on an elaborated theory, which covers a broad range of phenomena. This is promising with respect to the extensibility of the approach. It is an important advantage of the approach that it generates conceivable explanations of emotions that allow for well-directed system reactions.

The approach is not restricted to handle classical emotions, but extends to other affective states. Also it is not restricted to states that are displayed nonverbally. Affective verbal feedback, such as *I like this*, can be explained along similar lines.

References

J. Ang, R. Dhillon, A. Krupski, E. Shriberg, and A. Stolcke. Prosody-Based Automatic Detection of Annoyance and Frustration in Human-Computer Dialog. In: *Proc. ICSLP-2002*, pp. 2037–2040, Denver, CO, 2002.

A. Batliner, K. Fischer, R. Huber, J. Spilker, and E. Nöth. How to Find Trouble in Communication. *Speech Communication*, 40:117–143, 2003a.

A. Batliner, C. Hacker, S. Steidl, E. Nöth, and J. Haas. User States, User Strategies, and System Performance: How to Match the One With the Other. In: *Proc. An ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, pp. 5–10, Chateau d'Oex, Switzerland, August 2003b.

A. Batliner, V. Zeißler, C. Frank, J. Adelhardt, R.P. Shi, E. Nöth, and H. Niemann. We Are Not Amused – But How Do You Know? User States in a Multi-Modal Dialogue System. In: *Proc. EUROSPEECH-03*, vol. 1, pp. 733–736, Geneva, Switzerland, 2003c.

R.R. Cornelius. Theoretical Approaches to Emotion. In: *Proc. ISCA Workshop on Speech and Emotion: A Conceptual Framework for Research*, pp. 3–10, Newcastle, Northern Ireland, 2000.

C. Frank, J. Adelhardt, A. Batliner, E. Nöth, R.P. Shi, V. Zeißler, and H. Niemann. The Facial Expression Module, 2006. In this volume.

A. Ortony, G.L. Clore, and A. Collins (eds.). *The Cognitive Structue of Emotions*. Cambridge University Press, Cambridge, UK, 1988.

C.S. Peirce. Abduction and Induction. In: J. Buchler (ed.), *Philosophical Writings of Peirce*, pp. 150–156, Berlin Heidelberg New York, 1995. Springer.

R.W. Picard (ed.). *Affective Computing*. MIT Press, Cambridge, MA, 1997.

T. Portele. Interaction Modeling in the SmartKom system. In: *Proc. ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, Chateau d'Oex, Switzerland, August 2003. ISCA.

H. Prendinger, S. Descamps, and M. Ishizuka. Scripting Affective Communication With Life-Like Characters in Web-Based Interaction Systems. *Speech Communication*, 16:519–553, 2002.

W. Wahlster. SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: R. Krahl and D. Günther (eds.), *Proc. Human Computer Interaction Status Conference 2003*, pp. 47–62, Berlin, Germany, June 2003. DLR.

V. Zeißler, J. Adelhardt, A. Batliner, C. Frank, E. Nöth, R.P. Shi, and H. Niemann. The Prosody Module, 2006. In this volume.

Problematic, Indirect, Affective, and Other Nonstandard Input Processing

Michael Streit

DFKI GmbH, Saarbrücken, Germany
streit@dfki.de

Summary. Natural communication is accompanied by errors, misunderstandings, and emotions. Although in the last decade considerable research has evolved on these topics, human–computer dialogue applications still focus on the exchange of rational specifications of the tasks that should be performed by the system. In this chapter we describe a component that is devoted to processing problematic input that is characterized by the lack of a clear specification of the user's request. The following topics are discussed: interpretation of emotions and verbally communicated likes and dislikes, interpretation of indirect specifications, clarification of underspecified input, help on demand, robust navigation in multimodal dialogue and generation of error messages.

1 Introduction

Natural communication is accompanied by errors, misunderstandings and emotions. Consequently, human agents spend much effort in avoiding and clarifying misunderstandings, in ensuring the formation of common ground, and in accounting for affective states. Although, in the last decade considerable research has evolved on these topics (Traum, 1994; Hirst et al., 1994; Traum et al., 1999; Streit et al., 2004), human–computer dialogue applications still focus on the exchange of rational specifications of the tasks that should be performed by the system. The plan-based intention recognition approach, which does not demand direct specification of actions, nevertheless relies on utterances that may be recognized as rational contributions to a plan. Though the approach has been extended, e.g., for handling clarification subdialogues, it is not suited to process affective communication or to explain communicative reactions in absence of shared intentions. On the other hand, because the approach is difficult to realize, practical dialogue systems usually take speech acts as expressing the user's intention more or less directly.[1] On the *main road* of dialogue processing, the user is expected to provide specifications of her request in terms of the actions she wants to be performed or the goals she wants to achieve.

[1] Dialogue systems usually do not expect a complete specification. They ask for necessary parameters.

The conglomeration of attributes in the title of this chapter points to types of interaction, that are — more or less — *off the main road*, and are characterized by lacking a clear specification of what the user exactly is expecting from the system. Some topics presented in this chapter are rather close to the *main road*, such as clarification of underspecified input and the provision of help, others are further away from standard processing, such as the handling of emotions and indirect specifications. By the latter we understand requests where the agent does not specify an action or goal by providing their inherent parameters (e.g., the begin time of an entertainment). Instead she will specify her goal indirectly by relating it to other goals or objects.

Which topics are considered as belonging to standard processing is partly a design decision. But it seems to be clear that topics such as indirect specification or emotion interpretation need specialized knowledge and inference capabilities to be processed adequately, independent from being assigned to some *official* module or not. In SMARTKOM the issues listed below are handled in a special *nonstandard* processing component. There a several alternatives to integrate the processing of these issues, which is discussed in the next section.

The following topics are discussed in subsequent sections:

(1) Interpretation of emotions and verbally communicated likes and dislikes
(2) Interpretation of indirect specifications
(3) Clarification of underspecified input
(4) Help on demand
(5) Shallow model for robust navigation in multimodal dialogue
(6) Generation of error messages

The processing of emotion is presented in more detail in Streit et al. (2006), but we say something about the handling of verbally expressed likes and dislikes.

The processing of underspecified and indirect specification may be considered, at the first glance, as a similar problem. But the problems exist on different levels of analysis. In the first case we are faced with input that is incomplete and cannot be completed by analyzing discourse history. For instance, the user utters time expressions, which fits to many goal specifications, but none of them occur in the salient context. In this case we are looking for reasonable completions, and usually ask the user which alternative she has in mind, after checking if a misrecognition has been occured. In contrast, indirect specifications are complete intentions but cannot be mapped to a specification of a task directly. For instance, if the user asks the system to *show more crime movies*, the user delegates to the system to find out how to adjust parameters to the effect that more crime movies are found.

In the analysis of most of these topics, the consideration of context is much more important than the analysis of the semantic structure of the communication act, which is another common feature of the phenomena that are discussed here. This is also true for the topic of *multimodal dialogue navigation*, where the simple request to leave a subdialogue requires a thorough analysis of the multimodal dialogue structure to determine the state to return to.

2 Architectural Issues

There is a fundamental architectural alternative for implementing the handling of the topics listed above. One alternative consists in devoting an assistant component to care about events (input events in the first place) that cannot be handled on the standard strand of processing. The assistance is consulted if a certain input cannot be processed on the main strand. This is the architecture chosen in SMARTKOM (with the exception of the handling of pure emotions, where the nonstandard component takes initiative).

The other approach installs a metadialogue component that inspects every input event and its processing — including the events processed on the main strand — whether there is some indication to modify the processing (Purang et al., 2000). Normal processing could be canceled, e.g., if a misunderstanding is suspected, if the result is such that it probably does not satisfy the user's wishes, e.g., if it is empty, or if the communication act shows a certain emotional color which indicates that it should not taken literally.

In graphical user interfaces, there is the possibility to handle (at least some of) the issues mentioned in the introduction as a parallel strand of interaction, e.g., by devoting a special window to them. This approach does not seamlessly fit into the paradigm of dialogue (though in a multimodal system the approach is possible in principle). One could speculate about transferring this approach from graphical use interfaces to multimodal dialogue by introducing several artificial agents that tak care of different aspects of the interaction. But this far beyond the scope of SMARTKOM, which, e.g., lacks the turn-taking mechanism that would be needed if independent specialized agents interact. We only risk in the processing of pure emotion that the nonstandard component takes initiative independently.

In the metadialogue approach, it is basically the metadialogue component that decides whether there is a situation that needs special attention or not. In the other approach the standard processing dialogue component has to overtake the position to decide whether a problem occurs and when the subdialogue that handles the problem is finished.

For a self-contained understanding of this chapter, we sketch in one sentence the *main road* processing in SMARTKOM: The user provides a probably ambiguous or elliptic and even noisy multimodal specification of her goal and its parameters (i.e., the slots that belong to the goal). The user's specification is mapped to a formal specification of the task. This semantic specification is enriched by inheriting information from the discourse memory. The best scored hypothesis is taken as a specification of the task that has to be done. Input that does not represent an application goal or cannot be processed for technical reasons is delivered to the dynamic help module, which is the name of the module devoted for processing nonstandard input events.

The SMARTKOM approach brings about a formally strict control and clear division of initiative. It is decided by the dialogue management (i.e., the *action planning* module) if user input is in some sense nonstandard. The problem of this approach is that the knowledge about critical situations is situated in the nonstandard component.

Therefore, the responsibility for handling events is de facto based on a distinction of the structure and semantic content of the input. If it specifies an application task,[2] the action planner will handle it; otherwise it either addresses a nonstandard task, e.g., a help demand or an indirect specification, or it is incomplete (disregarding ill-formed representations).

In the SMARTKOM architecture, we must solve the problem of how the flow of processing resulting from subdialogues on the nonstandard strand can be controlled. We discuss this problem in Sect. 5.

3 Verbally Expressed Likes and Dislikes — Preference Subdialogues

Preferences for specific genres or actors are regarded by the presentation planning component.[3] Objects that contain features, which are explicitly preferred by the user, are presented at the start of a presentation, and features that are explicitly disliked are presented at the end. If the user verbally expresses a *like* or *dislike*, the semantic representation is passed to the nonstandard component. If the message explicitly addresses a feature that can be handled by the system's preference management (e.g., *I like crime movies*), the component will generate the offer to present objects in the future according to the preference. The system asks if it should store this positive or negative preference. If the dynamic help module detects that the preference is already stored, it does not newly generate a preference dialogue.

Verbally expressed likes and dislikes are closely related to positive or negative emotions, but they are not necessarily accompanied by an emotion (either the user has no emotion or displays no emotion). This is in agreement with the OCC model of emotions (Ortony et al., 1988), that we use for the interpretation of emotion. For instance, a dislike triggers anger if the user *blames the system* for showing her disliked objects, which is not necessarily the case. But even if the user does not blame the system for ignoring her (up to now) unknown preferences, it may promote further joy if the system proposes to regard her preferences in the future.

The problematic case is the handling of unspecific utterances such as "nice" or "I like this". On the surface, the unspecific case is elliptic or anaphoric. The identification of the objects of a like or dislike is not resolved by the SMARTKOM discourse analysis. The resolution is rather taken as a process of inferring an explanation of the positive or negative feedback, as we process in the case of pure emotions.[4] This is achieved by an abductive reasoning as is explained in more detail in the chapter on emotions. As with pure emotions, we use genres as the default feature for preferences.

[2] Including certain system-related tasks.

[3] Other possibilities to react on the user's likes and dislikes are discussed in Streit et al. (2006).

[4] In fact, the process of explanation may be interwoven with another anaphora resolution process, if one considers examples such as *I like this actor*.

4 Interpretation of Indirect Specifications (Relaxation)

Indirect specification, which is usually not an indirect speech act, is rather common in natural communication. For instance, it is quite normal to specify a goal not by its inherent parameters (which correspond in the SMARTKOM system to the slots of an object of the system's ontology), but by demanding that it obeys a certain relation to other objects or events. This leaves it to the addressee of the indirect (and usually vague) request to figure out appropriate actions for achieving the goal. We discuss indirect specifications that contain *vague* goals concerning database queries. These requests are basically handled by relaxation and occasionally by adding additional specifications which are provided by the user.

USER: Which crime movies are on TV this afternoon?
SYSTEM: *Shows the crime movies running this afternoon.*
USER: *(indirect request)* More of these!
SYSTEM: *System checks if there are any crime movies that afternoon on TV, which are not presented yet. Otherwise it relaxes the query specification*

In the example above, the user's indirect specification contains a reference to the set of results presented to the user beforehand. The specification adds up to extend this set. The user does not specify how this extension should be done, but there is a vague plea to find objects that are *similar* to the objects in the set.

We distinguish two basic reactions on the request. The first reaction only checks if there are any crime movies which are not presented yet. But there may be no more crime movies left. A more cooperative reaction is to extend the set by relaxing the specification. *Crime* is explicitly specified, therefore this genre is not a feature that should be relaxed. We use modification of time intervals as a default operation for achieving a modified result, which is similar to the objects that are already found.

There is another dimension, in respect to which *more* can be interpreted in two ways: as asking for a super set of the original set or as asking for additional items only. Taking into account the limited space for presentations, we take the second interpretation, by moving the time interval instead of extending it.[5]

The user may also combine an indirect request for relaxation with an additional direct specification:

USER: What's on TV this afternoon?
SYSTEM: *Shows a part of the afternoon program, containing one crime movie.*
USER: Show me more crime movies.
SYSTEM: *System checks if there are more crime movies that afternoon on TV, which are not presented yet. Otherwise it relaxes the query specification on the one hand and makes it more specific at the other hand.*

The dynamic help module constructs an new specification from the indirect (relational) request and the specification of the former request by using relaxation and

[5] This is a decision which is appropriate for SMARTKOM, but may be wrong for other systems in certain situations.

default unification [6] Query relaxation, as is described in the next section, is performed first. Default unification is used to add additional specifications provided by the user.

4.1 Query Relaxation

The concept of query relaxation is mainly explored in the context of relational databases and logic programming (Gaasterland et al., 1994). The basic operations are:

(i) rewriting a predicate with a more general predicate
(ii) rewriting a constant with a more general constant

We perform relaxation on constants, and also on sequences of constants and intervals, but not on predicates. Hence we distinguish three cases:

- atoms
- lists
- intervals

In SMARTKOM, predicates correspond to nonatomic types, constants correspond to atomic types. There is no hierarchy on constants (or atoms) in SMARTKOM. It is not possible to exchange atoms by more general atoms. Hence atoms may be deleted or exchanged in special cases. For lists, relaxation by deleting an element is an option. But experience with the list-valued features in the TV and cinema domain led us to choose the deletion or exchange of lists instead. The only intervals relevant for TV and cinema database queries are time intervals. As mentioned, intervals are moved rather than extended. This modification accounts for the *additional values* reading of more. This reading cannot achieved in general for atoms and lists.

5 Clarification of Underspecified Input

Semantic multimodal analysis in SMARTKOM delivers a set of alternative interpretations. The alternatives are either specified starting from the top of the ontology (*full* structures), or they are substructures that may be completed in many ways and may even be inconsistent. Discourse analysis extends these structures, but does not delete alternatives. In case of full structures, the intention recognition module selects one alternative as being the intention of the user. In case of substructures the intention module delivers a set of alternatives. Substructures may be completed in many ways, and sets of substructures may even be inconsistent. The main topic in this section is the handling of substructures.

Substructures are typically instantiation of types that are placed at a *low* level in the hierarchy. Substructures come either from incomplete parses, i.e., from partial

[6] Default unification as is used in the dynamic help module is different from the method used in the SMARTKOM discourse analysis and is described in Streit and Krieger (2004).

understanding, or from deliberately underspecified or highly elliptic user input. In the first case, misrecognition is a possible source of the partial understanding. The following utterances are examples of underspecified input:

(1) *Thrillers!*
(2) *Today evening!*
(3) *Heidelberg!*

In example 1 it is unclear if the user wants to get information about the TV program or if she is interested in cinema performances. In example 2 there is the same ambiguity.[7] In example 2 there is another ambiguity concerning the role of the substructure: It is unclear if the user refers to the begin time of an event or to the end time (or perhaps both). Similarly in example 3 it is unclear if, in the context of route planning, Heidelberg is the target or the start of the route. The role ambiguity in example 2 is solved by taking begin time as default. In example 3 the role ambiguity is considered as a topic of clarification dialogue.

In the first place, substructures are ambiguous with respect to the goal that the user wants to achieve by the help of the system. We understand *Goal* as it is used in the action-planning component. There the concept goal corresponds roughly to processes in the ontology (which is used for semantic representation). But a process may correspond to several finer-graded goals. Which goal comes into question depends on the specific substructures of the process.[8] For instance, an information search process concerning the TV program is assigned to a TV program browsing goal, while information search on cinema performances is assigned to a different goal. In most cases there are no role ambiguities or these are resolved by defaults. But in certain cases the role ambiguity is relevant (route planning).

5.1 Completion of Substructures

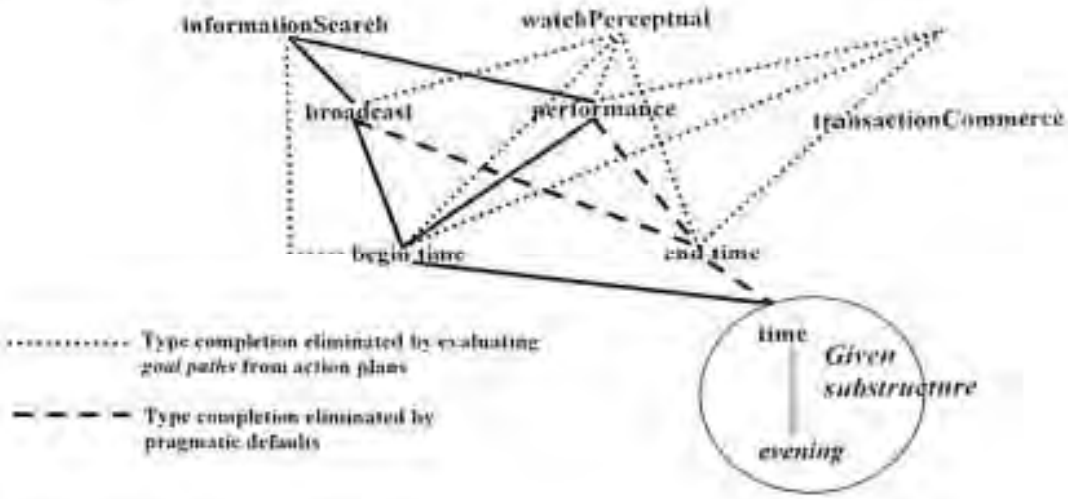
We want to generate the structures that are meaningful in terms of the goals that the user can pursue in the system. We want to achieve a minimal full structure that uniquely determines a goal. We do not add atomic values, except if this is necessary for determining the goal. The set of all full structures that contain the substructure has many members that represent no goal (in the sense of the action planning component). On the other hand, full structures that contain the substructure may be underspecified with respect to goals.

Technically, goals are related to ontological structures by paths of slots and types that must be contained in a structure to represent a goal. A path consists of slots and types; each slot is followed by the type of the fillers of the slot.[9] Though we

[7] In respect to the ontology there are many more ambiguities, but in respect to the possible applications these are the relevant ones.

[8] Alternatively, a process may be an abstract type, which has concrete subtypes that correspond to goals.

[9] This translates to feature structures if we take features instead of slots.



Relevant goal paths (simplified)

- informationSearch/broadcast/
- informationSearch/performance/

Alternative options, derived from the substructure (simplified)

- informationSearch/broadcast/beginTime/evening
- informationSearch/performance/beginTime/evening

Fig. 1. Ontology-based completion of substructures

distinguish here slots and types, they are not distinguished on the level of the internal representation of the ontology, due the internal definition of types by XML schemata.

Similarly, the parameters of a goal are characterized by paths. As a first step, we match parameter paths with substructures resulting in a parameter full structure by regarding defaults. Then we unify the minimal structure that is consistent with the goal path with the parameter structure.

In Fig. 1 we present a little cut-out of the ontology. The presented branches of the ontology are further simplified by omitting certain types and slots. Figure 1 shows how goal paths and defaults eliminate structures that are licensed by the ontology. If multiple substructures are present, we search for maximal consistent subsets of parameter structures and goal structures.

5.2 Clarification Subdialogue – Guided Interaction vs. Free User Initiative

Because underspecification is a possible indication for errors, we follow a cautious strategy[10] and first ask the user if the system understood correctly what she said, as is shown in Fig. 2.

[10] The dialogue strategy is determined in subdialogue plans of the dynamic help component and can easily be modified.

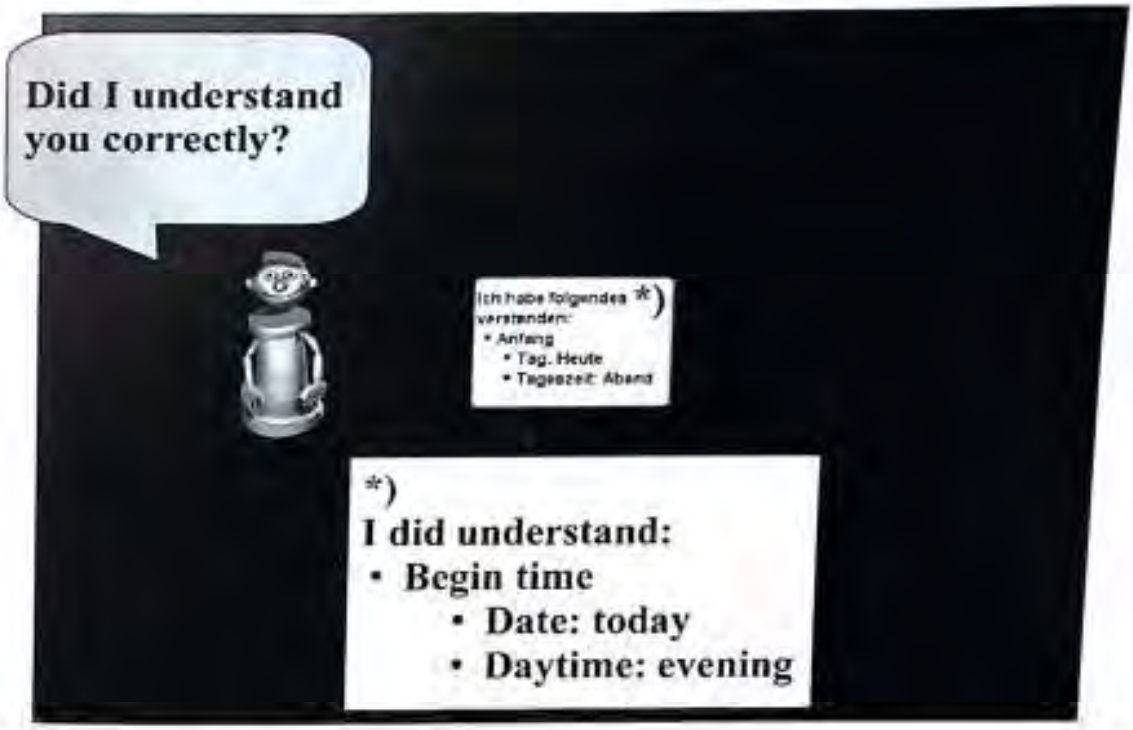


Fig. 2. First step of the substructure clarification dialogue

The user may react by stopping the subdialogue or by rephrasing her request. The user may correct the analyzed substructure by rephrasing. If there are more than one substructures she may select one by rephrasing it.

If the user has confirmed the analysis, the goals are presented that are compatible with the substructure (Fig. 3). In case of a role ambiguity we present alternative instantiations of the goal (Fig. 4).

We aim in performing the clarification subdialogue in a way that combines features of a guided dialogue paradigm with free user initiative. The user may:

(1) take the presented options as a menu to choose from
  (a) by pointing
  (b) by selecting the alternative by speech
(2) restate or modify her request
(3) express a new unrelated request
(4) stop the subdialogue

Stopping the subdialogue is processed by the dialogue navigation facility, as is presented later in Sect. 7. Alternative options are presented graphically by buttons. This is different from objects that are intended to be used by deictic pointing, which is normally accompanied by speech (Streit, 2001). The user may recognize from the graphic design that she may use them in a direct manipulative style.

Underspecification occurs if there is not sufficient context to solve the input fragment by discourse analysis. There is either no context at all, or discourse analysis has

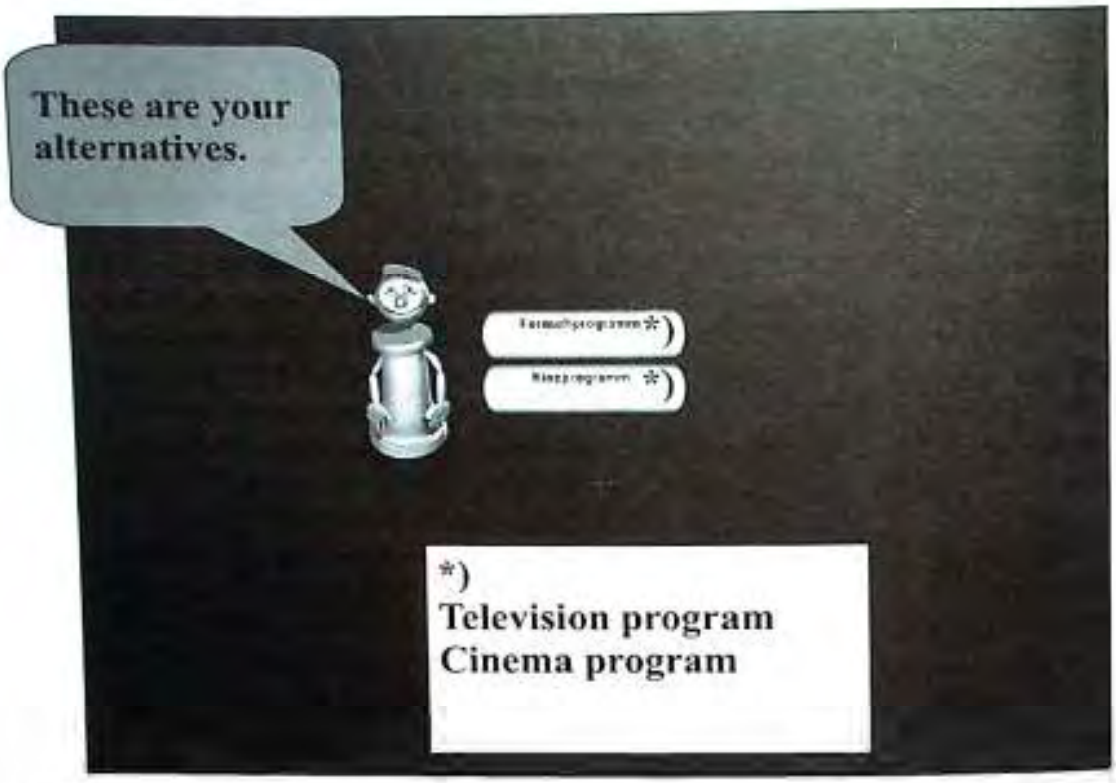


Fig. 3. Second step — presenting goals that are compatible with the substructure

decided that the substructure does not fit to any salient context. In that, it is usually assumed that any goal that is compatible with the substructure is a reasonable option, provided that the user has confirmed that the system did understand her utterance correctly. There are some exceptions to this rule. If the system has requested an answer of the user, dynamic help reacts with a repetition of the request, rather than with an analysis of the under-specified substructure. Also, the coverage of the scenarios is considered. In the mobile system only substructures that fit to functionalities in this scenario are considered.

5.3 Processing of Clarification Subdialogue

The dynamic help component provides full semantic representations of the alternative, including the parameters that are due to the substructures. These representations are published as expectations, to allow the analysis of spoken input by matching it with these representations. If the user chooses an alternative, she will probably not specify the whole content of the alternative. She will instead provide elliptic information that is suited to select an alternative. The representations of the alternatives are further directly assigned to the graphical presentations. This is more convenient to use than expectations. It enables the multimodal fusion component to provide a semantic analysis directly, which unburdens discourse analysis.

5.4 Problems and Coverage

Help on underspecification is realized for the home scenario and the mobile scenario. Since the public and the home scenarios are often presented together, the provision of help then appears incomplete. The presentation of alternatives in the mobile scenario shows how the substructures are integrated in the goal parameters of that goal (Fig. 4). This is neglected in the home scenario, which makes the alternatives less understandable. If many different substructures are recognized, it would be valuable if the user could choose alternatives, not only on the goal level, but also on the level of substructure understanding by pointing.

6 Help on Demand

There are three basic types of information that should be provided by help on demand. These basic types are:

- goals to achieve
- slots or roles that further specify the goal
- values that serve as fillers for slots

We focused on the provision of goal information in the mobile scenario and on value information in the home scenario, neglecting the information about slots.

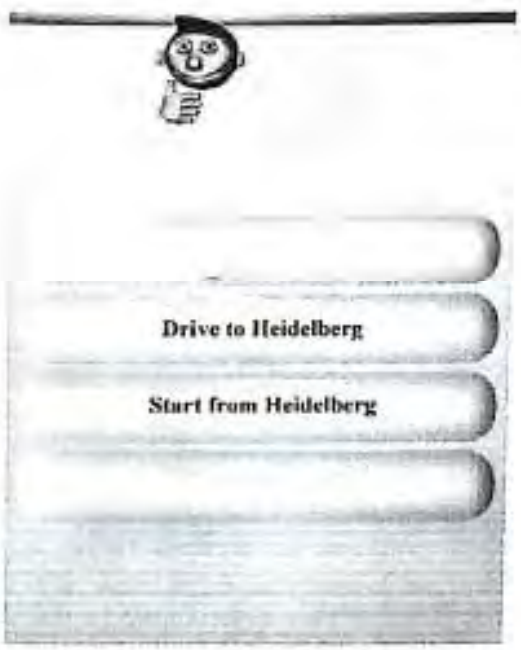


Fig. 4. Alternative readings of Heidelberg

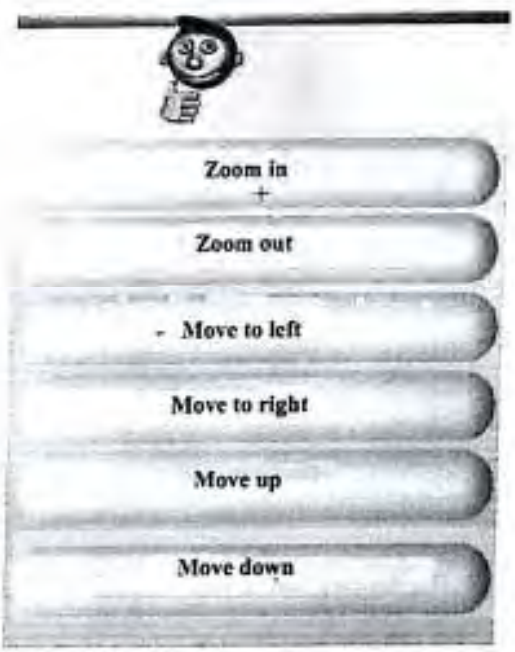


Fig. 5. Goals for navigating a map

6.1 Presentation of Goals

In SMARTKOM, the user can change her intention completely in almost any situation. In this respect, there is no context that strongly calls for presenting specific options, if the user asks nonspecifically for help. But there are certain goals that can always be performed as route planning, and others that can be performed only in a certain context, e.g., manipulating a map is only possible if there is a map to be manipulated. We call the first class of goals unconditioned goals, and the second class (pre)conditioned goals.

In the mobile scenario we realized the following strategy for the presentation of goal options: Conditioned goals are only presented if the preconditions are satisfied. In this situation conditioned goals are preferred. Unconditioned goals are not excluded, but are presented later, e.g., if the user repeats asking what she can do. Figure 5 shows the options for navigating in a map.

The conditioned vs. unconditioned classification is not a factual one, but is a classification in respect to the abilities of the main strand processing. For example, requesting a telephone call by specifying the name of the callee instead of her number needs access authorization to an address book. In this case the main strand cares about authorization and telephone number retrieval. If the user asks for reservation seats for a performance, standard processing does not care about identifying a unique performance, but it cares about identifying definite seats, by providing a seating plan.

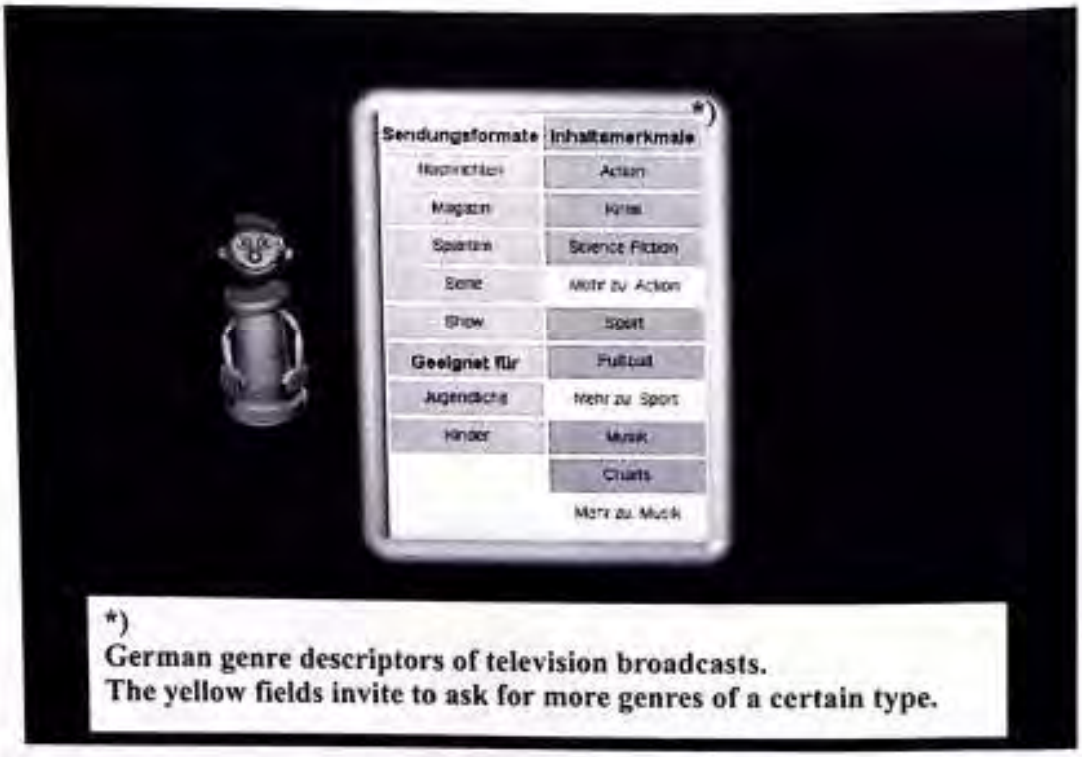


Fig. 6. Presentation of options for genres

6.2 Presentation of Values

Values are shown on explicit demand. Figure 6 shows the presentation of values for the role or type genre.[11] If the user asks nonspecifically for genres, the system provides genres for movies in a cinema program context and genres for TV broadcasts in a television context.

7 Dialogue Model for Navigation in Multimodal Dialogue

In a multimodal dialogue system, navigating is different in some respects from navigating in a pure speech dialogue system. If a subdialogue is finished in speech dialogue system, there is no artifact left. But in a multimodal system subdialogues leave artifacts on the screen. If the user wants to finish the subdialogue, these artifacts have to be removed. This is especially true if the user wants to finish the subdialogue not by proceeding with another topic, but, e.g., because she wants to see again some relevant information that was present before the subdialogue. It is obvious that restoring the presentations that belong to the state to which the user wants to return cannot be done by just restoring the state that precedes the current state. The subdialogue may consist of several successive presentations, the subdialogue may be preceded by another subdialogue, and subdialogues may be nested. Therefore a dialogue model is maintained that represents the subordination of dialogue parts. It was necessary to generate this model in a very short time. A shallow but robust model, which works very successfully, has been achieved by basically categorizing presentation acts into four classes: main contribution, information subdialogue contribution, help information, and clarification, and by imposing a subordination relation to these classes.

8 Generation of Error Messages

The generation of an error message is the default reaction to input that is not processed otherwise. In an earlier version of the system we distinguished error messages with respect to the stage of processing, where the error occured. For example, errors that occurred on the recognition level have been prompted by *I could not understand you acoustically*; errors on later levels of analysis have been prompted by *I could not understand you*. If the goal of the user had been analyzed successfully, but the actions to achieve this goal could not be identified or performed, this has also been commented upon. Also, the possibility that a result could not be presented has been considered. The stages where the error occurred are identified by the system's *watchdog*, which monitors the processing states of the module and infers an overall system state from the states of the single modules. Later, these error messages have been unified to the generic answer *I did not understand you*, except for presentation failures, which are commented by *an error has occurred*.

[11] The distinction between type (as filler of a slot) and slot (which is filled by items of a certain type) is not clearly marked in the internal representation of the ontology, but it also is not distinguished by the user as long as the relation between role and type is one-to-one.

9 Conclusion

Problematic, indirect, attentive nonstandard input has been explored by considering an extraordinarily broad range of phenomena. For certain problems, such as underspecified input, general solutions are developed. For some aspects, such as handling emotions, attentive user states, or indirect specifications, principled approaches are developed and implemented for a restricted range of phenomena. For other topics, such as navigation in multimodal dialogue, shallow but robust solutions are developed. One conclusion of the work is that architectural issues are relevant and may facilitate or impede a principled general approach for error recovery.

References

T. Gaasterland, P. Godfrey, and J. Minker. An Overview of Cooperative Answering. In: *Nonstandard Queries and Nonstandard Answers*, pp. 1–40, Oxford, UK, 1994. Clarendon Press.

G. Hirst, S. McRoy, P. Heeman, P. Edmons, and D. Horton. Repairing Conversational Misunderstandings and Non-Understandings. Technical report, University of Maryland, 1994.

A. Ortony, G.L. Clore, and A. Collins (eds.). *The Cognitive Structue of Emotions*. Cambridge University Press, Cambridge, UK, 1988.

K. Purang, D. Traum, D.V. Purushothaman, W. Chong, Y. Okamato, and D. Perlis. Meta-Reasoning for Intelligent Dialog Repair. Technical report, University of Maryland, 2000.

M. Streit. Why Are Multmodal Systems so Difficult To Build? About the Difference Between Ceictic Gestures and Direct Manipulation. In: *Cooperative Multimodal Communication*, Berlin Heidelberg New York, 2001. Springer.

M. Streit, A. Batliner, and T. Portele. Cognitive-Model-Based Interpretation of Emotions in a Multi-Modal Dialog System. In: *Proc. ESCA Tutorial and Research Workshop on Affective Dialog Systems*, Irsee, Germany, 2004.

M. Streit, A. Batliner, and T. Portele. Emotion Analysis and Emotion Handling Subdialogs, 2006. In this volume.

M. Streit and H.U. Krieger. Ellipsis Resolution by Controlled Default Unification for Multi-Modal and Speech Dialog Systems. In: *Proc. ACL Workshop on Reference Resolution and Its Applications*, Barcelona, Spain, 2004.

D. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, University Rochester, 1994.

D.R. Traum, C.F. Andersen, W. Chong, D. Josyula, Y. Okamoto, K. Purang, M. O'Donovan-Anderson, and D. Perlis. Representations of Dialogue State for Domain and Task Independent Meta-Dialogue. In: *Proc. IJCAI'99 Workshop on Knowledge And Reasoning In Practical Dialogue Systems*, 1999.

Part IV

Multimodal Output Generation

Realizing Complex User Wishes with a Function Planning Module

Sunna Torge* and Christian Hying**

Sony International (Europe) GmbH
Sony Corporate Laboratories Europe, Advanced Software Laboratory, Stuttgart, Germany

Summary. Recently, spoken dialogue systems became much more sophisticated and allow for rather complex dialogues. Moreover, the functionality of devices, applications, and services and the amount of digital content increased. Due to the network age (Internet, personal area networks, and so on) the applications one can control or wants to control change dynamically, as does the content. Therefore a dialogue system can no longer be manually designed for one application. Instead, a layer in between is required, which translates the functionalities of the devices such that they can be used via a generic dialogue system and which also resolves complex user wishes according to available applications. In this paper a module, called the function planning module, is described which builds a link between a dialogue system and an ensemble of applications. A planning component searches for the applications necessary to solve a user wish and the sequence of action which has to be performed. A formal description of the function planning module and its successful integration in the SMARTKOM prototype system are presented in this paper as well.

1 Introduction

Although human–machine interactions are ubiquitous, they are not always a very pleasing experience. An often-cited example is how inconvenient and error-prone it is to program a video cassette recorder (VCR). Another example is that currently users have a lot of trouble when they want to connect Bluetooth devices. The main problem with most of the user interfaces is that the user has to think in terms of devices and services (play, record). Instead, the user should be able to interact with a system in a natural way, as he would do with a human assistant. To give an example, the user might put a document on the table and simply say, "Send this to Mr. Green." In consequence, the system should deal with the devices and services for him. It should scan the paper with a document camera, find out how to reach Mr. Green by consulting an appropriate address book, and transmit the image either to an e-mail

* S. Torge is now with the Institute for Computer Science, Universität Freiburg, Germany; torge@informatik.uni-freiburg.de

** C. Hying is now with IMS, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Germany; christian.hying@ims.uni-stuttgart.de

client or a fax modem, possibly doing format conversion beforehand, etc. The user in this case has a complex wish involving several devices and several actions per device. He simply wants his wish to be solved; he does not want to care about the individual devices or the action sequences to be carried out. Similar problems arise in the case of copying videos or combining EPG access and personal calendars. Note that these tasks can only be solved by a system providing a central human–machine interface operating several devices.

Given a home network or personal area network, the problem we addressed within the SMARTKOM project is to enable a flexible and intuitive control of an ensemble of devices and services with a single intuitive human–machine interface. We focused on

- how complex user wishes can be served, and
- how plug and play can be realized on the side of the dialogue system.

In order to solve these problems different aspects need to be considered:

A traditional dialogue system used for the control of devices usually consists of an input understanding part, a dialogue manager, and the devices that are to be controlled. The simplest way to control the devices is to have a unique mapping of the user input to the appropriate control command. Given, e.g., a speech input "CD play", it can be uniquely mapped to the "play" *command* of a CD player (Rapp et al., 2000). However, this approach does not allow us to serve complex wishes like the above-mentioned examples, since the user input for complex wishes cannot be uniquely mapped to a single control command.

In order to gain more flexibility with respect to adding new applications, it is necessary to separate domain-dependent knowledge from domain-independent knowledge (Flycht-Eriksson, 2000; Thompson and Bliss, 2000; Han and Wang, 2000; Lemon et al., 2001).

Separation of domain-dependent and domain -independent knowledge can be obtained by introducing an additional module that serves as an interface between the dialogue manager and the devices to be controlled, including models of the devices (Pouteau and Arevalo, 1998). However, to serve complex user wishes it is not enough to consider models of single devices. Instead, some sort of reasoning on the provided services is necessary (Torge et al., 2002b,a; Heider and Kirste, 2002).

In Larsson et al. (2001), the use of precalculated plans for complex tasks is described. With this approach, however, the user is limited to those tasks where precalculated plans are available.

Concerning plug and play, there are existing physical solutions like Bluetooth and IEEE 1394 (e.g., iLINK, Firewire, see Hoffman and Moore (1995)), which offer plug and play on a hardware level. This has also to be reflected in the human–machine interface since new devices have to be controlled through the human–machine interface.

In order to overcome these sorts of problems within the SMARTKOM project, a multimodal dialogue system is enhanced with a planning component, called the function planning module. The function planning module consists mainly of a set of abstract models describing the functionalities of devices and services available

in the network and a reasoning component. Using these abstract models the human–machine interface can infer how to control the available services and devices. Henceforth, we use the term "devices" instead of "devices and services" for reasons of brevity. This article is organized as follows: First, we give an overview about the function planning module, which is followed by a detailed description of the single parts of the function planning module and a formal description of the concept. The next part of the article focuses on the realization of the function planning module within the SMARTKOM prototype system. We conclude with a section about future work.

2 The Function Planning Module

2.1 Overview

As described above, our focus is on serving complex user requests which may involve several devices. In order to handle requests like "Please record the film XYZ on Saturday," the system needs to find out

- *which* devices are necessary to serve the request
- *how* to control the devices

For this purpose a planning component and appropriate data structures for planning are needed.

We introduce a new module, the so-called function planning module, in order to solve the above described problems. The purpose of the function planning module is to allow the integration of the functionalities of several different devices and to support plug and play. In a traditional dialogue system (von Kuppevelt et al., 2000) the devices to be controlled by the system are controlled by the dialogue manager (Fig. 1). In contrast, the enhanced system (Fig. 2) provides a sort of intelligent interface between the dialogue manager and the devices to be controlled, namely the function planning module. The function planning module consists of

- an abstract model of the functionalities of a device (the so-called functional model) for each device in the network
- a reasoning component
- a plan processor

Furthermore, the function planning module is always aware of the current state of each device in the network.

Instead of controlling the devices directly from the dialogue manager, this allows us to formulate the requests given by the dialogue manager on an abstract level. Based on this abstract request, the function planning module first calculates a plan to serve the request and then performs the plan.

The dialogue manager is therefore independent of the real devices and is robust against changes of devices. The overall functionality of the given devices does not

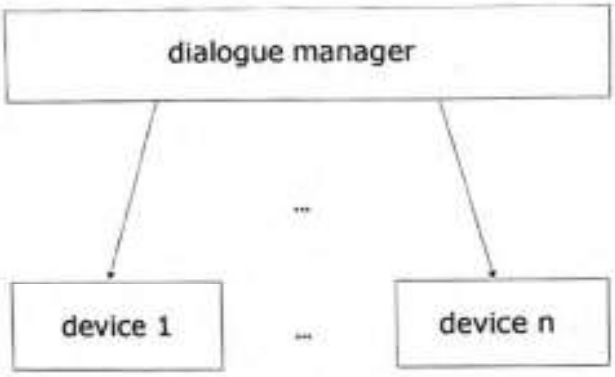


Fig. 1. Traditional dialogue system

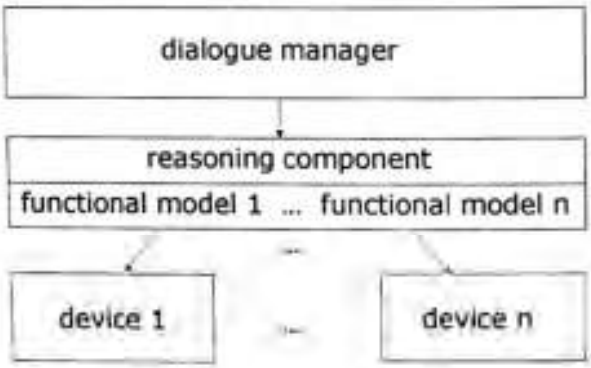


Fig. 2. Dialogue system with function planning module

need to be known in the dialogue manager, but it is deduced from the functional models of the given devices. This approach enables the system to serve complex user requests, and to be flexible and robust against changes in the network of controlled devices, i.e., it supports plug and play. In the following, we give a detailed description of the different parts of the function planning module, i.e., the functional model, the reasoning component, and the plan processor.

2.2 The Functional Model

The functional model is a two-layered model. It consists of an external model and an internal model, where the external model describes the knowledge necessary to find appropriate devices and the internal model describes the single device in more detail. The external and internal models are connected; however, they are used separately. This approach was chosen in order to reduce the search space.

2.2.1 External Model

The external model of a device models the knowledge that is necessary but also sufficient to decide whether a device is appropriate to serve a given request or not. Fur-

thermore, the external models of the devices allow the system to draw conclusions on which devices are to be combined in order to fulfill a request. The external model of a device describes the input data and the output data of the device. It is modeled as a binary relation. Each external model of a device corresponds to a functionality of the device. Note that a single functional model may consist of several external models, since one device may provide several functionalities. As an example consider, a VCR, which provides the functionalities "recording" and "playback." The external models corresponding to "recording" and "playback" are depicted in Figs. 3 and 4, respectively. However, a VCR as a single device is not capable of recording a film. A second device, e.g., a tuner, is necessary in order to provide the data to be recorded. That is, the user wish "record the film" is a complex wish, such that the system needs to combine two devices, namely a tuner and a VCR, in order to fulfill the wish. This information is encoded in the external models of the tuner and the VCR.

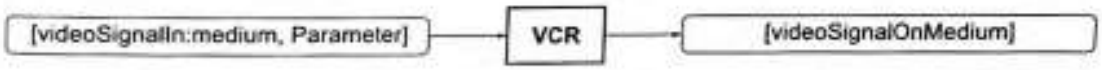


Fig. 3. External model of the VCR describing "recording"

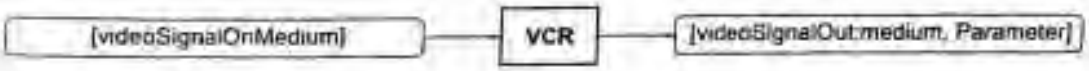


Fig. 4. External model of the VCR describing "playing"

2.2.2 Internal Model

The internal model of a device describes the possible states of the device and the actions that are necessary to bring the device from one state into another. Regarding the knowledge to be modeled, i.e., states and actions, finite state machines (FSM) are an appropriate means to build up an internal model.

The states in the internal model are partly annotated with incoming or outgoing data, respectively. This is an important point since these annotations ensure the connection between external and internal models of a device. In addition, the states also might be annotated with pre- or postconditions of each state. The internal model of a VCR is depicted in Fig. 5.

2.2.3 Formal Description of the Functional Model

The functional model can be formally described as follows:

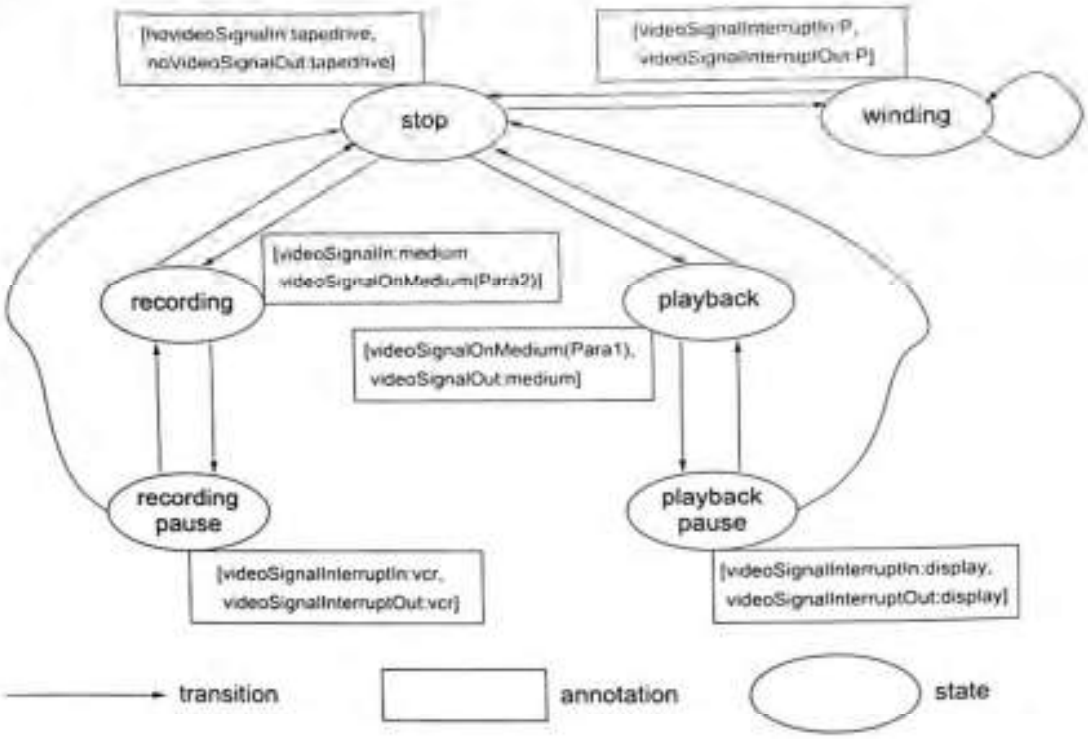


Fig. 5. Internal model of the VCR

Definition 1. (Functional Model)

A functional model $\mathcal{F}$ *is a tuple* $\mathcal{F} = (\mathcal{D}, \mathcal{S}, \mathcal{T}, i, o, pre, post)$ *where* $\mathcal{D}$ *defines the external models and* $\mathcal{S}, \mathcal{T}, i, o, pre, post$ *define the internal model.*

- $\mathcal{D}$ *is a finite set of* external models *with*

$$\mathcal{D} = \{(d_j^i, d_j^o) \mid d_j^i \in \mathcal{D}^i,\ d_j^o \in \mathcal{D}^o\}, \mathcal{D} \subseteq \mathcal{D}^i \times \mathcal{D}^o$$

- $\mathcal{S}$ *is a finite set of* states.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ *is a* transition function *and* $\mathcal{A}$ *a set of* actions *(including the empty action).*
- $i, o, pre, post$ *are* functions, *which informally spoken map additional information to a state* $s \in \mathcal{S}$. *They are defined as follows, where* ε *is the empty element:*

$$i : \mathcal{S} \rightarrow \mathcal{D}^i \cup \{\varepsilon\}$$
$$i : s \mapsto d^i$$
$$o : \mathcal{S} \rightarrow \mathcal{D}^o \cup \{\varepsilon\}$$
$$o : s \mapsto d^o$$

Let $\mathbf{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ *be a finite set of functional models with* $\mathcal{F}_i = (\mathcal{D}_i, \mathcal{S}_i, \mathcal{T}_i, i_i, o_i, pre_i, post_i)$, $i = 1, \ldots, n$ *and* $\mathcal{F} \in \mathbf{F}$, $\mathcal{F} = (\mathcal{D}, \mathcal{S}, \mathcal{T}, i, o, pre, post)$:

Let $\mathbf{D^i} = \bigcup_{j=1}^{n} \mathcal{D}_j^i$ *and* $\mathbf{D^o} = \bigcup_{j=1}^{n} \mathcal{D}_j^o$. *Then pre and post are defined as*

$$pre : \mathcal{S} \rightarrow \mathcal{P}(\mathbf{D^i} \times \mathbf{D^o}) \cup \{\varepsilon\} \times \{\varepsilon\}$$
$$pre : s \mapsto \{(d_j^i, d_j^o) \mid j = 1, \ldots, l, d_j^i \in \mathbf{D}^i, d_j^o \in \mathbf{D}^o\}$$

$$post : \mathcal{S} \rightarrow \mathcal{P}(\mathbf{D^i} \times \mathbf{D^o}) \cup \{\varepsilon\} \times \{\varepsilon\}$$
$$post : s \mapsto \{(d_j^i, d_j^o) \mid j = 1, \ldots, k, d_j^i \in \mathbf{D}^i, d_j^o \in \mathbf{D}^o\}$$

Note that the definitions of i and o only refer to the external models which are part of the functional model to be defined, whereas *pre* and *post* refer to the external models of all other functional models under consideration. Note furthermore that the definition of pre- and postconditions, namely *pre* and *post*, corresponds to the definition of the external models.

2.3 The Reasoning Component

Each of the devices of the given network is modeled by its functional model. In order to use this knowledge, the function planning module includes a reasoning component. This is necessary since, given a complex user request and given a network of devices, the following steps need to be executed in order to serve the user request:

1. Find out the device/devices which are necessary to perform the request.
2. Find out how to combine and to control the devices, i.e., generate an appropriate plan of actions to be executed.

Using the functional model of each device in the given network, the reasoning component first searches for the necessary devices and then generates a plan of actions for every device that is involved.

The functional models allows us to describe how devices can to be combined. This is done via the input and the output data of each device. Furthermore, it is possible to describe a temporal order in which the devices are to be controlled. This is done by formulating pre- and postconditions of single states of a device. Let us have a closer look to the algorithms.

In a first step, a complex task T is mapped to a tuple $(d_T^i, d_T^o) \in \mathbf{D^i} \times \mathbf{D^o}$ (see Definition 1). Let $\bigcup_{j=1}^{n} \mathcal{D}_j$ be the set of all external models which are currently available. Then, using the representation (d_T^i, d_T^o) of the task, the device search algorithm returns a set of external models $\mathcal{D}^T$ of the necessary devices:

$$\mathcal{D}^T \subset \bigcup_{j=1}^{n} \mathcal{D}_j, \mathcal{D}^T = \{(d_T^i, d_1^o), \ldots, (d_j^i, d_T^o)\}.$$

There are two cases to distinguish:

Case 1: $|\mathcal{D}^T| = 1$, i.e., one single device is returned and planning starts and ends with this single device.

Given $\mathcal{D}^T$ and the internal model of the device, during the next step the initial and the final states of the device are searched for. This search is based on the annotation functions i and o, which establish the connection between external models and the appropriate states of the internal model. As mentioned before, the functional model also includes the knowledge of the current state of each device. Therefore at that point the system knows about the current, the initial, and the final state to be reached by the device.

Thus, the next step is to search for a path from the current to the initial to the final state in the internal model of the device. Since the transitions of the internal model of a device are annotated with actions, a sequence of actions, i.e., a plan, is returned.

While searching for a path (i.e., a plan), pre- and postconditions of the states are checked. Informally speaking, the pre- and postconditions model the necessity of using further devices and how to use them. Since pre- and postconditions are formulated like the external models (see Definition 1), the same planning algorithm as described here is applied to fulfill them, which returns additional plans. These additional plans, corresponding to other devices that are necessary to fulfill the given complex task, finally are integrated in the overall plan.

Case 2: $|\mathcal{D}^T| > 1$, i.e., an ordered set of external models is returned. Each model uniquely corresponds to one device. And for each device the planning algorithm is performed as will be described for case 1.

It should be pointed out that the concept described above allows two possibilities to model the necessity of using several devices. First, it can be modeled by mapping the complex task to $(d_T^i, d_T^o) \in \mathbf{D}^i \times \mathbf{D}^o$, such that the device search algorithm returns an ordered set of external models (and also devices) as described above. Second, the concept of pre- and postconditions allows us to model a more flexible temporal order.

2.4 The Plan Processor

As described above, the reasoning component generates a plan, which consists of a sequence of actions. In order to fulfill a requested user wish, the generated plan needs to be executed. This is done by the plan processor, as we discuss next.

3 Realization in the Prototype

The function planning module was realized in the SMARTKOM prototype system (Wahlster et al., 2001). It provides an intelligent interface between the dialogue manager (called the action planning module, see Löckelt (2006)) and the application interface of the devices (Michelitsch et al., 2000).[1] For the current realiza-

[1] The application interfaces for the devices used in the SMARTKOM-Home scenario (Portele et al., 2006) were realized in the Embassi project, [2], and adapted to the SMARTKOM prototype.

tion these devices are loudspeakers, microphone, VCR, television, document camera, telephone, telefax, biometric authentication, address book, calendar, e-mail, car navigation, pedestrian navigation, and internal scheduler. According to the concept described above, the implementation of the planning module consists of: (i) the functional model (i.e., the external and the internal model) for each device, which contain information about the basic functionalities of the devices and about the commands which are required to invoke them; (ii) a representation of the current state of each device; (iii) a reasoning component, which involves a device search algorithm and a planning algorithm; and (iv) a plan processor, which executes the generated plans.

In the following sections the processing flow is first described. Second, the plan language, which is used in the functional models, is explained in detail. Finally, the main features of the planning algorithm and the plan execution are presented.

3.1 Processing Flow

As already mentioned, the function planning module provides an intelligent interface between the action planning module and the devices. In the following, the processing flow of requests issued by the action planning module will be described.

The action planning module requests an abstract and complex user wish to be fulfilled by the function planning module. The planner of the function planning module generates a plan to serve the request. It starts off from the current state of the devices and finds a path through the functional models that yields a configuration which provides the requested functionality. The path determines a sequence of commands which define the actual plan. Its execution is controlled by the plan processor.

The interaction between the function planning module and the devices is based on a command and response mechanism, where the initiative always lies by the function planning module. As a consequence, devices can only change their state according to the functional model, if they can relate the change to a command issued by the function planning module. Note that this does not mean that there is a one-to-one correlation between commands and state changes. A single command can also trigger a sequence of changes. Changes of states are reported to the action planning module only if there are according definitions in the functional models.

Because of the abstraction from specific devices, it is hidden from the action planning module which devices are involved in serving a particular request and how they are controlled. Nevertheless, we can distinguish between three complexity classes of plans within the function planning module. Many requests can be served by just sending one command to one device, e.g., looking up an entry in the address book. Some requests can be served by one device, but more than one command is needed. This is the case for biometric authentication: If an authentication method for a particular user is requested, the function planning module first checks whether that method is available for that user. If the test has been positive, it sends the command to initialize that authentication method. Otherwise the whole request is rejected.

The most complex treatment is required by those requests that are served by a couple of devices that need to interact in a well-defined way. Such a complex request is, for example, to program a VCR. It involves the internal scheduler, the calendar

application, a tuner, and the VCR. The programming request contains a time interval specifying when the recording procedure should be activated and information about the channel to be recorded. The function planning module checks by requesting the calendar application whether the VCR is programmed for the given time interval. If not, the request can be served. The scheduler is instructed to activate the recording procedure at the requested time interval with the given parameters. In order to carry out the recording procedure, the function planning module looks for an appropriate TV signal tuner, switches it to the specified channel, and then instructs the VCR to record the signal from the tuner.

3.2 Plan Language

The plan language is used to annotate the transitions of the internal model of a device. These annotations model the actions which have to be carried out successfully to pass the transition. The plan language consists of the following statements:

- action
- wait
- repeat_while_pending
- expect
- goal
- reached

The statement**action** models the actions to be taken in order to fulfill a complex wish. **wait** models the necessity of feedback of a device. **expect** is comparable with **wait**, but depending on the feedback of a device, the performance of the calculated plan will be performed, aborted, or alternative plans will be generated and performed. **repeat_while_pending** simulates a self-transition within a FSM, i.e., **repeat_while_pending** is a loop of the very same sequence of actions. It causes a repetition of the specified actions as many times as the application involved informs the function planning module that it is still working on the current request, i.e., the status of the request is pending. While processing a multiple response the current state of the device is the one before the transition annotated with a **repeat_while_pending** statement. The **repeat_while_pending** statement was introduced since the SMARTKOM system needs to handle requests with an *unknown* number of responses (e.g., navigation service). Multiple responses with a *fixed* number of responses can be modeled with a fixed number of **wait** statements. **goal** and **reached** are used for comparing the current state of the device while performing a plan and the state based on which the plan was calculated. This is necessary since it might happen that some devices change their states independently during the performance of a plan. If so, the calculated plan must be updated (plan repair).

A generated plan consists of a sequence of statements in the plan language. The length of a plan is defined as the number of statements in the plan. The length of a plan is always a finite natural number.

3.3 Planning and Plan Processing

The planning algorithm and the plan processor implemented in the SMARTKOM system allow conditional planning, parallel processing, scheduling a task, and interrupting the performance of a task. These features are described in this section.

The planning algorithm comprises a depth-first search. It allows conditional/heuristic planning by using the **expect** statement of the plan language. A generated plan will be performed as long as the feedback of a device is the one as expected. Otherwise the calculated plan will not be performed any longer. Instead an alternative plan will be performed. This alternative plan either already exists, i.e., is precalculated, or will be generated based on the current states of the system and devices that are involved (plan repair). As a simple example, consider a biometrics application. The generated plan is based on the expectation that the authentication is successful. This is modeled via an expect statement. If it is not successful, the generated plan is aborted, and a precalculated plan is performed in order to inform the user about the rejection.

The SMARTKOM system provides some services that last over a certain time period like, e.g., the navigation service. Using such a service, the user should still be able to request other services from the SMARTKOM system. In order to do that, the function planning module has the capability to serve different tasks in parallel. Given a task, the reasoning component calculates a plan to serve the task. If at some point the performance of the plan depends on some feedback of a device, the calculated plan is stored as long as the system is waiting for feedback. While waiting for feedback the system is able to serve other user wishes, i.e., the reasoning component calculates another plan and the plan processor is executing the calculated plan. In this case the plan processor is performing two plans in parallel. As an example, consider a tour guide application, which provides information to the user during some time. While still using the tour guide, the user wants to call a friend, which is another task performed by the system. Without interrupting the tour guide, the user is able to call and continue using the tour guide after finishing the telephone call.

Another feature of the function planning module that is realized in the SmartKom system is the possibility to serve a task at a certain time, i.e., scheduling a task. For this purpose the function planning module contains a scheduler, which is regarded as an additional virtual device. This implies that the scheduler is modeled with a functional model, consisting of an external and an internal model, exactly like all the other devices and applications controlled by the system. Based on this concept, scheduling a task is modeled as a functionality that arises by using one or several devices together with the scheduler. As an example, consider the task "programming a VCR." The functionality "recording" is combined with the functionality "scheduling a task X." The process is as follows:The device scheduler causes sending the request "record(Parameter)" to the function planning module at time when the recording should start. As usual, the function planning module then generates a plan, which is performed in order to fulfill this task.

Plans can be interrupted before they have been executed completely. The possibility of interruption has to be defined explicitly in the functional models by intro-

ducing extra states and transitions. Those extra transitions bring the according device into the original state. (By *orginal state* we mean the state which the device was in before the execution of the plan started.) Furthermore, they define actions that take the rest of the plan from the agenda, and if necessary, also bring other devices into their original state.

4 Future Work

Future work includes several topics. Planning and performing a plan within the function planning module could be refined and enhanced with additional features. The existing concept of the function planning module supports plug and play. However, there is still some work to do in order to realize plug and play in an existing system. A more theoretical research topic is to consider complexity. In the following, future work will be described in more detail.

Planning and performing a plan are implemented as two subsequent procedures. Making them interactive with each other would enhance the functionality of the function planning module. If a device reported a change of state, the plan processor should check the consistency of all plans with respect to that updated state. If necessary, an inconsistent plan should be substituted by a new plan calculated by the reasoning component. As an effect, the devices would gain the possibility to change their state autonomously. Furthermore, this change would enable self-transitions in the functional model.

The interruption of plans is realized by extra states and transitions in the functional model. A more general concept is desirable, such that a plan can be interrupted and discarded at any point and that its effects are taken back automatically, so that the original states of the devices are restored.

In a complex network of devices and applications, a given user request might be accomplishable in several ways. First of all, the function planning module could generate alternative plans representing alternative solutions. These solutions either could be presented to the user, such that the user could choose the solution he likes the best. Or, alternatively, the actions that are the components of the generated plans could be annotated with costs. Depending on the calculated costs of a generated plan, the system itself decides on the most appropriate plan to be executed.

When designing the function planning module, one of the goals was to support plug and play. This caused a strict separation between the functional models of devices and any reasoning component, such that any device can be enhanced with its functional model and plugged into the network. However, in order to support plug and play within a network of devices controlled with a multimodal dialogue system, the systems not only needs to know about functionalities of a given device but also, e.g., about words and pronunciations the user can use in order to ask for additional functionalities. Therefore additional information needs to be added to the functional models of the devices.

Another issue is to consider complexity. In Domshlak and Dinitz (2001) complexity results are shown for a multiagent system. The models used in that work are

comparable to the functional model. Thus the complexity results may also apply to the work described in this paper.

Acknowledgments

This research was conducted within the SMARTKOM project and was partly funded by the German Federal Ministry of Education and Research under Grant 01 IL 905 I7. The authors would like to thank their colleagues within Sony, Stefan Rapp, Martin Emele, and Ralf Kompe, for fruitful discussions about the ongoing work, helpful hints concerning the implementation, and helpful comments on earlier drafts of this paper. They would also like to thank their colleagues within the SMARTKOM project, Horst Rapp, Kerstin Reichel, and Victor Tabere (MID Dresden), and Axel Horndasch (Sympalog Erlangen) for the pleasant and successful collaboration.

References

C. Domshlak and Y. Dinitz. Multi-Agent Off-Line Coordination: Structure and Complexity. In: *Proc. 6th European Conference on Planning*, 2001.

A. Flycht-Eriksson. A Domain Knowledge Manager for Dialogue Systems. In: *Proc. 14th ECAI*, Berlin, Germany, 2000.

J. Han and Y. Wang. Dialogue Management Based on a Hierarchical Task Structure. In: *Proc. ICSLP-2000*, Beijing, China, 2000.

T. Heider and T. Kirste. Supporting Goal-Based Interaction With Dynamic Intelligent Environments. In: *Proc. 15th ECAI*, Lyon, France, 2002.

G. Hoffman and D. Moore. IEEE 1394: A Ubiquitous Bus. In: *Proc. COMPCON*, San Francisco, CA, 1995.

S. Larsson, R. Cooper, and S. Ericsson. menu2dialog. In: *Proc. Workshop Knowledge and Reasoning in Practical Dialogue Systems at IJCAI*, Seattle, WA, 2001.

O. Lemon et al. The WITAS Multi-Modal Dialogue System 1. In: *Proc. EUROSPEECH-01*, Aalborg, Denmark, 2001.

M. Löckelt. Plan-Based Dialogue Management for Multiple Cooperating Applications, 2006. In this volume.

G. Michelitsch, C. Settele, P. Hilt, and S. Torge, S. und Rapp. EMBASSI: Overview of the Architecture for a Multimodal User Interface for Future Home Entertainment Systems. In: *Proc. 10th Sony Research Forum*, pp. 156–161, Tokyo, Japan, 2000.

T. Portele, S. Goronzy, M. Emele, A. Kellner, S. Torge, and J. te Vrugt. SmartKom-Home: The Interface to Home Entertainment, 2006. In this volume.

X. Pouteau and L. Arevalo. Robust Spoken Dialogue System for Consumer Products: A Concrete Application. In: *Proc. Int. Conf. on Speech and Language Processing*, Sydney, Australia, 1998.

S. Rapp, S. Torge, S. Goronzy, and R. Kompe. Dynamic Speech Interfaces. In: *Proc. Workshop Artificial Intelligence in Mobile Systems at the 14th ECAI (AIMS 2000)*, Berlin, Germany, 2000.

W. Thompson and H. Bliss. A Declarative Framework for Building Compositional Dialog Modules. In: *Proc. ICSLP-2000*, Beijing, China, 2000.

S. Torge, S. Rapp, and R. Kompe. The Planning Component of an Intelligent Human Machine Interface in Changing Environments. In: *Proc. Workshop on Multi-Modal Dialogue in Mobile Environments*, Bad Irsee, Germany, 2002a.

S. Torge, S. Rapp, and R. Kompe. Serving Complex User Wishes With an Enhanced Spoken Dialogue System. In: *Proc. Int. Conf. on Speech and Language Processing*, Denver, Colorado, 2002b.

J. von Kuppevelt, U. Heid, and H. Kamp (eds.). *Best Practice in Spoken Language Dialogue Systems Engineering*, vol. 6(3,4) of *Journal of Natural Language Engineering*. Cambridge University Press, Cambridge, UK, 2000.

W. Wahlster, N. Reithinger, and A. Blocher. SmartKom: Multimodal Communication with a Life-like Character. In: *Proc. EUROSPEECH-01*, vol. 3, pp. 1547–1550, Aalborg, Denmark, September 2001.

Intelligent Integration of External Data and Services into SmartKom

Hidir Aras[1], Vasu Chandrasekhara[2], Sven Krüger[2], Rainer Malaka[1], and Robert Porzel[1]

[1] European Media Laboratory GmbH, Heidelberg, Germany
{hidir.aras, rainer.malaka, robert.porzel}@eml.villa-bosch.de
[2] Quadox AG, Walldorf, Germany
{vasu.chandrasekhara,sven.krueger}@quadox.de

Summary. The SMARTKOM multimodal dialogue system offers access to a wide range of information and planning services. A significant subset of these are constituted by external data and service providers. The work presented herein describes the challenging task of integrating such external data and service sources to make them semantically accessible to other systems and users. We present the implemented multiagent system the corresponding knowledge-based extraction and integration approach. As a whole these agents cooperate to provide users with topical high-quality information via unified and intuitively usable interfaces such as the SMARTKOM system.

1 Introduction

SMARTKOM is a multimodal dialogue system that offers access to a wide range of information and planning services. In contrast to many previous systems, it integrates services from multiple domains and in its current state covers over 50 different specific tasks (Wahlster, 2003, 2002; Wahlster et al., 2001). Some of these tasks are simple control applications such as that of sending a fax. Other applications are more complex and introduce open discourse areas. For instance, in the tourism domain, sights, streets, and locations can be subject to novel and unanticipated user requests. In the TV application, all sorts of topical and constantly changing broadcast information, actors, and genres are part of the SMARTKOM world. This information cannot be held locally by the system but rather must be gathered ad hoc from various Internet resources.

If Internet information resources are to be made accessible for a dialogue system such as SMARTKOM, which allows for natural language and gesture interaction, it is not sufficient to represent information as flat HTML or text pages. Rather, semantic information is needed, because users can refer to presented information and may ask additional questions such as "tell me more about this." In fact, we need to access the Web resources as a knowledge base in the sense of the semantic Web (Berners-Lee

et al., 2001). This means that the unstructured and semistructured data in the Web must be semantically annotated while it is integrated into the SMARTKOM system. The integration of semantic Web information typically involves multiple levels:

- representation of content (usually XML-based) and locations (URLs)
- knowledge modeling/ontologies
- integration/translation of content using different ontologies
- brokering of actual content with respect to user queries

There are a number of XML-based content languages. For SMARTKOM, M3L as a multimodal markup language has been used as a system-wide format (Gurevych et al., 2003a; Herzog et al., 2003). The URLs for particular Internet resources are predefined and contain a variety of third-party information on the SMARTKOM domains such as electronic program guides (EPG) for TVs, cinema programs, and movie information. Such resources on the Web have become increasingly popular, leading to incredibly large amounts of information published online.

However, finding the information on a particular subject, and evaluating the quality and correctness of the data is still a difficult process. Several services ranging from search engines to manually preprocessed catalogues address the problem of mapping the wealth of online information. But because of the distributed and rapidly changing nature of the Internet, different ontological information models, and unclear separation of data and format in the underlying Web-publishing technology, there is no trivial solution for an intelligent knowledge broker.

While brokers typically select and compile data from different sources, translators are needed to encode the meaning of the varying information originating from different repositories in a commonly used scheme, such as SMARTKOM's Multimodal Markup Language (M3L). Knowledge repositories and ontologies must be fused for this purpose. While the SMARTKOM ontology (Gurevych et al., 2002; Porzel et al., 2003) defines the target knowledge representation, each data source may come up with its own respective legacy data model.

In this paper, we describe the integration of external data into the SMARTKOM system. For this purpose, we developed a multilayer architecture that realizes a semantic Web-like access to selected third-party information providers that do not yet provide such metadata. The architecture is implemented as an agent system that also allows for caching mechanisms to enable SMARTKOM to respond to user queries in an offline mode.

2 Semantic Data Integration

Information sources are already available online in global information systems and have been developed independently. Consequently, semantic heterogeneity in terminology, structure, and the context of information can arise. This has to be properly managed to effectively exploit the available information independently of the source from which it has been extracted. The goal of information extraction and integration techniques is to construct synthesized, uniform descriptions of the information

extracted from multiple, heterogeneous sources and to provide the user or interface system with a uniform query interface. Moreover, to meet the requirements of global Internet-based information systems, it is important to develop tool-based techniques to make information extraction and integration semiautomatic and as scalable as possible. With today's tools, a tourist planning a trip to Heidelberg cannot simply download a map from the Web showing all the Italian restaurants close to the Castle, even though the information may well be available on the Web as maps of Heidelberg, lists of restaurants, and of tourist sites. The problem is not information distribution but information integration. Building tools to simplify access to the wealth of available information constitutes a significant challenge to computer science.

2.1 Logical Layers of Data Integration

In our prototypical implementation within the SMARTKOM demonstrator, we realized such a knowledge and service brokerage system. Our approach for an intelligent knowledge and service broker system within SMARTKOM is organized on three layers (Fig. 1):

- the interface layer for translating queries and responses
- a mediation layer that links to brokers and translators to the repositories
- a service layer that accesses the individual Web repositories and databases

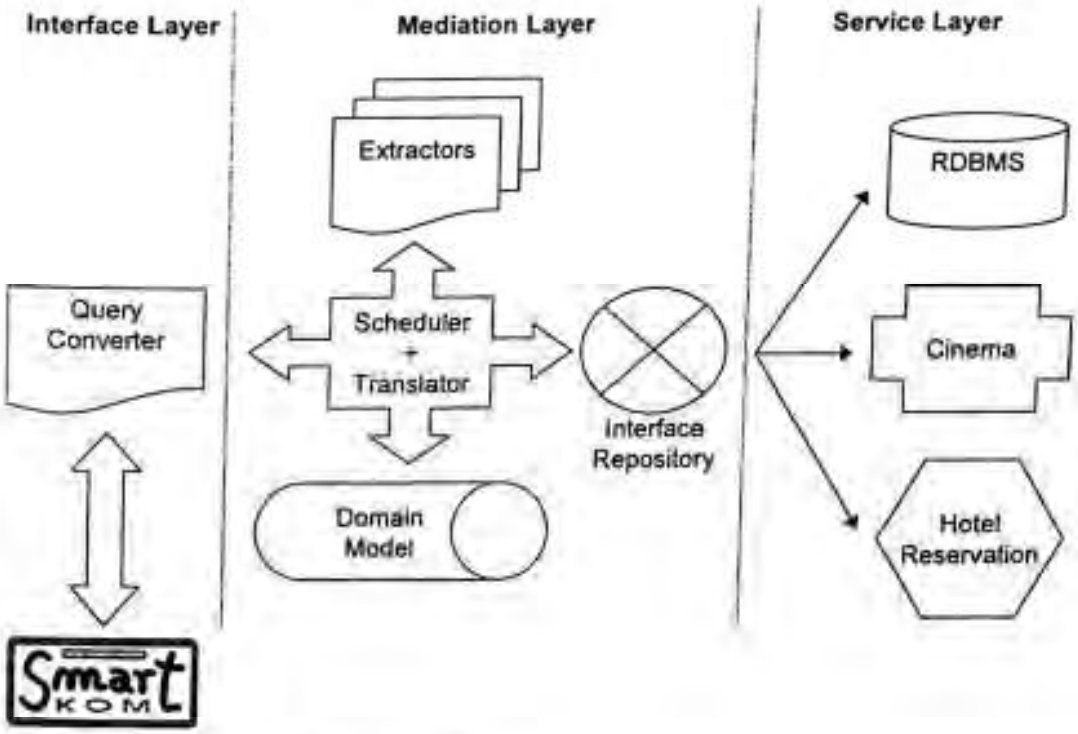


Fig. 1. Logical layers of data integration

On the interface layer, the information integration system is linked to the SMARTKOM semantic interfaces. Therefore, we prepared an in-depth domain model from which we derived a database schema. Furthermore, the model was also integrated into SMARTKOM's M3L standardization effort. This allows for efficient translation of user requests into XML-based schema instances that represent user queries and can be further analyzed by the integration layer.

Since different sources on the same information type may apply quite different taxonomic models on their information structure, the mediation layer must include semantic translators to integrate those external models. After the extraction and translation process, the question of the quality of the information is raised. By using sampled manual feedback and a computed confidence score, the more trustable information is passed back to the SMARTKOM system. The broker system exposes a uniform query interface; the queries are specified in M3L. We have constructed a generic query engine, adoptable to any ANSI SQL 92 compatible database. Queries must include selection and projection criteria. Metadata information for the construction of schema conform markups can be externally supplied or can be automatically extracted from the database schema.

The service layer includes a multiple-protocol handler and an interface repository to technically access external resources. For our system, we allowed for more than just Web resources that connect HTML pages via HTTP. The framework is more flexible and also integrates other protocol and content formats such as a CORBA-enabled hotel reservation system or standardized relational databases that come along with their own metadata model. However, standard Web pages without metadata are also included. It should be noted that in HTML, the term *markup* only describes annotations within the document that instruct a compositor how a particular passage should be printed or laid out. As the formatting and printing of texts was automated, the markup language was extended to cover all sorts of special markup codes inserted into electronic texts to govern formatting, printing, or other processing. In order to segregate the important information data from the printing or processing data we included source-specific extractor scripts, based on a high-level regular expression language. As more Web resources offer content data in XML, the requirement for data segregation will diminish.

2.2 Agent-Based Realization of the Layers

The three layers describe the information flow in the system. For the practical realization, however, it has to be considered whether these layers should be built into a static or more flexible component-based software architecture (Ding et al., 2001; Wooldridge, 1994). It has been shown in many scenarios that an agent-based software design is advantageous in many ways (Poslad et al., 2001). Introducing an agent-oriented paradigm to the system, we can model the tasks as cloneable agents as user, broker, and resource agents. The user agents locate brokers on the basis of costs and evaluate user preferences. In the middle layer, broker agents find the best resources for a request, evaluate match-rates and costs. The resource agents extract and translate the appropriate source, evaluate hit-rates, and remember costs.

The advantages of the agent approach are as follows (Wooldridge, 1994):

- Autonomy and adaptivity: Tasks delegated to the agent are solved independently, without immediate demand for user attention. By incorporating specialized learning algorithms, an agent can intelligently adapt to requests. Complex inquires can be solved dynamically using different partial tasks depending on external availabilities, thus leading to goal-oriented behavior.
- Cooperation: An intelligent agent is able to communicate with the surrounding environment, with other agents, with infrastructure facilitators, and, if equipped with a graphical interface, also with the user. Agent cooperation can be achieved by direct or broadcasted communicative events— speechacts— which are formulated in a high-level Agent Communication Language (ACL), but more importantly also by indirect dependencies leading to self-organization models based on stigmergy.
- Proactivity: An agent is able to start an action without having been initiated by direct requests. Proactive functions could be based on environment changes or internal state changes.
- Mobility: In a distributed environment, entities communicate mainly by request/response schemes. With a mobility-enabled distributed agent platform, agents can move dynamically from system to system, transferring code and interior states. This gives rise to a new application paradigm, for example, roaming agents with delegated responsibilities.

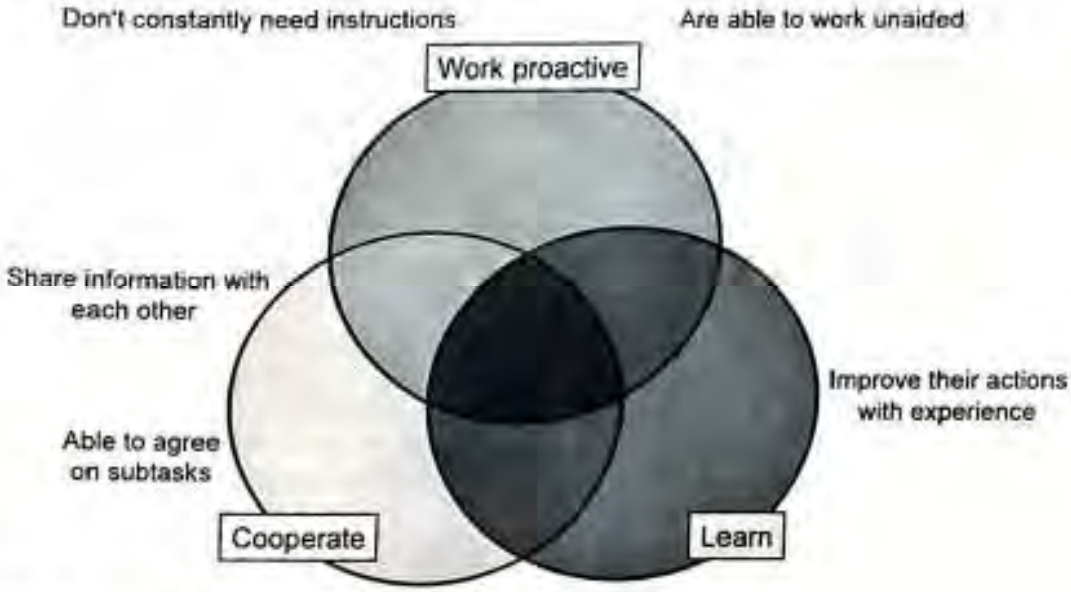


Fig. 2. Conceptual advantages of intelligent agents

The conceptual advantages of intelligent agents make them particularly interesting for the complex task of integration of data from heterogeneous sources that

consist of several subtasks as described here. The implemented architecture of such a system consists of a set of software agents to allow cooperative, adaptive, and proactive behavior of the overall system (Fig. 2). With this approach, we aim at developing intelligent tools for the integration of information extracted from multiple heterogeneous sources. In our system, we work with a fixed set of external information sources. However, in order to effectively exploit the numerous sources available online in global Internet-based information systems, some challenges remain to be solved. The main problems involved are:

- the identification of semantically related information (that is, information related to the same real-world entity in different sources)
- the heterogeneity of semantically related information

For the SMARTKOM demonstrator, we rolled out a working knowledge and service broker system completely implemented in a Java environment. Redundant resources for cinema, film, actors, EPG, yellow pages, ticket reservation, hotel information, and reservation domain were connected. Furthermore, parts of the Deep Map historic content and a geo-map agent for Heidelberg were integrated (Malaka et al., 2000; Malaka and Zipf, 2000).

The actual integration of external data in SMARTKOM realizes a transparent bridge from SMARTKOM to Web services and databases. As described in the previous sections, the system is organized in three layers, each of which, in turn, is implemented in multiple software agents.

2.2.1 Interface Layer Agents

Although the agents on the interface layer are implemented very differently, the service provided by them is the same, i.e., offering interfaces to external clients enabling them to benefit from different agent system services. Users have various options available for using the agent system. They can choose between using JSP sites, mail requests, or a graphical user interface. The SMARTKOM system communicates via a blackboard pool architecture and is based on a PVM architecture (Herzog et al., 2003). A PVM bridge and a connector agent were constructed to enable both systems to communicate. This agent required special adaptation to the SMARTKOM pool architecture. In future systems, all agents should be capable of dynamically locating the optimal broker agent on a cost basis and registering the user's habits for the adaptation.

2.2.2 Mediation Layer Agents

The agents assigned to this middle layer (so-called broker agents) have to address the following tasks:

- solve complex inquiries dynamically by decomposing them into separate partial tasks depending on external availabilities
- locate agents that provide the requested service

- eliminate redundancies in the answers of different service agents by using sampled manual feedback and a computed confidence score
- provide a facility for translating various metadata models of external sources into a global domain schema

The current broker agents implement three main components. First, the XML–Java data-binding component dynamically translates XML into Java objects and back by using marshaling and unmarshaling mechanisms. The second component realizes a resource location protocol. The purpose of this is to determine the best resource agents by using a catalogue service, the so-called directory facilitator, where all resource agents have to register with the respective costs involved, e.g., response time, quality of content, etc. The third part of the broker agent is a merging and integration facility developed to eliminate redundant data from the answers of service agents.

2.2.3 Service Layer Agents

The service layer encompasses two different kinds of agents: database agents and agents responsible for providing access to different Web sources. Such wrapping agents implement source-specific translators to generate HTTP requests from the incoming XML requests that are understandable for the Web source. It is on the basis of these requests that the URL requester retrieves the corresponding HTML-based data. In order to segregate important information from printing or processing data, we included source-specific extractor scripts based on a high-level, regular, and fault-tolerant expression language. In the future, as an increasing number of Web resources offer content data coded in XML the requirement for data segregation will lose its importance. The database integrated into our system at present provides content periodically extracted from various Web sources about films, actors, cinema performances, and broadcasts. We have integrated this database into the system to enhance request performance, to avoid redundant operations, and to ensure the complete functionality of various services for offline operations as well. To provide an interface consistent with the SMARTKOM domain representations, a generic SQL–XML query engine was developed. This engine makes it possible to answer an XML query containing selection criteria and projection statements defining the resulting XML syntax structure. By using the metadata structure of the database, we were successful in developing an absolutely generic solution adaptable to any ANSI SQL 92 compatible database. This means that we can reuse this generic XML–SQL engine to integrate other SQL-conformable databases, e.g., sights or architectural databases.

3 Implementation of the Data Extraction and Integration Process

We will now give some more implementation details and examples of the methods implemented in the various agents on each layer. The prototypical scenarios of the SMARTKOM system have a special emphasis on the cinema and EPG domain, where most examples originate.

3.1 Development of the Domain Models for Messages und Databases

The first step for semantically integrating SMARTKOM external data is the provision of internal domain models. For the fields of cinema and EPG, these models were modeled such that they share many concepts such as actors or genres. The initial version of the model was built for the XML Document Type Definition (DTD) language.[1] An analogous version in XML schema[2] was developed for the current version of XSD, taking into account the W3C specifications from the working draft of the XML Schema Definition Language. The necessary translation between DTD and XMLS can be done using propriety parsers and tools. The initial schema-based models for the respective domains were also transformed into the SMARTKOM ontology (Gurevych et al., 2003b). From the ontology, in turn, tools allow for automated generation of corresponding XML schema (Gurevych et al., 2003a) from the ontology.

Based upon this model, a relational database has been set up configured using Cloudscape, a Java RDBMS. This database seemed appropriate because it supports portability, has a small footprint, and allows the persistent storage of Java objects.

For the domain of geographical data, necessary for many tourist applications, the data integration is done by specific geospatial agents who access geoservers that follow OpenGIS standards, which are also semantic models and also conform with the ontology.In this way, objects from one domain like a "cinema" can be linked to objects in the other domain in space (and time).

3.2 Building the Infrastructure and Middleware

As discussed above, we chose a multiagent system (MAS) for building the necessary infrastructure and middleware for the information integration platform. We evaluated a number of Java-based MAS and decided to use JADE,[3] which supports the important OMG[4] standard (known from CORBA) and the FIPA[5] standards that aim for promotion of a commercial standardization of the Foundation for Intelligent Physical Agents (FIPA). Distributed objects using, e.g., CORBA, RMI, DCOM, or RPC, communicate with each other based on the principle of local transparency, while the agent communication is done asynchronous by exchanging messages that are semantically divided into three layers:

- **communication layer:** describes the communication parameters, i.e., sender, receiver, unique communication identification, etc.
- **message layer:** defines the *performatives* (borrowed from *Speech Act Theory* (Searle, 1969), for example, *ask-if, tell, reply*, etc.
- **content layer:** the "real" message that is exchanged and encapsulated by the communication protocol

[1] http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm
[2] http://www.w3.org/TR/xmlschema-0
[3] http://sharon.cselt.it/projects/jade
[4] http://www.omg.org
[5] http://www.fipa.org

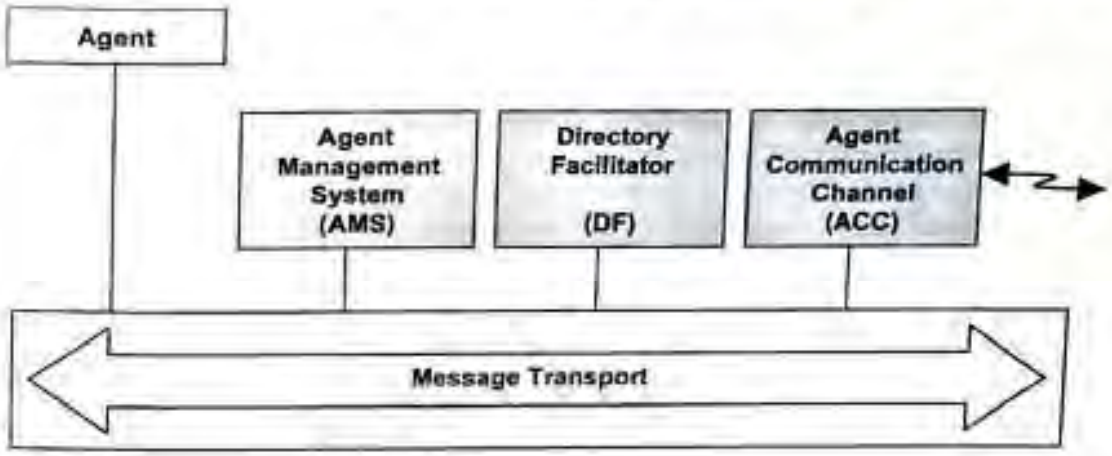


Fig. 3. Message transport

Using semantic messaging one can map multiagent contract negotiations, as in "Call-For-Proposal," without having specific API knowledge about the counterpart (Fig. 3). The communication layer is mostly defined by the MAS itself; the content layer reflects the domain models used here. The message layer links content to communication by describing for the agents what action they should perform or telling other agents what action has been performed.

3.3 Programming of Specific Data Extractors

The World Wide Web has evolved into a public information space. It no longer consists only of simple, hyperlinked documents, but more and more of queryable sources that deliver their responses in semistructured form. Product catalogs, financial services, weather news, commented publications, software collections, conference calls, etc., represent typical examples. The structured information provided by these resources are "pressed" into a pattern that is based upon the navigate-and-show paradigm (hyperlinking) of the Web. Such sources contain mostly overlapping or complementary information that is presented with different syntax and semantics. In general, one can identify the following types of access to data (Web content):

- **static HTML pages**, where information is provided by prepared pages in the HTML layout.
- **dynamic HTML pages** that are similar to static ones, but data and layout are mixed up, and target content can be queried in a directed way (using <form GET/POST/>).
- **static or dynamic XML pages** present content in XML-structured data format, either statically or dynamically (analogously to static and dynamic HTML). HTTP is used technically, and target content mostly refers to the underlying DTD.
- **distributed (object-oriented) interfaces** (RPC, IIOP/CORBA, DCOM, Java-RMI). The access is performed through well-defined, standardized interfaces.

- **direct database connectivity** over DB-API (Oracle Call Interface, ODBC, or JDBC). The specific query statement can be formulated in SQL.

The unstructured HTML sources require an extensive manual preprocessing to allow reasonable data extraction. As mentioned above, the most interesting data is today available in HTML sources. Different groups work on heuristics to analyze semistructured documents, but this is only applicable to certain scenarios (for example, XWRAP[6] or DERG[7]). A typical example of such HTML text from a cinema Web site is:

```
  [...]
  <TR><TD VALIGN="top" WIDTH="10"><IMG SRC="/img/dot.gif"
WIDTH="10" HEIGHT="10"></TD><TD ALIGN="left"
WIDTH="100%"><FONT FACE="Arial, Helvetica" SIZE="2"><A
HREF="/filminfo/a/abbuzze.htm" TARGET="inhalt">Abbuzze - Der
Badesalzfilm</A></FONT></TD></TR>
  <TR><TD VALIGN="top" WIDTH="10"><IMG SRC="/img/dot.gif"
WIDTH="10" HEIGHT="10"></TD><TD ALIGN="left"
WIDTH="100%"><FONT FACE="Arial, Helvetica" SIZE="2"><A
HREF="/filminfo/a/abgefahren.html"
TARGET="inhalt">Abgefahren</A></FONT></TD></TR>
  [...]
```

The information is provided by prepared pages in HTML layout. The example shows how layout and data are mixed. The specific extractor has to use a directory of URLs to intelligently extract the right target content. In a typical XML document the information is well-structured and content is semantically annotated:

```
<channel>
  <title>
    moreover ... Indian subcontinent news
  </title>
 [...]
  <description>
    News headlines harvested every 15 minutes
  </description>
  <item>
    <title>
      China Denies Charges of Supplying Missile Tech to Pakistan
    </title>
    <link>
      http://c.moreover.com/click/here.pl?r8036262
    </link>
    <description>
      New York Times Jul 4 2000 11:25AM
    </description>
```

[6] http://www.cc.gatech.edu/projects/disl/XWRAPElite

[7] http://www.deg.byu.edu

```
  </item>
  <item>
[...]
```

The XML.ORG[8] site tries to support standardization efforts for different vendors and application domains. The XML Tree[9] indexes content providers who provide their contents in pure data formats, unfortunately approximately 50 vendors for German content, mostly in WML form for WAP devices (WML is structured, but also mixes up layout and data).

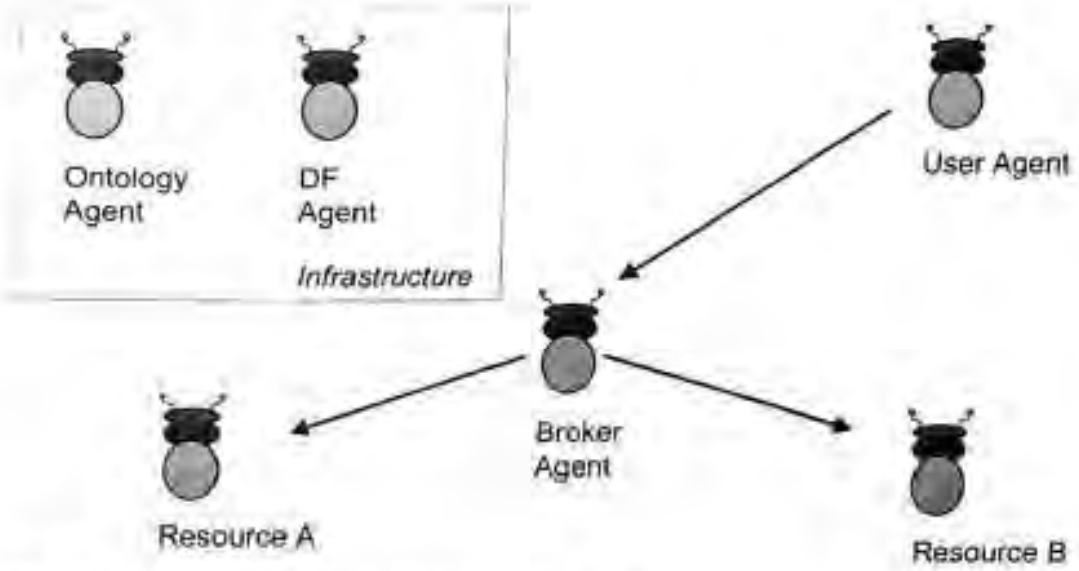


Fig. 4. Resource, broker, user and infrastructure agents

After a detailed feasibility study we decided to use JEDI 1.2.1,[10] a Web extraction toolkit (Huck et al., 1998). In order to use mediators and wrappers, JEDI only requires standard Web browser technology. All system components have been implemented fully in Java. The wrapper consists of a rich, error-tolerant parser. The source structure of documents (nesting included) can be described using attributed rules. The execution of these rules results in a network of objects that represent the documents that are the subject of the data extraction process. The parser is able to process incomplete or ambiguous source descriptions by using a simple approach, i.e., efficiently choosing the most specific rule combination out of the entire applicable set of rules. If a part of the document does not allow the parser to apply a rule, only the necessary part is skipped, and not the whole document. Beside the rules for structuring, the parser also provides methods for navigation and form-based queries

[8] http://xml.org
[9] http://www.xmltree.com
[10] http://www.darmstadt.gmd.de/oasys/projects/jedi

to information sources. It can also be extended by Java predicates and methods. Several tools are available for mediation in XML. Exolab's Castor tool was chosen to bind XML schema to Java objects.[11]

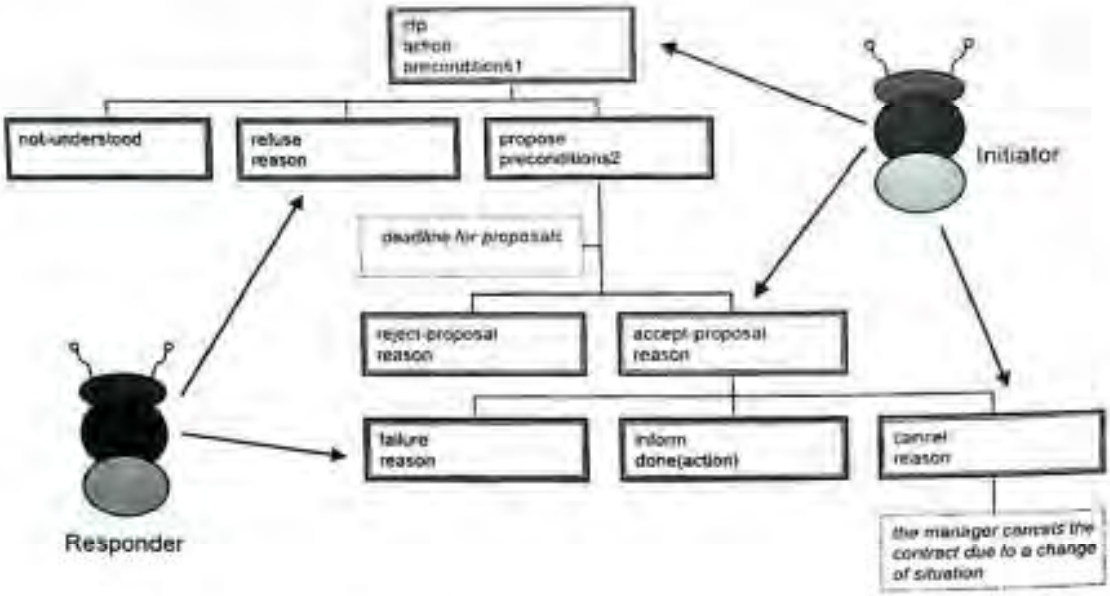


Fig. 5. The call for proposal protocol

3.4 Formulation of the Processing Logic of Queries

Information and external services from different sources from the Web are represented by resource agents (Fig. 4). A broker agent then initiates the negotiation process to access the service by using the call for proposal (CFP) protocol (Fig. 5). Available service agents can be found using the directory facilitator (DF) service. This procedure allows for individual handling of each resource. The scheduler, respectively, the query engine is distributed among the different agents. The broker is also the bridge to the interface/user agents. An ontology agent provides the system with ontological information.

The CFP protocol allows two agents to negotiate service interaction. A broker can start to search for service agents by broadcasting a CFP for a requested action and can parameterize the CFP with a set of preconditions. Responding agents have the choice between a proposal to react or to refuse the call in case they understand the request. If they propose a response, they also give a set of preconditions. After a deadline for the CFP, the initiating agent collects the responses for the CFP and decides to accept or reject certain offers from responding agents. The agents who get

[11] Xerces, http://xml.apache.org; XQL, http://xml.darmstadt.gmd.de/xql; Castor, http://castor.exolab.org.

an accept-response for their offer will either deliver a result or fail, and will inform the initiator accordingly. If the response takes too long or due to other circumstances is not needed any more, the initiator can also asynchronously cancel the execution of the request.

3.5 Definition of the Query and Response Syntax and Semantics

We have connected the resources into our prototypical demonstrator that feature content about cinematography, hotel reservations and geographical information modeled in the corresponding actor, film EPG, hotel, and map query DTDs. We will illustrate the function of these models with examples from the cinema domain. Consider, for example, a user asking whether the film *Gladiator* is running in the city of Viernheim tomorrow. The corresponding query syntax is defined by a generic DTD shown below:

```
<?xml version="1.0"?>
<!ELEMENT cr          (day,time?,town,title?,actor?)>
<!ELEMENT day         (#PCDATA)>
<!ELEMENT time        (#PCDATA)>
<!ELEMENT town        (#PCDATA)>
<!ELEMENT title       (#PCDATA)>
<!ELEMENT actor       (#PCDATA)>
```

Based on this syntax, the M3L-corresponding representation of the user's intent (see also Wahlster et al. (2001); Wahlster (2003)) is transformed into the specific query given below:

```
<?xml version='1.0'?>
<!DOCTYPE cr SYSTEM "CRquery.dtd">
<cr>
    <day>morgen</day>
    <time>14</time>
    <town>viernheim</town>
    <title>Gladiator</title>
    <actor/>
</cr>
```

Based upon such a query, the processing logic described above yields answers that are annotated with their source and contain the corresponding information, as shown in the example below:

```
<entry>
  <url>
    http://www.film.de/asp/content/film.asp?film_id=4361
  </url>
  <titel>
    Gladiator
  </titel>
  <ort>
```

```
      Viernheim
    </ort>
    <kinos>
      <kino zeit="8. Juli 2000 - 16.45, 20.00, 23.15 Uhr">
        Kinopolis Rhein-Neckar
      </kino>
    </kinos>
  </entry>
```

Employing the ontology-based semantic translators, this answer is translated back into M3L and enabled the action and presentation planning component of SMARTKOM to respond to the user's query with topical and trustworthy information.

4 Conclusion

We have shown herein how the SMARTKOM dialogue system can offer its users access to a wide range of information and planning services that stem from external sources, and how an intelligent external data and service integration platform can provide users with topical high-quality information via an unified and intuitively usable multimodal dialogue system. It has been demonstrated that this can be achieved in a prototypical implementation based on a mutliagent system which can cooperate on extracting, integrating, and ensuring the quality of the services and information requested by the user.

While there is still room for improvement, e.g., better adaptivity toward the quality of the resources, more intelligent fetching strategies, and the employment of dynamic and learning semantic schema translators for handling HTML resources, we anticipate to connect more semistructured, XML-based Web resources in the future. Still, HTML resources will continue to pose a difficult challenge, as their content is quickly outdated and their presentations often undergo redesign. To further stabilize and increase the quality and quantity of the information content at our disposal, agreements with networks of content providers, or federations of agent-based service providers such as in the Agentcities[12] framework, constitute promising future directions.

Projects such as Agentcities are aiming to build worldwide, publicly accessible test beds for the deployment of agent-based services. We, therefore, joined such an open initiative, where the core effort will be devoted to interoperability at the service level, in particular the development of ontologies and service descriptions defining a new open standard. Finally, the SMARTKOM approach to external data and service integration described herein shows that important features such as quality negotiations and redundancies brought about by the employment of a layered multiagent system as well as the semantic extraction and integration capabilities thereof yield the desired added value of homogeneous and intuitive access to heterogeneous legacy systems and sources.

[12] http://www.agentcities.org

References

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

Y. Ding, C. Kray, R. Malaka, and M. Schillo. RAJA — A Resource-Adaptive Java Agent Infrastructure. In: *Proc. 5th Int. Conf. on Autonomous Agents*, pp. 332–339, Montreal, Canada, 2001.

I. Gurevych, S. Merten, and R. Porzel. Automatic Creation of Interface Specifications from Ontologies. In: H. Cunningham and J. Patrick (eds.), *Proc. HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, pp. 59–66, Edmonton, Canada, 2003a. Association for Computational Linguistics.

I. Gurevych, R. Porzel, E. Slinko, N. Pfleger, J. Alexandersson, and S. Merten. Less Is More: Using a Single Knowledge Representation in Dialogue Systems. In: *Proc. HLT-NAACL'03 Workshop on Text Meaning*, pp. 14–21, Edmonton, Canada, May 2003b.

I. Gurevych, R. Porzel, and M. Strube. Annotating the Semantic Consistency of Speech Recognition Hypotheses. In: *Proc. 3rd SIGdial Workshop on Discourse and Dialogue*, pp. 46–49, Philadelphia, PA, July 2002.

G. Herzog, H. Kirchmann, S. Merten, A. Ndiaye, P. Poller, and T. Becker. MULTIPLATFORM Testbed: An Integration Platform for Multimodal Dialog Systems. In: H. Cunningham and J. Patrick (eds.), *Proc. HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, pp. 75–82, Edmonton, Canada, 2003. Association for Computational Linguistics.

G. Huck, P. Fankhauser, K. Aberer, and E.J. Neuhold. JEDI: Extracting and Synthesizing Information From the Web. In: I.C. Society (ed.), *Proc. 3rd IFCIS Int. Conf. on Cooperative Information Systems*, pp. 32–43, New York, 1998.

R. Malaka, R. Porzel, A. Zipf, and V. Chandrasekhara. Integration of Smart Components for Building Your Personal Mobile Guide. In: *Proc. Workshop Artificial Intelligence in Mobile Systems, ECAI 2000*, Berlin, Germany, 2000.

R. Malaka and A. Zipf. Deep Map: Challenging IT Research in the Framework of a Tourist Information System. In: D.R. Fesenmaier, S. Klein, and D. Buhalis (eds.), *Proc. 7th. Int. Conf. on Information and Communication Technologies in Tourism (ENTER 2000)*, pp. 15–27, Berlin Heidelberg New York, 26–28 April 2000. Springer.

R. Porzel, N. Pfleger, S. Merten, M. Loeckelt, I. Gurevych, R. Engel, and J. Alexandersson. More on Less: Further Applications of Ontologies in Multi-Modal Dialogue Systems. In: *Proc. 3rd IJCAI 2003 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, 2003.

S. Poslad, H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf. CRUMPET: Creation of User-Friendly Mobile Services Personalised for Tourism. In: *Proc. 2nd Int. Conf. on 3G Mobile Communication Technologies*, pp. 28–32, London,UK, 2001.

J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK, 1969.

W. Wahlster. SmartKom: Fusion and Fission of Speech, Gestures, and Facial Expressions. In: *Proc. 1st Int. Workshop on Man-Machine Symbiotic Systems*, pp. 213–225, Kyoto, Japan, 2002.

W. Wahlster. SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: R. Krahl and D. Günther (eds.), *Proc. Human Computer Interaction Status Conference 2003*, pp. 47–62, Berlin, Germany, June 2003. DLR.

W. Wahlster, N. Reithinger, and A. Blocher. SmartKom: Multimodal Communication with a Life-like Character. In: *Proc. EUROSPEECH-01*, vol. 3, pp. 1547–1550, Aalborg, Denmark, September 2001.

M. Wooldridge. Intelligent Agents. In: G. Weiss (ed.), *Multi-Agent Systems — A Modern Approach to Distributed Artificial Intelligence*, pp. 17–20, Cambridge, MA, 1994. MIT Press.

Multimodal Fission and Media Design

Peter Poller and Valentin Tschernomas

DFKI GmbH, Saarbrücken, Germany
{poller,tscherno}@dfki.de

Summary. This chapter describes the output generation subsystem of SMARTKOM with special focus on the realization of the outstanding features of the new human–computer interaction paradigm. First, we start with a description and motivation of the design of the multimodal output modalities. Then we give a detailed characterization of the individual output modules, and finally we show how their collaboration is organized functionally in order to achieve a coherent overall system output behaviour.

1 Design of Interaction and System Output in SmartKom

One of the main tasks of the SMARTKOM project was the investigation of the coordinated use of different modes and modalities in human interaction aiming at symmetrically exploring them to support the intuitive multimodal access to knowledge-rich services in a mixed-initiative system. SMARTKOM merges three user interface paradigms: natural language dialogue, graphical interface, and gestures (incl. facial expressions) in a symmetric use (Wahlster, 2003), i.e., seamless fusion and mutual disambiguation of multimodal input, as well as multimodal fission and adaptive presentation layout on the output side. According to these overall system requirements, the project first concentrated its activities on the design of a superordinated completely new human–computer interaction metaphor that allows for natural multimodal access to all the different applications in the three scenarios.

Interactions with the SMARTKOM system are based on the *situated delegation-oriented dialogue paradigm (SDDP)*, which means that the user delegates a task to a virtual communication assistant that is visualized as a life-like character. This dominating completely new SDDP-style of natural collaborative interaction with a virtual communication assistant in SMARTKOM, in conjunction with the three different scenarios, implies a series of essential requirements on the design (and their implementation in the system) of various prominent characteristic features of the SMARTKOM system:

- There is no direct interaction of the user with any output element independent of its respective medium and modality.

- The system persistently conforms to the design principle "no presentation without representation" (Wahlster, 2003), i.e., all output elements are (independent of their modality) internally represented using corresponding Multimodal Markup Language (M3L) expressions potentially referrable by the user.
- Consequently, all output graphics are visual widgets for output presentation elements, also potentially referrable by the user.
- The planning and realization of output presentations natually depends on the respective scenario and the available output media and modalities.
- The system constantly presents adaptive perceptual feedback that represents its internal processing state.
- All system output is based on context-dependent modality fission.
- SMARTKOM realizes symmetric multimodality, i.e., all input modalities are also available for output, and vice versa.
- In conformity to the SDDP paradigm, the communication assistant Smartakus is a de facto personification of the system.
- All multimodal capabilities of the system are integrated into the same reusable and efficient dialogue shell: the core dialogue processing engine.

In the following sections we show the various explicit and implicit impacts of the above-mentioned features on the design of interaction metaphor of SMARTKOM as well as various aspects of that metaphor. We give special focus to the respective consequences on the design of the multimodal system output, i.e., its modalities and media and their coordinated use.

1.1 SDDP

The before-mentioned conditions imply that system interactions in SMARTKOM must move beyond the traditonal windows, icons, menus, pointing device (WIMP) interaction with keyboard and mouse on a desktop. Rather, user input is based on natural human conversation means: speech, gestures, and mimics. In addition to that, in line with the guidelines elaborated in Oviatt (2003), SMARTKOM goes yet another step further as it breaks radically with traditional mouse- and keyboard-based interaction via desktop. Instead, interactions with the system are based on the SDDP, which means that the user delegates a task to a virtual communication assistant that is visualized as a life-like character. This, in turn, is done by natural human communication means, i.e., the user input is exclusively performed in a purely natural way, namely by the coordinated use of speech, gesture, and mimics, just as humans do in real life.

Only this way is it possible that the user gets a homogeneous and pleasing interaction experience through communication with the anthropomorphic personalized interaction agent Smartakus to whom the user virtually delegates the task to be solved. In SMARTKOM, both communication partners collaborate during the problem-solving process in which the personalized agent accesses the background services. The agent may ask back for more information and finally presents the results on the output channels.

This kind of human–machine interaction cannot be done in a simple command-and-control style for more complex tasks. Instead, a collaborative dialogue between the user and the agent elaborates the specification of the delegated task and possible plans of the agent to achieve the user's intentional goal.

Consequently, human–computer interaction in SMARTKOM is put on a completely new footing. The focus of interaction rests on the cooperative communication with the animated agent about the fulfillment of the respective service the user is interested in. Unlike the traditional desktop interaction metaphor, which is based on WIMP interaction, the content of the graphical user interface of SMARTKOM is radically reduced to only those elements (e.g., graphics) that are relevant to the user to the extent that the visible presentation contents on the screen exclusively consist of the graphical presentation elements such that the screen background becomes transparent. The presentation agent Smartakus itself guides the user through a presentation and comments on and points to objects as needed.

The user delegates a task to Smartakus and helps the agent, where necessary, in the execution of the task. Smartakus accesses various digital services and appliances on behalf of the user, collates the results, and presents them to the user. Therefore, all further design aspects must naturally be subordinated to this fundamental interaction style.

1.2 No Presentation without Representation

Interactions in SMARTKOM are managed based on representing, reasoning, and exploiting models of the user, the domain, the task, the context, the media, and the modalities themselves. Consequently, the system follows the underlying principle not to process and present anything without representing it. So, for all multimodal inputs and outputs a common representation approach based on the XML language M3L is used. Whatever the communication with the system is about (including the output widgets themselves that are used for graphical presentations), it is always internally represented.

This way, the SMARTKOM system properly follows the principle that — independent of the respective modality — user references to all system output elements must be generally possible. With respect to the design of visible output, this rules out graphical elements or fonts that are too small to be unambiguously referenced by a user gesture on the respective display.

1.3 Smartakus as the Personification of the System

The key design feature of SMARTKOM is the realization of the SDDP paradigm, which results in using the life-like character Smartakus as a kind of virtual "personification" of the system the user is communicating with. In this sense, the communication metapher opens up a large range of performances that constitute full system output presentations that may even go beyond human capabilities. Consequently, a SMARTKOM system presentation is usually designed as a kind of presentation show that is organized as follows: Depending on the current position of Smartakus on the

screen, the graphical elements on it, and the space and positions of the graphical elements to be presented. Smartakus first "jumps" to a new position (leaving enough space). Then the graphical elements are brought up, and finally Smartakus comments on the graphic content, and performs appropriate pointing gestures and speech animations that are synchronized with the audio output signal.

1.4 The Three SmartKom Scenarios

There are three different instances/scenarios of SMARTKOM that differ in the respective applications and hardware. In this section we shortly recapitulate them in order to show their respective key influence factors on design aspects of different output elements.

SMARTKOM "Public" is a system that can be used as a public multimodal information kiosk. Contactless gestures of the user are rendered possible by an infrared camera (the Siemens Virtual Touch screen, SiViT), which tracks the user's hand moving across a video projection of the graphical output on a tablet screen. Naturally, this constellation may contain the complete spectrum of application types, modalities, and hardware functionalities. All input devices and input modalities are available. Because of the large display, the dominating output element in system presentations is graphics. Furthermore, there is enough space to show Smartakus with its full body and let it move around on screen to optimize its positioning during output presentations.

In the SMARTKOM Home scenario, a comparatively small touch-sensitive Tablet PC (Fujitsu Stylistic) is used to show the visual output of SMARTKOM. Here, the system is used as an information system at home, e.g., as an EPG and for device control (TV set, video cassette recorder). Input gestures are performed with a pen on the touch screen. There is no camera intergrated to interpret facial expressions or for scanning. SMARTKOM Home is a reduced system instance whose output device permits mobile interaction at home.

SMARTKOM Mobile is an instance of the SMARTKOM system with its display on a touch-sensitive PDA (Compaq iPAQ) that is used as a mobile travel companion to access tourist information and navigation (car and pedestrian). The very small display requires us to reduce Smartakus to only those body parts of it that are animated, i.e., one hand (for pointing gestures) and the head (for lip-synchronized speech output). Naturally, there is a correspondingly adapted gesture repertoire such that Smartakus' multimodal output capabilities are fully preserved, i.e., moving around, synchronized lip movements, and pointing gestures.

1.5 Adaptive Visual Perceptive Feedback

The dominating role of the communication assistant Smartakus in the interaction with the SMARTKOM system requires the realization of a high degree of adequacy, naturalness, and liveliness (not particularly during output presentations but rather at times of output inactivity) that is constantly perceivable over time. This does not that much of an effect on the agent behavior during system output presentation. Much

more time passes by due to communication pauses or while the system is processing the last user input.

Simple observation of human–human communication shows that as long as temporary communication pauses are accompanied with feedback signals, it is even natural and polite to give the communication partner enough time to think about his next utterance. Transferring this observation to human–computer interaction, particularly to multimodal dialogue systems, means that especially those systems that have slower reaction times of several seconds have to focus on giving the user permanently adequate and adaptive perceptual feedback during the processing of the user input about the respective state or progress of this processing. Fortunately, concerning the realization of such a feedback, the SMARTKOM system is equipped with various multimodal output capabilities that can be used for that purpose in a natural way.

Accordingly, apart from the entire interaction facilities, the SMARTKOM system additionally provides constantly updated adaptive perceptual visual feedback. In line with the SDDP paradigm, the feedback is given indirectly by the animated agent, which simultaneously performs conformable gestures (Poller and Reithinger, 2004). In parallel to that, adequate graphic elements, e.g., a flashing light during input processing, are additionally presented on the display to reinforce the desired effect.

All in all, the realization of perceptual feedback is mainly achieved by the animated agent Smartakus by its capability to dynamically adapt its behavior depending on the respective system state. Figure 1 shows the eight most important behaviour patterns of Smartakus in form of the respective main characteristic posture.

Fig. 1. Some state-reflective behavior patterns of the animated agent Smartakus

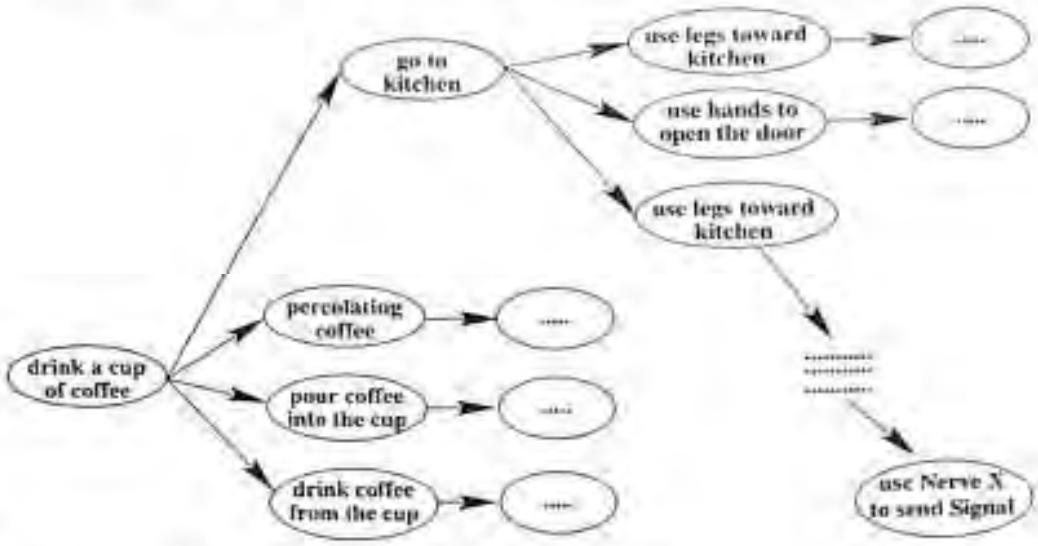


Fig. 2. Creating a plan by dividing the task by human mind

1.6 Dynamic Context-Dependent Modality Fission

Multimodal system output of SMARTKOM consists of synchronized presentations of appropriate graphics and speech output accompanied and moderated by corresponding Smartakus gestures and postures, including cross-modal references.

A crucial computational step is the distribution of the output elements over the available modalities (fission) that is based on a dynamic, context-dependent procedure taking care of several influencing factors, e.g., the dialogue context, the available media and modalities, or user preferences.

For example, in the lean-backward mode of the Home scenario, the user does not look at the display, which means that neither graphics nor gestures can be used. Consequently, all output elements must be adapted to be presented monomodally by speech only without any cross-modal references. The display itself is cleared in such a case, and a special "deactivation" screen is shown.

1.7 Symmetric Multimodality

An important and novel feature of the user interaction in SMARTKOM is symmetric multimodality, which means that all input modalities that may be part of a user input (speech, gesture, facial expression) are, in turn, available as modalities of system output presentations as well. The system uses speech, graphics, gestures, and facial expressions on the output side as well as on the input side.

In accordance with that, for users interacting with SMARTKOM, the system must be "personalizable" in the sense that there is the humanlike communication partner of the user. This is realized as the animated agent Smartakus to which — according to the SDDP paradigm — all tasks are delegated to by the user.

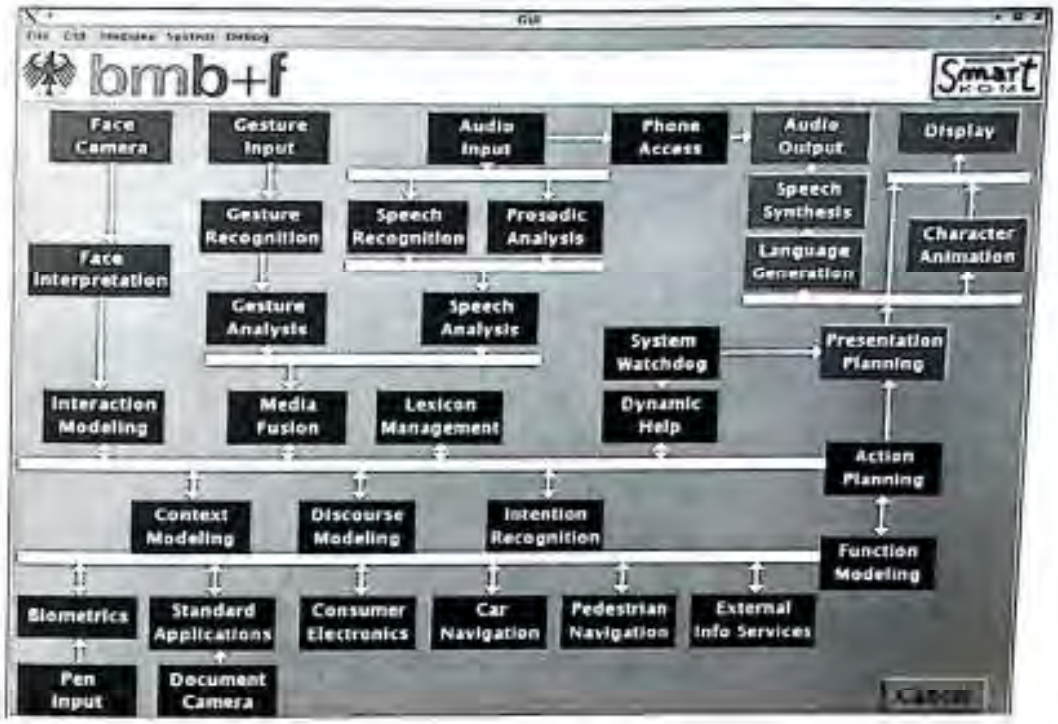


Fig. 3. Fission modules highlighted on the SMARTKOM GUI

2 Media Fission and Presentation Planning

The best way to communicate with a human is to use different information media simultaneously. For example, the use of graphics and speech in a presentation gives a better understanding level than a graphics-only presentation. The partitioning of a presentation into tasks for different media is called "multimedia fission."

2.1 Multimodal Media Fission

For better understanding of multimedia fission, take a look at the human being itself. The example task is to drink a cup of coffee, but there is no atomic human action like this. The human mind first divides this task into appropriate subtasks like going into the kitchen or percolating coffee. Each of the subtasks is divided further into sub-subtasks, and so on until the tasks become atomic, like sending signals over a nerve X to muscle Y. So a complete plan for elementary actions is computed (see Fig. 2 for an example) to be used for completing the given task.

In the same way, the human mind would act if the task were to inform someone verbally about something. In this case, the main subtasks are the modality-specific parts for gesture, mimics, and speech. So the mind would perform a "multimodal presentation" of the information that the human has to tell.

On different levels, some of the subtasks can, of course, overlap with each other, so they have to be synchronized. For example, while speaking the mouth should be used simultaneously with the tongue. Otherwise nobody can be understood.

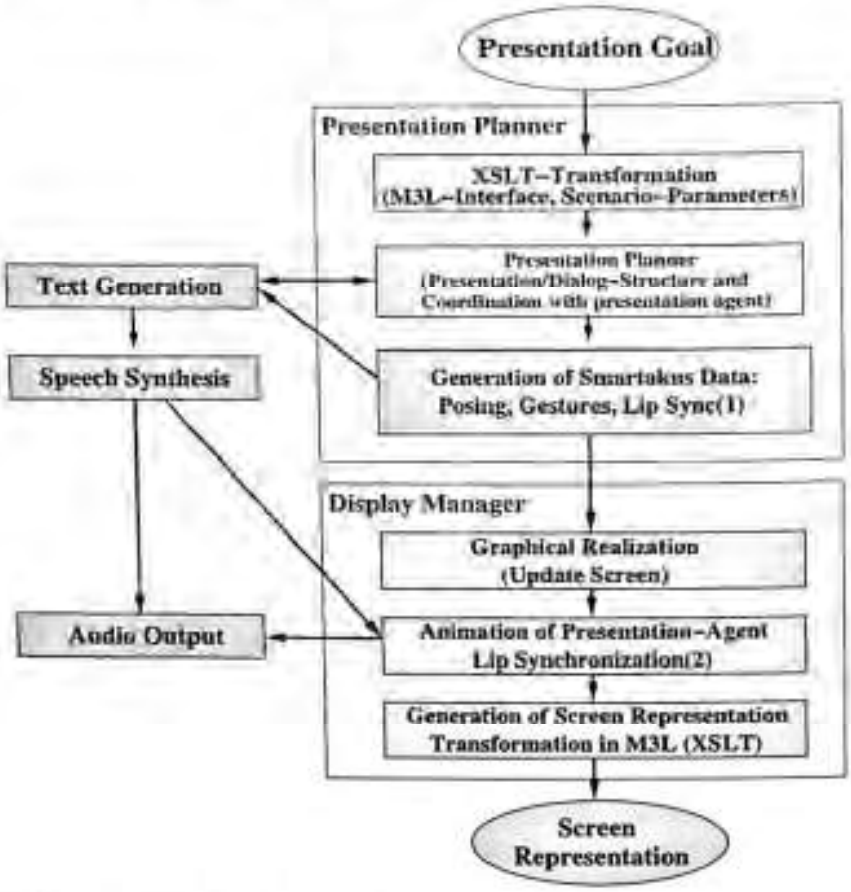


Fig. 4. Architecture of the fission modules

Accordingly, the task of multimodal fission consists of two major steps:

Partitioning: This is the division of the tasks into appropriate subtasks, including the generation of the corresponding evaluation plan. In SMARTKOM this task is performed by the presentation manager (Sect. 2.2).

Evaluation and synchronization of the subtasks: After that, the atomic subtasks are evaluated according to the generated plan. While evaluating the subtasks are synchronized to avoid presentation failures. This task is job of the display manager (Sect. 3)

SMARTKOM has more ways to present information than the human. Besides gestures, mimics, and speech, SMARTKOM also uses graphics, which can be presented on different graphic devices. This makes multimodal fission more complex such that the tasks are distributed between several modules as follows (see also Sect. 3): The presentation manager generates complete plans for graphics, gesture, and mimics output, while the plan for speech output is generated on an abstract high level only. The text generator creates a middle-level plan, i.e., a plan containing content-to-speech representations of the sentences to speak. Finally, the speech synthesis gener-

```
<presentationTask>
 <subTask>
  <presentationGoal>
   <inform> ... </inform>
   <abstractPresentationContent>
    <taskSpecification>
     <intentionContent>...</intentionContent>
    </taskSpecification>
    ...
    <result>
     <broadcast id="bc1">
      <channel> <name>EuroSport</name> </channel>
      <beginTime>
       <time> <at>2000-12-05T14:00:00</at> </time>
      </beginTime>
      <endTime>
       <time> <at>2000-12-05T15:00:00</at> </time>
      </endTime>
      <avMedium>
       <title>Sport News</title>
       <avType>sport</avType>
       <rating>0</rating>
      </avMedium>
     </broadcast>
    </result>
   </abstractPresentationContent>

   <interactionMode>leanForward</interactionMode>
   <goalID>APGOAL3000</goalID>
   <source>generatorAction</source>
   <realizationType>graphicAndSpeech</realizationType>

  </presentationGoal>
 </subTask>
</presentationTask>
```

Fig. 5. M3I document containing input for presentation

ates the low-level actions, i.e., the audio data (plus phoneme information that is used for lip synchronization, see Sect. 4.1).

Also, several modules are used for the evaluation of a presentation plan. The display manager is used to evaluate graphics, mimics, and gestures. For presenting graphics, dynamic and static data like pictures and output texts are combined on the graphics output device. For gestures and mimics, the character animation computes animation sequences for the animated agent Smartakus in such a way that the human mind can perceive them as natural, humanlike gestures and mimics. Speech and audio tasks are evaluated by the audio module. The synchronization of the particular output modality streams is done by the display manager (Fig. 4).

Fig. 6. Widgets for presentations in SMARTKOM-Public and Home

2.2 Presentation Manager

The main task of the presentation manager is the planning of a multimodal presentation that consists of two parts: static gesture-sensitive graphical elements and a corresponding multimodal animation of Smartakus, including gestures referring to objects with aligned audiovisual speech output.

All information that flows between modules is represented as an XML document conforming to the schema-based language M3L, which was developed in SMARTKOM (Herzog et al., 2003). The input of the presentation planner consists of a M3L document that is sent by the action planner or the dynamic help component. It contains an abstract representation of the system intention to be presented multimodally to the user.

2.2.1 Input for Presentations

The first computational step being performed on the M3L input document (see an example in Fig. 5, which specifies a "Sport News" broadcast as presentation goal) is a transformation of the document into the special input format of the core planning component PrePlan (André, 1995; Tschernomas, 1999) by application of an appropriate XSLT stylesheet. The use of stylesheets ensures flexibility with respect to the input structure of the presentation planner. Similarly, different situation-dependent XSLT stylesheets that reflect different dialogue situations are used (one stylesheet per situation). A dialogue situation is defined by the set of parameter for the presentation planner.

Fig. 7. Presentation of touristic sights in SMARTKOM-Mobile

Some examples for the presentation parameters are the current scenario (SMARTKOM-Mobile, SMARTKOM-Home, SMARTKOM-Public), the display size (SiViT and Tablet PC have a resolution of 1024×768 pixel while the PDA has only 240×320 pixels), output language (German, English), available user interface elements (e.g., lists for TV or cinema movies, seat maps, virtual phone or fax devices), user preferences (preference for spoken output while using SMARTKOM-Mobile in a car), available output modalities (graphics, gestures, speech) and design styles (e.g., different background colors for Mobile and Public, see Figs. 6 and 7).

The stylesheet transformations add scenario-specific or language-specific data to the knowledge base of the presentation planner. For example, translations for labels and a logical description of the available screen layout elements are inserted.

2.2.2 Presentation Tasks

The presentation planner starts the planning process by applying a set of so-called presentation strategies (see Fig. 8 for an example), which define how the facts are presented in the given scenario. Based on constraints, the strategies decompose the complex presentation goal into primitive tasks, and at the same time they do the media fission step depending on available modalities, which means they decide which part of the presentation should be instantiated as spoken output, graphics, or gestures of our presentation agent Smartakus. Also, they choose appropriate graphical elements like lists or slide shows.

```
(define-plan-operator
 :HEADER (A0 (Show-Sights ?sightlist))
 :CONSTRAINTS
   (BELP (layout ?layout canshow picture))
 :INFERIORS (
     (A1 (SetLayout ?layout))
     (A2 (seq-forall (?url) with
       (BELP (objectpicture ?sightlist ?url))
         (AI (Add-Picture-URL ?layout ?url))))
     (A3 (Add-Description ?layout ?url))
     (A4 (Persona-Speak))
  ))
```

Fig. 8. A presentation strategy in SMARTKOM

For simple presentation tasks, like showing the phone, the presentation planner creates commands for a graphical element (the phone itself), a speech comment, and a gesture script for Smartakus for pointing at the phone while commenting. In contrast to simple presentations, there are complex presentation tasks that depend on more input and modality parameters. For example, while planning of a TV program, the presentation planner chooses the appropriate graphical element depending on the amount and type of input data. If there is only one broadcast to show, a lot of background information about it can be presented (e.g., directors, actors, and description). The presentation planner must also take into account what the user likes (e.g., if the user likes action films but not the channel ARD, action films will be shown first while broadcasts on channel ARD will be shown at the end or not shown at all). If some information parts are not available, the place reserved for this information can be used to extend other information parts (e.g., if there is a film description, only some actors are shown, but if there is no description available, all actor names are presented. In Mobile, the whole space is used for text output on absence of pictures for a sight description).

After choosing the graphical presentation, appropriate speech and gesture presentations are generated. The gesture and speech form is chosen depending on the graphically shown information. That is, if the graphically presented information is in the focus of a presentation, only a comment is generated for speech output. The goal of the gesture presentation is then to draw the users's attention to the appropriate graphical element. If there is no graphically presentable information or it is insufficient to completely fulfill the presentation goal, more speech is generated.

In SMARTKOM-Home, there is a special mode (called the lean-backward mode) where the user may turn away from the screen, e.g., for watching TV. In that case the visual output modality is not available anymore, so the presentation planner switches to exclusively use speech output to inform the user. However, the presentation agent remains visible on screen just to indicate that the system is still active and that the display can be reactivated.

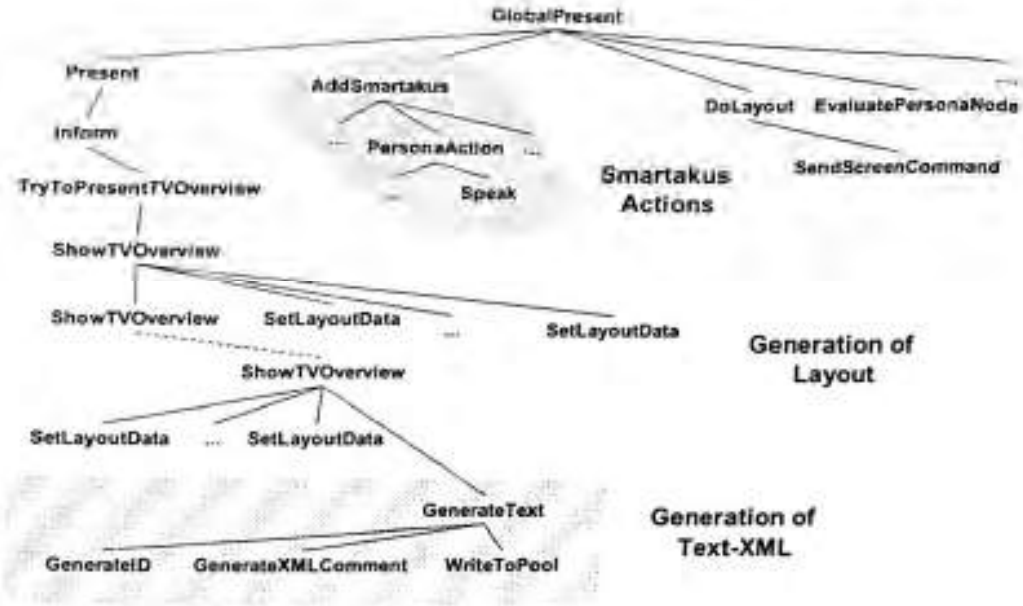


Fig. 9. Presentation plan in SMARTKOM

The presentation planner takes also the size of the graphical display into account. While in SMARTKOM-Home and in SMARTKOM-Public multiple graphical elements can be shown on the screen at the same time, in SMARTKOM-Mobile there is only space for one central graphical object to show on the screen due to the small screen.

In order to solve a presentation task, the Presentation Planner uses a top-down approach with declarative knowledge bases: one static and one dynamic knowledge base. The static knowledge base describes, for example, existing presentation alternatives and dictionaries for text translations. The dynamic knowledge base is derived from the presentation planner input on every presentation task. The presentation planner automatically decomposes the presentation task into subtasks, sub-subtasks, and so on. An important subtask of this planning process is the generation of animation scripts for the presentation agent Smartakus, which have to be related and coordinated with the graphical presentations. The main task of the presentation planner here is the selection of gestures that appropriately supplement the graphical output. The presentation planner composes a special script for Smartakus that is based on the script-language of the PERSONA system (Müller, 2000) on an abstract level. In order to generate speech output, a corresponding speech generation request is sent to the text generator (Becker, 2002) to produce the utterance. The speech synthesis module (Schweitzer et al.) reads this representation and generates the audio data plus the corresponding phoneme sequence for lip synchronization.

The final result of the planning process is a hierarchical plan that reflects the structure of the presentation (Fig. 9). This resulting planning tree is evaluated in order to create a presentation script that is finally passed on to the display manager, which renders the resulting presentation on the available output media and modalities.

3 Graphics and Gesture Generation

To show the generated presentation on a graphic-based medium, the display manager must first generate the layout of all graphical elements as well as configure the presentation agent Smartakus.

3.1 Generation and Layout of Graphical elements

After the presentation script is generated, it is sent to the display manager. The display manager evaluates this script and performs the computational steps as follows.

The display manager first chooses the appropriate graphical layout elements from the knowledge base. According to the planned presentation, these layout elements are dynamically filled with actual data (e.g., movie titles, channel logos, maps) with respect to space restrictions of the chosen graphical elements. This also includes the update of status information that is graphically on-screen, like the display of the virtual cellular phone or the virtual console for TV and video devices.

Since in SMARTKOM multiple graphical objects can be shown at the same time, a layout manager must be used to compute and optimize the arrangement of the objects on the screen and to remove unnecessary objects from the preceding presentation. For that purpose, all layout elements are classified with constraints as follows:

MUST-objects: objects that must be shown in the current context
SHOULD-objects: objects that are not essential but are useful in the current context
CAN-Objects: objects that are currently placed on screen, but are already out of the current context

The classification of the graphical objects is done by the presentation planner while generating the presentation tree (see Sect. 2). The presentation planner defines MUST- and SHOULD-objects for the current presentation task. All other objects that are already on the screen are classified as CAN-objects.

Additionally, constraints for every graphical object can contain boundary information for one and up to four sides. So, some objects can only be placed on a fixed position while others may float within a region. The former is, e.g., essential for the document scanning application, in which the scanning camera has only a limited rotation angle such that all documents must be placed inside the same predefined area. In the document scanning application, a video camera is used to scan a document. After this, the document data is sent via e-mail or a fax machine.

For some objects like status bar for TV and video recorder, it is useful to exclude them from the layout process, because they must always be visible at the same position independent from other objects. This feature can also be set through layout constraints.

Finally, after finishing the graphical presentations, the display manager computes a screen representation in M3L format (see an example in Fig. 10). This representation is published to the SMARTKOM system in order to permit the identification of all objects visible on screen, especially for the gesture analysis to relate gestures to on-screen objects to the real-world objects they represent.

```
<?xml- version="1.0"?>
<presentationContent>
   [...]
  <abstractPresentationContent>
    <performance id="PP325">
      <beginTime> 2003-06-04T20:15:00</beginTime>
      <avMedium>
        <avType>action</avType>
        <avType>scienceFiction</avType>
        <title> Matrix: Reloaded </title>
      </avMedium>
      <cinema>
        <name>Saal 1</name>
        <partOf>
          <movieTheater>
            <name>Lux/Harmonie</name>
          </movieTheater>
          [....]
        </partOf>
      </cinema>
    </performance>
  </abstractPresentationContent>
  [...]
  <panelElement>
    <label id="PN14">
      <boundingShape>
        <leftTop>
            <x>0.0546875</x>
            <y>0.6484375</y>
        </leftTop>
        <rightBottom>
            <x>0.40625</x>
            <y>6898020834</y>
        </rightBottom>
      </boundingShape>
      <contentReference>PP325</contentReference>
      <text>Matrix: Reloaded:  Action, Science Fiction;
            (Lux/Harmonie,     20:15 Uhr)</text>
    </label>
  </panelElement>
  [....]
```

```
abstractPresentationContent>
  <mapLocation id="PP717">
    <locationName> Lux/Harmonie </locationName>
    <objectTypes>
      <objectType> cinema </objectType>
    </objectTypes>
    <geometries>
      <point>
        <x> 3478601.0 </x>
        <y> 5474947.0 </y>
      </point>
    </geometries>
  </mapLocation>

  [...]

</abstractPresentationContent>

  [...]

  <label>
    <boundingShape>
      <leftTop>
          <x>0.732421875</x>
          <y>0.39322916666666663</y>
      </leftTop>
      <rightBottom>
          <x> 0.7421875 </x>
          <y> 0.40625 </y>
      </rightBottom>
    </boundingShape>
    <contentReference>PP717</contentReference>
    <text> Lux/Harmonie </text>
  </label>

  [...]

</presentationContent>
```

Fig. 10. Simplified example of the screen representation

Fig. 11. Some life-like characters used in different systems

3.2 Presentation Agent Smartakus

Life-like characters offer great promise for a wide range of applications (Rist et al., 1997; Cassell et al., 2000; Perlin and Goldberg, 1996; Noma and Badler, 1997; Ball et al., 1997) since they make presentations more lively and entertaining and allow for the emulation of conversation styles common in human–human communication. In virtual environments, animated agents may help users learn to perform procedural tasks by demonstrating them (see different agent characters in Fig. 11). Furthermore, they can serve as a guide through a presentation and can release the user from orientation and navigation problems.

In SMARTKOM, according to the SDDP interaction metaphor (Sect. 1.1 above), the presentation agent Smartakus acts as a dialogue partner and represents the whole SMARTKOM system for the user. Smartakus has the role of a real presenter. He points to objects, speaks to the user, and shows mimics, i.e., realizes the multimodal communication means of the systems.

The presentation agent Smartakus is based on the PERSONA system (Müller, 2000). He is instructed by the presentation planner at an abstract level. Abstract complex commands are dynamically converted to simple agent scripts, which are used to control the animated character.

While converting, constraints are used to generate special so-called BETWEEN-gestures if needed. BETWEEN-gestures are automatically generated gestures for a smooth transition between two agent postures. These gestures are used to smoothly combine different gestures within one fluent animation.

The knowledge source for the behaviours and gestures of Smartakus is a large catalog of predefined agent behaviour patterns. These categorized agent behaviours

Fig. 12. Animation of Smartakus in Home/Public scenarios

form the knowledge base of the presentation subsystem for Smartakus. Smartakus itself is statically modeled with 3D Studio Max (Autodesk, US) as a 3D life-like character. But for efficiency reasons, the deeply 3D-modeled gestures, postures, and behaviours are technically rendered as animated GIF files. During run time, the animation of Smartakus is dynamically generated using the elements of this catalog by visually combining elementary gestures to larger animation sequences (see Figs. 12 and 13) in conjunction with inserting appropriate BETWEEN-gestures wherever needed.

Because of the different display sizes in the SMARTKOM scenarios, two Smartakus characters were developed:

Full Smartakus is a complete character. It has a head, body, and two arms with hands. There are more gestures for it in the catalog, and it can do gestures more precisely which makes the animation computationally more difficult.

Reduced Smartakus consists only of a smaller head and one hand. This minimizes the data size to be transferred and makes more places available on the screen for showing information data without cutting any modalities. Lip-synchronized speech output, including gestures and mimics, are equally possible.

Both characters use the lip and gesture synchronization for presenting contents. And both characters can move on the screen to avoid overlapping with information.

4 Output Synchronization and Media Coordination

All deictic gestures have to be synchronized with the display of the corresponding graphical output to which they are related. Also, the acoustic speech output has to be synchronized by appropriate lip movements of the communication assistant Smartakus.

The display manager is responsible for the realization of visible output, while the audio output module realizes speech output. But audio output and visual output are

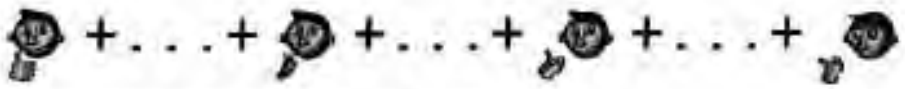


Fig. 13. Animation of Smartakus in Mobile scenario

performed in principle independently from each other and can even be processed and realized on different machines. Therefore to present both modalities simultaneously and synchronously, we have to coordinate them over time to get lip-synchronized output.

4.1 Lip Synchronization

The lip synchronization in SMARTKOM (Poller and Müller, 2002) is based on an underspecified mapping between acoustic and visual units (so-called phonemes and visemes). In our system a viseme is defined as a specific mouth position picture. Due to the cartoonlike character of Smartakus (neither tongue nor teeth are visible) only a limited variety of mouth/jaw positions or movements are possible at all (Fig. 14). Consequently, the only parameters that are relevant in our case to describe mouth positions are the lip rounding and the jaw opening.

Fig. 14. Possible visemes for lip synchronization

We found that almost every phoneme has one single corresponding viseme, while only a few of them (plosives and diphtongs) have to be mapped to at least two visemes to visualize their articulation appropriately (Schweitzer et al.). Thus, in such cases the mapping partly becomes a one-to-many mapping in the sense that one phoneme can be mapped to more than one viseme. Furthermore, the mapping has to be partly underspecified in lip rounding and jaw opening as well to be able to take articulation effects into account. Since the audio output module and display manager are two separate modules in SMARTKOM that in principle work independently from each other, the idea to synchronize lip movements with speech output is to synchronize the individual time points at which corresponding acoustic and visual events occur as exactly as possible.

The speech synthesis module in SMARTKOM not only produces audio data but also produces a detailed representation of the phonemes and their exact time points (in milliseconds) inside the audio signal. Based on this representation and the phoneme–viseme mapping mentioned above, the character animation module generates a lip animation script for Smartakus that is then executed during speech output.

The animation script is generated by a procedure iterating over the phonemes to consecutively specify the concrete viseme(s) and the exact point in time at which they have to be shown in the output animation stream. In terms of Fig. 14 the procedure always tries to select a viseme that has a common borderline with the previous viseme whenever possible (also by inserting intermediate visemes, if necessary).

4.2 Coordinated Lip and Body Animation of Smartakus

During presentation, Smartakus often points to graphical objects on the screen to draw the user's attention to them. To make such presentations more lively and natural, Smartakus is pointing while speaking. For this purpose the body animation (which is performing the pointing gestures) is separated from head (and visemes) animation (which is performing the lip animations). Additionally, heads with visemes for left, front, and right side exist. During pointing the body often moves into a posture for which the head must also be moved horizontally or vertically to realize a natural posture switch of the entire agent. Therefore, for every body posture and every head posture an image with one ground point is generated to be able to calculate body and head coordinates. Such ground point images can be easily generated in advance from the 3D model of Smartakus, such that the synchronization of heads and bodies can be done fully automatically at run time (Fig. 15).

Since the display manager, character animation module, and the audio device of the SMARTKOM system may run on different computers, the start of the output of the audio data and the start of the lip animation must be synchronized in a special manner. A fixed point in time at which the audio signal has to start is defined dynamically, and at exactly the same time the animation is started by the PERSONA player.

4.3 Visual Perceptive Feedback

In order to increase the liveliness of the character, the character shows idle-time actions like breathing or waving. Another job of the presentation agent is to show the actual system state (also called global system working state) to the user (see Fig. 1 above again for details).

Here is a short explanation of the agent behaviour patterns that are assigned to some of the global system working states (Poller and Reithinger, 2004).

IDLE: In idle state, the agent shows its liveliness by breathing regularly and blinking its eyes.

LISTENING: In the listening state, the animated agent purposefully turns toward the user in order to give him its best attention while he is, e.g., speaking to the system.

UNDERSTANDING: In the understanding state, the agent signals the user that the system is currently analyzing his input. The agent moves its right hand to its temple to scratch with its index finger.

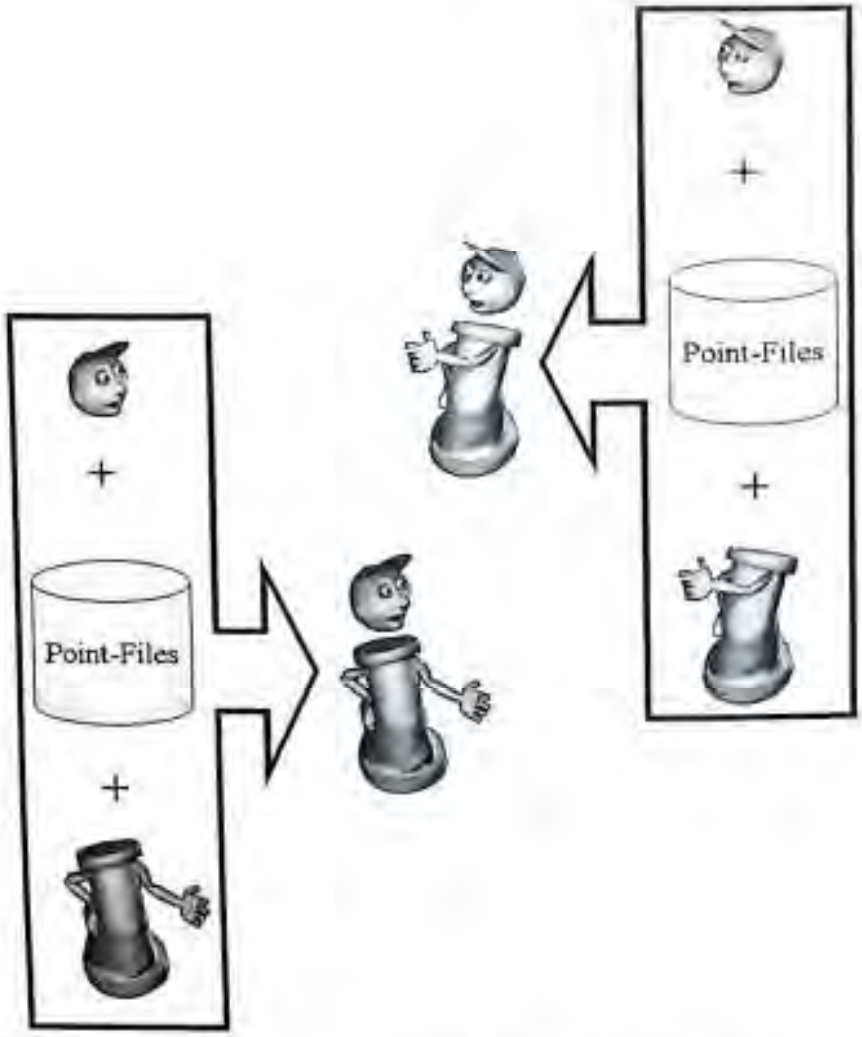


Fig. 15. Gestures are generated out of body parts using reference points

UNDERSTOOD: The upraised index finger is used to signal that the user input was comprehended (global system state "understood").

BARGE-IN: In the "bargeIn" interaction state, an upraised hand is used to show the user that the system is still processing the last input and will not accept barge-in data. Currently, our system cannot interact in a barge-in mode.

PROMPTING: If there were communication pauses, this gesture is used in the idle state to prompt the user implicitly to continue his interaction with the system.

PROCESSING: This gesture is used in the "processing" state to indicate that the system is currently accessing its knowledge sources to fulfill the user request.

MOVE: This gesture is used as part of the output presentations to enable the agent to move around on the screen.

These agent behaviors are shown as soon as the the respective state of the system/conversation is reached.

5 Conclusion

In this chapter we described the design of the novel human–computer interaction in SMARTKOM in line with the *situated delegation-oriented dialogue paradigm (SDDP)* and presented in a detailed compilation the multimodal system output planning and generation process. As shown, this is a very complex procedure containing the modality fission, i.e., the distribution of the output elements over the available output modalities, the generation of the respective modality streams, and finally the coordination and synchronization of all output streams within one multimodal presentation. Consequently, this procedure was divided into a bundle of actively communicating and cooperating modules.

The SMARTKOM solution of coordinating and synchronizing multiple output modality streams in a pleasing and natural user interaction situation was significantly determined by the principle of "interacting though delegation." Having a conversational communication with a virtual communication assistant on screen to which all tasks are delegated brings completely new human-computer interaction methods to light for further investigation.

References

E. André. *Ein planbasierter Ansatz zur Generierung multimedialer Präsentationen.* PhD thesis, Universität des Saarlandes, 1995.

G. Ball, D. Ling, D. Kurlander, J. Miller, and D. Pugh. Lifelike Computer Characters: The Persona Project at Microsoft. In: J.M. Bradshaw (ed.), *Software Agents*, Menlo Park, CA, 1997. AAAI/MIT Press.

T. Becker. Practical, Template–Based Natural Language Generation With TAG. In: *Proc. TAG+6*, Venice, Italy, May 2002.

J. Cassell, J. Sullivan, S. Prevost, and E. Churchill. *Embodied Conversational Agents.* MIT Press, Cambridge, MA, 2000.

G. Herzog, H. Kirchmann, S. Merten, A. Ndiaye, P. Poller, and T. Becker. MULTIPLATFORM Testbed: An Integration Platform for Multimodal Dialog Systems. In: H. Cunningham and J. Patrick (eds.), *Proc. HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, pp. 75–82, Edmonton, Canada, 2003. Association for Computational Linguistics.

J. Müller. *Persona: Ein anthropomorpher Präsentationsagent für Internet-Anwendungen.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2000.

T. Noma and N.I. Badler. A Virtual Human Presenter. In: *Proc. IJCAI-97 Workshop on Animated Interface Agents: Making Them Intelligent*, pp. 45–51, Nagoya, Japan, 1997.

S. Oviatt. Multimodal Interfaces. In: J.A. Jacko and A. Sears (eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, pp. 286–304, Mahwah, NJ, 2003. Lawrence Erlbaum.

K. Perlin and A. Goldberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds. *Computer Graphics*, 28(3), 1996.

P. Poller and J. Müller. Distributed Audio-Visual Speech Synchronization. In: *Proc. ICSLP-2002*, Denver, CO, 2002.

P. Poller and N. Reithinger. A State Model for the Realization of Visual Perceptive Feedback in SmartKom. In: *Proc. Interspeech 2004, (ICSLP, 8th Int. Conf. on Spoken Language Processing)*, Jeju Island, Korea, 2004.

T. Rist, E. André, and J. Müller. Adding Animated Presentation Agents to the Interface. In: *Proc. 1997 Int. Conf. on Intelligent User Interfaces*, pp. 79–86, Orlando, FL, 1997.

A. Schweitzer, G. Dogil, and P. Poller. Gesture-Speech Interaction in the SmartKom Project. Poster presented at the 142nd meeting of the Acoustical Society of America (ASA). http://www.ims.uni-stuttgart.de/~schweitz/documents.shtml. Cited 11 April 2006.

V. Tschernomas. *PrePlan Dokumentation (Java-Version)*. Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany, 1999.

W. Wahlster. SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: R. Krahl and D. Günther (eds.), *Proc. Human Computer Interaction Status Conference 2003*, pp. 47–62, Berlin, Germany, June 2003. DLR.

Natural Language Generation with Fully Specified Templates

Tilman Becker

DFKI GmbH, Saarbrücken, Germany
becker@dfki.de

Summary. Based on the constraints of the project, the approach chosen for natural language generation (NLG) combines the advantages of a template-based system with a theory-based full representation. We also discuss the adaption of generation to different multimodal interaction modes and the special requirements of generation for concept-to-speech synthesis.

1 Introduction

The natural language generation (NLG) module of SMARTKOM is part of the kernel dialogue system and as such is used in all scenarios and applications. Its design is guided by the need to (i) adapt only knowledge sources when adding a new application and (ii) generalizing the knowledge sources from the applications. The NLG module generates not just plain text but rather generates complete syntactic structure and discourse information to supply highly annotated information structures for the concept-to-speech approach of SMARTKOM see also Schweitzer et al. (2006) on Speech Synthesis.

A further project constraint was the need to adapt the kernel dialogue system quickly to new applications. Initially, this could be achieved by generating fixed system responses for simple dialogues. However, these then needed to be extended to a flexible, "full" generation system.

The obvious choice for the initial system is a so-called template-based NLG system. Usually such a system is perceived as the opposite of full generation. However, as we will argue in the next section, a template-based NLG system can indeed be seamlessly extended into a full system. In SMARTKOM, we have taken this approach. The perceived differences between so-called full and template-based NLG systems are also discussed, e.g., in van Deemter et al. (1999).

The NLG system of SMARTKOM is based on lexicalized tree-adjoining grammar (LTAG) with full feature structures as its syntactic representation formalism. Development in the NLG module was mainly for German, but for some applications the templates and rules were also ported to English.[1]

[1] This work was carried out by Johno Bryant and David Gelbart at ICSI, Berkeley.

2 Fully Specified Templates

The design decision to use template-based generation in SMARTKOM is driven by practical considerations: We wanted generator output as early as possible in the project, and there were various applications with a considerable amount of nonoverlapping genres and relatively constrained linguistic coverage, down to a fixed list of utterances in the first systems. On the other hand, simple string concatenation is not sufficient. For example, the integration of concept-to-speech information, especially in the way the synthesis component of SMARTKOM is designed, calls for an elaborate syntactic representation, i.e., phrase structure trees with features, to guide the decisions on prosodic boundaries. At least since Reiter (1995) (also see Becker and Busemann (1999)), the use of templates and "deep representations" is not seen as a contradiction. Picking up on this idea, the generation component in SMARTKOM is based on fully lexicalized generation (Becker, 1998), using whole parts of a sentence together as one *fully specified template* that is then represented not as a string but rather as a partial TAG derivation tree (Fig. 1).

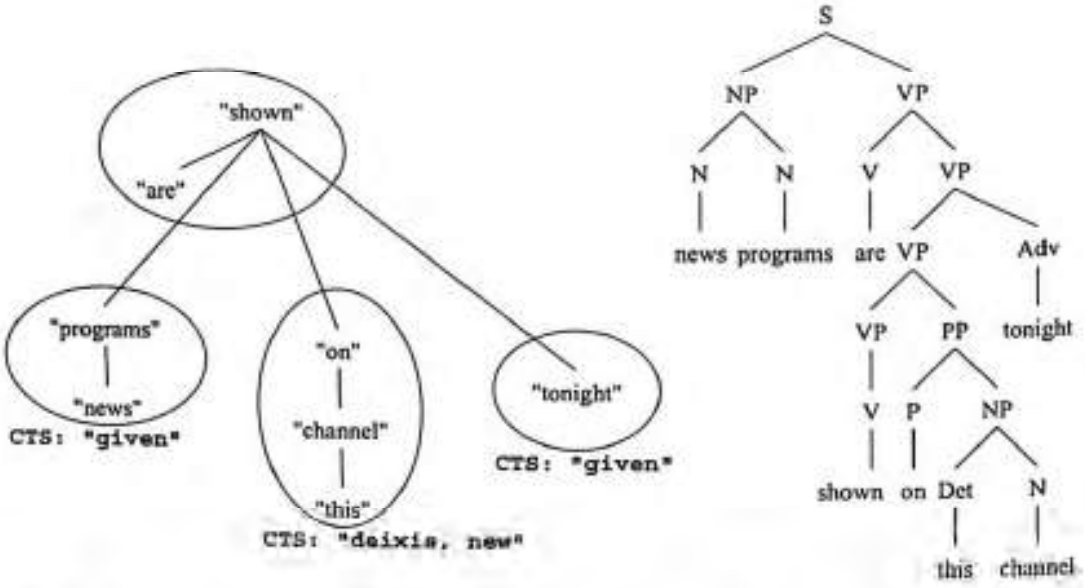


Fig. 1. Derivation tree with partial CTS (concept-to-speech) markup. Each ellipse is a fully specified template. The sentence-planning process combines such templates to a complete derivation tree, which when executed results in the derived tree shown to the *right* (shown without the feature structures attached to each node)

2.1 Specifying Only Relevant Representations

With this approach to specify intermediate levels of representation (which commonly only exist inside the NLG module), the question remains whether *all* levels of repre-

sentation have to be specified. Clearly, this is desirable but not necessary in SMARTKOM. Thus only the levels of dependency and phrase structure are represented fully. Dependency is necessary to guide the top-down generation approach, and phrase structure is necessary for (concept-to-)speech synthesis. However, there is nothing preventing the later inclusion of other levels of representation, e.g., adding a level of semantic representation (Scheffler, 2003; Joshi and Vijay-Shanker, 1999) to allow for the inclusion of a separate realization module and which might also be used, e.g., by a sentence aggregation component.

3 From Templates to a Full Grammar

The TAG formalism (Joshi et al., 1975; Joshi and Schabes, 1997) offers an elegant way to reconcile the two notions of templates and deep representations. A TAG grammar consists of a set of partial phrase structure trees, so-called elementary trees. These trees are then combined to form the final, so-called derived tree by two elementary operations, adjunction and substitution. The applied operations are recorded in a graph called the derivation tree.

In a lexicalized TAG (LTAG), each elementary tree contains only one leaf node with a lexical element. There, the derivation tree has one node for each lexical element of the sentence and represents a dependency structure between the words. Note also that in an LTAG the grammar is, in fact, a kind of extended lexicon.

However, the pure TAG formalism has no constraints on the number of lexical elements in one elementary tree. In fact, even in LTAG, idiomatic expressions are usually represented as elementary trees with multiple lexical elements. So, for the initial version of the NLG system, we used large elementary trees that contained the complete expressions of the templates. Later versions of the grammar and generation rules then used ever smaller trees until eventually all elementary trees were, in fact, lexicalized in the sense of an LTAG.

3.1 Development Steps

In this section, we describe the intermediate steps that led the development over the time of the project from the initial version with large templates (typically full sentences) to the final version using a fully lexicalized TAG grammar. The work in Scheffler (2003) describes an approach to further extend the lexicalized TAG grammar with a flat semantics to allow for an extraction of a separate syntactic realizer from the NLG module of SMARTKOM.

The development from the initial version went from (i) simple templates to (ii) phrasal templates to (iii) feature-structure-enriched trees to a (iv) system based on a fully lexicalized TAG grammar:

(i) **Simple templates:** The initial SMARTKOM generation component was an instance of a simple template-based NLG system. Our "grammar" consisted of fully specified phrase structure trees at the sentence level that were augmented

with attributes specifying, e.g., sentence mode and part of speech for CTS purposes. Note that no derivation beyond tree selection is necessary when the selected tree is already a complete utterance.[2]

Of course, even the first version went beyond a "canned text" system, and we already needed a planning operator that could instantiate the simple templates (actually, overwrite leaf nodes in the derived tree with strings taken from the generator input XML document).

(ii) Phrasal templates: As soon as we encountered scenarios that required more variability, we began to disjoin some sentence-level templates into smaller templates, keeping most features untouched.

Accordingly, the corresponding planning rules grew more complex. In order to capture linguistic generalizations, we began to delegate the planning of noun phrases and (possibly optional) modifiers to subrules and began to extract the corresponding NP elementary trees.

(iii) Feature-based TAG: As a next step, all trees were annotated with top and bottom feature structures (Vijay-Shanker, 1987), which include, e.g., case, argument status (subject/object/adjunct/ppobject) and discourse information for the synthesis component. The simple realizer was enhanced with a unifier, enabling us to make use of feature checking, percolation and structure sharing (co-indexing, reentrancy). A morphology module was prepared that could easily be integrated.

(iv) A full grammar: The final step then was moving to a fully lexicalized grammar, further disjoining the TAG templates. The resulting set of lexicalized TAG trees can be considered a partial grammar for the current SMARTKOM scenarios. For better maintainability, we decided to name and group them in the vein of the XTAG grammar (XTAG Research Group, 2001). Since we have implemented the standard definition of (L)TAG, the free-word-order phenomena of German are captured only by including all ordering variants as separate elementary trees in our grammar.

Our architecture left room for all of these extensions that were introduced gradually while the generation component was maintained in a working condition. There was no need to fork an experimental or "next generation" development branch or even to start a new architecture from scratch. Linguistic knowledge was added monotonically, i.e., once it was encoded, it did not become obsolete.

The description of our approach so far has concentrated on the Realizer part of the NLG system for which we have executed the development from fully specified templates to a "full" grammar rigorously. Input to the NLG component, however, is on a language- independent conceptual level. Mapping directly from this nonlinguistic input to fully specified German dependency (derivation) trees proved to be quite a challenge for the planning rule authors. As our templates developed into lexicalized TAG trees, planning rules became fine-grained, and a principled way of writing and organizing them had to be found.

[2] An utterance in the SMARTKOM generator is either a single sentence or a dialogue turn that consists of sentences connected by discourse functions.

3.2 Dependency and Speech Markup

Templates in template-based NLG systems are commonly surface strings with gaps that are filled from the input information. In the SMARTKOM NLG system, templates are specified on the level of derivation trees rather than on derived trees (which contain the surface string on their leaves) or even surface strings directly. This has several advantages. In the context of concept-to-speech synthesis, it is necessary to add markup to entire phrases which need not even be contiguous in the surface string. This can be done easily by adding the information to the corresponding node in the derivation tree from where it is percolated and automatically distributed to the corresponding parts of the utterance when constructing the derived tree and the string from the derivation tree. Such markup relates to parts of the output that have to be (de)emphasized, parts that refer to objects in the graphical presentation and must be coordinated with a pointing gesture.

Figure 1 shows a derivation tree with speech-relevant markup on some nodes. Besides mere convenience in the markup,[3] the additional power of TAG allows us to distribute semantically connected markup to discontinuous parts in the final string. Since formal complexity is a very different issue in generation than in parsing, we are open to use extensions from the standard TAG formalism as in Rambow (1994) or Gerdes and Kahane (2001), which might be necessary for German.

4 Adding Levels of Representation

As we have shown in the previous sections, the templates for template-based generation need to have a representation that is as detailed as possible, in order to make generation very flexible, while at the same time preserving its advantages. For relatively restricted input, on the other hand, it is not necessary to fully specify all the levels of representation that are relevant to natural language generation. In this section we will discuss how a modularized generator architecture can be extended by adding new levels of representation.

As the number of possible outputs increases, so does the complexity of the set of planning rules that map from the conceptual input to the level of TAG derivation trees. At some point, this rule set needs to be organized by adding additional representational levels. An extended new architecture is shown in Fig. 2. The new architecture contains two new, linguistically well-defined levels of representation, semantics and argument structure. This helps the modularization of rules, because they can be strictly linguistically motivated (i.e., levels of linguistic representation are not processed all in one step, as previously).

The individual steps and the representational levels of the new generator will be developed as follows:

[3] For example, XML-style opening and closing parentheses can be integrated into the trees and thus are realized by a single marked node vs. the situation in a classical context-free-based string-expanding template generator, where opening and closing elements have to be given independently—a typical source for errors.

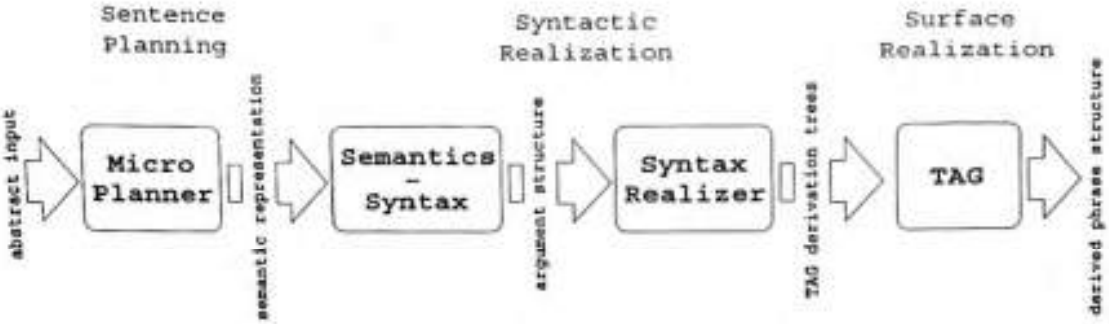


Fig. 2. The improved architecture of the system, with additional levels

Microplanner

Output of the Microplanner will be a flat semantic representation, such as MRS (Copestake et al., 1999). The planner maps the domain-dependent relations of the abstract generator input onto more general semantic relations. The semantic representation must also include additional information of pragmatic and/or situational content, such that this kind of information is available in further steps.

Semantics–Syntax Interface

The representation at the semantics–syntax interface level is argument structure, i.e., a dependency tree enriched with feature annotations at the nodes. Edges are labeled with the kind of dependency relation that holds between the two nodes, e.g., "temporal adjunct," "argument1," "argument2." The arguments are understood more as underlying deep syntactic roles (agens, patiens, etc.) than as concrete forms of realization of those (subject, object, etc.).

During this phase, lexicalization must be carried out, at least as far as part of speech and dependency structure are affected. The heuristics that guides us in defining what specifications are made during the lexicalization and in the argument structure, and which are left open as syntactic or morphological variants to be decided later in the syntactic realizer, is to prefer later specification to earlier specification, as long as this does not inhibit the generation process in any significant way. Choices that will be left for later steps in the generator architecture include synonyms, auxiliaries, active/passive, prepositions, etc.

Syntactic Realization

In this phase, the concrete syntactic representation for an utterance is generated. Argument structure is traversed and appropriate TAG elementary trees are found. This traversal can be done top-down (as, for example, in McCoy et al. (1992)). Edges of the argument structure usually translate into edges in the derivation tree (a tree is substituted or adjoined into its mother of the argument structure), but not always, as is clear for adjectives (multiple adjunctions), etc.

Surface Realization

The derivation trees and derived trees are formally equivalent, because all operations and their order are fully specified in the derivation tree. So in this step, the tree selections, adjunctions and substitutions planned previously and encoded in the derivation tree are actually executed, and feature unification and thus feature percolation (needed for morphology) takes place.

5 Adaptive Multimodal Generation

An important decision in the presentation of system output in a multimodal dialogue system like SMARTKOM is its distribution over the available modalities. Some of the relevant factors are:

- *Appropriateness of media:* Larger sets of systems results, e.g., from a database query, are most conveniently displayed as tables or lists in the graphics modality, leaving a commenting function for the speech modality. Short feedback on the execution of user commands, e.g., switching the TV set to another channel, can best be presented in speech.
- *User preferences:* Given an appropriate user model, personal preferences, e.g., about the level of verbosity in speech output, can be used.
- *Redundancy:* Whenever appropriate, information should be repeated in other modalities, which can resolve ambiguities for the user.

An additional interesting property of SMARTKOM in this context is the definition of various *interaction modes*:

- *Lean-forward mode:* in the home scenario, when the user is sitting on the couch, leaning forward, i.e., looking at and touching the screen, listening and talking
- *Lean-back mode:* in the home scenario, when the user is not looking at the screen, i.e., the interaction is restricted to audio only
- *Theater mode:* in the mobile scenario, when the user uses only the screen
- *Privacy mode:* also in the mobile scenario, when the user uses gestures (and potentially also speech) for input but wants only screen but not audio output from the system, which could be overheard by others

Our implementation of lean-forward and lean-backward in SMARTKOM puts the presentation design into the NLG module: The current mode is given as a parameter, and the initial planning stage selects the presentation templates and the corresponding data. Consider the following example from the home scenario:

User: *What's on TV tonight?*

In lean-forward mode, the display will show a selected list of films, using the screen as the preferred modality for mass data, filling it with the maximum of programs. This maximum depends on the graphical design decisions and available

screen area. In this case, speech is used only as a redundant metainformation, explaining the status of this list of films and implicitly giving feedback about the system's understanding of the user request: *Here you can see a list of the films that are on TV tonight.*

In lean-back mode, the display cannot be used, and the communicative constraints of speech have to be considered. In this case, speech is again used to give feedback about the system's understanding of the user request, but only the first three[4] programs are presented: *I have, for example, the following three programs: "Das Boot" is shown on channel PRO7 at 8pm, ...".*

The selection between these presentations is made within the NLG module, which contains separate sets of planning rules for the different presentation modes. Note that the output differs in the amount of information conveyed, thus making a request for more information more likely in the lean-back case. This could be made explicit in changing the expectations published by the action planner, however, SMARTKOM's ability to deal with mixed initiative dialogue allows the system to handle all possible user reactions in an equal manner.

6 Tools

Currently we have editors available[5] for the planning rules and for the TAG-tree templates. Both build on XML representations of the knowledge bases and present them in an easily accessible format: a directory structure as known from the Windows Explorer for the set of trees and a graphical tree editor for the TAG trees.

In an ideal development environment, with a (partial) TAG grammar in existence, the rule editor can be enhanced with a TAG-parser: To add a new template to the generator, the user can type in an example sentence, have it parsed, select the correct parse, mark (delete) the variable parts, keeping the fixed part and add the remainder of the rule. Thus rules can be created without ever writing trees by hand, thereby avoiding inconsistencies in the set of templates.

7 Conclusion

Modular design and a reasonable amount of additional work (for fully specifying the templates) make template-based generation systems scalable, even up to a "full" system, while still keeping the advantage of rapid development of initial systems. We showed that extension of the generator is possible along multiple axes: making the templates smaller to get a grammar, extending the syntactic representation and adding new representational levels to the generator architecture.

[4] The exact number is a parameter to be determined by experiments and also a possible parameter for a user model.

[5] Thanks are due for the implementation and also further work on the NLG system by Jan Schehl, Quan Nguyen, Michael Kaißer, Christian Pietsch, and Tatjana Scheffler.

Thus, the SMARTKOM NLG system supports rapid prototyping without closing the path to a step-by-step development of full system. The TAG grammar formalism is particularly well suited to this approach since all relevant information is kept in the elementary trees, which can contain either just one lexical item (LTAG) or multiple words up to entire sentences. This setup also supports the needs of the concept-to-speech synthesis approach in SMARTKOM. Finally, the multimodal setup raises new issues in adapting the NLG system in its multimodal setting to the various interaction modes defined for SMARTKOM.

Acknowledgments

The author is indebted to Tatjana Scheffler and Christian Pietsch for their work on the NLG module and their contributions to early versions of parts of this chapter.

References

T. Becker. Fully Lexicalized Head-Driven Syntactic Generation. In: *Proc. 9th Int. Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada, August 1998.

T. Becker and S. Busemann (eds.). *May I Speak Freely? Between Templates and Free Choice in Natural Language Generation*, Bonn, Germany, September 1999. Workshop at the 23rd German Annual Conference for Artificial Intelligence (KI '99).

A. Copestake, D. Flickinger, I.A. Sag, and C. Pollard. Minimal Recursion Semantics. An Introduction. Draft, 1999.

K. Gerdes and S. Kahane. Word Order in German: A Formal Dependency Grammar Using a Topological Hierarchy. In: *Proc. ACL 2001*, Toulouse, France, 2001.

A. Joshi and Y. Schabes. Tree-Adjoining Grammars. In: G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, vol. 3, pp. 69–124, Berlin Heidelberg New York, 1997. Springer.

A.K. Joshi, L.S. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Science*, 10(1), 1975.

A.K. Joshi and K. Vijay-Shanker. Compositional Semantics for Lexicalized Tree-Adjoining Grammars. In: *Proc. 3rd Int. Workshop on Computational Semantics*, Utrecht, The Netherlands, January 1999.

K. McCoy, K. Vijay-Shanker, and G. Yang. A Functional Approach to Generation With TAG. In: *Proc. 30th ACL*, pp. 48–55, Newark, DE, 1992. Association for Computational Linguistics.

O. Rambow. *Formal and Computational Models for Natural Language Syntax*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1994. Also vailable as Technical Report ICRS-94-08 from the Institute for Research in Cognitive Science.

E. Reiter. NLG vs. Templates. In: *Proc. 5th European Workshop in Natural Language Generation*, pp. 95–105, Leiden, The Netherlands, May 1995.

T. Scheffler. A Semantics Interface and Syntactic Realizer for TAG. Master's thesis, University of the Saarland, Germany, 2003.

A. Schweitzer, N. Braunschweiler, G. Dogil, T. Klankert, B. Möbius, G. Möhler, E. Morais, B. Säuberlich, and M. Thomae. Multimodal Speech Synthesis, 2006. In this volume.

K. van Deemter, E. Krahmer, and M. Theune. Real vs. Template-Based Natural Language Generation: A False Opposition? In: T. Becker and S. Busemann (eds.), *May I Speak Freely? Between Templates and Free Choice in Natural Language Generation*, KI-99 Workshop, Bonn, Germany, 1999.

K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1987.

XTAG Research Group. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, The Institute For Research In Cognitive Science (IRCS), Philadelphia, PA, 2001.

Multimodal Speech Synthesis

Antje Schweitzer[1], Norbert Braunschweiler[1], Grzegorz Dogil[1], Tanja Klankert[1], Bernd Möbius[1], Gregor Möhler[1], Edmilson Morais[1], Bettina Säuberlich[1], and Matthias Thomae[2]

[1] Institute of Natural Language Processing, University of Stuttgart, Germany
{antje.schweitzer,braunnt,dogil,klankert,moebius,moehler,emorais,bettina.saeuberlich}@ims.uni-stuttgart.de

[2] DaimlerChrysler AG, Research and Technology, Ulm, Germany
thomae@ei.tum.de

Summary. Speech output generation in the SMARTKOM system is realized by a corpus-based unit selection strategy that preserves many properties of the human voice. When the system's avatar "Smartakus" is present on the screen, the synthetic speech signal is temporally synchronized with Smartakus visible speech gestures and prosodically adjusted to his pointing gestures to enhance multimodal communication. The unit selection voice was formally evaluated and found to be very well accepted and reasonably intelligible in SMARTKOM- specific scenarios.

1 Introduction

The main goal of the speech synthesis group in SMARTKOM was to develop a natural sounding synthetic voice for the avatar "Smartakus" that is judged to be agreeable, intelligible, and friendly by the users of the SMARTKOM system. Two aspects of the SMARTKOM scenario facilitate the achievement of this goal. First, since speech output is mainly intended for the interaction of Smartakus with the user, most of the output corresponds to dialogue turns generated by the language generation module (Becker, 2006). Therefore, most speech output can be generated from linguistic concepts produced by the language generation module ("concept-to-speech synthesis", CTS) instead of from raw text ("text-to-speech synthesis", TTS). The advantage of CTS over TTS is that it avoids errors that may be introduced by linguistic analysis in TTS mode. Second, the CTS approach narrows down the SMARTKOM synthesis domain from a theoretically open domain to a restricted domain, which makes unit selection synthesis a promising alternative to diphone synthesis for the SMARTKOM application.

Multimodality introduces additional requirements for the synthesis module. The visual presence of Smartakus on the screen during speech output requires lip synchronization. Furthermore, Smartakus executes pointing gestures that are related to objects which are also referred to linguistically. These pointing gestures influence

the prosodic structure of the utterance and necessitate temporal alignment of the gestural and linguistic modes. Another momentous requirement was that the graphical design of Smartakus was given before the voice database was recorded. This entailed that the appropriateness of the speaker's voice for Smartakus could be included as an important factor in the speaker selection process.

In developing the synthesis voice for Smartakus, we pursued the following strategy: After the speaker selection process, a diphone voice was developed first. This voice served both as a starting point for implementing a unit selection voice by the same speaker tailored to the typical SMARTKOM domains, and as the default voice for external open domain applications that require TTS instead of CTS. The diphone voice and the unit selection voice were both evaluated in the progress of the project.

This chapter is organized as follows. We focus on the prosody generation in CTS mode in the subsequent section. The speaker selection process is described in Sect. 3. Sect. 4 concentrates on the unit selection voice. Lip synchronization and gesture-speech alignment are discussed in Sect. 5. Finally, the two evaluation procedures are described in Sect. 6.

2 Concept-to-Speech Synthesis

The motivation for CTS synthesis is the view that the linguistic content of an utterance determines its phonological structure and prosodic properties. It has been shown that prosodic structure can reflect aspects of syntactic structure, information structure, and discourse structure (Culicover and Rochemont, 1983; Selkirk, 1984; Cinque, 1993; Hirst, 1993; Abney, 1995; Büring, 1997; Mayer, 1999). The challenge in TTS is that text represents only a very reduced version of the full linguistic content of an utterance. It not only lacks marking of higher-level linguistic structure, but may also be ambiguous with respect to syllabic and segmental structure due to abbreviations and homographs. All these properties have to be inferred from the text in TTS. The idea of CTS is to use the full linguistic structure of an utterance, i.e., the original "concept," instead of its raw textual representation. This structure is available in dialogue systems that generate utterances dynamically. In SMARTKOM, it is available with some exceptions: many utterances contain material retrieved from external databases, such as movie titles or geographical names. Although the overall structure of such utterances is known, the internal structure of the retrieved items is unknown. They may contain abbreviations, material in unknown languages, or, particularly in the case of movie titles, may even have their own internal linguistic structure.

The main advantage of CTS in SMARTKOM is therefore the availability of higher-level linguistic structure, which influences the prosodic structure of an utterance. Cinque (1993) gives a detailed account of how syntactic structure determines the default location of sentence stress. We have implemented an algorithm motivated by Cinque's findings. The prediction of prosodic structure including pitch accent and boundary types from linguistic structure is described in more detail in Schweitzer et al. (2002). Here we only give a brief description of the concept structure, the pre-

diction of phrasing, and the implementation of Cinque's account for accent placement.

2.1 Concept Input

Concepts in SMARTKOM contain information on three linguistic levels. The highest level of annotation used for prosody prediction is the **sentence level**. Sentence mode (declarative, imperative, yes/no-question, or wh-question) is annotated on this level. This kind of information is mainly required for the prediction of boundary tones.

The next-lower level is **syntactic structure**. Syntactic trees in SMARTKOM are binary branching, and they may include traces resulting from movement of syntactic constituents. They are generated from smaller tree segments within the tree-adjoining grammar framework (Becker, 1998). Semantic and pragmatic information is integrated into the syntactic structure as follows. For each node of the syntactic tree, its argument status (subject, direct or indirect object, prepositional object, sentential object, or adjunct) and its information content (new vs. given) can be specified. Deixis is also specified on the syntactic level. Deictic elements occur when Smartakus executes pointing gestures referring to objects on the screen.

The lowest level of annotation is the **lexical level**. On this level, material that originates from database queries is inserted. The domain and language of this material are annotated if available. An example of the syntactic and lexical levels of a concept structure is given in Fig. 1.

2.2 Prediction of Prosodic Phrases

The first step in prosody generation is the prediction of prosodic phrase boundaries. There are two levels of phrases: Intonation phrases are terminated by major breaks ("big breaks", BB) and can be divided into several intermediate phrases, which in turn are separated by minor breaks (B).

Syntactic structure has been shown to be useful in the prediction of prosodic phrasing (Schweitzer and Haase, 2000). Particularly the insertion of prosodic phrase breaks between topicalized constituents in the *Vorfeld* (i.e., constituents preceding the finite verb in verb-second sentences) and the rest of the sentence has proved to be a common phenomenon in natural speech, if the material in the Vorfeld is long enough (Schweitzer and Haase, 2000). The Vorfeld corresponds to a syntactic constituent, a maximal projection, that is in the specifier position of another maximal projection (depending on the syntactic theory a verbal, inflectional, or complementizer projection). Prosodic breaks can also occur between constituents in the *Mittelfeld*. Another observation is that breaks are less likely between heads and complements than between heads and adjunct constituents. In any case, the longer the constituents are, the more likely the breaks are inserted. Usually, the inserted breaks are minor breaks; occasionally even major breaks occur.

These observations motivate the two rules in (1) and (2), which insert optional minor breaks. The [±B] feature indicates that a break can be inserted at the end of

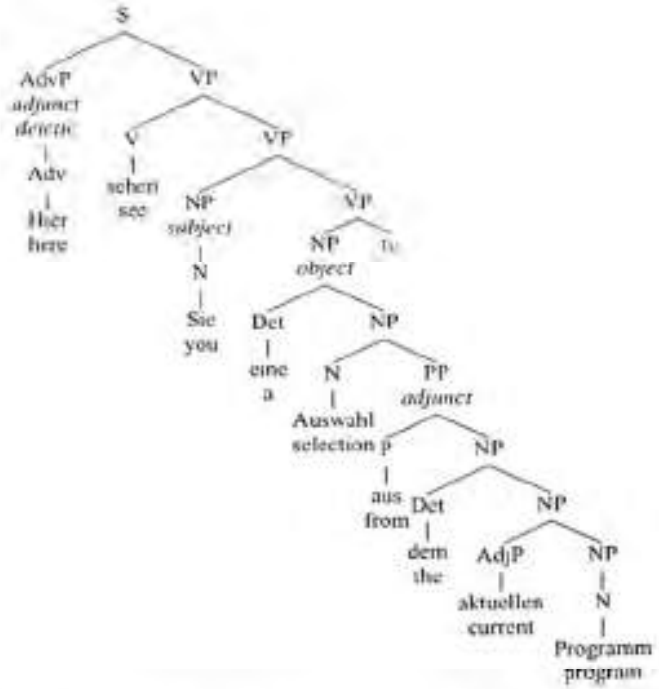


Fig. 1. Integration of additional information into the syntactic and lexical structure of the sentence *Hier Sehen Sie Eine Auswahl Aus Dem Aktuellen Programm*. In this example, deixis (*deictic*) and argument status (*adjunct*, *subject*, *object*) are added. These values are indicated in *italics*

the respective constituent. The rules each have two variants, (a) and (b), which are mirror images of each other.

(1) a. XP
YP[±B] XP

b. XP
XP[±B] YP

(2) a. XP
X[±B] YP
adjunct

b. XP
YP[±B] X
adjunct

The first rule states that maximal categories (the YPs in (1)) that are daughters of other maximal categories (the dominating XPs in (1)) can be separated from their sister node by a minor break. S constituents are also treated as maximal projections.

Since we do not distinguish X-bars from XPs, rule (1) applies to any maximal projection that is not the sister of a head. Examples of the application of (1) are the insertion of boundaries between topicalized constituents and the VP as well as between adjacent constituents within the VP. The second rule allows breaks to be inserted between the head of a phrase (the X in (2)) and its sister node, but only if the sister node is an adjunct. Thus, phrase boundaries between a head and its argument are excluded.

Deictic elements often trigger additional minor phrase breaks. A pilot study on material from 26 speakers showed that deictic expressions, i.e., expressions that were accompanied by pointing gestures, were usually marked by a phrase break or an emphatic pitch accent or both. This effect is modeled by inserting mandatory minor breaks preceding and following deictic expressions.

The result of the phrase break insertion for the sentence in Fig. 1 is shown in (3). Mandatory phrase breaks are (trivially) at the end of the utterance, and after the deictic AdvP *hier*, indicated by the [+BB] and [+B] features, respectively. Optional phrase breaks are inserted after the NP *Sie* according to rule (1a), and after the noun *Auswahl* according to rule (2a). These optional breaks are marked by the feature [±B] in (3).

(3) Hier [+B] sehen Sie [±B] eine Auswahl [±B]
aus dem aktuellen Programm [+BB]

In a second step, a **harmonization algorithm** selects candidates from the set of possible combinations of prosodic phrases. Candidates whose mean phrase length lies in a given optimal range and which show an even phrase length distribution are favored over other candidates. Thus, the observation that the insertion of breaks depends on the length of the resulting phrases is accounted for, and sequences of phrases that are unbalanced in terms of number of syllables per phrase are avoided if possible. The optimal range for the mean phrase length was found to be more than 4 to less than 11 syllables.

For the example in Fig. 1, the optimal candidate is shown in (4a). The other candidates are given in (4b) through (4d). Syllable number per phrase, mean phrase length, and variance are indicated in italics. Rule 4b is discarded because its mean phrase length is not in the optimal range. From the remaining three candidates, (4a) is chosen because it has the smallest variance.

(4) a. Hier [+B] sehen Sie eine Auswahl [+B] aus dem aktuellen Programm [+BB]
syllables: 1, 7, 8; mean: 5.33; variance: 9.55

b. Hier [+B] sehen Sie [+B] eine Auswahl [+B] aus dem aktuellen Programm [+BB]
syllables: 1, 3, 4, 8; mean: 4; variance: 6.5

c. Hier [+B] sehen Sie [+B] eine Auswahl aus dem aktuellen Programm [+BB]
syllables: 1, 3, 12; mean 5.33; variance: 22.89

d. Hier [+B] sehen Sie eine Auswahl aus dem aktuellen Programm [+BB]
syllables: 1, 15; mean 8; variance: 49

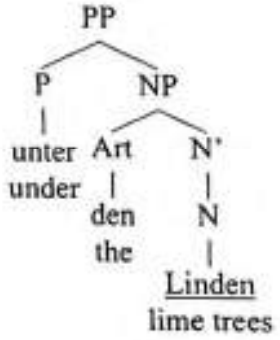
Finally, boundary tones are assigned to each predicted major phrase boundary. For sentence-internal phrase boundaries, a rising boundary tone is assigned to indicate continuation. In all other cases, the boundary tone depends on the sentence mode.

2.3 Accent prediction

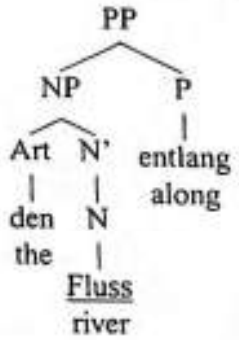
The default location of sentence stress is determined by the syntactic structure according to Cinque (1993). We have adapted this procedure to predict the default accent location for each prosodic phrase. Additionally, semantic factors can cause deaccentuation. Pitch accent types depend on the information content of the accented word, on its position in the phrase, and on sentence mode.

According to Cinque (1993), the default accent is on the syntactically most deeply embedded element, as illustrated by the prepositional phrases in (5) (from Cinque (1993)). The underlined words are the most deeply embedded elements, and they are accented by default.

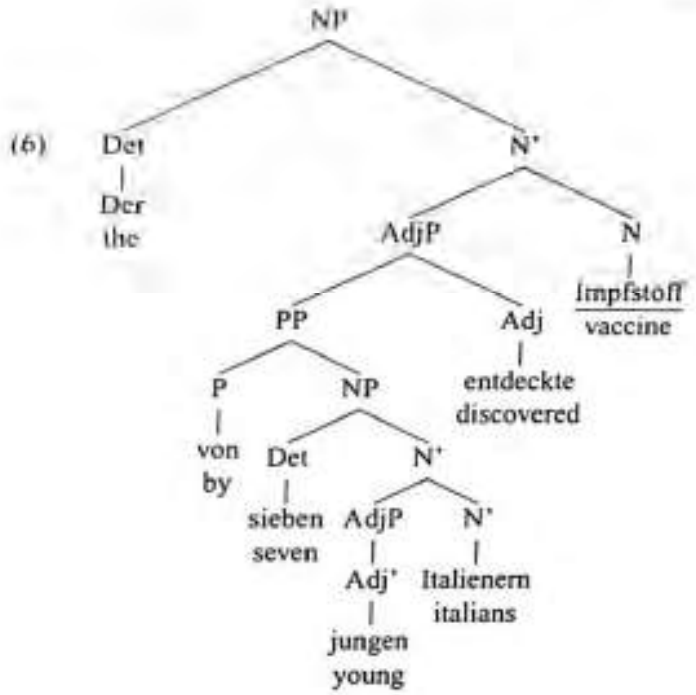
(5) a.



b.



However, depth of embedding on the nonrecursive side is irrelevant, as shown in (6) (from Cinque (1993)): In neutral accentuation, the pitch accent falls not on the overall most deeply embedded element, *Italienern*, but on *Impfstoff*. This is because NPs are right-recursive. Depth of embedding, according to Cinque (1993), is only counted on a path along the X-bar axis (e.g., connecting XP and X', X' and X') and on the recursive side of each projection XP (e.g., connecting X' to a YP embedded on the left side, if XP is a left-recursive category; or connecting X' to a YP embedded on the right side, if XP is a right-recursive category). The main path of embedding is the path that reaches the top node. The overall most prominent element is the most deeply embedded element on the main path of embedding. In constituents on



the nonrecursive side, depth of embedding determines the locally most prominent element in the constituent, but its depth of embedding is irrelevant for the location of the main stress.

This procedure had to be modified for two reasons. First, in the syntactic structures used in SMARTKOM, there are no X-bars. Thus, the main path is along an axis connecting XPs with embedded XPs, or connecting an XP with a maximal projection YP on the recursive side of the XP, if YP is a sister to the head X of XP. Second, large syntactic trees will usually be split into smaller units by the phrase prediction algorithm. In the phrases that do not contain the globally most prominent element according to the definition above, we still need to assign an accent to the locally most prominent element. The resulting procedure works as follows. In each phrase, the element with the smallest number of branches on the nonrecursive side is accented. If there are several elements with the same number of branches, the last one is accented. Depending on the information structure of an utterance, accentuation can deviate from the default accentuation: Words are deaccented if they are marked as "given" in the respective context, and narrow focus moves the accent from the default location to the focused constituent.

For each accented element, its accent category depends on its position in the phrase, its information content, and the sentence mode. We use a subset of the pitch accent inventory of the German ToBI labeling system as described in Mayer (1995), viz. L*H as a rising accent, H*L as a falling accent, and L*HL as an emphatic accent. For the diphone voice, the type of accent determines the template used for modeling the fundamental frequency contour (Möhler and Conkie, 1998). For the unit selection

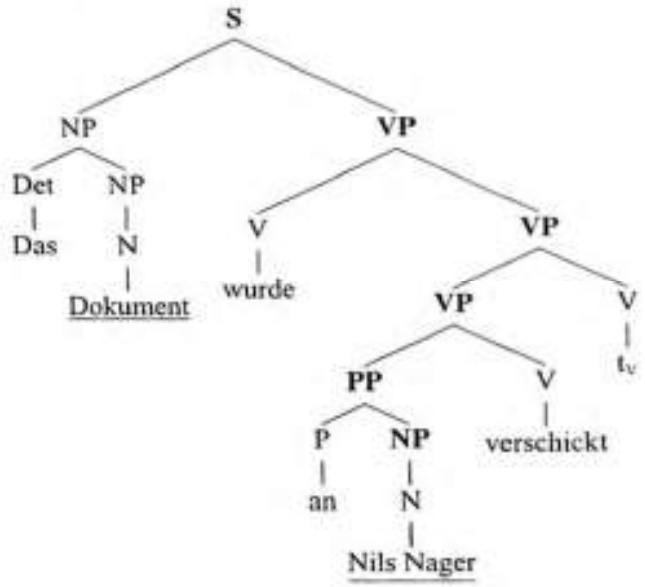


Fig. 2. Syntactic structure for the example "Das Dokument wurde an Nils Nager verschickt" (*The document was sent to Nils Nager*). The main path of embedding is marked by the nodes in *boldface*. The default accents for the two phrases are assigned to the underlined words

voice, it restricts the candidate set to candidates realized with a similar accent (see Sect. 4).

2.4 An Example

The complete prosody prediction algorithm is illustrated by the example in (7) and (8). An optional phrase break is inserted between the topicalized object *Das Dokument* and the finite verb *wurde*. The harmonization algorithm selects (8a) because (8b)'s mean phrase length exceeds the upper limit of 11 syllables per phrase and is therefore not considered in the selection step. Otherwise, it would have been preferred over (8a) because its variance is smaller.

(7) Das Dokument wurde an Nils Nager verschickt.
The document was to Nils Nager sent
The document was sent to Nils Nager.

(8) a. Das Dokument [+B] wurde an Nils Nager verschickt [+BB]
syllables: 4, 8; mean: 6.00; variance: 4.00
b. Das Dokument wurde an Nils Nager verschickt [+BB]
syllables: 12; mean: 12.00; variance: 0.00

The syntactic structure of (8) is shown in Fig. 2. The main path of embedding is indicated by the nodes in boldface. For the first phrase, the default accent is assigned to the noun *Dokument* because the path from the top of the tree to the noun contains only one branch (connecting S to the NP on its left) that is neither on the recursive

side nor on the X-bar axis. The branches connecting NP to NP and NP to N are on the recursive side because NPs are right-branching. The path to the determiner *Das* contains two branches that are on the nonrecursive side: the branches connecting S to NP and NP to Det, respectively. In the second phrase, the name *Nils Nager* is on a path exclusively along the X-bar axis or along branches on the recursive side. It is therefore accented.

Since the sentence is a declarative sentence, it is terminated by a falling boundary tone. The accented element in the second phrase is assigned a falling accent for the same reason. The accent in the first phrase is predicted to be rising because the sentence continues across the intermediate phrase boundary between the two phrases. Thus, the prosodic structure for (8) is as shown in (9):

(9) Das Dokument wurde an Nils Nager verschickt.
L*H - H*L L%

3 Speaker Selection

Several constraints have to be met in the speaker selection process. On the one hand, users' expectations include not only intelligibility but also more subjective properties such as agreeableness, pleasantness, and naturalness. Adequacy of the voice for the target application may be even more important than subjective pleasantness. For instance, Smartakus is a small blue-colored cartoon-like character reminiscent of the letter "i". This visual appearance did not seem to go well with one particularly deep candidate voice, which was rated high by listeners only when presented independent of Smartakus. There are additional, more technical and practical, requirements, such as the experience of the speaker, which can decisively reduce the time needed for the recordings, but also foreign language skills, which are required for some nonnative diphones, as well as the speaker's availability over a longer period of time.

The subjectively perceived properties of a diphone voice are currently not predictable from the speaker's natural voice. The prediction is less difficult for unit selection voices because they preserve the characteristics of the original voice much better by reducing the number of concatenation points and the amount of signal processing. However, the number of concatenation points may be similar to that in diphone synthesis, in which case the subjective voice quality is almost as hard to predict as in the diphone case. To ensure that the selected speaker's voice is suitable for diphone synthesis in the sense that the resulting diphone voice is still judged to be agreeable, we built a test diphone voice for each speaker. To this end, a small diphone set was recorded that covered the diphones required for synthesizing three short sentences. The speaker with the best test voice was selected.

Since recording and building a diphone database is very time-consuming, even for the rather small set of diphones needed for the test voices, we split the speaker selection process into two phases. In the first phase, we asked speakers to record some SMARTKOM-specific material. This material included a short dialogue typical

of a SMARTKOM domain, a list of (nonsense) diphone carrier words, and three short excerpts from movie reviews in German, English, and French. Some speakers sent in demo tapes, and some were recorded in an anechoic recording booth at our lab. Altogether, we collected demo material from 40 speakers, 29 female and 11 male. For each voice, some representative sentences were selected and rated for their subjective qualities in an informal evaluation procedure. Most participants in this rating procedure were colleagues from our institute.

In the second phase, the ten best speakers from the first phase, four male and six female, were invited to our lab to record the diphone set required for our three test sentences. The diphones were manually labeled and afterwards processed by the MBROLA[3] group at Mons. We carried out a formal evaluation with 57 participants: 20 participants were experienced, and 37 were "naive" with respect to speech technology. The three target sentences were synthesized for each speaker using different signal processing methods (MBROLA (Dutoit et al., 1996), PSOLA (Moulines and Charpentier, 1990) and Waveform Interpolation, (Morais et al., 2000)) and different prosody variants (the speaker's original prosody vs. prosody as predicted by our TTS system, with the pitch range adapted to the respective speaker's pitch range in the latter case). Some of the stimuli were presented as video clips showing Smartakus speaking, but without correct lip synchronization. Participants were asked to rate the stimuli for naturalness on a five-point scale from −2 to +2, where −2 corresponded to "not natural" and +2 corresponded to "very natural." Mean scores were calculated for every stimulus.

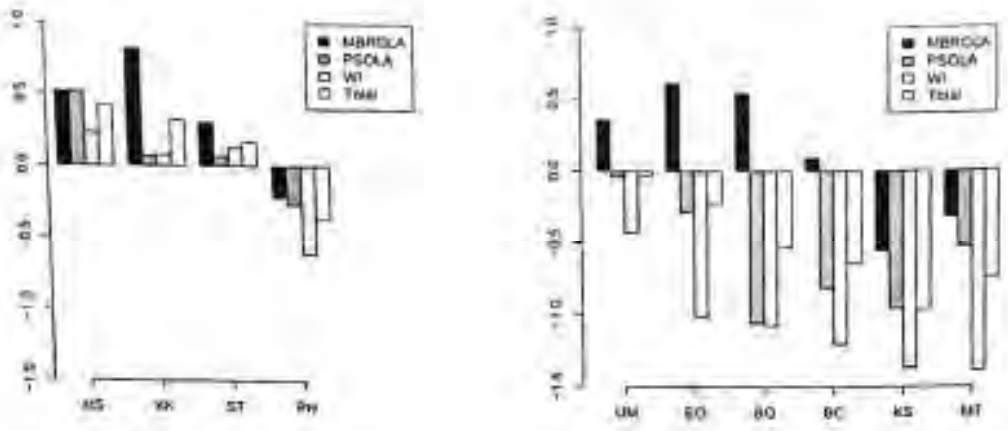


Fig. 3. Mean scores of audio stimuli for the male voices (*left panel*) and the female voices (*right panel*), broken down by signal processing method. Speakers are indicated on the x-axis, mean scores on the y-axis. Averaged over the different methods (*white bars*), the MS voice was rated the most natural, but when looking at MBROLA voices only (*black bars*), KK's voice was clearly better

[3] http://tcts.fpms.ac.be/synthesis/

The most important outcome of the evaluation procedure was that the subjective ranking of speakers was different for the two steps. For instance, the left panel in Fig. 3 shows that for the best four male speakers from the first step, the MBROLA diphone voice of KK, who was originally ranked third, was judged to be the most natural diphone voice, and the second most natural diphone voice was from MS, who was originally ranked fourth. Other signal processing methods yielded different rankings; in these cases, the MS diphone voice was judged to be the most natural. Similar effects are evident in the ranking of the female diphone voices in the right panel of Fig. 3. This confirmed our expectation that the subjective quality of the diphone voice does not correlate directly with the subjective quality of the original voice.

It is evident in Fig. 3 that MBROLA turned out to be the best signal processing method in all cases. Male voices were generally rated better than female voices, especially for signal processing methods other than MBROLA.

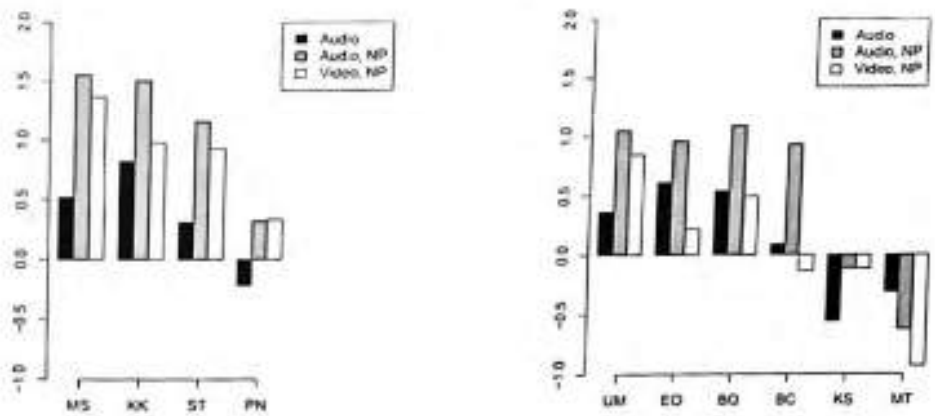


Fig. 4. Mean scores for MBROLA voices presented in audio mode, averaged over natural and rule-based prosody (*black bars*), with natural prosody (NP) only (*gray bars*), and for MBROLA voices presented in video mode with natural prosody only (*white bars*). All except one voice (MT) are rated better when the speaker's natural prosody is used in the audio-only condition. In the video condition, the ranking is different. MS is rated best in video mode. Generally, the male voices are preferred in this mode

In spite of Smartakus' relatively androgynous features, there was an even stronger preference for the male voices in the video clips. This is illustrated in Fig. 4. Only the results for the MBROLA voices are presented here, which were again rated better than the other voices. In the video condition, only the speakers' original prosody was used, which was transplanted onto the diphone voices. To assess the influence of the natural prosody in the ratings of the video stimuli, we included the ratings for audio stimuli with natural prosody in the diagram. The preference for MS in the video condition was not due to the natural prosody: While MS and KK were rated

similarly good for those stimuli, MS was clearly preferred in the video condition. Thus, as alluded to above, the speaker rated best for the audio-only stimuli was rated much lower for the audio–video stimuli, presumably because of his low pitch, which did not seem to go well with the cartoonlike features of Smartakus.

Based on these results, MS was selected as the speaker for SMARTKOM. We recorded both a diphone and a unit selection database of this speaker. The diphone voice was used in the project during the development of the unit selection voice, and it was also used as a baseline synthesis voice in the evaluation of the unit selection voice.

4 Restricted Domain Unit Selection Synthesis

The SMARTKOM domains are restricted but not limited: Utterances are generated from a number of lexicalized partial syntactic trees (Becker, 1998, 2006), but open slots are filled with names, proper nouns, movie titles, etc., from dynamically changing external and internal databases. The vocabulary is therefore unlimited, although it is biased toward domain-specific material. The predominance of domain-specific material calls for a unit selection approach with a domain-specific speech database to ensure optimal speech synthesis quality for frequent phrases. However, since the vocabulary is theoretically unlimited, domain-independent material must be taken into account as well. This is especially important because the vocabulary shows typical "large number of rare events" (LNRE) characteristics (Möbius, 2003). Although each infrequent word on its own is very unlikely to occur, the probability of having an arbitrary infrequent word in an utterance is very high.

Domain-specific and domain-independent materials pose different requirements for the unit selection strategy. Domain-specific phrases may often be found in their entirety in the database. In this case, it may be unnecessary to even consider candidates made up of smaller noncoherent units. Domain-independent material, on the other hand, will usually have to be concatenated from much smaller units, such as single segments, demanding a carefully designed database with optimal coverage and a selection algorithm that can handle larger amounts of possible candidates. Therefore, a hybrid approach was implemented combining two existing strategies (Schweitzer et al., 2003). It is described in the following section. Details on the construction of the unit selection corpus are presented in Sect. 4.2.

4.1 Unit Selection Strategy

Current unit selection approaches mostly use segments (Hunt and Black, 1996; Black and Taylor, 1997; Breen and Jackson, 1998) or subsegmental units such as half-phones (Beutnagel et al., 1999; Conkie, 1999) or demiphones (Balestri et al., 1999) as the basic unit. For each unit in the target utterance, several candidates are selected from the speech database according to criteria such as segment identity and segmental and linguistic context. For each candidate, its *target cost* expresses how well it matches the specification of the target unit. For each pair of candidates, their

concatenation cost measures the acoustic distortion that their concatenation would cause. Then the sequence of candidates is chosen that simultaneously minimizes target and concatenation costs. Since there is no distortion for originally adjacent units, longer stretches of successive units are favored over single nonadjacent units, reducing the number of concatenation points and rendering a more natural voice quality. We will call this a bottom-up approach because, starting from the segmental level, the selection of complete syllables, words, or phrases arises indirectly as a consequence of the lower concatenation costs for adjacent segments.

Such an approach faces two challenges. First, target costs and concatenation costs must be carefully balanced. Second, for frequent units the candidate sets can be very large, and the number of possible sequences of candidates grows dramatically with the number of candidates. For performance reasons, the candidate sets must be reduced, at the risk of excluding originally adjacent candidates.

One way to achieve the reduction of unit candidate sets is to cluster the units acoustically in an offline procedure and to restrict the candidate set to the units of the appropriate cluster (Black and Taylor, 1997). We will refer to this method as the acoustic clustering (AC) approach. The idea is to cluster all units in the database according to their linguistic properties in such a way that the acoustic similarity of units within the same cluster is maximized. In other words, the linguistic properties that divide the units into acoustically similar clusters are those properties that apparently have the strongest influence on the acoustic realization of the units in the cluster. During synthesis, the linguistic context determines the pertinent cluster. All other units are ignored, which reduces the number of candidates.

Some approaches (Taylor and Black, 1999; Stöber et al., 2000) use a different strategy. Candidates are searched top-down on different levels of the linguistic representation of the target utterance. If no candidates are found on one level, the search continues on the next-lower level. If appropriate candidates are found, lower levels are ignored for the part of the utterance that is covered by the candidates. For the phonological structure matching (PSM) algorithm (Taylor and Black, 1999), candidates can correspond to various nodes of the metrical tree of an utterance, ranging from phrase level to segment level, while Stöber et al. (2000) use only the word and segment levels. Both approaches are designed for limited domains and benefit from the fact that most longer units are represented in the database. The advantage of such a top-down approach is that it favors the selection of these longer units in a straightforward way. If candidates are found on levels higher than the segment level, this strategy can be faster than the bottom-up approaches because there are longer and therefore fewer unit candidates. Still, particularly on the segment level, candidate sets may be very large.

The LNRE characteristics of the SMARTKOM vocabulary with a limited number of very frequent domain-specific words and a large number of very infrequent words originating from dynamic databases suggested a hybrid strategy that integrates the two approaches described above. The PSM strategy ensures high-quality synthesis for frequent material by directly selecting entire words or phrases from the database. If no matching candidates are found above the segment level, which will typically

be the case for domain-independent material, the AC approach serves to reduce the amount of candidate units.

Our implementation of the PSM algorithm differs from the original implementation (Taylor and Black, 1999) in some aspects. First, the original algorithm requires candidates to match the target specification with respect to tree structure and segment identities, but they may differ in stress pattern or intonation, phonetic or phrasal context, at the expense of higher target costs. This reflects the view that a prosodically suboptimal but coherent candidate is better than the concatenation of smaller non-coherent units from prosodically more appropriate contexts. We kept the matching condition more flexible by more generally defining two sets of features for each level of the linguistic hierarchy. *Primary features* are features in which candidates have to match the target specification (in addition to having the same structure), while they may differ in terms of *secondary features*. Mismatch of secondary features causes higher target costs, just as the mismatch of prosodic features increases the unit score in the original algorithm. The primary features typically are the unit identity and the classification of prosodic events occurring on the respective unit. Secondary features are mostly positional features expected to have a strong influence on the acoustic realization of the unit. More details can be found in Schweitzer et al. (2003).

Another, more important, difference from the original PSM algorithm is that candidate sets can optionally be reduced if their size exceeds a certain threshold. In this case, the candidate set is filtered stepwise for each secondary feature, thereby excluding candidates that do not agree on the respective feature, until the size of the candidate set is below the threshold. However, the PSM search is not performed below the syllable level because the initial candidate sets would be too large. Instead, the AC algorithm (Black and Taylor, 1997) takes over on the segment level, adding candidates for those parts of the target utterance that have not been covered yet.

As for the final selection of the optimal sequence of units, candidate units found by either search strategy are treated in the same way, i.e., they are subject to the same selection procedure. Thus, longer units are treated just as shorter units in that the optimal sequence of candidates is determined by a Viterbi algorithm, which simultaneously minimizes concatenation costs and target costs. Concatenation costs for two longer units are the concatenation costs for the two segments on either side of the concatenation point.

4.2 Text Material Design and Corpus Preparation

The requirements for the contents of the database are again different for domain-specific vs. domain-independent material. For the limited amount of domain-specific material, it is conceivable to include typical words in several different contexts (Stöber et al., 2000) or even to repeat identical contexts. In contrast, for the open-domain part a good coverage of the database in terms of diphones in different contexts is essential, as emphasized by van Santen and Buchsbaum (1997) and Möbius (2003).

We followed van Santen and Buchsbaum (1997) by applying a greedy algorithm to select from a large text corpus a set of utterances that maximizes coverage of

units. The procedure was as follows. First, the linguistic text analysis component of the IMS German Festival TTS system[4] (Schweitzer and Haase, 2000) was used to determine for each sentence in a German newspaper corpus of 170,000 sentences the corresponding phone sequences as well as their prosodic properties. We built a vector for each segment including its phonemic identity, syllabic stress, word class, and prosodic and positional properties. Thus, we obtained a sequence of vectors for each sentence. Additionally, we determined the diphone sequence for each sentence. Sentences were then selected successively by the greedy algorithm according to the number of both new vectors and new diphone types that they covered. For German diphone types that did not occur at all, we constructed sentences that would contain them, added these sentences to the corpus, and repeated the selection process. This ensured that at least a full diphone coverage was obtained, and at the same time the number of phoneme/context vector types was increased.

We added 2643 SMARTKOM-specific words and sentences to the domain- independent corpus. They included excerpts from demo dialogues, but also domain-typical slot fillers such as people's names, place names, numbers, weekdays, etc. Movie titles, many of them in English, constituted the largest group of domain-specific material, partly to make up for the omission of English phones in the systematic design of the text material.

The speech database was recorded using the same professional speaker as for the diphone voice and amounted to about 160 minutes of speech. The automatically generated transcriptions were manually corrected according to what the speaker had said together with the corresponding orthographic notation. The hand-corrected transcriptions were then used for sentencewise forced alignment of the speech signal on the segment, syllable, and word levels. Pitch accents and boundary tones were automatically predicted from the orthographic notation and were subsequently corrected manually.

The corrected version of the database contains 2488 diphone types. 277 of the 2377 originally predicted types were not realized in the database, mostly because of incorrect predictions; instead, 388 additional types occurred. Similarly, the database had been predicted to cover 2731 out of 2932 phoneme/context vector types from the complete text corpus. 687 of these were not realized in the recorded database, whereas 791 new ones occurred, which yields 2835 types. Of these new vector types, only 10 belong to the 201 vectors that had been in the complete text corpus but not in the subset selected for the recordings.

These figures show that more than 90% of the diphone types were covered as expected, and many new types involving foreign phonemes were added. As for the coverage of phoneme/context vectors, the situation is more complex. Combinatorially, 19 440 phoneme/context vector types are possible. We estimate that no more than 4600 are theoretically possible because the context properties are not independent. For instance, boundary tones only occur on phrase-final syllables. Some consonants are phonotactically not allowed in syllable onsets, others not in the rhyme, and vowels are in the rhyme per definition. Also, pitch accents are always realized on

[4] http://www.ims.uni-stuttgart.de/phonetik/synthesis/index.html

syllables with syllabic stress, and function words usually have no pitch accent. However, only approximately 60% of these 4600 types were covered, even with a careful database design. One reason for this is that some of these types are so rare that they do not occur even in large corpora (Möbius, 2003). Apart from that, coverage of phoneme/context vectors was problematic because many of the predicted vectors were incorrect. This was partly due to foreign language material in the text corpus, which could not be adequately dealt with using the monolingual German lexicon; also, unknown words, mostly compounds, abbreviations, and acronyms, had often been predicted incorrectly. We expect that the prediction of context vectors can be significantly improved if foreign material is reliably marked as such in a preprocessing step. However, the prosodic contexts are difficult to predict, and often several alternative realizations are possible. Giving the speaker additional directions concerning intended prosodic realizations, on the other hand, may add too much load in supervising the recordings and moreover might result in unnatural realizations.

5 Lip and Gesture Synchronization

It has been shown that visual segmental information can enhance segmental speech perception (Massaro, 1998). Vice versa, inconsistencies between visual and acoustic information can significantly decrease intelligibility to the extreme that the segmental identity is compromised: MacDonald and McGurk (1978) demonstrated that acoustic [ba] is perceived as /da/ when presented with the visual information of [ga]. Thus, correct lip synchronization is an important issue in multimodal speech synthesis.

5.1 Lip Synchronization

In contrast to lip synchronization for more human-looking avatars, which may require modeling of various parameters such as the position of the jaw, the upper and lower lip, teeth, tongue tip, and tongue root, only two parameters are necessary for Smartakus because of his cartoonlike features: jaw opening and lip rounding. His teeth and tongue are never visible.

We used a simple mapping procedure to map phonemes to so-called *visemes*. Visemes are visually contrastive speech elements (Fisher, 1968). In our view, visemes are sets of feature-value pairs, where the features correspond to the different articulators and the values indicate their target positions. Movement results from interpolating between the target positions specified by the visemes. Each phoneme is represented by one or more visemes. Visemes can be underspecified regarding particular features. In this case, the value of the feature consists of a range of possible values. Underspecified visemes inherit the missing values from the context, which allows us to model coarticulation.

In SMARTKOM, only two features are specified for visemes: lip (or jaw) opening and lip rounding. Four degrees of lip opening (closed, almost closed, mid, open) and two degrees of lip rounding (unrounded, rounded) are differentiated. Visemes corresponding to vowels are fully specified for both opening and rounding. Figure 5

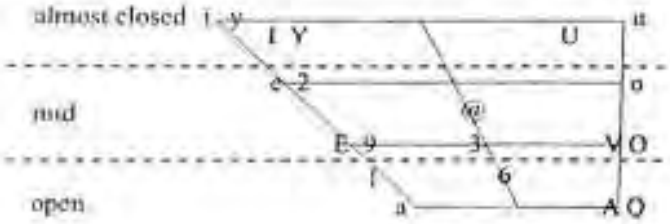


Fig. 5. IPA vowel space diagram of the vowels used in SMARTKOM synthesis, in SAMPA notation. The lip opening degree for each vowel is indicated on the *left*. The *dashed lines* indicate the boundaries between different degrees of opening

shows the vowels used in SMARTKOM arranged in the International Phonetic Association (IPA) vowel diagram, in SAMPA[5] notation. The position of each vowel in the IPA diagram reflects the tongue position in articulation. Thus, the vertical position of the vowels indicates tongue height, and the horizontal position indicates the front–back position of the tongue. To map vowel positions in the diagram to lip opening degrees, we stipulate that tongue height and lip opening correlate, but that the horizontal position is irrelevant for lip opening. The resulting mapping from the position in the diagram to the opening degree of the corresponding viseme is indicated on the left of Fig. 5. Schwa vowels (/6/, /@/) are an exception: They are usually realized in a reduced way, in which case the correlation between lip opening and tongue height seems to be less strong. Both schwas are therefore realized with almost closed lips. Rounding trivially follows the phonological specification of the respective vowels. Diphthongs are represented by a sequence of two visemes corresponding to the visemes representing the underlying vowels.

Visemes for **consonants**, on the other hand, may be underspecified. This is motivated by the hypothesis that consonants whose place of articulation is to the back can be articulated with an almost closed jaw or with an open jaw. Consonants with an anterior place of articulation, however, can only be articulated with a relatively closed jaw. Also, most consonants can be articulated with rounded or with unrounded lips. Thus, visemes corresponding to consonants are unspecified with regard to lip rounding, and the farther back their place of articulation is, the higher is the degree of underspecification for jaw opening. In Table. 1, the mapping from place of articulation to jaw opening degree is again demonstrated by an IPA chart containing the consonants used in SMARTKOM.

After mapping phonemes to visemes, the resulting sequences of partly underspecified visemes and the corresponding time intervals relative to the beginning of the speech signal are passed to the presentation manager (Poller and Tschernomas, 2006), which resolves underspecification and concatenates video sequences corresponding to the visemes.

[5] SAMPA is a common standard used for convenience instead of the IPA notation.

Table 1. IPA chart for the consonants used in SMARTKOM, in SAMPA notation (*upper part* of the table) and mapping to ranges of jaw opening degrees (*lower part*). Places of articulation are arranged in the chart from front (labial) to back (glottal). Possible jaw opening degrees are 0 (closed), 1 (almost closed), 2 (mid), and 3 (open). The range of values depends on the place of articulation of the corresponding phoneme. Velar, uvular, and glottal consonants exhibit the highest degree of underspecification for jaw opening: The range of jaw opening degrees is from 1 to 3

	Place of articulation								
	Labial	Labiodental	Dental	Alveolar	Postalveolar	Palatal	Velar	Uvular	Glottal
Plosives	p b	–	t d			–	k g	–	?
Nasals	m	–	n			–	N	–	–
Fricatives	–	f v	T D	s z	S Z	C	x	R	h
Approximants	–	–	l r			j	w	–	–
	Jaw opening								
Minimal	0	1	1				1		
Maximal	0	1	2				3		

5.2 Gesture–Speech Alignment

Smartakus may execute gestures while he is speaking. In this case, temporal alignment of speech and gesture is required. Pointing gestures also influence prosody, as mentioned in Sect. 2.

Building on the sketch model (de Ruiter, 2000), speech is synthesized independently of the temporal structure of the accompanying gesture. Instead, the gesture is executed in temporal alignment with the speech signal. According to de Ruiter (2000), gestures can be divided in three phases, viz. the preparation phase, the stroke phase, and the retraction phase. The stroke phase is the core phase, which accompanies corresponding speech material. Preparation and retraction phases of gestures can be adjusted to align the stroke phase with the relevant speech material. In SMARTKOM, most of the gestures occurring during speech are pointing gestures, which accompany deictic elements in the linguistic structure. In this case, the timing information for the deictic material is passed to the presentation manager to enable alignment of the stroke phase with the corresponding deictic element.

6 Evaluation of the Speech Synthesis Module

Due to the complexity of multimodal systems, it is difficult to evaluate single components because they are not designed to perform in a stand-alone mode, isolated from other system components with which they interact. Also, the performance of the system as a whole, not the performance of its modules, is decisive when it comes to user acceptance or usability. Consequently, the SMARTKOM system has been evaluated extensively as a whole (Schiel, 2006).

However, in addition to an end-to-end evaluation of the complete system, the evaluation of its speech synthesis component is necessary to give more detailed, possibly diagnostic, insights into potential synthesis specific problems. This can be difficult since the boundaries between system components are often not clear-cut

from a functional point of view. In SMARTKOM, language generation and synthesis are strongly linked. Without language generation, simulating concept input for CTS synthesis is tedious. But if concept input is generated automatically for synthesis evaluation purposes, the language generation component is implicitly evaluated together with the synthesis module. A second problem is that the appropriateness of the synthesis voice for Smartakus cannot be evaluated without the animation component.

To detect possible synthesis-specific problems, we carried out evaluations of the synthesis module, detached as far as possible from the SMARTKOM system, at two times. The first evaluation took place early in the project and served to verify that the diphone synthesis voice produced satisfactory intelligibility; the second evaluation was carried out in the last project phase to assess the quality of the new unit selection voice, particularly in comparison with the diphone voice. Table 2 shows an overview of the tasks performed in the evaluation procedures.

Table 2. Overview of the tasks performed by participants in the evaluation procedures. The general type of task is indicated in the *left column*. The table lists text material, viz. normal text (open domain), semantically unpredictable sentences (SUS), or SMARTKOM specific material (SK), and the voices used to generate the stimuli, viz. diphone voice or unit selection voice (US)

	First evaluation		Second evaluation			
			Pilot study		Full experiment	
	Material	Voice	Material	Voice	Material	Voice
Dictation	SUS	Natural	–	–	SUS	Diphones
		Diphones				US
	SK	Diphones				
Listening comprehension	–	–	–	–	Open domain	Diphones
						US
Subjective impression	SK	Diphones	SK	Diphones	SK	US
				US		

6.1 First Evaluation

The first evaluation involved a total of 58 participants, which can be classified in two groups. The first group comprised 39 students from the University of Ulm. These subjects are referred to as "naive" because they reported having no prior experience with speech synthesis or language processing. The second group consisted of employees of DaimlerChrysler at Ulm, who were experienced with regard to speech technology. All participants completed three dictation tasks: one with SMARTKOM-specific utterances rendered by the diphone voice, one with semantically unpredictable sentences (SUS (Benoit et al., 1996) recorded from a speaker, and one using SUS stimuli synthesized by the diphone voice.

The SMARTKOM-specific dictation task was intended to verify that the intelligibility of the diphone voice was satisfactory for the use in SMARTKOM. The participants transcribed nine system turns in a continuous dialogue between the system

and a user. Ninety three percent of these system turns were transcribed without any errors; 4% involved obvious typing errors; and in 2% of the transcriptions there were errors that can probably be attributed to memory problems rather than to intelligibility. These figures show that the diphone voice offers excellent intelligibility for normal speech material.

The SUS dictation tasks are perceptually more demanding because the linguistic context does not provide any cues in cases of locally insufficient intelligibility. The tasks thus aimed at testing the intelligibility of the diphone voice under more challenging conditions. The sentences were generated automatically using five different templates, which are listed in Table 3. The material to fill the lexical slots in the templates came from lists of words selected from CELEX (Baayen et al., 1995) according to their morphological and syntactic properties. The lists were randomized before generating the SUS stimuli. All lexical items were used at least once, but in varying combinations.

Table 3. Overview of syntactic templates used for the generation of SUS stimuli. The table shows the lexical slots in the templates corresponding to the constituents in each of the templates. Although not explicitly stated here, noun phrases were also congruent in gender, and the complements of transitive verbs and prepositions were in the appropriate case

Template	Constituent	Lexical slots
S V O	subject	determiner (sg.) + noun (sg.)
	verb	transitive verb (3rd person sg.)
	object	plural noun
S V PP	subject	determiner (sg.) + noun (sg.)
	verb	intransitive verb (3rd person sg.)
	adjunct PP	preposition + determiner (acc. sg.) + noun (acc. sg.)
PP V S O	adjunct PP	preposition + determiner (dat. sg) + noun (dat. sg.)
	verb	transitive verb (3rd person sg.)
	subject	determiner (nom. sg.) + noun (nom. sg.)
	object	determiner (acc. sg.) + noun (acc. sg.)
V S O!	verb	transitive verb (imperative pl.)
	subject	"Sie"
	object	determiner (acc. sg.) + noun (acc. sg.)
V S O?	verb	transitive verb (3rd sg.)
	subject	determiner (nom. sg) + noun (nom. sg)
	object	determiner (pl.) + noun (pl.)

The SUS task using natural stimuli immediately preceded the task with the diphone stimuli. It served to estimate the upper bound of scores in such a task. The subjects transcribed 15 stimuli in each of the 2 tasks. For the natural stimuli, the sentence error rate was 4.9%. Of these, 0.6% were obvious typing errors. The error rate for the synthesized stimuli was 33.9%. Again, 0.6% were typing errors. The error analysis for the diphone stimuli showed three relatively frequent error types. One concerned the confusion of short and long vowels. This can probably be attributed to the duration model used for determining segmental durations, which had been trained

on a speech corpus from a different speaker. We replaced this model with a speaker-specific model trained on the unit selection voice data later in the project. Another problem was that sometimes the subjects did not correctly recognize word boundaries. We expect that in these cases listeners should also benefit from the improved duration model. The other two types of errors concerned voiced plosives preceding vowels in word onsets, and voiced and voiceless plosives preceding /R/ in the same position. We claim that the latter is a typical problem in diphone synthesis: The two /R/-diphones concatenated in these cases are two different positional variants of /R/, viz. a postconsonantal variant, and an intervocalic variant.

After performing the dictation tasks, participants were asked for their subjective impression of the diphone voice. They rated the voice on a five-point scale ranging from −2 to +2 for each of the two questions *"How did you like the voice?"* (−2 and +2 corresponding to "not at all" and "very much," respectively), and *"Did you find the voice easy or hard to understand?"* (−2 and +2 corresponding to "hard" and "easy", respectively). Subjects also answered "yes" or "no" to the question *"Would you accept the voice in an information system?"*. The results strongly indicate that non-naive subjects generally rated the voice better than naive subjects. The mean scores for the first two questions broken down by experience with speech technology were +0.53 and +1.37 for non-naive participants, and −0.21 and +0.67 for naive participants, respectively. Of the non-naive subjects, 95% said they would accept the voice in an information system, whereas only 72% of the naive subjects expressed the same opinion. In summary, the first evaluation confirmed that the diphone voice yielded satisfactory results.

6.2 Second Evaluation

The second evaluation focused on the unit selection voice. Here the diphone voice served as a baseline for the dictation and listening comprehension tasks. The actual evaluation was preceded by a pilot study on the acceptability of the unit selection voice versus the diphone voice specifically for typical SMARTKOM utterances. The subjects in this pilot study were students from Stuttgart and their parents. The younger student group and the older parent group each consisted of 25 participants. Subjects listened to 25 SMARTKOM-specific dialogue turns in randomized order, both rendered in the unit selection voice and in the diphone voice. Afterwards, they were asked to answer the questions *"How do you judge the intelligibility of the synthesis voice?"* and *How do you judge the suitability of this voice for an information system?"* on a five-point scale ranging from −2 ("very bad") to +2 ("very good"). There was a similar effect observable between the younger and the older group as in the first evaluation between the non-naive and the naive group. The younger group was more tolerant to diphone synthesis regarding intelligibility: The mean scores for the diphone voice were +0.83 for the younger group and +0.51 for the older one. The unit selection voice was rated significantly better by both groups; the mean score was +1.76 in both cases. The results for the question regarding the suitability of the voices in an information system show that the unit selection voice is strongly preferred. Mean scores were clearly below zero for the diphone voice (−1.21 and −1.33

for the younger and the older group, respectively), and clearly above zero for the unit selection voice (+1.79 and +1.23 for the younger and the older group, respectively).

In the following evaluation, 77 subjects participated, none of which had taken part in the earlier evaluations. Three tasks were completed in this evaluation. Participants first transcribed SUS stimuli. The stimuli were taken from the first evaluation, but they were synthesized using both the diphone and the unit selection voices. The results are comparable to the earlier results: The sentence error rate was 27% including typing errors for the diphone voice (earlier: 33%). This shows that the diphone voice has gained in intelligibility compared to the first evaluation. For the unit selection voice, however, the error rate was 71%. This is due to the fact that the SUS stimuli contained only open domain material. The unit selection voice was designed for a restricted domain with prevailing SMARTKOM-specific material (Sect. 4). In this respect, completely open domains are a worst-case scenario, in which the synthesis quality must be expected to be inferior to that of SMARTKOM-specific material. Additionally, at the time of conducting the evaluation, the speech database was still in the process of being manually corrected. Informal results obtained at the end of the project, i.e., two months after the formal evaluation and after extensive manual correction of prosodic and segmental corpus annotations, indicate that the subjective synthesis quality especially for open domain material has improved since the completion of the evaluation.

After completing the SUS dictation task, participants were presented three video clips showing the SMARTKOM display during a user's interaction with Smartakus. The user's voice had been recorded by a speaker. The system's voice in the video clips was the unit selection voice, synchronized with Smartakus's lip movements and gestures. Subjects were asked to answer three questions by adjusting a sliding bar between two extremes. The three questions were "*How do you judge the intelligibility of the voice?*", with possible answers ranging from "not intelligible" to "good"; "*How natural did you find the voice?*", with answers between "not natural at all" and "completely natural"; and "*How did you like the voice?*", with answers between "not at all" and "very well." The results for the three answers were 71% for intelligibility, 52% for naturalness, and 63% for pleasantness. These figures show that in the SMARTKOM-specific contexts, the unit selection voice is very well accepted and judged to be satisfactorily intelligible. This confirms the results obtained in the pilot study for audio-only stimuli.

In the last task, the listening comprehension test, the subjects listened to four short paragraphs of open domain texts. After each paragraph, they were asked three questions concerning information given in the text. Two texts were rendered using the diphone voice, and two using the unit selection voice. The results were again better for the diphone voice, with 93% of the answers correct, while 83% were correct for the unit selection voice. In this context, both voices were rated lower than in the SMARTKOM-specific task. The scores for intelligibility, naturalness, and pleasantness were 53%, 34%, and 42% for the diphone voice, and 23%, 22%, and 26% for the unit selection voice, respectively. Again, we expect much better results after the manual correction of the speech database.

7 Conclusion

To summarize, the superiority of the unit selection voice is evident for the SMARTKOM domain. This was confirmed by the pilot study and the SMARTKOM-specific part of the second evaluation. The quality of the diphone voice improved between the first and the second evaluation. We attribute this effect mainly to the new duration model obtained from the unit selection data of our speaker. The ongoing manual correction of the unit selection database is evidently effective. Subjectively, the synthesis quality has improved since the completion of the second evaluation. However, this will have to be confirmed in more formal tests.

Future work will focus on the extension of our unit selection approach from the restricted SMARTKOM domain to open domains in general. The experience gained in working with the SMARTKOM unit selection voice suggests that accuracy of the database annotation is crucial for optimal synthesis quality. Also, the strategy to deal with large numbers of unit candidates as they often occur in open domain sentences without excluding potentially good candidates will need some more attention in the future.

Acknowledgments

The significant contributions of Martin Ernst (DaimlerChrysler, Ulm) and Gerhard Kremer, Wojciech Przystas, Kati Schweitzer, and Mateusz Wiacek (IMS) to the synthesis evaluations are gratefully acknowledged.

References

S.P. Abney. Chunks and Dependencies: Bringing Processing Evidence to Bear on Syntax. In: *Computational Linguistics and the Foundations of Linguistic Theory*, Stanford, CA, 1995. CSLI.

H. Baayen, R. Piepenbrock, and L. Gulikers. *The CELEX Lexical Database – Release 2 (CD-ROM)*. Centre for Lexical Information, Max Planck Institute for Psycholinguistics, Nijmegen; Linguistic Data Consortium, University of Pennsylvania, 1995.

M. Balestri, A. Pacchiotti, S. Quazza, P.L. Salza, and S. Sandri. Choose the Best to Modify the Least: A New Generation Concatenative Synthesis System. In: *Proc. EUROSPEECH-99*, vol. 5, pp. 2291–2294, Budapest, Hungary, 1999.

T. Becker. Fully Lexicalized Head-Driven Syntactic Generation. In: *Proc. 9th Int. Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada, August 1998.

T. Becker. Natural Language Generation With Fully Specified Templates, 2006. In this volume.

C. Benoît, M. Grice, and V. Hazan. The SUS Test: A Method for the Assessment of Text-To-Speech Synthesis Intelligibility Using Semantically Unpredictable Sentences. *Speech Communication*, 18:381–392, 1996.

M. Beutnagel, M. Mohri, and M. Riley. Rapid Unit Selection From a Large Speech Corpus for Concatenative Speech Synthesis. In: *Proc. EUROSPEECH-99*, vol. 2, pp. 607–610, Budapest, Hungary, 1999.

A.W. Black and P. Taylor. Automatically Clustering Similar Units for Unit Selection in Speech Synthesis. In: *Proc. European Conference on Speech Communication and Technology*, vol. 2, pp. 601–604, Rhodos, Greece, 1997.

A.P. Breen and P. Jackson. Non-Uniform Unit Selection and the Similarity Metric Within BT's Laureate TTS System. In: *Proc. 3rd Int. Workshop on Speech Synthesis*, pp. 373–376, Jenolan Caves, Australia, 1998.

D. Büring. *The Meaning of Topic and Focus – The 59th Street Bridge Accent*. Routledge, London, UK, 1997.

G. Cinque. A Null Theory of Phrase and Compound Stress. *Linguistic Inquiry*, 24 (2):239–297, 1993.

A. Conkie. Robust Unit Selection System for Speech Synthesis. In: *Collected Papers of the 137th Meeting of the Acoustical Society of America and the 2nd Convention of the European Acoustics Association: Forum Acusticum*, Berlin, Germany, 1999.

P.W. Culicover and M.S. Rochemont. Stress and Focus in English. *Language*, 59(1): 123–165, 1983.

J.P. de Ruiter. The Production of Gesture and Speech. In: D. McNeill (ed.), *Language and Gesture*, pp. 284–311, Cambridge, UK, 2000. Cambridge University Press.

T. Dutoit, V. Pagel, N. Pierret, F. Bataille, and O. van der Vrecken. The MBROLA Project: Towards a Set of High Quality Speech Synthesizers Free for Use for Non Commercial Purposes. In: *Proc. Int. Conference on Spoken Language Processing*, vol. 3, pp. 1393–1396, Philadelphia, PA, 1996.

C.G. Fisher. Confusions Among Visually Perceived Consonants. *Journal of Speech and Hearing Research*, 11:796–804, 1968.

D.J. Hirst. Detaching Intonational Phrases From Syntactic Structure. *Linguistic Inquiry*, 24:781–788, 1993.

A.J. Hunt and A.W. Black. Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database. In: *Proc. IEEE Int. Conf. on Acoustics and Speech Signal Processing*, vol. 1, pp. 373–376, Munich, Germany, 1996.

J. MacDonald and H. McGurk. Visual Influences on Speech Perception Process. *Perception and Psychophysics*, 24:253–257, 1978.

D.W. Massaro. *Perceiving Talking Faces: From Speech Perception to a Behavioral Principle*. MIT Press, Cambridge, MA, 1998.

J. Mayer. Transcribing German Intonation – The Stuttgart System. Technical report, University of Stuttgart, 1995.

J. Mayer. Prosodische Merkmale von Diskursrelationen. *Linguistische Berichte*, 177:65–86, 1999.

B. Möbius. Rare Events and Closed Domains: Two Delicate Concepts in Speech Synthesis. *Int. Journal of Speech Technology*, 6(1):57–71, 2003.

G. Möhler and A. Conkie. Parametric Modeling of Intonation Using Vector Quantization. In: *Proc. 3rd Int. Workshop on Speech Synthesis*, pp. 311–316, Jenolan Caves, Australia, 1998.

E. Morais, P. Taylor, and F. Violaro. Concatenative Text-To-Speech Synthesis Based on Prototype Waveform Interpolation (A Time-Frequency Approach). In: *Proc. Int. Conf. on Spoken Language Processing*, pp. 387–390, Beijing, China, 2000.

E. Moulines and F. Charpentier. Pitch-Synchronous Waveform Processing Techniques for Text-To-Speech Synthesis Using Diphones. *Speech Communication*, 9: 453–467, 1990.

P. Poller and V. Tschernomas. Multimodal Fission and Media Design, 2006. In this volume.

F. Schiel. Evaluation of Multimodal Dialogue Systems, 2006. In this volume.

A. Schweitzer, N. Braunschweiler, T. Klankert, B. Möbius, and B. Säuberlich. Restricted Unlimited Domain Synthesis. In: *Proc. European Conference on Speech Communication and Technology*, pp. 1321–1324, Geneva, Switzerland, 2003.

A. Schweitzer, N. Braunschweiler, and E. Morais. Prosody Generation in the SmartKom Project. In: *Proc. Speech Prosody 2002 Conference*, pp. 639–642, Aix-en-Provence, France, 2002.

A. Schweitzer and M. Haase. Zwei Ansätze zur syntaxgesteuerten Prosodiegenerierung. In: *Proc. 5th Conf. on Natural Language Processing – Konvens 2000*, pp. 197–202, Ilmenau, Germany, 2000.

E. Selkirk. *Phonology and Syntax – The Relation Between Sound and Structure*. MIT Press, Cambridge, MA, 1984.

K. Stöber, P. Wagner, J. Helbig, S. Köster, D. Stall, M. Thomae, J. Blauert, W. Hess, R. Hoffmann, and H. Mangold. Speech Synthesis Using Multilevel Selection and Concatenation of Units From Large Speech Corpora. In: W. Wahlster (ed.), *Verbmobil: Foundations of Speech-to-Speech Translation*, pp. 519–534, Berlin Heidelberg New York, 2000. Springer.

P. Taylor and A.W. Black. Speech Synthesis by Phonological Structure Matching. In: *Proc. EUROSPEECH-99*, vol. 2, pp. 623–626, Budapest, Hungary, 1999.

J.P.H. van Santen and A.L. Buchsbaum. Methods for Optimal Text Selection. In: *Proc. European Conference on Speech Communication and Technology*, vol. 2, pp. 553–556, Rhodos, Greece, 1997.

Part V

Scenarios and Applications

Building Multimodal Dialogue Applications: System Integration in SmartKom

Gerd Herzog and Alassane Ndiaye

DFKI GmbH, Saarbrücken, Germany
{herzog,ndiaye}@dfki.de

Summary. We report on the experience gained in building large-scale research prototypes of fully integrated multimodal dialogue systems in the context of the SMARTKOM project. The development of such systems requires a flexible software architecture and adequate software support to cope with the challenge of system integration. A practical result of our experimental work is an advanced integration platform that enables flexible reuse and extension of existing software modules and that is able to deal with a heterogeneous software environment. Starting from the foundations of our general framework, we give an overview of the SMARTKOM testbed, and we describe the practical organization of the development process within the project.

1 Introduction

The realization of fully fledged prototype systems for various application scenarios constituted an integral part of the research and development activities within the SMARTKOM project (Reithinger et al., 2003; Wahlster, 2003). Building such large-scale multimodal dialogue applications poses a significant technical challenge and requires a principled approach for succesful system integration.

Figure 1 provides an overview concerning the technical components of the SMARTKOM demonstrator system that covers all application scenarios, including SMARTKOM-Public, Home, and Mobile as well as the English system. The instrumented vehicle, which is not permanently available for demonstrations, can also be connected to this demonstrator setup. Alternatively, standard audio components and a separate display can be employed to simulate the in-car equipment. Furthermore, a GPS simulation can be used for SMARTKOM-Mobile to demonstrate incremental route guidance. The overall system incorporates more than 40 software components from 11 different project partners. Customized SMARTKOM installations exist at all partner sites. For example, two laptops and a handheld computer are sufficient for a compact instantiation of SMARTKOM-Mobile.

The difficulty of system integration is high and is often underestimated. The distributed nature of a large-scale research activity like SMARTKOM imposes severe

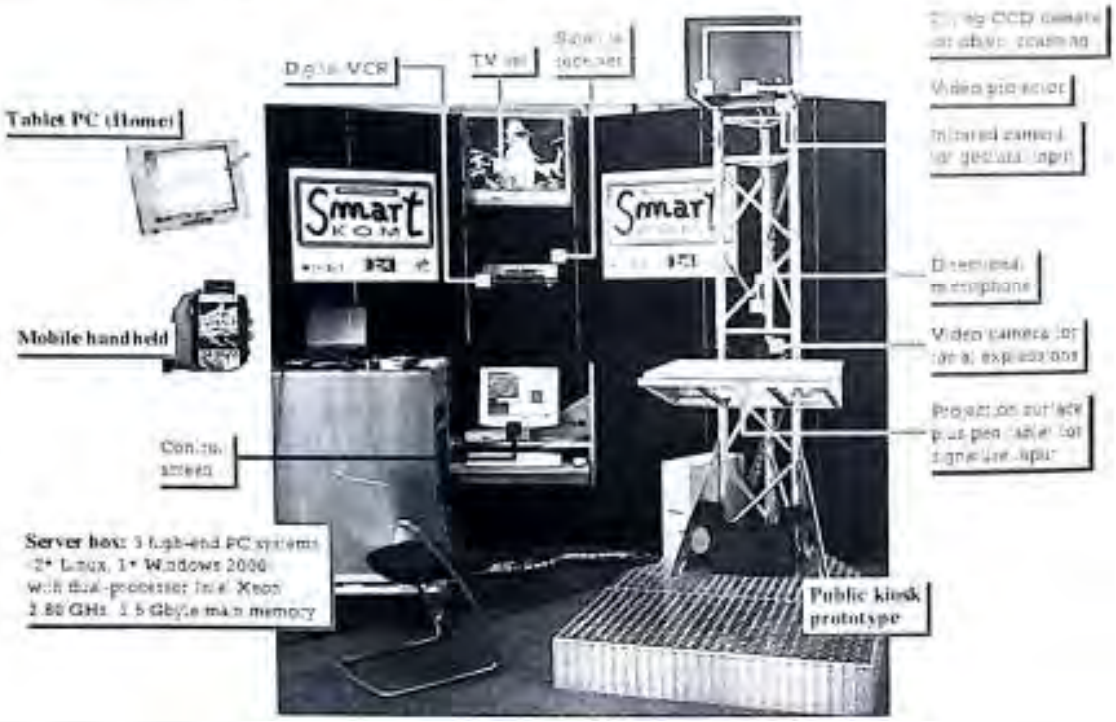


Fig. 1. SMARTKOM reference installation for demonstrations

constraints on the development of an integrated system. This kind of joint effort is characterised by a lot of variance in code development across teams. Each workgroup is accustomed to its preferred software environment and development practices. Given the need for optimal use of available project resources, it is also not feasible to start an implementation from scratch. The resulting heterogeneity of contributions from different project partners impedes the realization of a monolithic system. Thus adequate technical and organizational means are required to assemble heterogeneous components into an integrated and fully operational system.

The following section presents some basic considerations that govern our technical approach for system integration. Section 3 then provides a compact overview of the SMARTKOM testbed, our specific software integration platform that has been employed to realise the multimodal dialogue system prototypes. In addition to technical issues, organizational aspects are of particular importance for a succesful research and development project. Section 4 describes how the development and integration process has been organized within the SMARTKOM project.

2 A Framework for Software Integration

Typical aspects of large-scale system development have to be considered during the construction of advanced intelligent user interfaces. The following key factors influence the technical approach to be taken:

- strong need to reuse and extend existing software
- pressure to build initial prototypes rapidly
- significant number of distinct components from different sources
- heterogeneous software environment, including various programming languages and different operating systems
- high variability of target configurations for the integrated dialogue system applications

A distributed system constitutes the natural choice to realize an open, flexible, and scalable software architecture, able to integrate heterogeneous software modules implemented in diverse programming languages and running on different operating systems. A key point of our specific integration framework is the idea of having a component-based system, composed out of executable modules. Our approach proposes a modularization of the dialogue system into distinct and independent software modules to allow maximum decoupling. A main advantage of such a component architecture (Hopkins, 2000) is the fact that component integration and deployment are independent of the development life cycle, and there is no need to recompile or relink the entire application when updating with a new implementation of a component.

The kind of component-based software engineering proposed here requires an integration platform that provides the necessary software infrastructure to assemble the large-grained components into a multimodal dialogue system. Such an integration platform can be characterized as a software development kit and consists of the following main parts:

- a run-time platform for the execution of a distributed dialogue application, including also means for configuring and deploying the individual parts of the complete multimodal dialogue system
- application programming interfaces (API) for linking into the run-time environment and for interprocess communication
- tools and utilities to support the whole development process, including also installation and software distribution

In general, different options are available for the interconnection of individual components (Bernstein, 1989; Emmerich, 2000; Geihs, 2001). Distributed processing with remote procedure calls or remote method invocation follows the client–server paradigm and using these techniques, each component has to declare and implement a specific API to make its encapsulated functionality transparently available for other system modules. Asynchronous message-based communication, however, better supports the need for scalability, flexibility, and decoupling.

Two main schemes of message-oriented middleware can be distinguished. Basic *point-to-point* messaging employs unicast routing and realizes the notion of a direct connection between message sender and a known receiver. The more general *publish/subscribe* approach is based on multicast addressing. Instead of addressing one or several receivers directly, the sender publishes a notification on a named message queue, so that the message can be forwarded to a list of subscribers. This kind of distributed event notification makes the communication framework very flexible, as

it focuses on the data to be exchanged and it decouples data producers and data consumers. The well-known concept of a blackboard architecture (Erman et al., 1980) and distributed shared memory approaches that are based on the notion of tuple spaces (Carriero and Gelernter, 1989) follow similar ideas.

In order to ensure syntactic and semantic interoperability definite contracts among components are required. Using message-based communication, the interface of a specific component can be characterized by the admissible message contents that it is able to process and to produce. The so-called *extensible markup language*, XML, has meanwhile evolved into a standard tool for the flexible definition of application-specific data formats for information exchange. Thus we propose to operationalize interface specifications in the form of an XML language. XML-based languages define an external notation for the representation of structured data and simplify the interchange of complex data between the separate components of an integrated multimodal dialogue system.

For the technical realization of an integration platform, we postulate basic design principles that form the conceptual foundation of our integration framework:

- Asynchronous message-passing is more flexible than component-specific remote APIs and better fits parallel processing in a distributed system.
- Blackboard-like publish/subscribe messaging is a generalisation of point-to-point communication that supports decoupling of components and allows better modularization of data interfaces.
- Well-designed XML languages support the definition of declarative data models for information interchange and semantically rich interface specifications.
- The proposed framework does not rely on centralised control. Instead of that, individual components take care if coordination is needed. Required synchronization information is modeled explicitly within data interfaces.
- There should be no hidden interaction between components. All data have to be exchanged transparently via the provided communication system to ensure traceability of processing behaviour.

3 The SmartKom Testbed

Taking into account the design aspects outlined in the previous section, it becomes obvious that available off-the-shelf middleware is not enough to provide a flexible architecture framework and a powerful integration platform for large-scale dialogue systems. These considerations motivated our investment into the development of a suitable system integration platform.

SMARTKOM relies on the so-called *multiplatform testbed* (Herzog et al., 2003), a software solution that originates from our practical work in various research projects. The testbed, which is built on top of open-source software (Wu and Lin, 2001), provides a fully-fledged integration platform for dialogue systems and has been improved continuously during the last several years (Bub and Schwinn, 1999; Herzog et al., 2004; Klüter et al., 2000).

3.1 Middleware Solution

Our specific realization of an integration platform follows the architecture framework and the design principles described in the previous section. Its main characteristics are:

- component architecture
  - modularization into distinct and independent software modules to allow maximum decoupling
  - encapsulation of heterogeneous, large-grained software components into a uniform module structure
- event-based architectural style
  - very flexible publish/subscribe messaging
  - implicit invocation (event-driven) to support decoupling of components
  - highly adaptable content format specifications for each data pool, i.e., named message queue

Instead of using custom programming interfaces, the interaction between distributed components within the testbed framework is based on the exchange of structured data through messages. The SMARTKOM testbed includes a message-oriented middleware implementation that is based on the *parallel virtual machine* (PVM) software described in Geist et al. (1994). In order to provide publish/subscribe messaging with data pools on top of PVM, a further software layer called *pool communication architecture* (PCA) had to be added (Klüter et al., 2000).

Compared with commercial-grade middleware products, our low-cost solution does not consider issues like fault tolerance, load balancing, or replication. It does, however, provide a scalable and efficient platform for the kind of real-time interaction needed within a multimodal dialogue system. The PVM/PCA-based messaging system is able to transfer arbitrary data contents and provides excellent performance characteristics. Within SMARTKOM-Public, for example, it is even possible to perform a telephone conversation. Message throughput on standard PCs with Intel Pentium III 500 MHz CPUs is offhand sufficient to establish a reliable bidirectional audio connection, where uncompressed audio data are being transferred as XML messages in real time.

The so-called *module manager* provides a thin API layer for component developers with language bindings for the programming languages that are used to implement specific dialogue components. It comprises the operations required to access the communication system and to realise an elementary component protocol needed for basic coordination of all participating distributed components. The module manager implementation of the SMARTKOM testbed includes, in particular, APIs for C, C++, Java, and Prolog.

3.2 Technical Modules for System Execution

In addition to the functional components of the dialogue system, the run-time environment also includes special testbed modules in support of system operation.

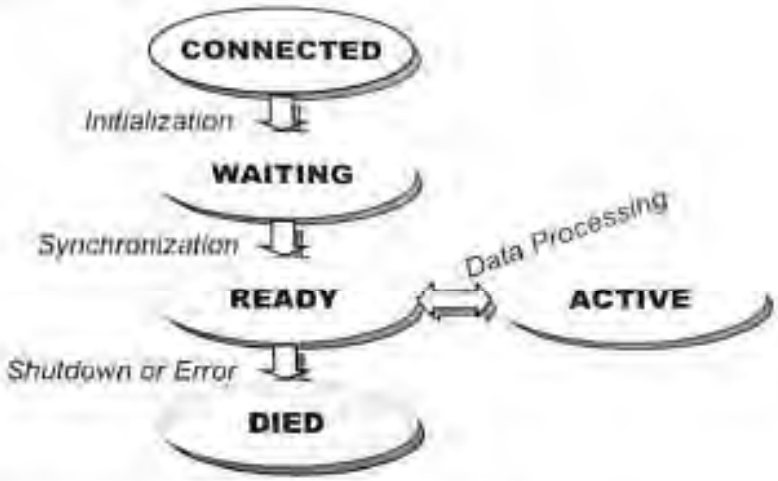


Fig. 2. Basic component states for module coordination

The *testbed manager* component is responsible for system initialization and activates all distributed components pertaining to a given dialogue system configuration. It also cooperates with the functional modules to carry out the elementary component protocol, which is needed for proper system start-up, controlled termination of processes and restart of single components, or a complete soft reset of the entire dialogue system. A state-transition network forms the basis for this protocol, which is also directly supported by the API functions of the module manager interface. As illustrated in Fig. 2, only a few basic component states need to be distinguished.

The testbed manager supports flexible run-time configuration of the application system. Module configurations can be declared using an XML-based specification. The modular concept allows start-up of individual components or subsystems for debugging purposes, e.g., to test a critical path from speech or gesture input to speech and graphic output.

The *testbed control graphical user interface (GUI)* shown in Fig. 3 constitutes a separate component that provides a graphical user interface for the administration of a running system. The adaptable testbed GUI provides a simplified view of the system architecture and offers the necessary means to monitor system activity, to interact with the testbed manager, and to manually modify configuration settings of individual components while testing the integrated system.

Both testbed modules and the functional components interact using the underlying communication system. Specific data pools are employed to exchange testbed-related control data vs. application data of the multimodal dialogue system. From the point of view of the component developer, the internal system pools are not directly visible, as they are hidden inside the module manager API implementation.

The SMARTKOM testbed provides automatic logging of module output messages and all communicated data. A further testbed module, the logging component, is employed to save a complete protocol of all exchanged messages—including system pools—for later inspection.