# Development of Advanced Dialog Systems with PATE

*Norbert Pfleger and Jan Schehl*

German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{pfleger,schehl}@dfki.de

## Abstract

Current commercial dialog systems show only limited capabilities with regard to the phenomena occurring in spontaneous, natural dialog. Many research prototypes, in contrast, are already able to deal with a great number of phenomena but lack the clarity and maintainability of commercial systems. In this paper we present a framework for developing advanced multimodal dialog systems designed to bridge this gap.

**Index Terms**: multimodal dialogue systems, commercial applications.

## 1. Introduction

Despite their commercial success, current commercial dialog systems show only limited capabilities with regard to natural dialog. Many of these systems do not support the resolution of referring or elliptical expressions and require reduced and simplified user commands. Moreover, true mixed-initiative dialog management is still only available in research prototypes and users have to deal with system-driven dialog management. Since the overall goal of employing a speech dialog system is to ease the interaction with computers, more powerful dialog systems are needed so that users do not need to adapt their natural conversational behavior to that of the dialog system.

While VoiceXML-based dialog systems typically consist of a set of well-defined VXML documents that are processed by a VXML-interpreter, state-of-the-art systems consist of a whole bunch of components of which each typically comes with its own knowledge base. This means that the price for the enhanced functionality and coverage of such systems is their general complexity and reduced maintainability. Our goal is to develop a framework for multimodal dialog systems that drastically reduces this complexity without reducing the functionality and coverage of dialog phenomena.

We present the basic architecture of an advanced dialog system that provides the flexibility and coverage of a research prototype combined with a structured and maintainable knowledge representation needed for commercial applications. Key to our approach is a production rule system called *PATE*. This system has been used in various research projects (e. g., VirtualHuman[1], Smartweb[2], COMIC[3], TALK[4], etc.) and also in an industry project to realize nearly all central processing components of a (multimodal) dialog system.

We will first introduce some concepts of ontology-based knowledge representation before we will give a brief overview of the PATE system. Then we will discuss the architecture of a prototype of an in-car multimodal dialog system developed within the TALK project where PATE is used to implement nearly the whole system.

## 2. Ontology-based Knowledge Representation

One of the crucial aspects within a dialogue system is the representation of knowledge (i. e., processing logic). The more intuitive and the more maintainable knowledge-sources are, the easier it is to enhance or adapt the system to new tasks. In our previous work we found that ontologies provide an appropriate framework for representing knowledge in multimodal dialogue systems (see [1]).

An ontology, in general, can be viewed as a controlled vocabulary that describes objects and relations between them in a hierarchical way. The concept of ontological knowledge representation originally belongs to the field of philosophy but the development of comprehensive ontologies has also been a research issue for quite a long time in the field of artificial intelligence. Today it is clear that it is nearly impossible to design an ontology that is capable of modeling the entire world while still remaining concise. This is the reason why we focus on specific sub-domains during the development of ontologies.

In our approach, we focus on a clear-cut separation between application specific knowledge and generic system specific knowledge which both represent individual sub-domains organized in meta-ontologies. The general idea is that we can re-use the dialogue-specific ontology and only need to adapt the application-specific ontology when the system needs to be adapted to a new application. The only restriction is that all sub-ontologies need to be consistent with an encompassing upper-model.

## 3. PATE - A Production Rule System based on Typed Feature Structures

The *PATE*[5] system is an extended production rule system for the development of advanced dialog systems [2]. The purpose of PATE is to provide an easy to use tool for the development of the processing logic of a dialog system. Key to PATE is that it can either be used to realize the entire dialog system within one instance or in a distributed way where several instances of PATE are employed to implement specialized sub-components of such a system. Due to space restrictions we can only present a brief overview of PATE. For a detailed documentation see [3].

[1] For VirtualHuman see http://www.virtual-human.org/
[2] For SmartWeb see http://www.smartweb-project.org/
[3] For COMIC see http://www.hcrc.ed.ac.uk/comic/
[4] For TALK see http://www.talk-project.org/

[5] PATE stands for *A Production Rule System based on Typed Feature Structures.*

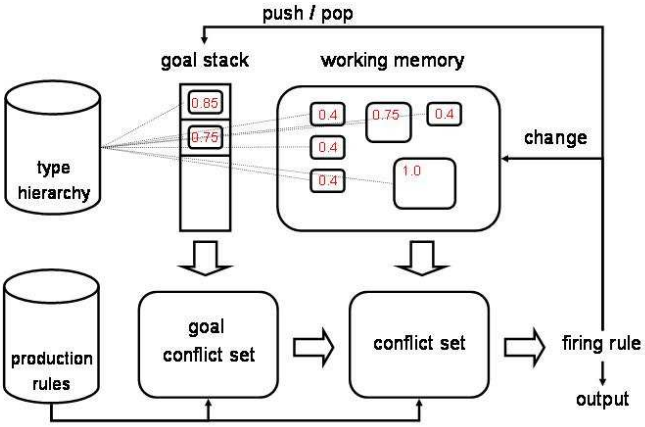


Figure 1: The basic architecture of the PATE system

### 3.1. General Characteristics of PATE

The general processing strategy is based on some ideas of the ACT-R 4.0 system of [4]. E. g., all incoming data is assigned an activation value before it is stored in the *working memory* (WM). This activation value represents the current accessibility of a *working memory element* (WME) and fades out in time. If the activation of a WME sinks below a specified threshold, this particular WME is not accessible anymore. Parallel to the working memory there is also a *goal*-stack which defines the focus of attention of the system – the WME on top of the goal stack defines the current focus of the PATE system.

As in a traditional production rule system there is a set of production rules that can be applied to the WMEs stored in the working memory (see figure 1). Each production rule consists of three components: (i) a weighting , (ii) a condition part (comprising a set of conditions that must be fulfilled) and (iii) an action part (comprising a set of actions that will be carried out if the rule fires). The weighting will be used during conflict resolution to compute an overall score for a rule. For writing and maintaining production rules, the PATE system also includes a built-in graphical user interface.

Another important aspect of the PATE system is its easy adaptation to new tasks or even entirely new dialog systems. All relevant information is stored in a global configuration file (containing general definitions of system-wide variables), a file defining the type system and a file defining the production rules. The entire PATE system is written in Java and is thus platform independent.

### 3.2. Data Representation

All internal data of the PATE system is encoded in typed feature structures (TFS; see [5]) as the internal representation language. TFS are an elegant way to represent complexly structured data so that it is still readable for humans. Another important advantage of TFS is that it can be used to represent data that was originally encoded by means of an ontology-based representation language. The way we use TFS to encode our internal data is similar to that of many ontology formats like DAML+OIL or RDF.

Key to our TFS based data representation are two operations—*unification* and *overlay* (see [6]) that determine the consistency of two typed feature structures and combine them if they are consistent. However, overlay uses its first argument as default (like the classic default unification) and returns a result in any case whereas unification returns `fail` in case of conflicting information. Besides this, overlay also generates a score reflecting how well the covering and the background fit together (see [7]). This score is used to assess the score of a condition where overlay is used. Overlay is of particular interest in case the user provides slightly conflicting information, for example, by saying *"Show me the Johnny Cash version of this song"* while selecting the Song *One* by *U2*. Unifying the representations for the two songs would result in a fail, while overlay would return an object representation of a *Johnny Cash* song with the remaining features of the *U2* song, namely the song name.

### 3.3. Rule Execution

The processing logic of PATE is represented by means of a set of production rules. The condition part (or left-hand side) of each rule consists at least of one goal-condition and a (possibly empty) set of conditions that define which objects must (or must not) be present on the goal-stack and in the working memory. Conditions are tested using unification or overlay (as specified by the condition). Consider for example figure 2: This rule can only fire if there is an object of the type *Song* on the goal-stack that has at least the feature *genre* = Rock (the objects specified in a condition can of course be much more complex).

```
<rule name="SampleRule">
 <comments> checks whether there is a rock
        song on the goal stack </comments>
 <activation> 0.8 </activation>
 <conditions>
  <condition name="goal">
   <Song>
    <has_genre>Rock</has_genre>
   </Song>
  </condition>>
  <condition name="wme1">
   ...
  </condition>
 </conditions>
 <actions>
  <action name="pop"/>
  <action type="output" name="goal"/>
 </actions>
</rule>
```

Figure 2: Example rule, checking for a rock song on the goal stack

Once initialized and started, PATE runs in a continuous loop. At the beginning of each cycle, a set of all rules that can potentially fire—given the current configuration of the working memory and the goal stack—is computed (this is called the *conflict set*). PATE then selects the best scored instantiated rule and executes its actions. Generating the conflict set is done in two steps. First the goal conflict set is initialized with all rules whose goal condition

| | |
|---|---|
| U: | Show me the Beatles albums. |
| S: | I have these four Beatles albums. [shows a list of album names] |
| U: | Which songs are on this one? [selects the Red Album] |
| S: | The Red Album contains these songs. [shows a list of the songs] |
| U: | Play the third one. |
| S: | [music plays] |

Table 1: A typical interaction with SAMMIE.

is fulfilled. In a second step, the remaining conditions are matched in a way that the conflict set contains only those rules whose condition part is fulfilled. The selection of the firing rule is done by computing an individual score for each instantiated rule and the one that scored highest. In case there are more than one, the firing rule is selected randomly but reproducibly.

# 4. SAMMIE - An In-Car Multimodal Dialog System

We now present the SAMMIE system as an example for an advanced multimodal dialog interface that provides state-of-the-art interaction capabilities of a research prototype combined with a uniform and maintainable knowledge representation needed for commercical applications. SAMMIE is a multimodal dialog system, developed by DFKI in collaboration with Saarland University within the EU-funded project TALK. It provides multimodal access to an in-car MP3 player through speech and haptic input with an Ergocommander, a button that can be turned, pushed down and pushed sideways in four directions. System output is presented by speech and a graphical display integrated into the car's dashboard. An example of the system display is shown in figure 3. The SAMMIE system allows the user complete freedom in their interaction with the system. Input can be made through any modality and is not restricted to answers to system queries. On the contrary, the user can provide new tasks as well as any information relevant to the current task at any time. Note that the user is also free in their use of multimodality: Deictic references (*Play this title* while pushing the Ergocommander button) are possible, and even cross-modal references as in *Play the third song (on the list)* are understood. Table 1 shows a typical interaction with the SAMMIE system.

## 4.1. Architecture

The SAMMIE system architecture follows the classical approach of a pipelined architecture [8]. Figure 3 illustrates the modules and their interaction: Modality-specific recognizers and analyzers provide semantically interpreted input to the multimodal fusion module that interprets them in the context of the other modalities and the current dialog context. The dialog manager decides about the next system move based on its task model, the current context and also on the result from calls to the MP3 database. The multimodal fission module then generates an appropriate message to the user by planning the actual content, distributing it over the available modalities and finally co-ordinating and synchronizing the output. Modality-specific output modules generate spoken output and an update on the graphical display. All modules interact with the Discourse Module in which all context information is stored. Within this architecture all of the core tasks, namely discourse modeling, interpretation/fusion, dialog management, linguistic and presentation planning, are modeled by a plan-based approach, using PATE. Due to space restrictions, we just give a short overview of two PATE-based modules of the SAMMIE system.

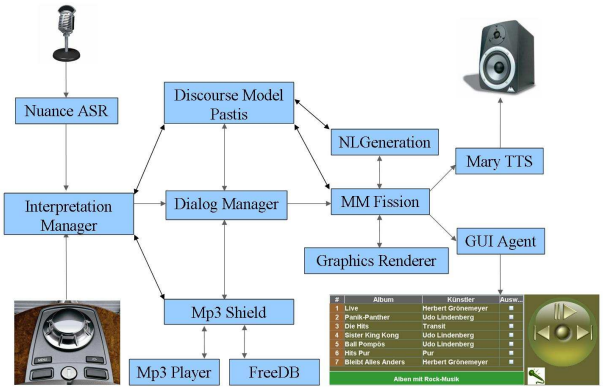


Figure 3: Sammie System Architecture

## 4.2. Dialog Management

In SAMMIE, we are following an approach that models the interaction on an abstract level as collaborative problem solving (CPS) [9] and adds application specific knowledge to the possible *tasks*, available *resources* and known *recipes* for achieving the goals. To this end we developed a PATE-based dialog manager that implements the formal CPS model. The basic building blocks of this model are problem solving (PS) objects, which define the upper level of the CPS-specific sub-ontology of the system and which we term *abstract PS objects*. There are six abstract PS objects in our model from which all other domain-specific PS objects inherit, namely *objective*, *recipe*, *constraint*, *evaluation*, *situation* and *resource*. These abstract objects are used to model problem-solving at a domain independent level and are taken as arguments by all update operators which implement conversation acts [10]. The model is then specialized to a domain by inheriting and instantiating domain-specific types and instances from the PS objects. Note that the operators do not change with domain since the reasoning is done on a domain independent level.

## 4.3. Multimodal Fission

In SAMMIE, multimodal fission is realized by two modules: The PATE-based *Turn Planner*, which is responsible for content planning and media allocation, and the *Output Manager* which coordinates the output and synchronizes the modalities. In general, the turn planner takes a set of CPS-specific conversational acts from the dialog manager and maps them to modality-specific communicative acts. Relevant information on how content should be distributed over the available modalities (speech and/or graphics) can be obtained through the discourse module which provides different types of context information for this case, e.g., the user's cognitive load or the modality on which a user is currently focused on[6]. Furthermore, a set of PATE production rules is used to determine

[6]In order to facilitate system interaction within the in-car scenario we modeled *user expertise* and a user's *cognitive load* to permit a more

which kind of information should be presented through which of the available modalities. Therefore, we developed an upper-level presentation planning ontology which models most of the planning steps on a domain independent level. To this end, additional reasoning is realized by presentation planning specific plan operators but also domain independent plan operators.

### 4.4. System Ontology

The whole SAMMIE ontology currently consists of 87 domain specific and 261 dialog/system specific classes. The latter part is represented in five sub-ontologies which correspond to the five PATE-based modules. See figure 4 for a screenshot of the SAMMIE ontology. Due to the concept of multiple inheritance provided by the type system, domain specific concepts are linked to appropriate upper-level concepts of the different domain independent sub-ontologies. This means, we can model different views onto *real-world* domain specific knowledge and allows to view, e. g., a *song* as a *browsable-object* so that we can generalize within the turn planning library over objects a user can browse. But these objects can also be seen as a *media-object* or a *PS-object* which are abstract concepts the dialog managment uses for planning and execution. Thereby PATE provides an efficient and elegant way to create more abstract/generic planning rules which eases the development of new applications.

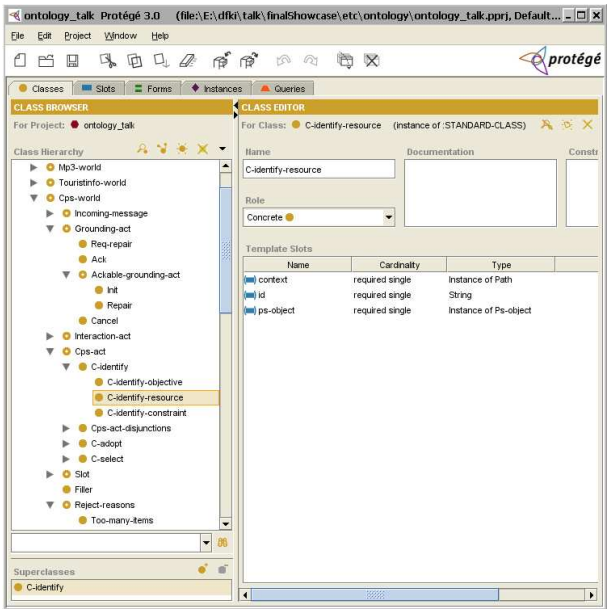


Figure 4: Sammie Ontology

## 5. Conclusion and Discussion

We presented the production rule system PATE which is an easily handable tool for the development of advanced dialog systems. The PATE engine has been used in various dialog systems to implement a variety of components. We gave a brief overview of the SAMMIE in-car dialog system which is the first system where all key components are based on PATE. Our approach is based on the notion of a central ontology-based knowledge representation that is used by all components of the system. One aspect of our current work is to develop a framework where generic application independent PATE rules are combined with domain specific rules that are automatically generated from the application specific part of the ontology in order to ease the process of adapting the system to new applications.

context-adaptive presentation planning, but we haven't yet fully elaborated the rule based processing of these factors as this part of our ongoing work.

## 6. Acknowledgements

This research is funded by the German Ministry of Research and Technology (BMBF) under grant 01 IMB 01A (VirtualHuman), 01 IMD 01A (Smartweb) and by the EU 6th Framework Program under grant, project No. IST-507802 (TALK). The responsibility lies with the authors.

## 7. References

[1] Jan Alexandersson, Tilman Becker, Ralf Engel, Markus Löckelt, Elsa Pecourt, Peter Poller, Norbert Pfleger, and Norbert Reithinger, "Ends-based Dialogue Processing," in *Proceedings of the Second International Workshop on Scalable Natural Language Understanding*, Boston, MA, 2004.

[2] Norbert Pfleger, "Context Based Multimodal Fusion," in *Proceedings of the Sixth Int. Conference on Multimodal Interfaces (ICMI'04)*, State College, PA, 2004, pp. 265–272.

[3] Benjamin Kempe, *PATE – a production rule system based on activation and typed feature structure elements*, http://www.dfki.de/ kempe/pate.pdf, 2004.

[4] John R. Anderson and Christian Lebiere, *The atomic components of thought*, Erlbaum, Mahwah, NJ, 1998.

[5] Bob Carpenter, *The logic of typed feature structures*, Cambridge University Press, Cambridge, England, 1992.

[6] Jan Alexandersson and Tilman Becker, "The Formal Foundations Underlying Overlay," in *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, Tilburg, The Netherlands, February 2003.

[7] Norbert Pfleger, Jan Alexandersson, and Tilman Becker, "Scoring functions for overlay and their application in discourse processing," in *KONVENS-02*, Saarbrücken, September 2002.

[8] H. Bunt, M. Kipp, M. Maybury, and W. Wahlster, "Fusion and coordination for multimodal interactive information presentation: Roadmap, architecture, tools, semantics," in *Multimodal Intelligent Information Presentation*, O. Stock and M. Zancanaro, Eds., vol. 27 of *Text, Speech and Language Technology*, pp. 325–340. Kluwer Academic, 2005.

[9] Nathan J. Blaylock, *Towards Tractable Agent-based Dialogue*, Ph.D. thesis, University of Rochester, Dept. of Computer Science, August 2005.

[10] Nate Blaylock and James Allen, "A collaborative problem-solving model of dialogue," in *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Laila Dybkjær and Wolfgang Minker, Eds., Lisbon, September 2–3 2005, pp. 200–211.