An Ontology-Based Adaptation Framework for Multimodal Interactive Systems

Matthias Bezold

Institute of Information Technology, University of Ulm, Germany, and
Elektrobit Automotive GmbH, Erlangen, Germany
matthias.bezold@uni-ulm.de

Abstract. One approach for improving the usability of interactive systems is adapting them to user behavior, which can be accomplished by adaptation rules. The advantage of rules is that they are explicit and intuitive, but their expressivity depends on the richness of the underlying data model. In this paper, a framework for the adaptation of interactive systems is presented that relies on a uniform ontology-based information representation, for instance for the system and the user model. Such a description can then be employed by the adaptation rules. By adding semantic information, the scope of the rules is widened. Moreover, special emphasis is put on the dynamic aspects of interactive systems, mainly the interaction of the user with the system and system events. Exemplary rules used in an interactive TV prototype illustrate this framework.

Keywords: Adaptive interactive systems, Knowledge base, Ontology, Interactive systems engineering, Rule-based adaptation.

1 Introduction

User groups of interactive systems become more and more diverse. For instance, home entertainment systems or automotive infotainment systems are operated by old and young, well-educated and uneducated people, who have different capabilities and expectations toward the systems. Therefore, there is a need for approaches to create systems usable by a wide range of different users. Making these systems adaptive to each individual user is a solution that has been a matter of research for many years [1,8]. Adaptive systems observe the user and improve themselves by deriving adaptations from the user's behavior.

There are numerous standards for the definition of interactive systems, such as UIML[1] or XUL[2] for graphical and VoiceXML[3] for speech-based systems, and even more research projects. But not all of them are apt for multimodal systems, which can be controlled by more than one modality at the same time. Statecharts [7], also known from the Unified Modeling Language (UML)[4], offer a sound formalism to describe both

[1] User Interface Markup Language (UIML): http://www.uiml.org/
[2] XML User Interface Language (XUL): http://www.mozilla.org/projects/xul/
[3] VoiceXML: http://www.w3.org/TR/voicexml20/
[4] Unified Modeling Language (UML): http://www.uml.org/

P. Forbrig and F. Paternò (Eds.): HCSE/TAMODIA 2008, LNCS 5247, pp. 205–212, 2008.

graphical and speech-based systems. While there are more sophisticated descriptions for instance for speech dialogue systems, such as frame- or agent-based approaches (cf. [9]), these are intended for speech-based systems only and not usable for graphical systems. A statechart model consists of a number of hierarchical states and event-triggered transitions connecting these states. Graphical components, composed of a hierarchy of graphical elements, and speech components, consisting of speech output and grammars for speech input, are attached to states and activated when the respective state is entered. For this work, the statechart-based commercial modeling tool EB GUIDE Studio [5] was extended by an adaptation framework. The tool includes a simulation component, which is used as a dialogue manager.

In order to perform appropriate adaptations in interactive systems, an adaptation framework is required for the development of prototypes and deployed systems. In this work, an adaptation framework is presented that employs rules to define the adaptations. These rules operate on information stored in a knowledge base that describes the system and the user. Special emphasis is put on the dynamic parts of the system, which play a significant role in interactive systems, i.e., the interaction of the user with the system and system events.

This paper is structured as follows. First, Section 2 introduces ontologies and how they are used to create a uniform description of all information relevant for the adaptation. Next, the use of rules on top of the knowledge representation to describe adaptations is discussed in Section 3. Finally, related work is presented in Section 4 and future work is outlined in Section 5.

2 The Semantic Layer

In rule-based adaptive systems, the adaptation rules have to rely on the information provided by the underlying data model. Therefore, the more information is provided by the data model, the more powerful and comprehensive the adaptation rules can be. Moreover, a common representation for all data is needed to make it available to the

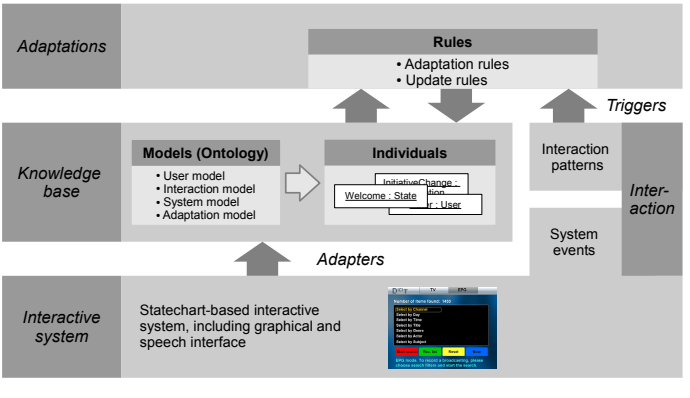

Fig. 1. The architecture of the adaptation framework comprises a knowledge base layer and adaptation rules on top of it. Adaptation rules are triggered by the interaction of the user with the system.

adaptation rules in a uniform way. For this purpose, a semantic layer covering all necessary information was added on top of the interactive system. The different components of the framework architecture shown in Fig. 1 are discussed in the following sections.

Since the information used by the adaptation rules has to be available at runtime of the system, a formalism with extensive instantiation support is required for the knowledge representation. The Web Ontology Language (OWL)[5] provides, in addition to class definition constructs, support for the dynamic instantiation of the models by means of individuals. Hence, OWL is employed in this work by means of the Jena framework[6], a Semantic Web library.

2.1 Ontologies

An ontology is a description of a certain domain, which is expressed as a hierarchy of classes covering all relevant parts of a domain. Classes have a set of properties, which can either be primitive types, such as a string or a number, or references to other individuals. Individuals are stored as so-called triples, which consist of a subject (an individual), a predicate (a property), and an object (a primitive value or another individual). For instance, the name "Welcome" of the state with the ID "State42" is described through the "hasName" property by the following triple:

```
(State42 hasName ''Welcome'')
```

A knowledge base consists of a set of classes and a number of instances of these classes. In the following sections, the different models contributing to the knowledge base are presented and the creation of instances of these models is discussed.

2.2 The Models – System, User, Adaptation, and Interaction Model

As a basis for intuitive and powerful adaptation rules, information about different aspects of the system is required, comprising information about the system, the user and the user's system configuration, the interaction of the user with the system, and information about possible adaptations. Each of these areas is implemented as a model consisting of a set of OWL classes with a number of properties. Most of these classes can be reused for different systems, only application-specific classes, such as annotated information or the interaction, have to be defined for each system.

The *system model* is a technical description of the system, comprising the statechart model and a description of the graphical and speech components. Moreover, to further enhance the scope of the knowledge base, additional semantic information can be annotated to the model. For instance, all elements, e.g. states, buttons, or speech output prompts, have a "type" property that describes their purpose, such as "help".

The *user model* describes the different users of the system. On the one hand, it comprises factual information about the user, such as name or preferences. On the other hand, the user model covers the configuration of the system for the user, e.g. whether certain entities are enabled. Since only the user model can change during the execution of the system, only changes to the user model have to be stored.

[5] Web Ontology Language (OWL): http://www.w3.org/2004/OWL/

[6] The Jena framework: http://jena.sourceforge.net

The *interaction model* describes the interaction of the user with the system by defining interaction patterns out of low-level system events. These low-level events can be state changes in the statechart model, dialogue manager events, input device events (e.g., remote control or input button), or speech input from the speech recognition system. All events have parameters for additional information, such as the name of the button for input device events, which can be used in the rules.

The interaction patterns are not part of the ontology, but are described by means of a Deterministic Finite Automaton (DFA). Sequences, repetitions, and alternatives combine low-level events or again other patterns into interaction patterns. For instance, a menu entry selection by moving the cursor up and down and pressing the OK button can be described by the pattern "(button up OR button down)* AND button OK", with the "*" denoting a repetition. Context information can be used to refine interaction patterns and to connect them to different parts of the system, e.g. differentiating between selections in different menus of the system.

The *adaptation model* describes the adaptations that can be applied to the system. Adaptations are expressed as rules, which are discussed in the following section.

2.3 Instantiating the Models

In order to use these models at runtime of an interactive system, they have to be populated with instances of the model classes, called individuals. For this purpose, special adapter modules are used to initialize the knowledge base accordingly when the system starts. For instance, the "Statechart Ontology Adapter" creates a knowledge base individual for every state in the system definition and fills the properties accordingly, such as the name, whereas the "User Model Adapter" loads information about the users from the user model.

The dialogue manager consults the knowledge base during the execution of the system. For instance, if a state or a speech output prompt is disabled in the knowledge base, it will be skipped. Therefore, the definitions of the respective elements do not have to be updated and the information needs to be stored only in the knowledge base.

3 Adaptations

This section discusses how adaptations are performed. Adaptation rules can exploit the the information provided by the knowledge base. Contrary to statistical approaches, rules are well-predictable and explicit, which is especially important for graphical systems.

3.1 Adaptation Rules

Rules are used for two different purposes. First, the information in the knowledge base can be inferred from the interaction of the user with the system, e.g. by defining a rule that updates the current state of the dialogue system from (low-level) state change events. Second, adaptations can be performed by rules, e.g. by disabling certain elements of the system. Adaptation rules are connected to entries of the adaptation model in order to make this information part of the knowledge base.

The rules used in this framework consist of three parts. First, the *event* part is used to define a trigger for a rule by connecting the rule to a pattern from the interaction model (e.g. menu selection) or a low-level event (e.g. a state change). If rules have no trigger, they are executed at predefined points, such as system startup or user change. Second, the *condition* part can be used to define additional criteria that have to be fulfilled to execute the body of the rule. Third, the *action* part contains the rule body, usually consisting of knowledge base updates.

Knowledge base updates are composed of two parts, the query part and the update part. Since the Jena rule engine only supports monotonic updates, i.e., does not support modifications of existing triples, and the non-monotonicity is important to allow updates to the user model, a custom rule system is used. The query part of the rule is transformed into a SPARQL[7] query. Parameter values from the event, e.g. the name of the new state for a state change event, are passed to the query as bound variables. The update part is a list of triples that will either be updated in the knowledge base determined by the subject and the predicate, or created if no matching triple exists. Updates can contain functors that perform computations, such as incrementing a value.

The evaluation of rules is carried out within a transaction, performing all updates to the knowledge base only after all triggered rules have been evaluated. This is necessary for consistency reasons. For instance, if rule A updated a value read by rule B, which is evaluated after A, the (implicit) order of execution would be relevant. But making the order explicit would increase the complexity unnecessarily.

3.2 Exemplary Adaptation Rules

Two adaptation rules are shown in this section, updating the knowledge base and performing a simple adaptation respectively. The rules in the actual system use XML as a notation, but for clarity reasons, a simplified notation is used. Variable names start with a "?", properties with a lower case character, and instances with an upper case character. The examples are taken from an adaptive home entertainment model that includes an electronic program guide.

In Fig. 2, an update rule is shown that is triggered when the current state in the statechart model changes. Therefore, a "state change" event trigger is used. In the actions part, a query is defined that retrieves the state element in the knowledge base

```
events: state change: ?name
conditions: none
actions:
    query:
        (?state hasName ?name)
        (?status isAboutEntity ?state)
        (CurrentSession hasUser ?currentUser)
        (?status isAboutUser ?currentUser)
        (?status hasUseCount ?oldUseCount)
    updates:
        (?status hasUseCount addOne(?oldUseCount))
```

Fig. 2. Knowledge base update rule that increments the use counter of a state in the user model

[7] SPARQL Query Language: http://www.w3.org/TR/rdf-sparql-query/

```
events: pattern: "UserIsLost"
conditions: none
actions:
    query:
        (CurrentSession hasUser ?currentUser)
        (?element hasType DialogueType_Help)
        (?status isAboutEntity ?element)
        (?status isAboutUser ?currentUser)
    updates:
        (?status isEntityEnabled "true")
```

Fig. 3. An adaptation rule that enables "help" elements (states, speech output prompts, etc.) if a user is lost (defined by the interaction pattern "UserIsLost")

determined by the name of the state (?name), selects the current user, her status triple for the state, and the old use count. The "updates" section updates the use count, which is computed from the old value using the "addOne" functor.

A (simplified) adaptation rule is given in Fig. 3. This rule enables help when a user seems to be lost, which is defined by the interaction pattern "UserIsLost", e.g. defined as "random scrolling" (not shown). "Help elements" are defined by annotations to the system model, which are available through the value "DialogueType_Help" of the "hasType" property.

4 Related Work

Ontologies have been used in multimodal systems for other purposes than adapations, such as supporting semantic coherence checking [6] in the SmartKom project or domain reasoning in the dialogue manager [4] in the Talk project. Moreover, ontologies have been used for modeling interactive systems. In [10], an ontology is used in addition to UML in the development process of multimodal interactive systems. The high-level model description is only available at design time to provide development support and for platform mapping, but not at runtime of the system.

Sophisticated adaptation architectures have been presented in the domain of adaptive hypertext systems. A formal definition of an adaptive hypermedia system is presented in [3]. The adaptation component consists of a set of rules expressed as first-order logic statements using a language called TRIPLE. The adaptations rely on a semantic annotation of the document space. Another framework for adaptive systems in presented in [2]. It relies on OWL for describing the system and the domain and the SWRL rule language to express the adaptations. The ODAS domain ontology [11] is used in conjunction with adaptation rules in an adaptive hypertext portal. The authors reason that rules provide a better transparency and controllability for users than statistical adaptation methods. The ontology provides a knowledge foundation for the adaptation rules and contains different models, such as a system model, a task model, and a resource model.

These adaptation architectures have only been applied to hypertext systems, but not to interactive systems in general, which are richer with regard to their interaction possibilities. Hence, this approach is better suited for interactive systems, since interaction patterns and system events can directly be connected to the adaptations rules.

5 Conclusions and Future Work

This work presented a framework for adapting dialogue systems to user behavior that is based on a knowledge base and adaptation rules. The knowledge base, which is defined by a set of OWL models and populated by means of adapter components, covers all aspects that are relevant for the adaptations. The expressivity of adaptation rules benefits from this information. Since the interaction of the user with the system is a vital part of adaptive systems, special emphasis was put on this issue by defining interaction patterns that trigger rules. These interaction patterns are defined out of low-level system events, such as input device or speech input events. Exemplary rules for knowledge base updates and adaptations were given to illustrate the use of this framework.

There are two main directions for future work. First, the description of the user interaction can be improved by adding a task model and connecting it to the interaction model. A task model describes what a user can do with an interactive system on an abstract task level. Thus, the expressivity of adaptation rules can benefit from this additional information. Second, the development of adaptive systems can benefit greatly from defining a list of adaptations and formalizing them as adaptation patterns. Based on existing research on human-computer interaction patterns [12], concrete adaptation patterns can be included in the ontology and connected to the interaction and task models. These patterns can then be added to the model by the system designer or executed automatically.

References

1. Browne, D., Totterdell, P., Norman, M. (eds.): Adaptive User Interfaces. Academic Press Ltd., London (1990)
2. Carmagnola, F., Cena, F., Gena, C., Torre, I.: A Multidimensional Framework for the Representation of Ontologies in Adaptive Hypermedia Systems. In: Bandini, S., Manzoni, S. (eds.) AI*IA 2005. LNCS (LNAI), vol. 3673, pp. 370–380. Springer, Heidelberg (2005)
3. Dolog, P., Henze, N., Nejdl, W., Sintek, M.: Towards the Adaptive Semantic Web. In: Bry, F., Henze, N., Małuszyński, J. (eds.) PPSWR 2003. LNCS, vol. 2901, pp. 51–68. Springer, Heidelberg (2003)
4. Garcia, G.P., de Amores Carredano, J.G., Portillo, P.M., Marin, F.G., Marti, J.G.: Integrating Owl Ontologies With a Dialogue Manager. In: Procesamiento del Lenguaje, pp. 153–160 (2006)
5. Goronzy, S., Mochales, R., Beringer, N.: Developing Speech Dialogs for Multimodal HMIs Using Finite State Machines. In: 9th International Conference on Spoken Language Processing (Interspeech), CD-ROM (2006)
6. Gurevych, I., Porzel, R., Malaka, R.: Modeling Domain Knowledge: Know-How and Know-What. In: Wahlster, W. (ed.) SmartKom - Foundations of Multimodal Dialogue Systems, pp. 71–84. Springer, Heidelberg (2006)
7. Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Sci. Comput. Program. 8(3), 231–274 (1987)
8. Jameson, A.: Adaptive Interfaces and Agents. In: Human-computer Interaction Handbook, 1st edn., pp. 305–330. Erlbaum, Mahwah (2003)
9. McTear, M.F.: Spoken Dialogue Technology: Towards the Conversational User Interface. Springer, London (2004)

10. Obrenovic, Z., Starcevic, D., Devedzic, V.: Using Ontologies in Design of Multimodal User Interfaces. In: Rauterberg, M., Menozzi, M., Wesson, J. (eds.) INTERACT 2003. IOS Press, Amsterdam (2003)
11. Tran, T., Cimiano, P., Ankolekar, A.: A Rule-Based Adaption Model for Ontology-Based Personalization. In: Wallace, M., Angelides, M.C., Mylonas, P. (eds.) Advances in Semantic Media Adaptation and Personalization. Studies in Computational Intelligence, vol. 93, pp. 117–135. Springer, Heidelberg (2008)
12. van Welie, M., van der Veer, G.C.: Pattern Languages in Interaction Design. In: Rauterberg, M., Menozzi, M., Wesson, J. (eds.) INTERACT 2003. IOS Press, Amsterdam (2003)