Journal of Ambient Intelligence and Smart Environments 2 (2010) 369–387
DOI 10.3233/AIS-2010-0081
IOS Press

A framework for adapting interactive systems to user behavior

Matthias Bezold [a,b,*] and Wolfgang Minker [a]
[a] *University of Ulm, Institute for Information Technology, Ulm, Germany*
[b] *Elektrobit Automotive Software, Erlangen, Germany*
Email: matthias.bezold@uni-ulm.de, wolfgang.minker@uni-ulm.de

Abstract. One advanced feature of user interfaces in smart environments is adaptation, the ability of the interface to improve itself for an individual user based on an observation of the user's behavior. Adaptation to user behavior comprises two steps: drawing conclusions from the user-system interaction and adapting the interface accordingly. In this article, the user-system interaction is regarded as a sequence of basic events, which are emitted by the interactive system. Different user modeling algorithms use the information from these basic events to extract new knowledge. Finally, adaptations are triggered based on this information.

Since adaptations rely on the expressiveness of the underlying data model, a semantic layer is introduced as an abstraction of the interactive system. The system-independent logic of the adaptations is defined on top of this ontology-based layer, whereas the system-specific execution is defined separately. A set of reusable adaptations is defined as design patterns and the adaptations are integrated into the framework. We present an adaptation framework that comprises a user modeling component and an adaptation component. In order to show the technical feasibility of the framework, we created a reference implementation that comprises all presented components. In addition, the implementation serves as a test bed for a user evaluation of the adaptations. For this purpose, test subjects fulfilled tasks with test systems that implement the adaptations.

Keywords: Adaptive interactive systems, adaptations, user modeling, ontologies, adaptation patterns, evaluation of adaptations

1. Introduction

Computer systems increasingly become part of our everyday lives. For instance, cell phones, television systems, and dashboard interfaces in the car have become ubiquitous companions. At the same time, the range of functions and complexity of these systems is increasing. The living room is transforming from a room with isolated appliances into a connected and smart environment that offers a multitude of functions. The smart environment knows the user's preferences, such as television channels, music selection, and lighting, and configures the room accordingly for an individual user. Similarly, the automobile radio has grown into a connected infotainment system that loads different kinds of information from online sources. At the same time, the variety of the users increases, with young and old, experienced and inexperienced people using these interactive systems. Thus, new approaches have to be found to enable people to operate these increasingly complex interfaces.

One approach for improving the usability of these interactive systems is to enable the interface to adapt itself to an individual user. Adaptations modify the interactive system to the user during the interaction with the interactive system. For this purpose, the interactive system observes the user and derives adaptations from the user-system interaction. In doing so, the interactive system better reflects individual capabilities and needs of a user [19]. Adaptations support beginners by guiding them throughout the interaction and experts by assisting them with repetitive tasks. This work addresses multimodal interactive systems, i.e., interactive systems that may be controlled with more than one modality, such as graphics, speech, and gestures.

*Corresponding author.

We regard the interaction of a user with an interactive system as a sequence of basic events. The interactive system emits these events, but sensor readings from the smart environment (such as the location of the user or the current room temperature) may be integrated as well. In order to describe the user, the interactive system first extracts meaningful information from this sequence. This includes a description of the actions a user performs and information about selected values. Next, user modeling algorithms perform further derivations and computations, such as predicting the most likely next user action or learning user preferences. Numerous user modeling algorithms have been presented in the literature. In this work, we present how different algorithms may be integrated into this framework. After user modeling algorithms have extracted and computed information about the user, this information serves as a trigger for adaptations.

Adaptations rely on a comprehensive description of the interactive system, the user, and the user-system interaction. An adaptation component may only consider information in the adaptation decision that is provided as a data model. The adaptation component in this framework employs an ontology to create a semantic representation of the interactive system, the user-system interaction, and the adaptation. Based on this abstraction, the adaptation logic is defined in a reusable and system-independent way. However, the appearance of these adaptations in a specific system is defined in a system-dependent way. We defined a set of generic adaptations for interactive systems as design patterns. These adaptations are applicable to a wide range of interactive systems and cover both graphical and voice interfaces.

In this article, we discuss an adaptation framework. After introducing the concepts and components of the framework, we present a reference implementation to show the technical feasibility of this approach. The framework covers both user modeling and adaptations. The user modeling component extracts information from basic events and forwards them to an adaptation component. Figure 1 gives an overview of this framework. On top of the interactive system, an ontology-based semantic layer provides an abstract semantic representation of the interactive system as well as other aspects of the user-system interaction. The interactive system sends basic events caused by the user-system interaction to the user modeling component. The user modeling component in turn triggers an adaptation component and the adaptation component accesses the semantic layer to perform adaptations to the interactive system. A set of patterns forms the basis of the adaptations in the adaptation component. We created a reference implementation of the adaptation framework to show the technical feasibility of the approach. The implementation includes all components of the architecture. In order to investigate the adaptations, we carried out a user test that uses the reference implementation as a test bed. The evaluation not only examines the feasibility of the adaptations, but also explores the conditions under which the adaptations work best.

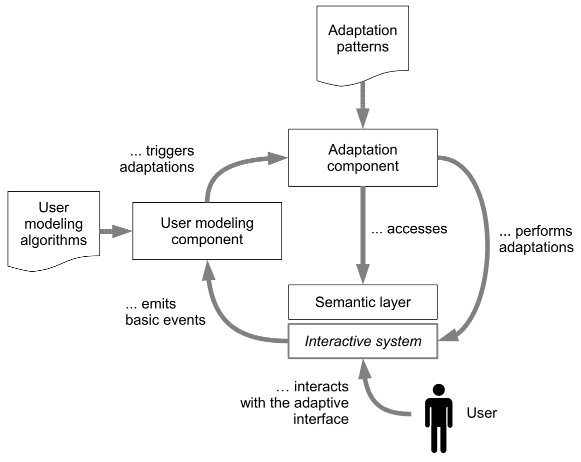

Fig. 1. The architecture of the adaptation framework comprises a semantic layer and an adaptation component on top of it. Adaptations are triggered by the user-system interaction.

This article is organized as follows. First, Section 2 gives an overview of related work. In Section 3, we present a framework for user modeling from a sequence of basic events. Section 4 introduces a semantic knowledge representation for the interactive system based on an ontology. Next, we describe the definition of adaptations on top of this knowledge representation and the execution of these adaptations in Section 5. Finally, we present a reference implementation of the adaptation framework and a user test of the adaptations in Section 6. Section 7 concludes this article.

2. Related work

In this section, a review of related work in the domains of user modeling architectures and adaptation frameworks is given.

2.1. User modeling architectures

Numerous approaches and systems for user modeling exist. This section introduces a selection of them and compares them to the approach presented in this article with regard to how inference of new information is performed.

Different generic user modeling architectures have been presented. The Doppelgänger User Modeling System [26] is a generalized user modeling system. Learning is supported by means of different techniques, such as linear prediction and Markov models. Contrary to our architecture, custom learning techniques cannot be added and the data accumulation is left to the application. In systems with logic-based representations, inferences are drawn by means of logical reasoning. In the BGP-MS system [21], the internal representation is converted to first-order logic to facilitate the use of logic reasoning. Moreover, inferences about the user can be drawn by rules. However, complex data types and computations such as Markov chains are not supported by the BGP-MS system. A user modeling framework for intelligent learning environments that implements both supervised and unsupervised learning is presented by Amershi and Conati [1].

Research on user modeling algorithms as well as machine learning provides a multitude of approaches, each suited for different applications. An extensive overview of predictive models for user modeling is for instance given by Zukerman and Albrecht [38]. Depending on the requirements of the interactive system, these algorithms by be implemented into this framework.

A number of user modeling systems employ semantic technologies. For example, Razmerita et al. [30] present a general architecture for a user modeling server that is based on semantic technologies. Various ontologies were proposed specifically for user modeling. Golemati et al. [13] and Heckmann et al. [18] demonstrate sophisticated examples of such ontologies. The ontology of our framework also includes user modeling information. However, the framework does not store complex data in the ontology, because ontologies cannot store raw data efficiently, such as matrices. Instead, this framework uses a bridging component to connect the user model to the knowledge base in order to better support complex data types. In doing so, the framework stores data efficiently and integrates data with an ontology at the same time.

2.2. *Adaptation architectures*

Ontologies have a broad area of application and have been used in multimodal systems for different purposes, such as supporting semantic coherence checking [16] in the SmartKom project or domain reasoning in the dialog manager [12] in the Talk project. Moreover, ontologies have been used for modeling interactive systems. For instance, Obrenović et al. [25] employ an ontology 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 as a knowledge representation for the interactive system itself. In this framework, the ontology is available at design time and runtime and creates an abstraction of the interactive system that is used to perform adaptations.

Sophisticated adaptation architectures have been presented in the domain of adaptive hypertext systems. Dolog et al. present a formal definition of an adaptive hypermedia system [9]. The adaptation component consists of a set of rules. These rules are stored as first-order logic statements using a language called TRIPLE. The adaptations rely on a semantic annotation of the document space. The ODAS domain ontology presented by Tran et al. [35] 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 frameworks employ logic-based reasoning or rules to express the adaptations, whereas our framework supports – in addition to adaptation rules – a more explicit representation of adaptations by embedding them in the knowledge base. Moreover, the focus of this framework is on interactive systems, which deal with a user interface rather than documents and their interconnection.

Several adaptation architectures for generic interactive systems exist. For example, Aragones et al. [2] present a semantic adaptation framework, called ACUITy. A controller component mediates between a UI engine and an ontology, which comprises information on the user, the user interface, and the domain. An interface is generated from an ontology-based definition of the application. Custom-tailored interfaces are created by incorporating information about past interactions. The adaptation framework presented in this article also exploits semantic technologies. However, ACUITy is focused on rich interfaces, whereas this adaptation framework focuses on embedded interfaces. In addition, this framework includes a set of specific adaptations. Another approach is the ISATINE framework [22], which decomposes the adaptation into seven stages of adaptation: goals for adaptation, initiative, specification, application, transition, interpretation, and evaluation. An implementation is presented that employs an agent-based architecture. The ISATINE framework supports user- and system-initiated adaptations. Our adaptation framework is focused on system-initiated adaptation and includes a set of adaptations. In addition, agent-based frameworks are less suited for interactive systems with limited computational resources.

3. User modeling

The process of observing a user and drawing conclusions from the user-system interaction is called user modeling. User modeling monitors user input and uses different algorithms to comprehend the user's behavior, for instance which actions the user performs. Thereafter, the interactive system predicts future user actions and learns user preferences. Since adaptations rely on user modeling as a trigger, this procedure is crucial for adaptive interfaces. In order to enable adaptivity for different interactive systems, a generic architecture for user modeling is required. User modeling comprises two phases: extracting information from basic observations and performing further derivations with different user modeling algorithms. Since the requirements of different adaptive interfaces with regard to user modeling algorithms are manifold, extensibility and generality are prerequisites for user modeling architectures. Numerous approaches for user modeling have been presented in the literature. On the one hand, generic algorithms have been discussed by Zukerman and Albrecht [38] or Amershi and Conati [1]. On the other hand, domain-specific algorithms have been used for particular tasks, such as modeling the behavior of museum visitors [37] or recommending shows in personalized digital television [3]. Therefore, developers of adaptive interactive systems may select the most appropriate algorithms from the literature. In this article, we present a framework that enables system designers to integrate user modeling algorithms into adaptive interactive systems more easily, rather than introducing new algorithms. Example algorithms demonstrate the workings and the feasibility of the user modeling component.

User input and the resulting (internal) reactions of the interactive system generate a sequence of low-level events. Sensor readings from the smart environment may contribute to this stream as well. Examples of such data are the location of a user, physiological readings, or the room temperature. Sensor readings are integrated into the user modeling process by converting them into low-evel events. Therefore, the user-system interaction is regarded as a sequence of basic events (cf. Dix [8]). User behavior is modeled by extracting information from this sequence, such as which action the user is performing or which values are selected most frequently. Based on this information, further derivations are computed, such as predicting user actions or learning preferences. The selection of the user modeling algorithms depends on the requirements of the adaptations.

In this section, we present a reusable and generic user modeling framework for interactive systems. The user modeling component consists of higher-level models that create new events from basic events and a user model, which infers new information with different algorithms. The architecture supports complex data types by means of an extension facility, which are needed to handle complex data for arbitrary algorithms efficiently.

3.1. User interactions as a sequence of events

When a user works with an interactive system, he or she wants to perform different actions in order to accomplish a specific goal. For example, a user might browse the electronic program guide in a digital TV system by means of speech commands and select a specific TV channel. The interactive system observes a speech utterance as well as system reactions to this input, for instance internal state or property changes. Similar events are emitted when the user presses a button on a remote control or operates an interactive system with touch-screen capabilities. A basic event represents each of these observations. Further interactions by the user thus create a growing sequence of events. The user modeling process extracts additional information from the sequence of basic events, such as detecting user actions, and computes further derivations, e.g. predicting the next user action.

The view of the system is limited to basic events, i.e., only what the interactive system observes as basic events becomes part of the user modeling process. To include additional sensors from the smart environment, the sensor readings are converted into basic events. An event is the basic element of the user-system interaction and consists of a timestamp, a type, and a type-

```
1 [1259582934671] utterance
            text={select channel CNN}
2 [1259582934671] event
            name={e.SelectChannel}
3 [1259582934672] state
            name={ChannelSelection}
  ...
4 [1259582934680] interaction
            name={ChannelSelection}
            channel={CNN}
5 [1259582934682] task
            name={ProgramGuide}
```

Fig. 2. Exemplary log lines, showing a speech utterance that changes the current channel (line 1), reactions of the system (lines 2–3), and further derivations from other components (lines 4–5).

specific set of parameters. These events are emitted by different modalities, such as speech and haptic input. An example is given in line 1 of Fig. 2: a speech utterance is represented by an event of the type "utterance" that has a property "text". This property contains the utterance of the user, in this case "select channel CNN". Lines 2 and 3 show system reactions to this input.

A user modeling component represents the central part for user modeling in this adaptation framework. This component consists of a generic communication bus, a set of information extraction models, and a user model for storing and deriving user-related data. An overview of the user modeling component and the connection to an interactive system is given in Fig. 3. A generic communication bus connects different components that contribute to the user modeling process. The interactive system sends events to the bus and different models process these events. A user model performs further computations. The wide-ranging requirements of different user modeling algorithms (cf. Section 2) with regard to data types and the algorithms themselves have to be reflected by a flexible and extensible architecture.

Different components of the interactive system exchange events through an event bus, which is conceptually similar to blackboard architectures (cf. Corkill [7]). Producers submit events to the bus. For instance, the interactive system submits events when it observes button presses, speech utterances, or state changes, such as lines 1–3 in Fig. 2. Other components subscribe to the event bus, either to all events or certain event types with specific parameters. Components are notified when appropriate events have been submitted.

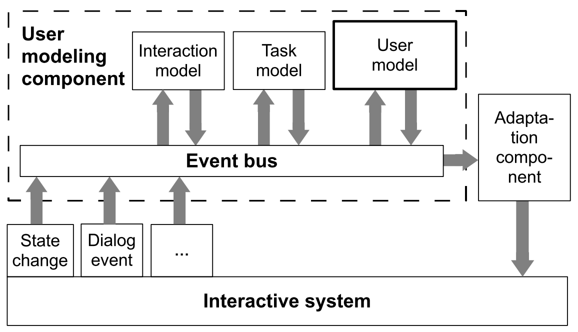

Fig. 3. The event architecture transmits events from different sources, such as the interactive system or higher-level components.

Thus, the event serves as a common and generic interchange format between different components.

The user modeling component extracts further information from the sequence of basic events, such as detecting user actions. For this purpose, higher-level models observe the sequence of events and submit new events to the event bus. Two example models shall illustrate this concept. These models build on each other and were implemented within this framework. First, an interaction model employs machine learning techniques to extract user actions from the sequence of events. We use an interaction model that recognizes user actions by means of probabilistic automata [36], which are trained from annotated log data. The interaction model not only detects user actions, but also extracts parameters of these actions, such as the name of the channel in a channel change action. Second, a task model observes the events from the interaction model and creates a new event when a task was recognized or finished. The task model defines possible user actions with a description similar to ConcurTaskTrees [29]. The interaction model describes the individual interaction steps. In Fig. 2, the interaction model observes different events (lines 1–3) and emits a new "interaction" event for a "ChannelSelection" user action (line 4), with the "channel" parameter being set to the respective value. The task model observes this action and concludes that the "ProgramGuide" task is performed, thus emitting a "task" event (line 5).

Sophisticated user modeling algorithms are implemented in a user model, which is part of the user modeling component. This model stores user-specific data, extracts information from the user-system interaction, and infers additional information. The user model is discussed in detail in the following paragraph. Finally, user modeling events trigger adaptations in an adaptation component. This lineup of components allows

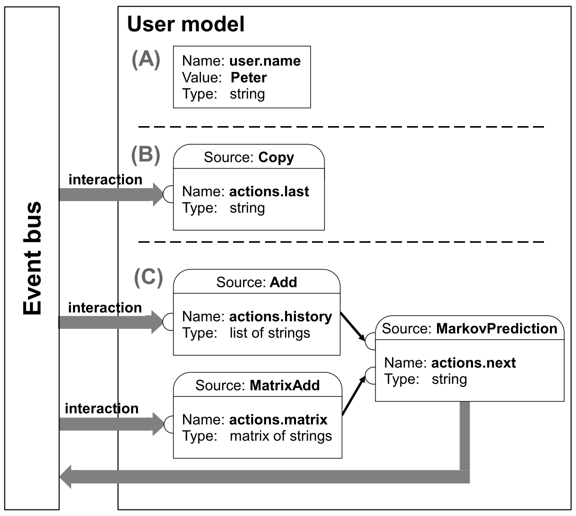

Fig. 4. An example user model with derivations.

a comprehensive description of user behavior, but additional models may be connected to the event bus. Therefore, a generic event bus connects different components and enables a description of user behavior from low-level events.

3.2. *The user model*

This section describes the user model, which serves two purposes: extracting information from events and deriving new information from existing information.

3.2.1. *Architecture*

The user model stores user-related pieces of information, for instance the user's name, the user's role (e.g. beginner or expert), or the list of the past interactions. The user model supports complex data types to represent data that is used by specific user modeling algorithms. These types include lists, maps, and matrices, but also arbitrary other types. User-defined data types are integrated by means of an extension facility.

Figure 4 illustrates the workings of the user model. In Fig. 4.A, an entry "user.name" of the type `string` stores the name of the user "Peter". Each entry is stored separately for the individual users and template roles define default values. For instance, the "beginner" role provides values for users who are not experienced with the interactive system.

3.2.2. *Derivations*

In addition to storing user-dependent data, the user model also performs derivations. For this purpose, it loads values directly from events and updates values based on events and other user model entries. Every user model entry may have a source that defines how the data is loaded or updated. A source consists of a data provider, which is either a user modeling event or another user model entry, and an operator that defines how new values are computed. The simplest operation is copying a value from the source to the entry. An example is shown in Fig. 4.B: the source `Copy` copies the name of the last user action from the respective "interaction" event into the "actions.last" entry of the user model. The architecture also allows the definition of more complex operations. Moreover, the outcomes of computations, which are stored in user model entries, again serve as sources for computations of other entries.

A more complex example shall illustrate the use of sources and complex data types in the user model in detail. We briefly introduce an action prediction algorithm and discuss the implementation of this algorithm in the user model. The algorithm is based on an algorithm for link prediction by Sarukkai [31] that employs first-order Markov chains, a tool for modeling sequential processes. The prediction algorithm works as follows. First, a numerical value is assigned to each user action to represent actions in vectors and matrices. The size of an action vector is equal to the number of possible interactions. In each vector, all indices are set to 0, except for the index of the corresponding interaction, which is set to 1. The last $N = 5$ interaction vectors are stored in a history vector h. A matrix A stores the probabilities a_{ij} that action j follows action i. A prediction for the next interaction is computed using Eq. (1).

$$s(t) = \sum_{j=1}^{N} d(j)h(t-j)A^j \tag{1}$$

$s(t)$ represents the next step and d is a dampening factor that decreases the weight of older actions. The transition matrix T is trained as follows. If the last action was i and an action j occurs, the value t_{ij} of T is incremented by 1. The probability transition matrix A is computed directly from T by dividing each value by the sum of the respective row.

An implementation of this algorithm serves as an example (see Fig. 4.C). The list of past interactions is stored in the user model entry "actions.history" and updated by the `Add` source that appends new interactions to the end of the list. Moreover, the matrix "actions.matrix", which corresponds to T, stores the number of transitions between two interactions, and the `MatrixAdd` operator updates the matrix when a new "interaction" event is observed. The `MarkovPrediction` source uses the prediction algorithm to compute the most likely next interaction based on these entries and stores it into the "actions.next" entry. For this purpose, this source converts the matrix "actions.matrix" into a probability matrix and combines it with the interaction history stored in "actions.history" to compute the prediction.

Thus, arbitrary derivations may be performed with values extracted from events by combining existing and possibly creating new operations. This includes for examples a model of the user's favorite channel or a prediction of the next action. Other algorithms, such as the algorithms presented by Zukerman and Albrecht [38], may be implemented similarly within this framework. An extension mechanism enables the definition of custom operators as well as specific data types for these operators.

4. A semantic representation of interactive systems

The user modeling component derives information from basic events. An adaptation component exploits this information to decision about adaptations in an interactive system. For this decision, the adaptation component requires information both about the interactive system, the user, and other topics. Since the interactive system can only consider information it has access to, both the amount of data and the representation formalism of this information are crucial for an adaptive interface. For this purpose, all relevant information has to be represented in a common formalism. Ontologies provide such a formalism for defining a domain. This work applies ontologies to the domain of adaptive interactive systems. The adaptation component is enabled to include all information in the adaptation decision by describing all facets of the interactive system using this formalism.

In this section, we present an ontology as the basis of an abstraction layer on top of the interactive system and the integration of this layer with an adaptation component. The ontology is available both at design time and at runtime of the interactive system.

4.1. Ontologies

An ontology [15,24] is a formal representation of a domain. The ontology describes the domain as a set of entities and the relationships between these entities. Each entity has properties, which use either primitive data types, such as string or number, or are connections to other entities. Another important property of an ontology is the ability to infer new information from existing information. For example, inference rules encode general knowledge about the domain.

The Web Ontology Language (OWL) [33] is a formalism for defining classes of entities with common sets of properties. Thus, items of the same kind are represented formally by one class and share the same properties. Inheritance creates a hierarchy of classes that inherit properties from their parent class. Individuals are instantiated from a class and all the properties defined by the class are filled in. A set of OWL classes forms the basis of a semantic layer. The ontology covers different parts of the adaptive interactive system, which are described by a set of OWL classes.

4.2. An ontology for interactive adaptive systems

The ontology used in this framework covers different facets of adaptive interactive systems, such as the system itself, the user, the user-system interaction, and the adaptations. The ontology consists of a set of OWL classes with an associated set of properties. For example, the name of a user represented by the "User" class is stored in a property called "hasName". The OWL classes form a hierarchy with a common base class called "DialogThing". Each part of the ontology, such as the description of the system or the user-system interaction, is again based on a common ontology class. For instance, all classes of the system description are derived from the "DialogSystem" class. Further knowledge is encoded in the OWL representation, for instance as restrictions to properties or general knowledge about the system by means of inference rules.

An overview of the ontology structure is given in Fig. 5. The system description defines the structure of the interactive system and the interface elements, such as graphical buttons and lists or speech components, as well as their properties. The dialog domain describes aspects of the domain that are relevant for adaptation, such as a list of channels in an interactive TV system. User modeling information reflects data specific to the current user. In addition, the user-system interaction is described by different user actions. Finally, the ontology comprises information on adaptations. The ontology is restricted to a declarative description of the in-

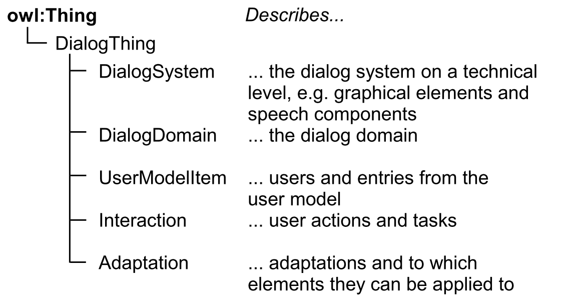
owl:Thing
Describes...
DialogThing
DialogSystem
... the dialog system on a technical level, e.g. graphical elements and speech components
DialogDomain
... the dialog domain
UserModelItem
... users and entries from the user model
Interaction
... user actions and tasks
Adaptation
... adaptations and to which elements they can be applied to

Fig. 5. The top-level classes in the class hierarchy of the ontology, including the root classes of the five sub-models.

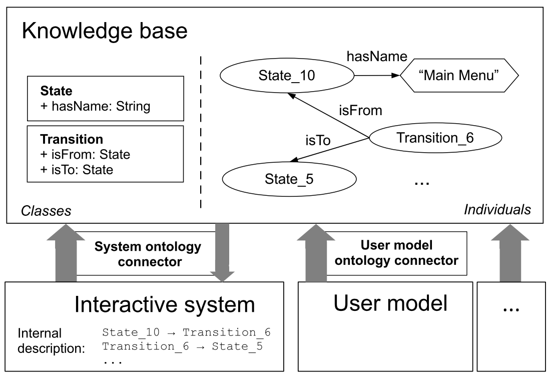

Fig. 6. Ontology connectors load information from the interactive system into a knowledge base, creating an individual for the respective elements of the interactive system.

teractive system, i.e., it describes the structure of the system, but not the workings.

4.3. Populating a knowledge base

In order to describe a specific system with the ontology, instances of the ontology classes are created to form a knowledge base. For this purpose, the corresponding OWL class is instantiated for each entity of the interactive system. Based on an existing description of the interactive system (e.g. from model-based development [32]), the knowledge base is populated automatically. A set of ontology connectors carries out the work of creating individuals for a specific part of the interactive system. In doing so, these connectors bring together data from different information representations, such as a model-based system description or a user model, into a common ontology-based formalism. For instance, a system ontology connector creates an individual for each graphical or speech entity of the system and interconnects them accordingly. Similarly, the user model connector loads information from the user modeling component. Connectors have to be created once for each specific component and may be reused for different interactive systems that use the same component. For example, if a connector was created for a model-based system description, the connector may be used for all systems that are based on this description. The system designer does not have to provide mapping information.

Figure 6 illustrates the concept of ontology connectors. This example includes the "State" and "Transition" classes that represent states and transitions respectively. The ontology connector creates individuals for all states and transitions from the internal system representation of the interactive system, in the example "State_10", "State_5", and "Transition_6". The individuals are connected and further properties, such as the "hasName" property, are filled in. Other connectors, such as a user model connector that loads information from the user model, load data from other components accordingly. Static information that does not change between different runs of the system (such as information about the domain) is imported at design time. Dynamic data (such as information about the current user) has to be loaded at runtime. When the framework starts (either at design time in the development environment or at runtime), existing data is loaded and the connectors produce a common representation of different aspects of the system. In addition to the data loaded by the ontology connectors, the system designer may provide manual annotation that is not available in other sources. For instance, this annotation may define which interface elements are for beginners or experts.

Queries provide access to this semantic representation. These queries return a set of ontology statements for an input set, e.g. all individuals for a certain class. The adaptation component performs such queries to determine which interface elements to select for an adaptation. On the other hand, the adaptation component also updates the semantic representation to alter the interactive system. For this purpose, the connectors observe the entities and update the interactive system accordingly.

4.4. Connecting the user model and the ontology

Ontologies have proven to be a useful tool for user modeling [18,30]. However, complex data types cannot be represented suitably and efficiently in an ontology. For instance, a matrix consists of a number of cells. In order to store a matrix in an ontology, a large number of statements are necessary and access to the matrix through these statements is inefficient. Instead, this framework employs a bridging component to connect user model entries to the ontology.

User model entries correspond to properties of individuals. For instance, the user model entry "user.name" introduced in Fig. 4.A corresponds to the property "hasName" of an individual of the "User" class. In order to connect user model entries to the ontology, the user model entry definition contains mapping information, i.e., the ontology class of the individual and how to retrieve the respective individual from the ontology. If the data changes, the corresponding individual of the ontology is updated, and the other way round. Only user model entries that have mapping information are synchronized with the ontology. In doing so, the information stored in the user model may also be employed in ontology reasoning, and at the same time, complex data types are used efficiently in conjunction with an ontology.

5. Performing adaptations

In this section, we present an adaptation component that employs the semantic representation of the interactive system to apply adaptations. User behavior and computations by the user modeling component trigger these adaptations. For this purpose, the results of the user modeling component are forwarded to the adaptation component as adaptation triggers. These triggers include a prediction of the next user action or an update of a model that describes a preference, such as a TV channel or an address book entry. After it received a trigger, the adaptation component selects an appropriate adaptation and an interface element and carries out the respective adaptation.

5.1. Defining adaptations

The description of adaptations consists of a system-independent and a system-dependent part. The system-independent part, which is called *adaptation selector*, defines adaptations on an abstract level and may be reused between different interactive systems that use this framework. It does not contain system-specific information on how to execute the adaptation. Adaptation selectors consist of a trigger, an ontology query, and a reference to an abstract adaptation. Adaptation selectors are connected to the user modeling component through event triggers, for instance the prediction of a user action by the user modeling component. When the respective trigger occurs, the adaptation selector is activated and uses a query to select an individual from the ontology that should be adapted. Finally, the adaptation selector has an abstract adaptation associated with it that should be executed on the selected individual. Figure 7 illustrates the definition of adaptations in this framework. The adaptation selector "A" is triggered by an action prediction and selects a graphical button that is connected to the predicted action. In this case, the abstract adaptation "ButtonEmph" is chosen.

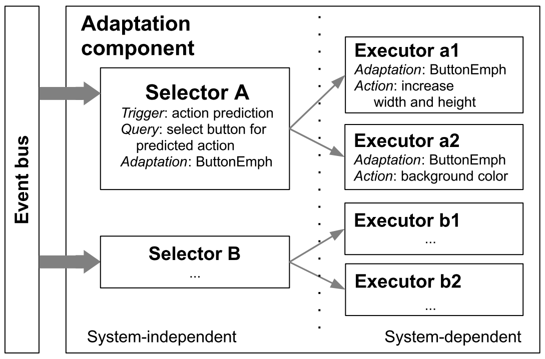

Fig. 7. The definition of adaptations is separated into system-independent adaptation selectors and system-dependent adaptation executors.

The system-dependent part of the adaptation description defines the execution of the selected adaptation on a specific interface element. For this purpose, the adaptation framework comprises a set of *adaptation executors*. Each of these executors defines how to execute a specific abstract adaptation on a specific interface element. The example in Fig. 7 provides two different adaptation executors for the abstract "ButtonEmph" adaptation. Adaptation executor "a1" visualizes the adaptation by updating the "width" and "height" properties to increase the size of the button. The adaptation executor "a2" performs the adaptation by changing the background color of the graphical button.

5.2. Predefined adaptation patterns

In order to support the integration of adaptations into interactive systems, we defined a set of standard adaptations as adaptation patterns. Patterns collect proven solutions for recurring problems. In the domain of software development, the design patterns by Gamma et al. [11] are the most well-known use of patterns. However, patterns also have been applied to domains such as interaction design (e.g. the interaction design patterns by Tidwell [34]). An important requirement for adaptations in interactive systems is not to break existing usability principles, such as the recommendations by the DIN 9241-110 norm [10]. In order to comply with principles such as predictability, learnability, and consistency, the adaptations do not alter the interactive system in a fundamental way, but enhance the interface in order to improve the user-system interaction.

The adaptation patterns were defined in a general way and the concept behind these patterns may be applied both to graphical and speech interfaces. In the following, a brief summary of the adaptation patterns is presented. The full pattern descriptions are presented by Bezold [5].

- **Component Emphasis:** The adaptation focuses the user's attention on a specific interface element by emphasizing it. For instance, a speech command may be emphasized when the available commands are read out. If the adaptation moves a speech output to the beginning or to the end of the list, the user notices the respective element. In a graphical interface, the adaptation may highlight a graphical button on the screen, for instance by increasing its size and selecting more noticeable colors, based on an action prediction by the user modeling component.
- **List Element Selection:** By emphasizing an element in a list, the most frequently selected or most likely elements in a list are put to the user's attention, thus supporting the selection of an element from the list. For example, a speech interface may offer a list of elements to the user by reading the list. Increasing the volume for these elements or adding acoustical markings recommends specific elements. In a graphical interface, the adaptation highlights elements in the list by adding graphical markings or selecting more noticeable colors.
- **Alternative Elements:** An adaptive interactive system supports the user by selecting the most appropriate among several alternatives for a certain interface element. For example, the system designer may provide different sets of speech output prompts in a speech interface, such as longer prompts for beginners and a short sound for expert users. In a graphical interface, the adaptation selects specific screens for beginners and experts. For instance, beginners could use a simple guided screen for destination input in a navigation system, whereas experts prefer a more complex screen.
- **Adaptive Help Presentation:** The interactive system explains features of the system by providing adaptive help. The help may address a feature the user is most likely to use next or that has not been used by the current user. For instance, a speech dialog system plays help prompts using speech output. A graphical interface shows a help message on the screen. In order to avoid user distraction, the adaptation may show an indication and lets the user open the help message.
- **Shortcut Area:** By providing a list of selections to the user in a separate "adaptation area", the user decides whether to use the shortcuts or not. Shortcuts can for instance be elements in a list or navigation shortcuts. In a graphical dialog system, a separate area of the screen may add shortcuts to user actions or frequently selected elements. (This adaptation is not applicable to speech-based dialog systems.)

Each of these patterns manifests itself in different ways. For instance, the "List Element Selection" adaptation emphasizes a graphical button or a part of the speech output. Therefore, a set of adaptation selectors was defined for each of these patterns to describe these manifestations. Since the adaptation selectors are defined in an abstract way based on the semantic layer, they may be reused between different interactive systems. In addition to the adaptation selectors, a set of default adaptation executors was defined. These generic executors only change properties that are common to all interactive systems. Custom adaptation executors may be added for a specific system, which support properties only available in that system. For instance, the background of a graphical element may be set to a specific value, although the background property is not present in all systems. Thus, adaptations are integrated seamlessly into interactive systems by defining custom adaptation executors.

5.3. *Executing adaptations*

After a set of adaptation selectors and executors were defined, the adaptation component applies them to an interactive system. In this section, we discuss how the adaptation component uses selectors and executors to perform adaptations and how the adaptation component decides about the application of adaptations. An adaptation selector is triggered through the event bus by the user modeling component. When the selector is activated, it first uses the ontology query to retrieve appropriate individuals for the adaptation. If the query returns elements, the selector recommends a certain adaptation for these candidates. The adaptation component then looks for an appropriate executor for the specific adaptation and interface element. If more than one appropriate executor is found, the adaptation component selects the one that is most appropriate for the user, considering for example previously applied adaptations and the experience of the user.

However, executing all possible adaptations would overwhelm the user and therefore, the adaptation component requires a strategy to decide which adaptations to apply. The framework provides two approaches for this decision: specification by the system designer and automatic decision by the adaptation component. In the first method for executing adaptations, system designer defines which adaptations should be applied. For this purpose, the system designer may enable adaptations at three different levels: globally, for an interface context, or for a specific interface element. If an adaptation is enabled globally, it is executed whenever possible. If an adaptation is enabled for a specific interface context (e.g. a graphical screen or a speech context), the adaptation is executed in this context, but not in others (unless it is defined for other contexts as well). Finally, if an adaptation is selected for a single interface element, it is executed on this element only, but not on others. In the second approach, the adaptation component automatically selects an appropriate combination of adaptations. The amount of adaptations is adjusted to the expertise of the user and more adaptations are activated in areas of the system (or tasks) that have not been used extensively by the user.

The remainder of this section demonstrates how the "Component Emphasis" adaptation introduced in Section 5.2 is executed in this framework. In a graphical system, each button executes a specific action. If the user modeling component predicts a user action, the "Component Emphasis" adaptation highlights the respective button to recommend it to the user. For this purpose, an adaptation selector reacts to an action prediction by the user modeling component by selecting all interface elements that execute the suggested action, in this case a graphical button. The adaptation selector also contains the information which abstract adaptation to use, in this case the "Button Emphasis" adaptation. The adaptation component then selects an adaptation executor for the "Button Emphasis" adaptation that is applicable to the specific button type used in the respective model. In this case, the adaptation executor highlights the graphical button by increasing its size and selecting more noticeable colors.

6. Evaluation

In order to show the feasibility of this framework, we carried out a user test. For this purpose, a reference implementation of this adaptation framework was developed to serve as a test bed. A separate adaptive interactive system was developed for each adaptation pattern and the systems were tested with users.

Evaluation frameworks for adaptive interactive systems recommend evaluating the individual parts separately, such as user modeling and adaptations. Brusilovsky et al. [6] and Paramythis et al. [27] argue in favor of a separate evaluation of the individual components of adaptive interactive systems. Paramythis and Weibelzahl [28] present a layered evaluation model for adaptive hypertext systems that decomposes the interactive system into different components. By evaluating the building block separately, the problems of the individual components become visible. If however an evaluation of the complete adaptive interactive system reveals problems, it is not possible to see if they were caused by the user modeling algorithms, the adaptations, or both. Furthermore, an adaptive interactive system might improve the user-system interactive, although problems in specific components persist. In addition, results from a separate evaluation of user modeling algorithms and adaptations are more general and may be transferred to other interactive systems, thus providing reusable evaluation evidence.

Therefore, this section presents an evaluation of the adaptation patterns from Section 5 independently of specific user modeling algorithms. The results from this evaluation may be transferred to other interactive systems that use the tested adaptations. The user modeling component presented in Section 3 represents a shell for specific user modeling algorithms, such as the algorithms introduced by Zukerman and Albrecht [38]. Since user modeling algorithms are more domain- and system-dependent than adaptations, an evaluation of user modeling algorithms is not included in this article.

6.1. Reference implementation

In order to show the technical feasibility of the framework, we created a reference implementation of the adaptation architecture as an extension of a modeling tool called EB GUIDE Studio [14], a model-based environment for developing multimodal interactive systems. The tool comprises a simulation component for executing the interactive system that serves as a dialog manager. The extension includes the components presented in this article, namely the user modeling component, the semantic layer, and the adaptation component.

The user modeling component describes the user-system interaction and provides information about user actions, such as detecting and predicting user actions. The user modeling component listens to events sent by the underlying interactive system and emits events based on the results of the user modeling procedure, which can be used by the adaptation component. The user modeling component also includes a user model for storing user-related data and computing inferences. Different user modeling algorithms were implemented to show the feasibility of this component. First, an action recognition algorithm employs probabilistic automata to detect user actions (see Section 3.1). Second, a task model tracks user behavior to predict actions or detect user errors (see Section 3.1). Third, a Markov chain-based algorithm is used to predict the next user action (see Section 3.2.2). An evaluation of the user modeling algorithms (see Bezold [4]) revealed that the average processing time of the user modeling framework per interaction event is less than 1 ms on recent hardware (Intel Core Duo CPU, 3 GHz). Thus, the integration of these different algorithms proves the feasibility of the user modeling component.

The semantic layer loads the ontology introduced in Section 4.2. As a foundation of the layer, a set of OWL classes was defined using the Protégé-OWL editor [20]. We created ontology connectors that instantiate the OWL classes and load data accordingly, for example from the model-based description provided by EB GUIDE Studio or from the user model. In doing so, the connectors form the semantic layer as described in Section 4.3. This layer is available both at design time and at runtime of the system. The semantic layer is implemented with the Jena framework[1], a Semantic Web library that supports OWL.

The adaptation component listens to triggers from the user modeling component and exploits information from the semantic layer. The adaptation patterns presented in Section 5.2 are integrated into the adaptation component. For this purpose, this component comprises a set of adaptation selectors for the adaptation patterns introduced in Section 5.2 and default adaptation executors. Moreover, the component implements the execution logic introduced in Section 5.3. In order to show the feasibility of the adaptation component, a user evaluation was conducted with different test systems that use the adaptation framework. In the remainder of this section, we introduce these test systems and present the evaluation results. Thus, we proved the technical feasibility of the framework with a reference implementation and by integrating example user modeling algorithms and the adaptation patterns into the framework.

The remainder of this paragraph summarizes the changes required to apply the adaptation framework to a previously non-adaptive interactive system to further illustrate the workings of the framework. What changes are required depends on the adaptations that should be used, since different adaptations have different requirements with regard to user modeling and semantic information. Therefore, not all of these steps have to be taken for every interactive system. With regard to user modeling, the interactive system has to emit events that can be observed by the user modeling component. In addition, a description of higher-level user behavior has to be prepared, for instance by means of a task model [29]. Next, semantic information has to be added to the model by means of annotation, such as whether an element is about help and is thus suitable for the "Adaptive Help Presentation" adaptation. Finally, adaptations are selected as introduced in Section 5.2, either by enabling adaptations for certain parts of the interactive system or by having the adaptation component decide about adaptations automatically.

6.2. Evaluation setup

A separate interactive test system was created for each of the adaptation patterns presented in Section 5.2. The test systems were developed in EB GUIDE Studio using the reference implementation of the adaptation framework. The test systems are automotive dashboard interfaces and a digital TV system.

Each of the interactive systems was tested by 20 users (16 for the "Alternative Elements" test system). All users tested both a non-adaptive and an adaptive version of the interactive systems in an alternating order. During the evaluation, the users fulfilled tasks that were handed to them on an instruction sheet and the users followed the individual steps closely. Since only

[1]The Jena Framework: http://jena.sourceforge.net

the adaptations are the focus of the evaluation, no actual user modeling was performed. Instead, the user modeling component recommended values according to the current task. In doing so, the results of the individual subjects could directly be compared. All systems were controlled with on-screen buttons, such as “OK”, “up”, “down”, etc. and the users clicked these buttons with a mouse.

The user sessions were recorded to extract objective measures, namely interaction time and number of interaction steps. For this purpose, the individual events from the user-system interaction were written to a file and an evaluation script extracted the respective measures automatically. In addition to objective measures, the attitude of users towards an interactive system plays an important role. Subjective measures were collected by means of questionnaires. The first part of the questionnaires addresses users’ general attitudes towards the interactive systems by means of the AttrakDiff questionnaire [17]. Users rate the attractiveness and usability by means of attribute pairs. However, some users had problems to create subtly graded ratings with these attribute pairs between the two conditions. The second part tests three hypotheses: adaptation confuses the user (H1), adaptation accelerates the interaction (H2), and adaptation supports the user (H3). A 7-item Likert scale with several questions each was used to investigate these hypotheses. The users filled in a questionnaire after each condition (adaptive and non-adaptive), which allows an assessment of the effects of the adaptations. In a final questionnaire, users were asked whether they preferred the adaptive or the non-adaptive version.

The aim of the evaluation is not only to investigate if adaptations improve the user-system interaction, but also to examine the conditions under which adaptations work well. Some adaptations do not show a significant improvement for all users, but only for a subgroup, such as beginners vs. experts or older users vs. younger users. In addition, the evaluation may reveal problems that should be avoided. Table 1 presents a summary of the evaluation results. In the following, the individual test systems and the results of the evaluations are discussed in detail.

6.3. Adaptation pattern: Adaptive help presentation

The “Adaptive Help Presentation” adaptation was tested using an interactive system that resembles a digital TV system with an electronic program guide. The user browses a list of TV shows and puts shows on a watch list. In addition, the user may change settings in a settings screen. The user controls the interactive system by means of an up and down button and three color buttons (red, green, and yellow). The meaning of the color buttons changes in every screen, with the respective functions being shown on the bottom of the screen. The adaptive version of the test system displays a yellow box with a concise help message in the upper right corner of the screen (see Fig. 8). The help messages relate to the current task. To improve the visibility of the help messages, the help box is faded in with a delay of 2 seconds.

The users performed different tasks in a given order: putting a show from the result list on the watch list (two times), removing an element from the schedul-

Table 1

An evaluation compared non-adaptive baseline versions to an adaptive versions of dialog systems, which implement adaptations defined by the design patterns in Section 5.2

	Subjective measures		Objective measures	
	Attractiveness	Usability	Interactions	Time
Adaptive Help Presentation	+	++	++	-
Component Emphasis	+	+	+	+
List Element Selection	++	+	++	++
Alternative Elements	+*	+	(n.a.)	(n.a.)
Shortcut Area	+	-	++	--

Legend: ++ = signif. better, + = better, - =worse, -- = signif. worse
*: For users younger than 46 years: ++

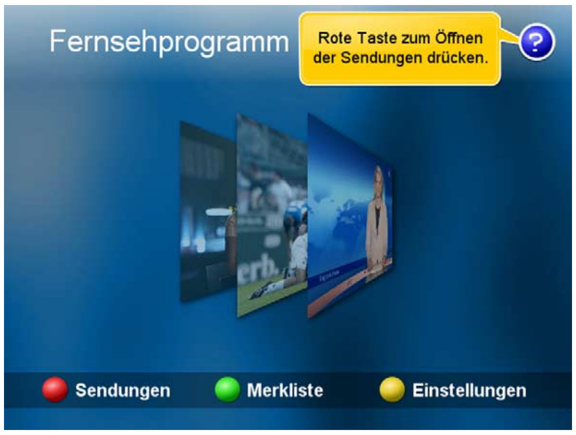

Fig. 8. Test system for the Adaptive Help Presentation adaptation. Help is provided in the upper right corner. The message reads: “Press red button to open list of shows.”

ing list (two times), changing settings (font size from regular to big), resetting the settings, and clearing the scheduling list. Similar tasks were used for the adaptive and the non-adaptive versions, but in a different order and with different values. However, the minimum number of interactions required to complete the tasks in both versions was constant.

Subjective Results

On average, the subjects rate the adaptive version as more attractive than the non-adaptive version, but not significantly. The usability of the adaptive version is assessed as significantly better ($p < 0.05$).

With regard to the three hypotheses, differences between expert users and non-expert users are observed. Whereas non-expert users regard the adaptation as not confusing, experts conceive the adaptation as neutral (H1, median value of 6.5 all users). Similarly, non-expert users believe that the adaptation accelerates the interaction, whereas experts see little influence of the adaptation on the performance (H2, median value of 2 for all users). Non-experts perceive the adaptation as supportive (H3, median value of 1.25 for all users). Expert users approve of this statement less. Therefore, non-expert users rate the adaptation significantly better than expert users. A majority of 13 out of 20 users prefer the adaptive version, five users prefer the non-adaptive version, and two users are undecided. However, the five users who prefer the non-adaptive version are expert users.

Objective Evaluation

As can be seen in Fig. 9, the adaptation significantly reduces the number of interactions ($p \leqslant 0.05$) compared to the non-adaptive version. Thus, users consult the adaptive help rather than using "trial and error" like in the non-adaptive version. However, the operating time of the adaptive version remains nearly constant compared to the non-adaptive version. We attribute this finding to a strong and significant ($p \leqslant 0.05$) learning effect between the first and the second version. Thus, regardless of whether the adaptive or the non-adaptive version is used first, the task is solved more quickly with the second version. In addition, the reading time of the help message affects the operating time of the adaptive version.

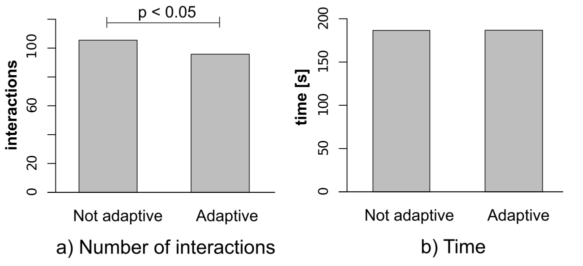

Fig. 9. Objective results of the "Adaptive Help" adaptation.

Discussion

The "Adaptive Help Presentation" adaptation allows users to reduce the number of interactions, but the interaction time remains constant. Beginners and advanced users regard the "Adaptive Help Presentation" adaptation as significantly more helpful and positive than expert users. Therefore, the adaptive help feature should only be enabled for beginners and advanced users and disabled for experts. A more detailed study, which eliminates the learning effect, is required to investigate the influence of adaptive help on the operating time.

6.4. Adaptation pattern: Component emphasis

The "Component Emphasis" adaptation draws a user's attention to specific interface elements through emphasis. The test system is part of a digital TV system, which allows the user to browse an electronic program guide (EPG). In order to cope with the vast number of shows, the user selects a number of search criteria, namely channel, time, date, and genre, to constrain the number of visible shows. After selecting a set of values, the user opens the result screen. The user controls the test system by means of up, down, "OK", and back buttons.

The adaptation inserts a yellow arrow on the left-hand side of the button that the user is supposed to select next according to the task (see Fig. 10). For each version, the user had to execute four tasks that each consist of selecting three different filter criteria and opening the result screen. For instance, one task instructed the user to select "BBC" for channel, "Friday" for day, and "documentary" for genre. The number of actions required for each task was constant.

Subjective Evaluation

Only a small improvement is observed with regard to attractiveness and usability. However, the adaptive version is judged very positively: The idea of the system as being confusing is rejected (H1, median value of 6.75). Users agree strongly with the notion that the adaptation decreases the interaction time (H2, median value of 1.00). Moreover, users agree that the adaptation supports them (H3, median value of 1.00). 70 % of the users (14 out of 20) prefer the adaptive ver-

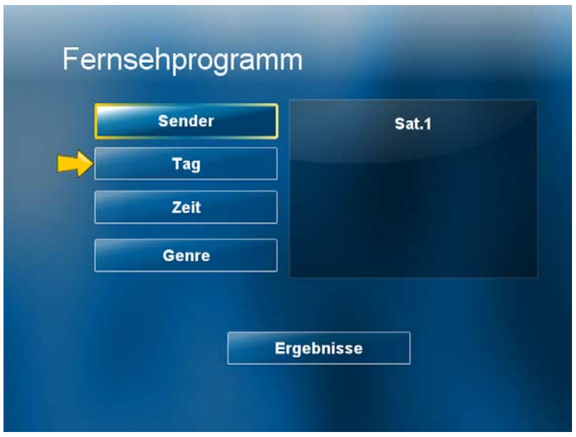

Fig. 10. Test system for the Component Emphasis adaptation. A lighter arrow indicates the recommended next button.

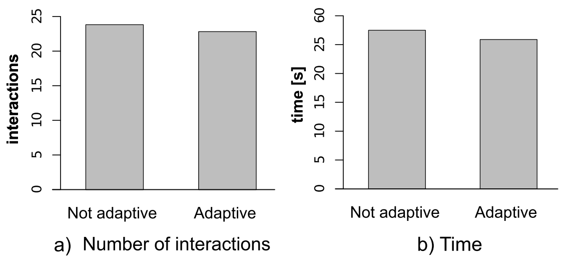

Fig. 11. Objective results of the "Component Emphasis" adaptation.

sion, 20 % (4 users) prefer the non-adaptive version, and 10 % (2 users) are undecided.

Objective Evaluation

Figure 11 presents the results of the object evaluation. Users performed four subtasks for the adaptive and the non-adaptive version. Both the average time and the number of actions improves with the adaptive version, but not significantly. However, the operating time as well as the number of actions decreased for 71 % of the users.

Discussion

Although the adaptation does not improve the perceived attractiveness and usability significantly, the users strongly rate the adaptation as not confusing, faster, and supportive. The number of average interactions and the operating time both decrease. The adaptation improves the objective measures for a majority of 71 % of the users.

6.5. *Adaptation pattern: List element selection*

The "List Element Selection" adaptation, which highlights frequently selected items in a list, was investigated with an interactive system that resembles an automotive dashboard interface. The system comprises an address selection menu with a list of 65 names in alphabetic order. A scrollbar on the left hand side indicates the current position in the list. The user presses up and down buttons to move the cursor and an "OK" button to select an item. A red rectangle and a yellow star highlight a list element in the adaptive version (see Fig. 12). In this test, the highlight corresponds to the next item in the task. In addition, a small yellow star indicates the position of the highlighted item in the scrollbar. The tasks instructed the subjects to select seven items from the list of names.

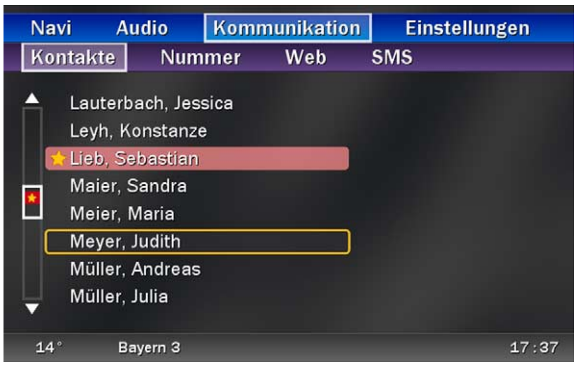

Fig. 12. Test system for the List Element Selection adaptation. A lighter background indicates the recommended next lst entry.

Subjective Evaluation

Users rate the attractiveness of the adaptive version as significantly better ($p \leqslant 0.05$) and the usability better, but without significance. With regard to the hypotheses, the adaptation is not perceived as confusing (H1, rejection with a median value of 6.50), but as faster (H2, agreement with median value of 1.00) and supportive (H3, agreement with a median value of 1.50). 19 out of 20 subjects (95 %) prefer the adaptive version.

Objective Evaluation

The results of the objective evaluation are presented in Fig. 13. Both the number of interactions ($p \leqslant 0.01$) and the total interaction time decreased significantly ($p \leqslant 0.05$).

Discussion

The "List Element Selection" proved to be a successful adaptation, which was significantly faster than the non-adaptive version. In addition, the users rated the adaptation positively in the questionnaire.

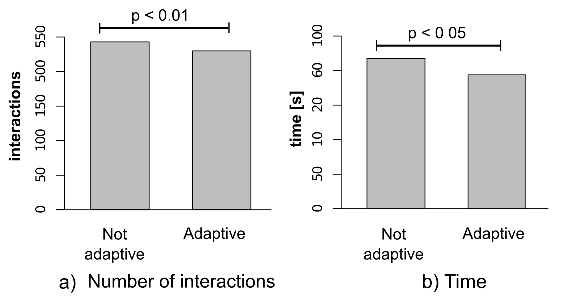

Fig. 13. Objective results of the “List Element Selection” adaptation.

6.6. Adaptation pattern: Alternative elements

The task of navigation destination entry in an automotive dashboard interface was selected for the evaluation of the “Alternative Elements” adaptation. Destination entry comprises the selection of city, street, and house number. The users selected each value from a list of 50 entries. Two slightly different interfaces were created for this system: The “expert mode” leaves more flexibility to the user by offering a menu in which the user can choose the selection order (see Fig. 14). The “beginner mode” enacts a fixed selection order of city first, street next, and house number last. Whereas both systems offer the same functionality, the level of control is reduced in the beginner mode. The system is controlled by up and down buttons for scrolling, an “OK” button for selection, and a back button. The adaptation automatically selects the appropriate version. The beginner mode was selected at the beginning of each session and switched to the expert mode after the user entered two destinations. A message was shown to indicate the change.

Since an investigation of the objective measures would compare the beginner to the expert mode rather than provide metrics of the adaptation (e.g. with regard to the number of interactions), the objective investigation was omitted for this adaptation.

Subjective Evaluation

Before filling in the questionnaire, the users were instructed not to rate the beginner or the expert mode, but the ability of the system to switch between the two modes. Both perceived attractiveness and usability increase for the adaptive version, but not significantly. Differences appear between the groups of users below and above the age of 46 years. For the group of users younger than 46 years, the improvement of the attractiveness becomes significant ($p < 0.05$). With regard to the three hypotheses, users disagree with the statement that the adaptation was confusing, but not strongly (H1, median value of 5.25) and do not regard the adaptation as faster than the non-adaptive version (H2, median value of 4.25). However, they agree that the system is supportive (H3, median value of 3.00). In the whole group, 11 out of 20 users (55 %) prefer the adaptive version over the non-adaptive version, 7 prefer the non-adaptive version, and 2 are undecided. However, in the group of subjects aged under 46 years, 10 out of 15 subjects (66 %) prefer the adaptive version, 3 prefer the non-adaptive version, and 2 are undecided.

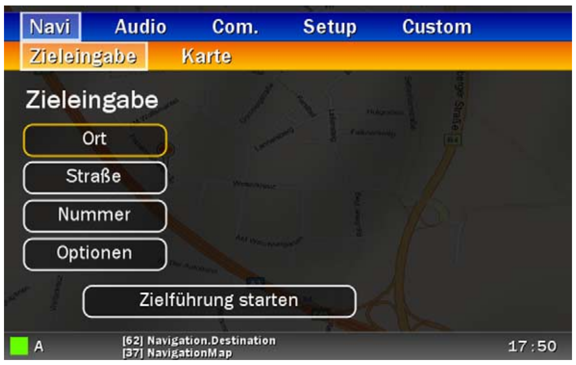

Fig. 14. Test system for the Alternative Elements adaptation. The main screen of the expert version is given.

Discussion

These findings indicate that users older than 45 years prefer a static interface, whereas younger users like the idea of a system that selects the most appropriate version for them. Some older users also stated that once they had learned to use an interface, they did not want to be forced to learn a different one. Younger users on the other hand like the idea of having a dynamic interface better. Interface changes should be communicated to the user, for instance by means of message boxes, in order to avoid confusion.

6.7. Adaptation pattern: Shortcut area

The “Shortcut Area” adaptation, which supports the user by offering frequently used items in a separate area of the interface, was evaluated with a system that resembles an automotive dashboard interface. The interface provides access to a “Music” menu, a “Contacts” menu, and a “Climate Control” menu. The system is controlled by clicking the buttons on the screen with a mouse. The adaptation predicts a sequence of user actions using an algorithm derived from Mannila et al. [23] and offers shortcut buttons on the screen (see

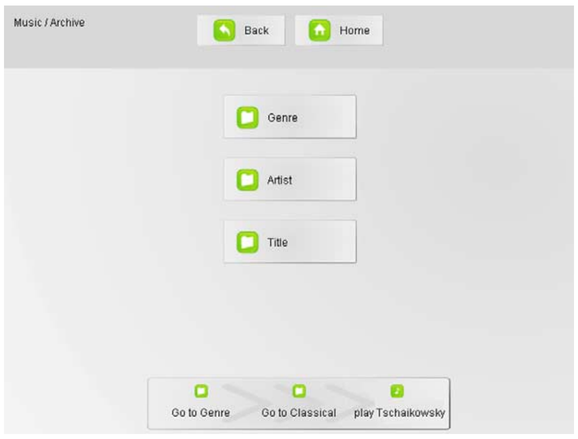

Fig. 15. Test system for the List Element Selection adaptation. Navigation shortcuts are provided on the bottom of the screen.

Fig. 15). The users clicked the buttons to execute the whole action sequence or a subsequence of it.

Users were given a number of repeating tasks, such as selecting a certain song from the "Music" menu or changing the climate settings. The users trained the algorithm first by executing four tasks from the assignment sheet. Each task consisted of a sequence of actions the users had to follow closely. After the training, the users executed each of the previously trained tasks twice in a given order. The decision whether to use the adaptations was left to the user's discretion.

This test system was not implemented using this adaptation framework, since the framework had not been finished when the evaluation was conducted. However, both the user modeling algorithm and the adaptations have been made part of the framework since then. A different user group than in the previously presented tests performed this evaluation.

Subjective Evaluation

Users regard the attractiveness of the adaptive version as significantly better ($p < 0.05$). While the mean value of the usability rating of the adaptive version is slightly better than the value of the non-adaptive version, some users rate the usability of the adaptive version as worse (no significance).

With regard to the hypotheses, the subjects do not feel confused, but do not reject this notion strongly (H1, rejection with a median value of 4.88). The users believe the adaptation lets them use the system faster (H2, agreement with a median value of 2.25) and feel supported by the adaptation (H3, agreement with a median value of 2.00). When asked which system they preferred, 10 users named the adaptive version, 2 subjects selected the non-adaptive version, and 4 persons did not select a version.

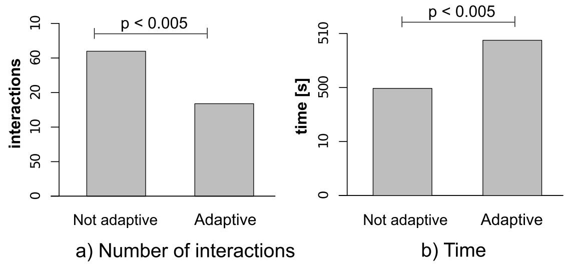

Fig. 16. Objective results of the "Shortcut Area" adaptation.

Objective Evaluation

With regard to the objective measures (see Fig. 16), users reduce the overall number of clicks significantly ($p < 0.001$) with the adaptive version. Therefore, users employ the adaptive shortcuts, although they were not forced to do so. However, the interaction time inversely increases significantly ($p < 0.001$), because the subjects compared the items in the adaptation area with the next steps on the worksheet. Thus, users employ the adaptive version by choice, but with a negative impact on the operating time. An evaluation with different tasks or a test with real world problems may reveal if this adaptation successfully reduces the interaction time as well.

Discussion

The evaluation of the "Shortcut Area" adaptation shows that subjects employ adaptive shortcuts by choice. A majority of the subjects prefer the adaptive version. The adaptation allows users to reduce the number of interaction steps significantly. However, the operating time increases due to the evaluation setup. Further evaluations therefore have to investigate if a different test setup leads to a reduced operating time.

7. Conclusion

In this article, we have presented adaptation as a solution for coping with increasing complexity of user interfaces in smart environments, such as interactive TV systems in the living room or infotainment systems in cars. For this purpose, a framework for adapting interactive systems to user behavior has been presented. The framework comprises both a component for user modeling and executing adaptations. User modeling is based on an observation of basic events and consists of two parts: extracting meaningful information from the stream of basic events and deriving new information. Different user modeling algorithms may be implemented within the framework. The user modeling results finally trigger the adaptation component.

A semantic representation of the interactive system as well as other information required for adaptation forms the foundation of the adaptation component. The semantic layer uses an OWL-based ontology, which is populated by means of different ontology connectors. These connectors load information about the interactive system into the ontology. The description of adaptations consists of system-independent adaptation selectors and system-dependent adaptation executors. Therefore, the adaptation logic may be reused between different systems, whereas the concrete execution of adaptations may be defined on a per-system basis. A set of adaptation patterns was defined to ease the integration of adaptive features into interactive systems. We showed the feasibility of the framework by creating a reference implementation of the architecture. This implementation includes all components of the architecture and different test systems prove the viability of the approach. A user test was carried out to investigate the adaptations. For this purpose, a separate test system for each of the five adaptations was created and an evaluation was performed with 20 test users. The results show that the framework allows the development of diverse adaptive interactive systems. The adaptations improve the user-system interaction, although some adaptations are successful only for a part of the users, such as beginners or younger users.

References

[1] S. Amershi and C. Conati. Unsupervised and Supervised Machine Learning in User Modeling for Intelligent Learning Environments. In *Intelligent User Interfaces (IUI) 2007*, pages 72–81. ACM, New York, NY, USA, 2007.

[2] A. Aragones, J. Bruno, A. Crapo, and M. Garbiras. An ontology-based architecture for adaptive work-centered user interface technology. In *2006 Jena User Conference*. Available online: http://labs.ingenta.com/2006/05/juc/, retrieved 2010-06-17, 2006.

[3] L. Ardissono, C. Gena, P. Torasso, F. Bellifemine, A. Difino, and B. Negro. *Personalized Digital Television*, volume 6 of *Human-Computer Interaction Series*, chapter User Modeling and Recommendation Techniques for Personalized Electronic Program Guides, pages 3–26. Springer, Dordrecht, The Netherlands, 2004.

[4] M. Bezold. Describing user interactions in adaptive interactive systems. In *First and Seventeenth International Conference on User Modeling, Adaptation, and Personalization (UMAP)*, volume 5535 of *LNCS*, pages 150–161. Springer, Heidelberg, Germany, 2009.

[5] M. Bezold. Towards formalized adaptation patterns for adaptive interactive systems. In *EuroPLoP 2009*, volume 566 of *CEUR (available from http://ceur-ws.org)*, pages B4–1–B4–18, 2009.

[6] P. Brusilovsky, C. Karagianidis, and D. Sampson. The Benefits of Layered Evaluation of Adaptive Applications and Services. In *Workshop on Empirical Evaluation of Adaptive Systems (EASy) 2001*, pages 1–8. Pedagogical University of Freiburg, Freiburg, Germany, 2001.

[7] D. D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, 1991.

[8] A. Dix, J. Finlay, and R. Beale. Analysis of User Behaviour as Time Series. In *Conference on People and Computers (HCI) 1992*, pages 429–444. Cambridge University Press, Cambridge, UK, 1993.

[9] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Towards the Adaptive Semantic Web. In *PPSWR 2003*, volume 2901 of *LNCS*, pages 51–68. Springer, Heidelberg, Germany, 2003.

[10] Europäisches Kommittee für Normung. Ergonomie der Mensch-System-Interaktion: Teil 110 Grundsätze der Dialoggestaltung, 2006.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley, Upper Saddle River, NJ, USA, 1995.

[12] G.P. Garcia, J.G. de Amores Carredano, P.M. Portillo, F.G. Marin, and J.G. Marti. Integrating Owl Ontologies With a Dialogue Manager. In *Procesamiento del Lenguaje (ISSN:1135-5948)*, pages 153–160, 2006.

[13] M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis. Creating an Ontology for the User Profile: Method and Applications. In *Conference on Research Challenges in Information Science (RCIS) 2007*. EMSI, Casablanca, Morocco, 2007.

[14] S. Goronzy, R. Mochales, and N. Beringer. Developing Speech Dialogs for Multimodal HMIs Using Finite State Machines. In *9th International Conference on Spoken Language Processing (Interspeech), CD-ROM*, 2006.

[15] T.R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, 43(5–6):907–928, 1995.

[16] I. Gurevych, R. Porzel, and R. Malaka. Modeling Domain Knowledge: Know-How and Know-What. In *SmartKom – Foundations of Multimodal Dialogue Systems*, pages 71–84. Springer, Heidelberg, Germany, 2006.

[17] M. Hassenzahl, M. Burmester, and F. Koller. Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In *Mensch & Computer 2003. Interaktion in Bewegung*, pages 187–196. B.G. Teubner, Stuttgart, Leipzig, 2003.

[18] D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff. Gumo – The General User Model Ontology. In *User Modeling (UM) 2005*, volume 3538 of *LNCS*, pages 428–432. Springer, Heidelberg, Germany, 2005.

[19] A. Jameson. *Human-computer Interaction Handbook*, chapter Adaptive Interfaces and Agents, pages 305–330. Erlbaum, Mahwah, NJ, USA, first edition, 2003.

[20] H. Knublauch, R.W. Fergerson, N.F. Noy, and M.A. Musen. The Protege OWL Plugin: An Open Development Environ-

ment for Semantic Web Applications. In *International Semantic Web Conference (ISWC) 2004*, volume 3298 of *LNCS*. Springer Verlag, Berlin/Heidelberg, Germany, 2004.

[21] A. Kobsa and W. Pohl. The User Modeling Shell System BGP-MS. *User Modeling and User-Adapted Interaction (UMUAI)*, 4(2):59–106, 1994.

[22] V. Lopez-Jaquero, J. Vanderdonckt, F. Montero, and P. Gonzalez. Towards an extended model of user interface adaptation: the isatine framework. In *Engineering Interactive Systems (EIS) 2007*. Springer, Berlin, Germany, 2007.

[23] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[24] N.F. Noy and D.L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, 2001.

[25] Z. Obrenović, D. Starĉević, and V. Devedẑić. Using Ontologies in Design of Multimodal User Interfaces. In *INTERACT 2003*. IOS Press, Amsterdam, Netherlands, 2003.

[26] J. Orwant. Heterogeneous Learning in the Doppelgänger User Modeling System. *User Modeling and User-Adapted Interaction (UMUAI)*, 4(2):107–130, 1994.

[27] A. Paramythis, A. Totter, and C. Stephanidis. A Modular Approach to the Evaluation of Adaptive User Interfaces. In S. Weibelzahl, D.N. Chin, and G. Weber, editors, *Workshop on the Empirical Evaluation of Adaptive Systems, held at User Model (UM) 2001*, pages 9–24, 2001. P.

[28] A. Paramythis and S. Weibelzahl. A Decomposition Model for the Layered Evaluation of Interactive Adaptive Systems. In *User Modeling (UM) 2005*, volume 3538 of *LNCS*, pages 438–442. Springer, Heidelberg, Germany, 2005.

[29] F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *INTERACT 1997*, pages 362–369. Chapman & Hall, London, UK, 1997.

[30] L. Razmerita, A.A. Angehrn, and A. Maedche. Ontology-Based User Modeling for Knowledge Management Systems. In *User Modeling (UM) 2003*, volume 2702 of *LNCS*, pages 213–217. Springer, Heidelberg, Germany, 2003.

[31] R. Sarukkai. Link Prediction and Path Analysis Using Markov Chains. *Computer Networks*, 33(1-6):377–386, 2000.

[32] D.C. Schmidt. Model-Driven Engineering. *IEEE Computer*, 30(2):25–31, 2006.

[33] M.K. Smith, C. Welty, and D.L. McGuinness. OWL Web Ontology Language Guide. Technical report, W3C, 2004.

[34] J. Tidwell. *Designing Interfaces*. O'Reilly, Sebastopol, CA, USA, 2005.

[35] T. Tran, P. Cimiano, and A. Ankolekar. A Rule-Based Adaption Model for Ontology-Based Personalization. In *Advances in Semantic Media Adaptation and Personalization*, volume 93 of *SCI*, pages 117–135. Springer, Heidelberg, Germany, 2008.

[36] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R.C. Carrasco. Probabilistic Finite-State Machines-Part I. In *Pattern Analysis and Machine Intelligence*, volume 27, pages 1013–1025. IEEE Computer Society, Washington, DC, USA, 2005.

[37] M. Zancanaro, T. Kuflik, Z. Boger, D. Goren-Bar, and D. Goldwasser. Analyzing museum visitors' behavior patterns. In *User Modeling (UM) 2007*, volume 4511 of *LNCS*, pages 238–246. Springer, Heidelberg, Germany, 2007.

[38] I. Zukerman and D.W. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction (UMUAI)*, 11(1-2):5–18, 2001.