A generic engine for interface adaptation

P.T. de Vrieze P. van Bommel Th.P. van der Weide

27th February 2004

1 Introduction

The area of adaptive systems is rapidly gaining scientific interest. Most research seeks to enhance human computer interaction by adapting the system to the user. This topic has already gained a lot of attention by various authors, see: [1], [2], [3], [4], [7]. System adaptation involves the use of incremental behaviour analysis for acquiring user models. It also involves adaptation of the system behaviour to the user model.

A main part of adaptive systems is the user model. However such a user model does not appear out of thin air. The model needs to be first initialised and then incrementally updated based on the user's actions. After that the user model needs to be used for system adaptation.

Here we distinguish three different functions: user model initialisation,

user model updating, and system adaptation. In figure 1 these three roles are depicted. These three roles together are fullfilled by an *adaptation engine*.

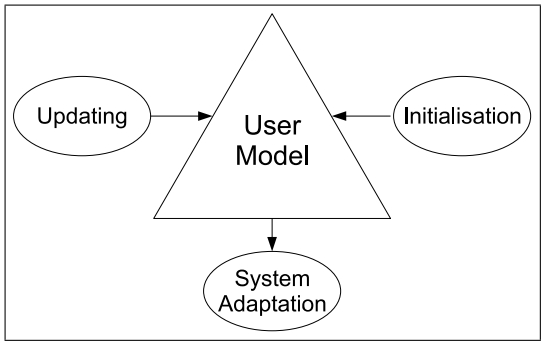


Figure 1: The three roles involved with the user model

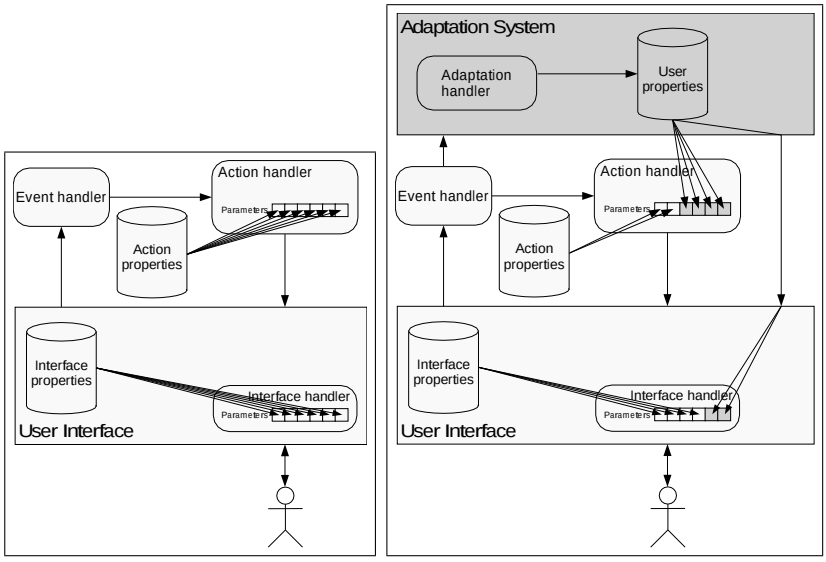
How this is done is described by an adaptation model.

In previous work [5, 6] we have gone into the issues involved in the development of adaptation models. Further we have described a framework for both classsification of and as a help in designing adaptation models.

In this paper we will describe the issues involved with such adaptation, how those issues reflect on a prototype we have developed, and the prototype itself.

2 Main concepts

A user modeling system is a system that shows adaptive behaviour concerning its interaction with the user . For explaining the difference between conventional systems, i.e interactive systems that do not employ user mod-



(a) A model of a normal interactive system

(b) A model of an interactive system with user modeling

Figure 2: Comparison of normal and user modeling systems

eling, (see figure 2(a)) and user modeling systems (see figure 2(b)) we first need to describe conventional systems in a suitable way. Then we need to describe user modeling systems, and compare them. In the next two sections we will describe both conventional and user modeling systems.

Conventional interactive systems (see figure 2(a)) can be seen as state machines that interact with a user. This interaction his handled by a user interface. Each user action can induce a state change, after which new user actions are possible.

In user modelling systems these user actions are also fed to the adaptation

system which, in some way, uses them to update the user model. This user model is then used for system adaptation.

In most cases one cannot perform the adaptation by directly storing events and retrieving them from the user model without any processing. In short we need processing that employs some kind of learning algorithm. There is however a choice where in the adaptation process the processing happens.

Concerning the location of the processing we can distinguish push, pull and hybrid strategies [5, 6]. A push strategy performs most processing before changing user model attributes. A pull model performs most processing at the point where information is needed about the user. A hybrid model tries to get on the middle ground in this. The choice of strategy is basically made when an adaptation model is created.

User adaptation is most effective when much information about the user can be gathered. The way most information can be gathered is to have many applications that support user modelling. To enable this it is important to to make it simple to integrate user modelling in these applications. The adaptation engine serves this purpose in providing a standard interface for user modelling.

3 Adaptation engine implementation

In our adaptation engine we have aimed to provide an infrastructure for easy creation and maintenance of adaptation models and the associated user models. The engine provides the infrastructure where there are mainly two functions that act as interface: `postEvent` and `askQuestion`.

The `postEvent` function is called by the system to pass events to the adaptation engine. The allowed events are described in the adaptation model that needs to be given as an argument when the adaptation engine is created.

Basically the event passing will initiate a rule evaluation procedure. This evaluation procedure retrieves the rules related to this event from the adaptation model. Checks whether their conditions apply and in case they do, they will be executed. The rule execution normally results into an update of the user model.

The `askQuestion` function can be called when a system wants to know about a property of the user. The possible questions are again defined in the adaptation model. The asking of a function is similar to calling a function in a functional programming language. A most minimal question will immediately return the value of an attribute in the user model. A more elaborate question will however use other intermediate questions and some calculation.

4 Adaptation engine description

To evaluate the properties of different styles of user modeling and adaptation we have implemented a prototype. This prototype provides an adaptation engine. This engine is only weakly linked with other parts of the system.

The main interfaces of the engine are formed by two functions: `postEvent` and `askQuestion`. The `postEvent` function is used to notify the adaptation engine of the events that happened in the application. These events in result in an update of the user model.

The `askQuestion` function is used by the application to ask information about the user. The answers to these questions are directly useable by the system for concrete decisions.

What is further required for the system is an adaptation model. This adaptation model describes the transformations to the persistent user model based on an event. Further it describes the persistent user model structure. Finally it describes the functions that calculate the answers to questions based on the persistent user model.

In our system questions are not allowed to have sideeffects. i.e. Questions are not allowed to update the user model. This makes it safe for systems to do multiple requests for the same information. It also implies that answers are stable given a user model. This allows for the caching of answers (not

implemented).

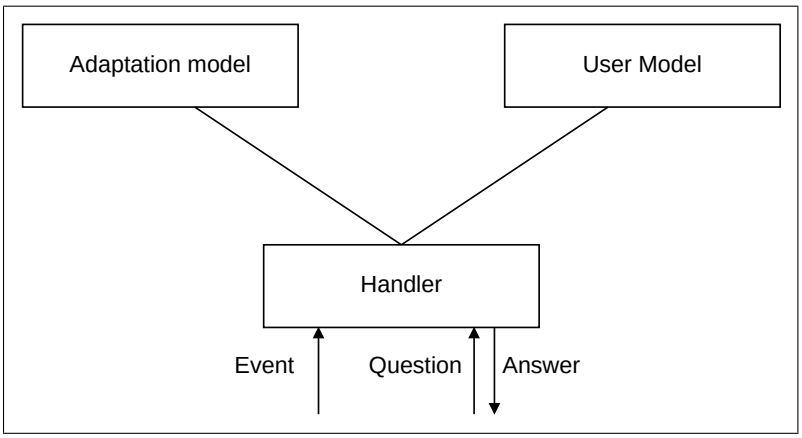


Figure 3: Basic architecture of the prototype

Besides the actual engine we have also implemented two applications. The first application is an adaptation model viewer (figure 4). This viewer can be used for simulating the functioning of an adaptation model. It allows an adaptation model to be loaded and events to be generated. It then shows both the persistent user model and the answers to all questions.

The second application is an adaptation model editor (figure 5). This editor allows the adaptation model to be edited more conveniently than by editing the xml source. Editing the source xml files is especially cumbersome because the adaptation model scripts are in xml format and as such need both xml escaping and explicit line ends. With the editor this is automatically taken care of.

Further the editor includes the possibility to perform a life test on the

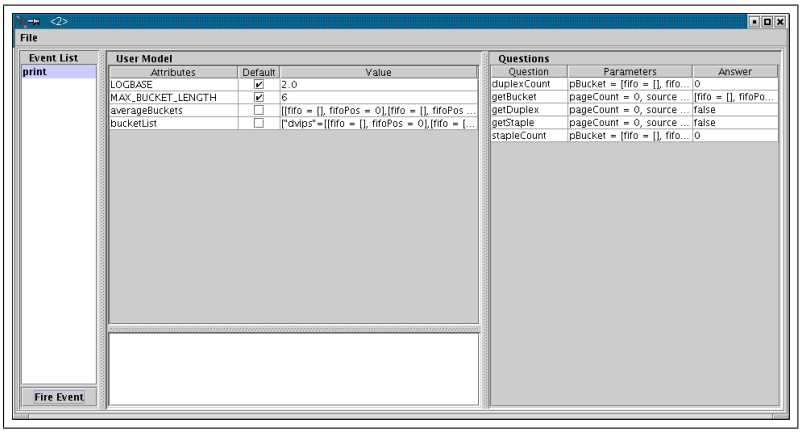


Figure 4: The adaptation model viewer

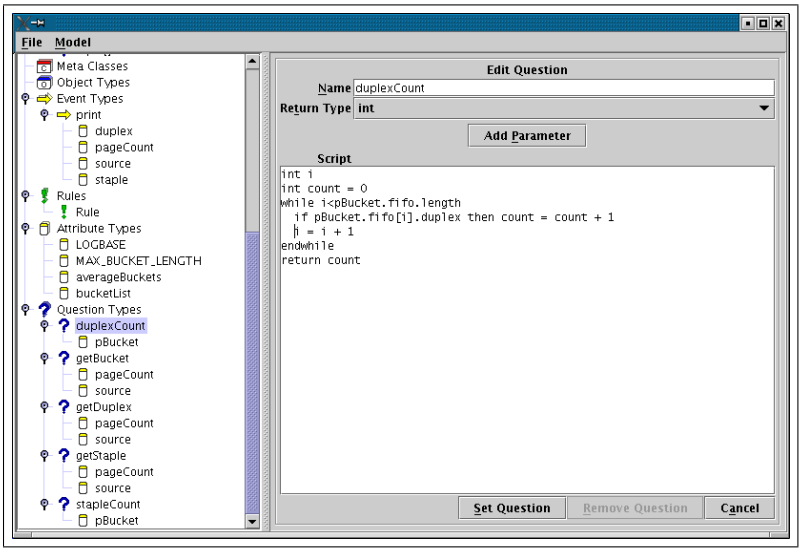


Figure 5: The adaptation model editor

adaptation model being edited, by invoking the adaptation model viewer on the current model.

5 Conclusion

References

[1] Johan Aberg and Nahid Shahmehri. User modelling as an aid for human web assistants. In M. Bauer, P.J. Gmytrasiewicz, and J. Vassileva, editors, *User Modeling: Proceedings of the Eight International Conference, UM01*, pages 201–203, Sonthofen, Germany, July 2001. Springer.

[2] ACM. The adaptive web. *Special Issue of Comm. o.t. ACM*, 45(5), May 2002.

[3] Brad Campbell and Joseph M. Goodman. Ham: a general-purpose hypertext abstract machine. In *Proceeding of the ACM conference on Hypertext*, pages 21–32, Chapel Hill, North Carolina, US, 1987. ACM Press.

[4] David N. Chin. Strategies for expressing concise, helpful answers. *Artificial intelligence review*, 14(4-5):333–350, October 2000.

[5] Paul de Vrieze, Patrick van Bommel, Jakob Klok, and Theo van der Weide. Towards a two-dimensional framework for user models. In *Proceedings of the MAWIS03 workshop attached to the OOIS03 conference*, Geneva, 09 2003.

[6] Paul de Vrieze, Patrick van Bommel, Jakob Klok, and Theo van der Weide. Adaptation in multimedia systems. *Multimedia Tools and Applications*, 2004. to appear.

[7] Josef Fink and Alfred Kobsa. User modeling for personalized city tours. *Artificial intelligence review*, 18(1):33–74, September 2002.