Towards a Two-Dimensional Framework for User Models

P.t. de Vrieze[1], P. van Bommel[1], J. Klok[2], and Th. van der Weide[1]

[1] University of Nijmegen
[2] Océ Research & Development

Abstract. The focus if this paper is user modeling in the context of personalization of information systems. Such a personalization is essential to give users the feeling that the system is easily accessible. The way this adaptive personalization works is very dependent on the adaptation model that is chosen.
We introduce a generic two-dimensional classification framework for user modeling systems. This enables us to clarify existing as well as new applications in the area of user modeling. In order to illustrate our framework we evaluate push and pull based user modeling.

1 Introduction

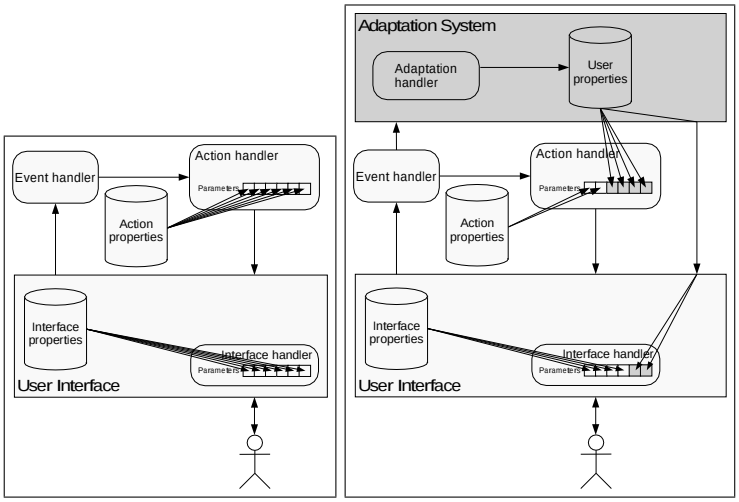
The research area of user modeling seeks to enhance human computer interaction by adapting the system to the user. This topic has already gained attention by various authors, see: [1], [2], [3], [4]. User modeling involves the use of incremental behaviour analysis for acquiring user models. It also involves adaptation of the system behaviour to the user model. For a background on system adaptation we refer to: [3], [5], [6].

The key part of a user modeling system is the user model. In order to know what a user model should look like it is necessary to know the adaptation methods that are going to be employed. The methods that do this are described in the adaptation model. This is a general model that describes how the user models need to be created, maintained and used.

We distinguish two kinds of adaptation models: a push adaptation model and a pull adaptation model. Those models are based on the direction of inference in the system. Further it is possible to combine both models into a hybrid adaptation model that combines aspects of both models. An example of a hybrid system can be found in.

While publications have described the use of both kinds of models and combinations of them, they have not explicitly evaluated the advantages and disadvantages of those models. We believe that this is important to be able to design user modeling systems better.

In this paper we analyse the differences between the push and pull adaptation models. For that it is important to first define what a user modeling system actually is, and which parts of a system can be seen as a part of the adaptation



(a) A model of a normal interactive system

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

Fig. 1. Comparison of normal and user modeling systems

system. For that reason we give an overview of user modeling systems in section 2. After that we will introduce a list of demands that user modeling system should satisfy. This list is then used in sections 5, 6, and 7 to evaluate the push, pull and hybrid adaptation models. Finally, in section 8 we will evaluate our framework and state possible points of further research.

2 Overview of User Modeling Systems

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 modeling, (see figure 1(a)) and user modeling systems (see figure 1(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 1(a)) can be seen as state machines that interacts 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 designing a user interface serveral choices have to be made concerning the looks and behaviour of the interface. Many of these choices are implicit or given by default choices from guidelines. For the sake of being able to compare

a conventional system with a user modeling system we asume that the choices are explicit. We call those choices interface properties.The interface properties determine both the behaviour and looks of the user interface.

In a conventional system user actions induce events. These events trigger system actions and interface changes. These actions and interface changes can differ based on the interface properties

In a system based on user modeling (see figure 1(b)), the behaviour of the various handlers may be affected by user properties in addition to the handler specific properties. See e.g. [4] and [7] for systems that show such a change of behaviour. Those user properties are supplied by the adaptation system. The user properties can be seen as questions asked by the system about a specific user property. As the adaptation system can be seen as the authority on the user, the questions should be in such a way that all inferrence happens inside the adaptation system.

As a consequence of the user properties influencing the handlers the user interface now takes into account the user model as its behaviour is determined by the user interface handler. The same goes for the action handler.

The user properties are provided by the adaptation handler. The adaptation handler generates these properties based on events fed to it by the event handler. The main point of user modeling is about how to go from these events to the user properties.

3 Further Analysis of User Modeling Systems

To evaluate user modeling systems it is very useful to have a clear method for comparing them. For this purpose we have developed a two-dimensional classification framework. Our framework looks at all kinds of user modeling systems and is not made by classification of existing systems. In this it differs significantly from the framework in [8].

Figure 2 presents the proposed framework. Along the horizontal axis is the inference process. It goes from the event model to the user model, and from the user model to the system concept model. The event model consists of the actual events generated by the system. The user model of the most system independent user properties, and the system concept model consists of all the user questions that can be asked by the system.

For certain user properties many derivation steps are necessary, and for others only a few. Because of this reason we model the progress in that process, not the steps. Further we define the model that is least system specific to be in the middle. For that reason all systems will have their highest point in the middle.

On the vertical axis we model system independence. At the start of the adaptation process, there are events generated by the system. These events are maximally system dependent. An example of such an event could be: “The user fills box 123 with a purple background”. We call the model here the *event model*.

For adaptation purposes the events generated by the system are not that relevant. An adaptation system wants to use specific cases to infer knowledge of

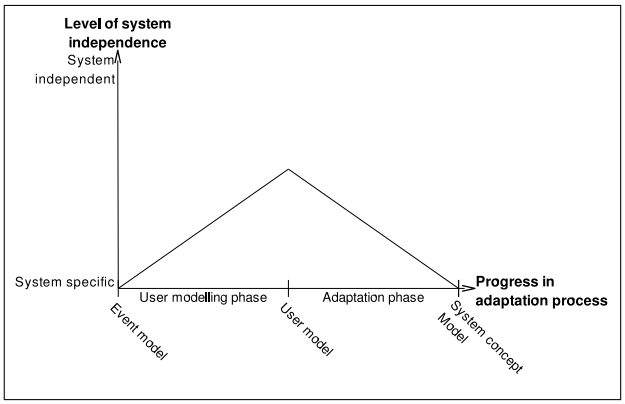


Fig. 2. A two-dimensional user modeling classification framework

the general case. This inference process goes in a number of steps. At one point a model is inferred that is most general. An example of knowledge that can be inferred here is: "The users favourite colour is purple". This is part of what we call the *user model*.

At a point where the user model is known, the system needs to know how this model fits into the questions a user modeling system might have. A user modeling system wants to know the answer on a question like: "What background color should a new box have?". In the adaptation phase of the system, the adaptation system will try to get system dependent answers based on the general knowledge from the user model. The model of answers to system questions is called the *system concept model*. The system concept model is where the user properties live.

We can use the framework of figure 2 to determine two properties of systems. Firstly, we can look at the hight of the triangle to determine how system specific an adaptation system is. For example in figure 3 we see the systems S2 and S4. S2 is more system independent than S4. This could mean that S2 can be more easily be extended to provide more or different adaptation. The second property we can distinguish is, where in the inference process a persistent model is stored. This is an important measure as the process is different before and after storage. Before storage a push process needs to be used to create the model. Push here means that the arrival of an event generates a waterfall of subsequent events that lead to updating the persistent model. We call this push adaptation. We will discuss the advantages and disadvantages of push based systems in section 5.

After storage we need to use a pull strategy to perform adaptation. This starts with the system requesting the value of a certain property from the adaptation system. For determining the value of this property the adaptation system might want to use the values of other properties that might also need to be calculated. This goes on until the persistent model is used. We call this pull adaptation.

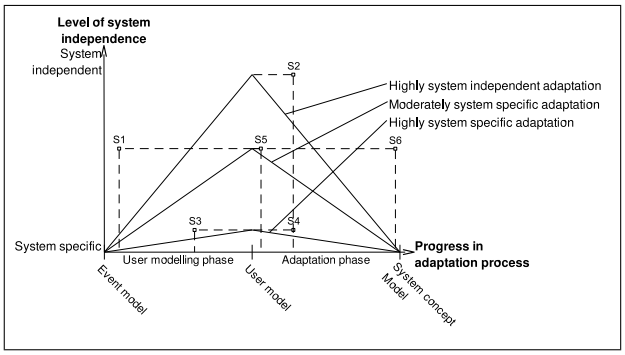


Fig. 3. Use of the two-dimensional classification framework

As an example of the use of the framework we look at figure 3. In figure 3 there are six systems with all different properties. System S1 is almost a purely pull-based system, as it's persistent model is created very early on in the inference process, while S5 is can be classified as a hybrid system and S6 is a rule-based system. The other systems are all different kinds of hybrid systems. Note that S5 is almost in the middle, but a system completely in the middle would be rather unrealistic.

Based on the locations of the systems in figure 3 we can say things about the systems, and especially their relations with eachother. As an example looking at systems S3 and S5 we can say that system S3 has a bias on pull modeling compared to S5 and that S3 is more system dependent than S5. This can be used to say things about these systems like: "the persistent model of S3 is probably relatively bigger than the persistent model of S5", "It is probably more easy to extend the adaptation system of S5 than to extend that of S3", and "The persistent model of S5 is less system dependent than that of S3".

4 Properties of a User Modeling System

In the framework from section 3 we saw that there is push adaptation and pull adaptation. In the comming sections we want to analyse the advantages and disadvantages of these adaptation strategies. To make an analysis we have identified a number of key properties of user modelling systems. Although some of these properties are not easily measured, we still believe they are important.

- *Adaptability.* The user should be able to manually adapt his model to a certain extend.
- *Speed.* The users' perception of the system's speed should not decrease.
- *Extensibility.* The system should be extensible while retaining the existing knowledge about its users.
- *Model size.* The model size should not grow too large.
- *Analysis possibilities.* The chosen kind of adaptation model should allow for all kinds of analysis techniques.

– *Privacy.* The system should be designed in such a way as to guarantee the highest possible level of privacy for the users.

Some of these properties are more important than others. It mainly depends on the application. We will not further discuss privacy as it depends mostly upon the application and very little on the adaptation model.

5 Push Adaptation Models

Push adaptation models are adaptation models, that let events propagate on to the values of a user model. Many systems that use push adaptation models use a rule-based model as employed in [9]. This paper describes the adaptation system of the AHA! system, a research system for creating adaptive hypermedia. These rule-based models are based on Active Database technology and as such inherit limitations from database systems.

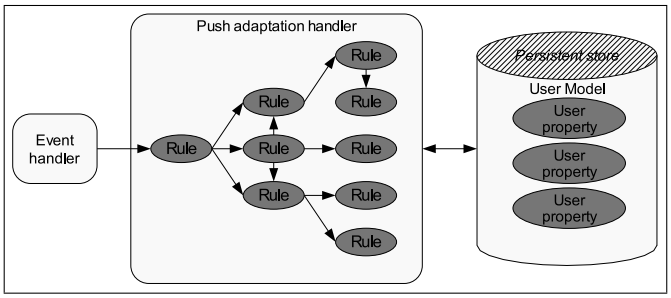


Fig. 4. A push adaptation model

There are several points to ECA rules. There is the possibility of endless recursion. Also there needs to be made a choice of techniques of achieving confluence. It should not be possible that equal starting models and equal events lead to different final user models.

One advantage of push adaptation is the fact that the contents of the user model are well aggregated. This has as advantage that those contents can be easily understood. Another advantage is that the relative size of the user model stays small, and that the size does not change during regular use of the system. This does however impede the possibilities for basing values of newly introduced attributes upon already seen behaviour of the user.

In this section we evaluate push based adaptation models based on the points from section 4.

– *Adaptability.* Because the user model stores end values it will be fairly easy for users to adapt the model to their wishes as the results of their changes are obvious and local. There could be too many possibilities for changes though.

– *Speed*. Provided that the amount of rules stays within limits there are no serious speed issues with push adaptation models.
– *Extensibility*. Push adaptation models are similar to database theory, and are often based on it. They have one problem that is similar to the problem of databases. Database systems are not good in data model change. This is the same with rule based adaptation models. At a moment that the adaptation model changes, values for new properties need to be calculated which can be expensive in terms of time.
– *Model size*. A push adaptation model has a user model with a limited size. This is because events are aggregated into the user model at the moment they happen.
– *Analysis possibilities*. The fact that event aggregation in rule based adaptation models happens at the moment the events happen makes it hard to impossible to perform time based analysis on user actions. Also aging (as weighing recent events higher than older events) is hard to implement.

From this point by point overview we can see that push adaptation models are especially good in the areas of model size and complexity. The weakest points lay in extensibility of the model.

Push adaptation models are very popular within the domain of educational systems. Those systems can be characterised by the fact that the user properties that need to be modelled are often (static/discrete/...). Push adaptation models are used in other system to though. Examples of push adaptation models can be found in: [9],[10],[11],[8].

6 Pull Adaptation Models

Pull adaptation models perform adaptation from a different direction than push models. In the extremity a pull adaptation model records all events in the user model. High level attributes are then derived based on lower level attributes and querying of the event record.

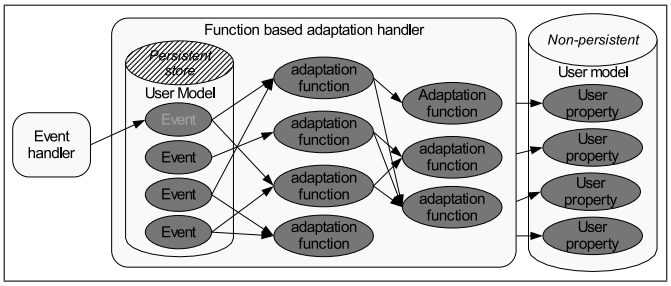


Fig. 5. A pull adaptation model

The pull model is based on calculation at the moment of the request. As such extension of the adaptation model is a lot easier than with push models.

One problem with the functional model though is the fact that the recorded data has very little value on itself. For adaptation purposes one would prefer to know concepts of user behaviour, not individual events. Push adaptation makes sure that concept generation needs to be done only once. Certain concept generation rules might be quite complex and would take a long time to recalculate on every use. To allow this for the pull model caching could be very helpfull.

- *Adaptability.* Pull models have problems with adaptability. This is caused by the fact that user models store huge amounts of abstract facts. One can not expect even experts to be able to make changes with predictable results in such a user model. An exception to this is that exclusion of time periods is easy in pull models. All events have a timestamp, and removal of facts just leads to different results of the functions.
- *Speed.* As user models that store events can get very big there is certainly the need to use extensive caching of intermediate results. The language used to query the user model could provide tools for incremental queries, where old results get enhanced with newer facts. Also the set of matching events can be stored to be used as a base for the query at a later time.
- *Extensibility.* The pull adaptation model scores very well on the point of extensibility. As abstract events are stored there will be many cases where new user attributes can be derived from behaviour before the attribute was introduced.
- *Model size.* Model size is a disadvantage of the pull adaptation model. With a little loss on model quality though old events could be aggregated into smaller parts or even discarded. If the amount of users of the system is not very high we don't believe there is a big problem on model size.
- *Analysis possibilities.* The pull adaptation model allows for more analysis possibilities. As all data in the user model is time stamped, time based analysis and aging are easy performed. There are no analysis possibilities in the push model that are not available in a pull model.

Pull based adaptation models are currently not common. They are especially utilised in cases where combinations of events need to be analysed to retrieve the goals of a user. A pull based adaptation model is for example used in [12]. In this article the interaction of users with a word processor is studied. This interaction is used to make recommendations to the user on doing things more efficient. Another example of pull models are attentive systems. They need to determine whether a user can be disturbed. These systems are highly dynamic and thus do not fit well with the static nature of the push model. Examples of these systems can be found in [13]. Other pull systems can be found in: [7], [14] and [15].

7 Hybrid Adaptation Models

Both adaptation models have their advantages and disadvantages. The push model for example might need workarounds for ages (as being dynamic properties

changing every second). The pull model is not very good at storing static user properties, and can be very space inefficient.

Looking at the two phases of the user modeling process we can see that while the model use phase is especially suited for a pull approach, the modeling phase is more directed towards a push approach. We can use this by using a hybrid adaptation model. Such a hybrid model can combine the advantages of both pure models. Basically the push model has a place in the user modelling phase and the pull model in the adaptation phase.

- *Adaptability.* By storing system independent user properties the hybrid system can offer the user clear high-level properties the user can change (not properties that are either too abstract events with unclear results (pull), or many system specific properties which have too localised results (push). This could mean that the adaptability of a hybrid system is better than both the rule based and functional approaches.

  This adaptability advantage could vanish if the rule based and functional models offer adaptability of intermediate concepts that are at the same position as the user properties of the hybrid model.
- *Speed.* Hybrid adaptation models should relieve many of the possible speed problems in the functional model as it can reduce the complexity of the event store in the functional model. It also avoids the rule explosion that comes with a big interrelated push model.
- *Extensibility.* The modeling process goes from very system specific events to less system dependent concepts. Those system independent concepts can be building blocks for extension. System dependent events cannot really do that. So there is no real loss in extensibility when using a hybrid model where concepts are stored that are less system dependent.
- *Model size.* In the hybrid model the model size can be significantly lower than the pull model as not single events are stored, but more high-level concepts.
- *Analysis possibilities.* As hybrid adaptation models allow for different adaptation strategies for different properties, they can retain most of the analysis possibilities that function-based adaptation models have. At the same time hybrid adaptation models can take advantage of properties of rule-based adaptation models where the analysis possibilities offered by a function-based approach is not necessary.

Hybrid adaptation models are more common than one would expect. They can often be found in systems where no special effort was put to the adaptation model. One area where they are almost unavoidable is the area of recommender systems. These systems tend to be focused on document–user matching techniques. Many of these systems make a single "user model" out of the event history of the user-system interaction (push). Those user models are then used at query time to make a rank of different recommendations (pull). Examples of recommender systems can be found in: [8],[16]

8 Conclusion

In this paper we have introduced a framework for classifying user modeling systems. With this framework we have shown that there are two basic categories of adaptation: rule-based adaptation and function-based adaptation. We have pointed out several examples of such systems.

Besides the rule-based and function based systems there is also a possibility for hybrid systems. We believe these hybrid systems can be able to solve the problems with both pure approaches, and combine their strong points.

We also pointed out that user modeling systems can have differing system dependence. This system dependence measure can be an indication of ease of extensibility of the system.

References

1. ACM: The adaptive web. Special Issue of Communications of the ACM **45** (2002)
2. Campbell, B., Goodman, J.M.: Ham: a general-purpose hypertext abstract machine. In: Proceeding of the ACM conference on Hypertext, Chapel Hill, North Carolina, US, ACM Press (1987) 21–32
3. Chin, D.N.: Strategies for expressing concise, helpful answers. Artificial intelligence review **14** (2000) 333–350
4. Fink, J., Kobsa, A.: User modeling for personalized city tours. Artificial intelligence review **18** (2002) 33–74
5. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction **6** (1996) 87–129
6. Cartwright, D.: Diy user profiling. `http://www.webdevelopersjournal.com/articles/user_profiling_diy.html` (2000)
7. Fleming, M., Cohen, R.: User modeling in the design of interactive interface agents. In: Proceedings of the seventh international conference on User modeling, Springer-Verlag New York, Inc. (1999) 67–76
8. Montaner, M., Lopez, B.: A taxonomy of recommender agents on the internet. Artificial intelligence review **19** (2003) 285–330
9. Wu, H.: A reference Architecture for Adaptive Hypermedia Applications. PhD thesis, Technical University of Eindhoven (2002) isbn: 90-386-0572-2.
10. Bra, P.D., Aerts, A., Houben, G., Wu, H.: Making generalpurpose adaptive hypermedia work. In: Proceedings of the WebNet Conference. (2000) 117–123
11. Brusilovsky, P., Cooper, D.W.: Domain, task, and user models for an adaptive hypermedia performance support system (2002)
12. Linton, F., Joy, D., Schafer, H.: Building user and expert models by longterm observation of application usage. In: Proceedings of the seventh international conference on User modeling, Springer-Verlag New York, Inc. (1999) 129–138
13. ACM: Attentive user interfaces. Special Issue of Comm. o.t. ACM **46** (2003) 30–72
14. Bull, S., McCalla, G.: Modelling cognitive style in a peer help network. Instructional science **30** (2002) 497–528
15. Virvou, M., Jones, J., Millington, M.: Virtues and problems of an active help system for unix. Artificial intelligence review **14** (2000) 23–42
16. Maglio, P.P., Barrett, R.: How to build modeling agents to support web searchers. In Jameson, A., Paris, C., Tasso, C., eds.: User Modeling: Proceedings of the Sixth International Conference, UM97, Springer (1997)