

Towards a Two-Dimensional Framework for User Models

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

12th June 2003

Abstract

The focus of 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 a lot of attention by various authors, see: [2], [3], [10], [13], [16]. 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: [6], [11], [13].

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. In particular it is necessary to know the which kinds of adaptation methods will be used. The methods used to do the adaptation is 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. First there is a the push adaptation model. Second there is the 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 parts of both models. An example of a hybrid system can be found in [8].

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 this is important to be able to better design user modeling systems.

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 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 (see [14] for a similar description, although there it is called intelligent user support). 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

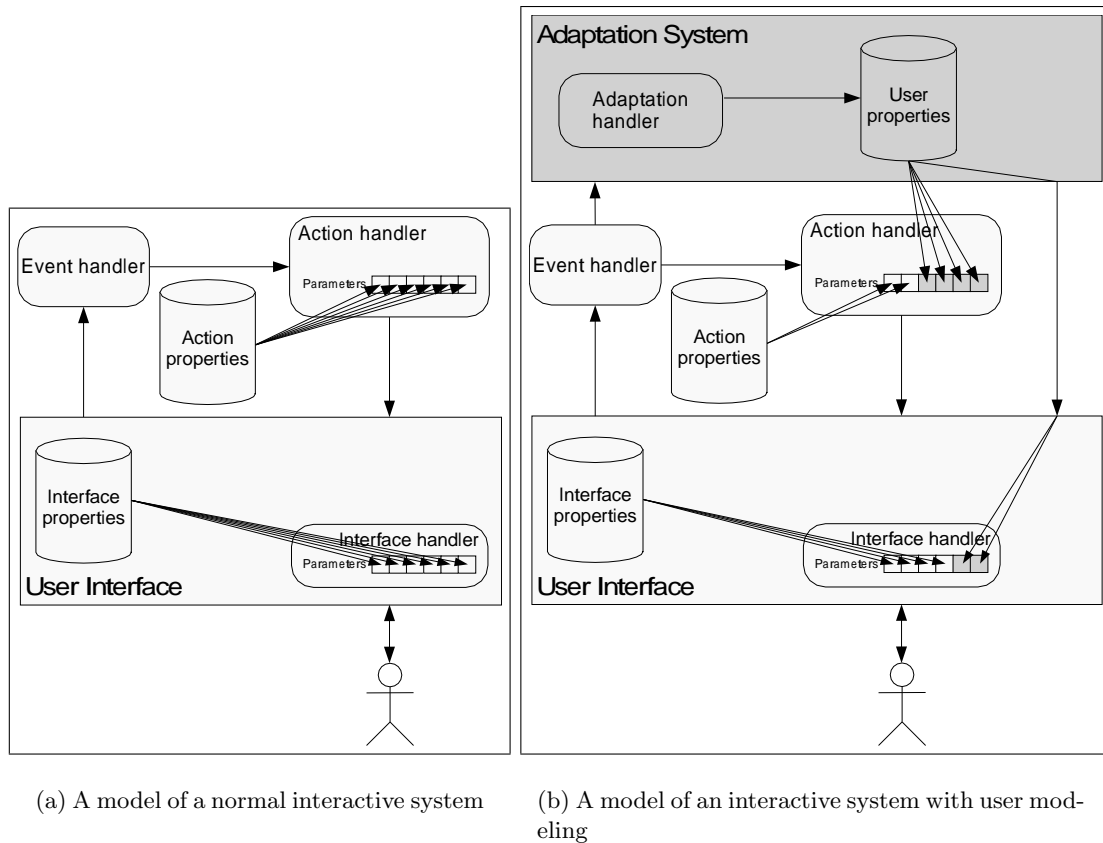


Figure 1: Comparison of normal and user modeling systems

(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.

2.1 Conventional interactive systems

Conventional interactive systems (see figure 1(a)) can be seen as state machines that interacts with a user. This interaction is handled by a user interface. Each user action can induce a state change, after which new user actions are possible. Looking at systems as state machines is also done in [4].

In designing a user interface several 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 assume that the choices are explicit. We call those choices interface properties.

The interface properties determine both the behaviour and looks of the user interface. As such the interface properties influence the events that are generated based on actions of the user. These events induce state changes in the system. This will trigger the event handler state. The events are now processed by the event handler which triggers the appropriate system actions. Some event might actually not involve any system actions in certain applications. For example for a text editor a mouse move event is not relevant.

The actions taken by the action handler upon the events passed to it are influenced by the action properties. For example in case there are two ways to approximate a certain value, the action properties determine which method is used. In most cases this choice is actually implicit and the system only allows one method.

The results yielded by the action handler are then presented to the user through the user interface. The behaviour of the user interface is determined by the interface properties.

Now that we have explained how events, actions, the user interface, and interface

properties work together in a conventional interactive system, in the next section we will introduce the model for user modeling systems.

2.2 User modeling systems

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. [16] and [17] 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 on a specific user property. As the adaptation system can be seen as the authority on the user, the questions should be as specific as possible to allow for making maximal use of the knowledge of the user.

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.

It is important to know that the distinction made in figure 1 is not absolute. Adaptive behaviour has such a broad impact on the functioning of a system that 1(a) is actually a model of an adaptive system where the adaptive part is cut away as much as possible.

In figure 1(b) there is a part of the system that is called the adaptation system. The process that is performed from the entering of events into the box to the emitting of user properties is called the “inference process”. The way this inference process is shaped is very important for the behaviour and possibilities of the system. This inference process is the basis of the framework that is introduced in the next section.

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 [21].

3.1 Two-dimensional user modeling classification framework

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 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?”.

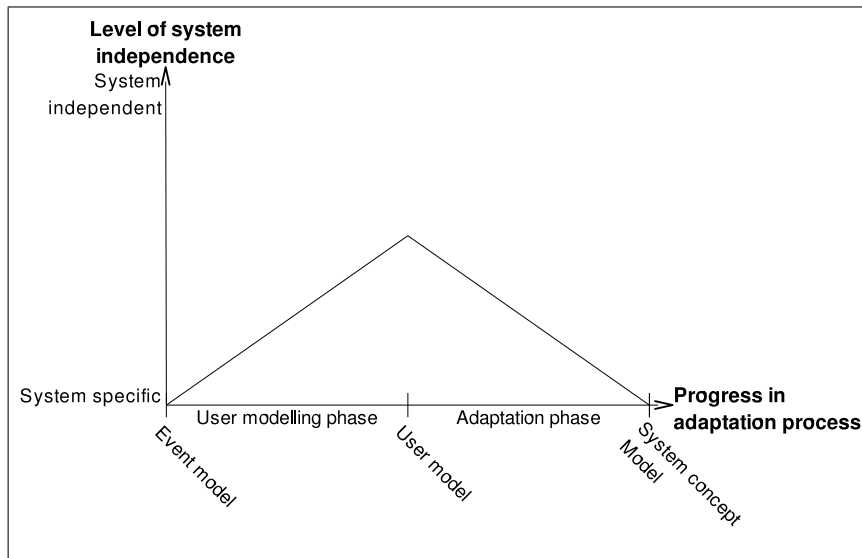


Figure 2: A two-dimensional user modeling classification framework

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.

3.2 Use of the framework

We can use the framework of figure 2 to determine two properties of systems. Firstly, we can look at the height 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. We will go further into system independence in section 3.3 The second property we can distinguish is, where in the inference process a persistent

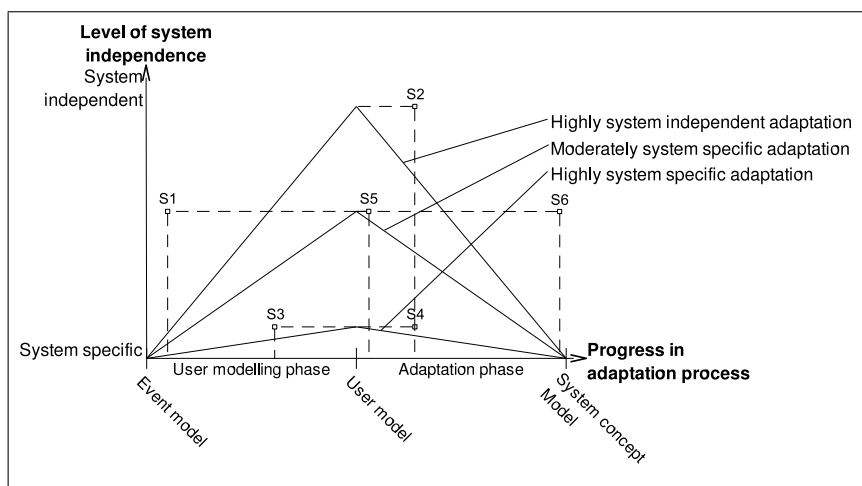


Figure 3: Use of the two-dimensional classification framework

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.

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".

3.3 Analysis of system independence

In this section we try to look at the meaning of system independence for an adaptation model. In general the idea is that the more system independent a user model is, the easier it is to use the model for other uses than the application it was originally intended for.

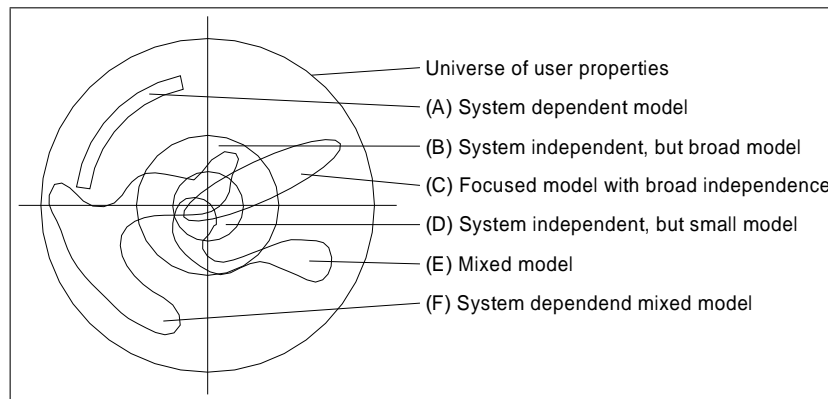


Figure 4: Example of different user models according to system dependence

To explain better what system independence is for a user model, we take a look at figure 4. This figure shows a "universe" in which we model the many-dimensional user models in a 2-dimensional way. The distance from the origin determines the level of system dependence. Close to the outer circle The dependence is maximal. At the origin the system dependence is minimal. The angle of a point from the origin represents the subject of that point. Similar subjects have similar angles.

In the context of figure 4, system A is a system that is system dependent. It does cover though many subjects. System B and D do cover all subjects and are system independent. Here system B is a superset of system D in that it also contains more system dependent properties.

System C is a system that is very focused on one subject, but is rather complete in that subject containing many properties. System E has a broader subject range, but within that range differs significantly in system specificity. System F could be seen as similar to the combination of systems C and A in that it is system dependent for many subjects, but for a certain subject range also is system independent.

The systems from figure 4 are examples of possible user models. We believe that although mapping user models into a two-dimensional model discards much information it does show that user models can be very diverse. It also shows that every property in the user model can have a different system dependence. We believe though that having the onedimensional degree of system independence that is used in the framework can express enough of the user model to be distinctive.

4 Properties of a user modeling system

In the framework from section 3.1 we saw that there is push adaptation and pull adaptation. In the coming 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.

- *Predictability.* The system should not give the user unpleasant surprises. If it would, the user would not feel “in-control” of the system, and abandon the system (or put the user modeling feature off if possible).
- *Adaptability.* The user should be able to manually adapt his model to a certain extend (controlled by the system administrator). This manual adaptation should never lead to an inconsistent user model.
- *Supportability.* A system that applies user modeling should still be supportable by a (corporate) support organization (help-desk). This could mean that a system has an option to temporary turn the user model off (reverting to a standard model).
- *Control.* The users should be in control of the system, not the reverse.
- *Speed.* The users’ perception of the system’s speed should not decrease. This basically means the responsiveness of the system needs to stay good.
- *Extensibility.* The user model/system should be extensible while retaining the existing knowledge about its users. For web-based systems / adaptive hypermedia this means it is desirable to have the option to to change/extend the system while running. That could mean that users starting a new session would get the new environment while users already in a session would keep the old environment.
- *Model size.* In environments where amount of user models on one server can be big, model size is very important. Also in those environments it is desirable to have a maximum size to each user model, or to have constant size user models.
- *Analysis possibilities.* Ideally the chosen kind of adaptation model should allow for all kinds of analysis techniques in a way that is as simple as possible.
- *Privacy.* The system should be designed in such a way as to guarantee the highest possible level of privacy for the users. This is especially true for internet-based systems, but is still valid for company internal systems. One way this might be achieved is to keep the user model at the user instead of at a central repository. To keep the model at the user poses some problems though, and would probably require parts of the functionality to be performed at the users side of the system.

The privacy problem of course is less when the system is not distributed, and only exists on the users computer. In that case there is no way for anyone else but the user, to get information about the user. For a discussion of privacy see [12]

- *Concurrency.* Depending on the kind of application the importance of a consistent behaviour towards reentrant or concurrent access varies. For an architecture though it is necessary to consider concurrent access of the user model. The need for this is pointed out in [15]. While there is no solution offered in either that article, or this article we state that concurrency is a factor that must be considered when evaluating an adaptation model.

The concurrency demand can be summarised in the ACID principle ([18]) as used in database theory. The acid principle has four pillars:

- *Atomicity:* A transaction is indivisible, and is only executed fully or not at all.
- *Consistency:* Any transaction must leave the user model in a consistent state regardless of failure or success.

- *Isolation*: A transaction may not be affected by concurrent transactions. This creates the necessity of locking.
- *Durability*: After successful completion of a transaction the changes should be persistent, even through system crashes.

The durability demand is not that relevant in the context of this paper. There are technical solutions to this problem, and any decent database system implements one. The solution necessary also doesn't change much on the user or adaptation model.

While privacy protection is essential in a good user modeling system, it does not put any specific demands on the user modeling system. Also the point of user model size, while very important is not included in the list of properties, as we believe it is already covered by items such as speed. Further it is essentially an implementation issue depending very much on the available hardware, and the setup of the system. We will include it though in our evaluation because there are considerable differences in model size between the different models.

Further we include the possibilities of analysis offered by both models. While both models can be used to a large extent to emulate behaviour of the other model we will not consider such use as this is rather complex, and changes the nature of the 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 adaptation model as employed in [23]. This thesis describes the adaptation system of the AHA system, a research system for creating adaptive hypermedia. rule based adaptation models are based on Active Database technology and as such inherit limitations from database systems. The rules are based on the Event Condition Action (ECA) rules from active database technology. ECA rules are invoked upon a certain event, specified in the rule. At occurrence of such an event the conditions are evaluated. If the conditions evaluate to **true** the actions get evaluated. The actions can then trigger other events which can trigger other rules etc.

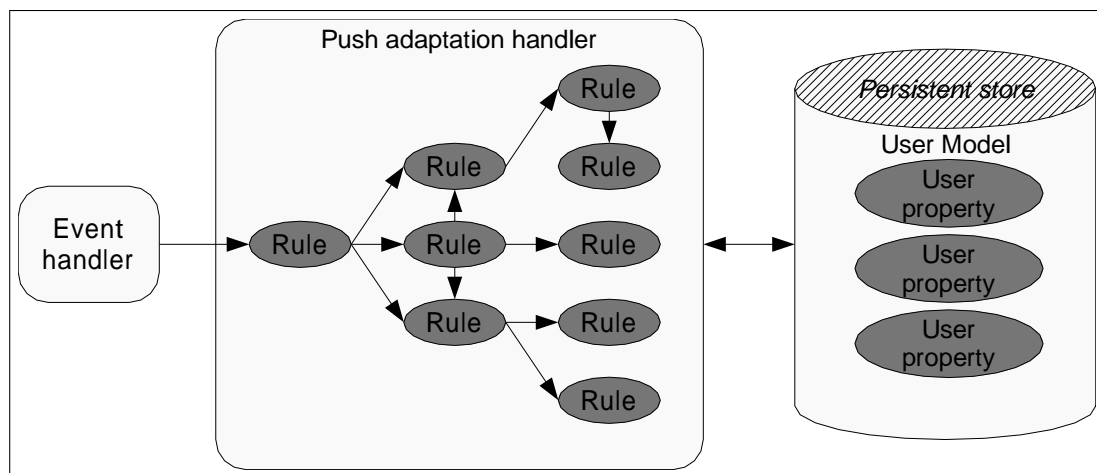


Figure 5: A push adaptation model

There are several points to ECA rules. There is the possibility of endless recursion (ending out in an infinite loop) which is obviously unwanted. 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 an 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. The use of predefined user profiles can be helpful here only if those profiles can be defined before use of the new property in the system. It is doubtful whether this is possible in most cases, except when a testing phase is used before introduction of a new domain model.

The techniques needed to achieve termination and confluence with rule based adaptation models complicate the adaptation system. There are possible ways of limiting the adaptation rules to avoid these problems.

The first possible way to achieve termination and confluence is through only allowing updates of attributes that are not dependent on the values of any properties. This makes events caused by change of properties almost pointless as it would mean there is no way for rules to depend on attributes in the user model.

A second way is to disallow any activation of rules as the result of updates of the user model. In this case there is no recursion possible. To assure confluence other measures need to be taken. One way is to impose an ordering on the rules.

A third way is to disallow cycles in rules. This would too solve the problem of recursion, but be less strict as the second way. Also here confluence is not taken into account.

A fourth way to combat the recursion problem is to allow only a limited number of rule invocations per user event. This will cut off any recursive rule invocation. The number of invocations needs to be fairly high so as to minimize the problems of unpredictability caused by sudden cutoffs of rule execution as different sequences of user actions might make cutoffs happen at different points in the execution path.

5.1 Point by point evaluation of push adaptation models

- *Predictability.* Push adaptation models, provided that there is no problem involving confluence, are in principle predictable. The main point in predictability is the design of the domain model. This is application / system specific.
- *Adaptability.* Because the user model stores high level concepts it will be fairly easy for users to adapt the model to their wishes. If concepts get too focused to particular system concepts this advantage might vanish as it becomes harder to influence the overall behaviour of the system.
- *Supportability.* As the amount of attributes in the user model is limited, support departments have an easier task to support the user model.
- *Control.* This point is mostly influenced by the domain model and other options within the system unrelated to the adaptation model. There is one point of control though that is very dependent on user model. That is the possibility to ask the system to undo certain behaviour, or even to ask it to exclude a certain time period from the model.

Undoing user models is possible with push adaptation models using snapshots and increments. This allows reconstruction of the user model at a certain state. Disregarding certain time periods from the model is impossible 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. In connection with that they inherit one problem of databases. Database systems are not good in data model change. This is the same with rule based adaptation models. Introduction of new attributes must go together with explicit defaults for the attribute involved. This can be hard as different user models might need different default values for new attributes. The use of user profiles here is hard as obtaining those profiles is a problem since there is no possibility to use existing user models as a base.

There are less problems with default values of new attributes when the new attributes are related to new concepts that are largely unrelated to attributes in the

existing user model. As an example let's consider the addition of a new concept. Suppose the access to this concept A is dependent to a attribute of another concept B (e.g. whether that concept is known to the user). At the point where concept B is known before concept A is a part of the user model there is a problem. Using only adaptation rules and default values there is no possibility, except to force a change of the attribute of concept B, to have perform the expected behaviour in this case (Allow A if B is known and disallow A if B is not known).

In the AHAM model [23], the approach is taken to actually evaluate the visibility of a certain concept at the moment it needs to be shown. While this "solves" the problem it introduces the problem that one still cannot now beforehand whether a link to this concept is warranted. We believe users should never see any links to concepts they cannot access.

- *Model size.* A push adaptation model has a user model with a limited size. This is because events are aggregated into a 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.
- *Concurrency.* Push systems work by updating the user model based on a rule and consequently executing all rules triggered by this update. To allow concurrency within push systems therefore all data properties that can change must be locked before the execution of the rule is started. Based on the rules a sizeable part of the model might need to be locked. This seriously limits the possibilities for concurrent access of the user model.

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.

5.2 Examples of push models

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: [23],[5],[7],[14],[21].

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 / user model.

The pull model can be seen as being functional. The functional model is not based on databases, but on mathematics instead. As such the model does not suffer from the problems of the push systems that make domain model extension hard.

With the choice the choice of a functional model there are no confluence problems at all. There can be a problem of termination. A solution to this could be to disallow recursion. Disallowing recursion in a functional model is not that problematic as disallowing recursion in a rule based model. In a functional model the results of the recursive function can often be calculated in other ways. There are also ways of proving termination of recursive functions (and indirect recursive functions).

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. Rule based adaptation makes sure that concept generation needs to be done only once. Certain concept generation rules might be quite

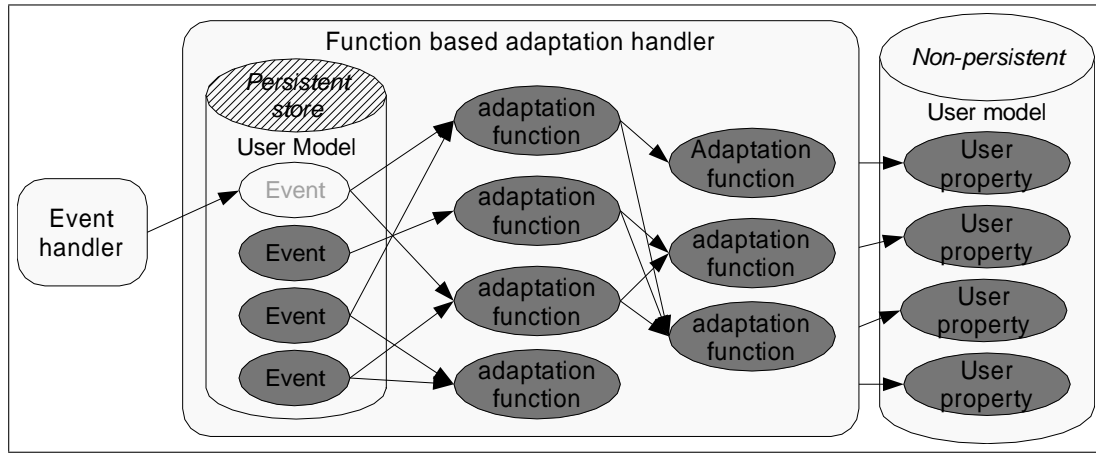


Figure 6: A pull adaptation model

complex and would take a long time to recalculate on every use. Caching could be very helpful here, but it very much depends on the functions involved.

6.1 Point by point evaluation of pull adaptation models

- *Predictability.* Pull models do not have inherent predictability problems. There are clear mathematical rules of preference in cases where confluence could occur.
- *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 predictable changes 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.

We believe though that there are certainly possibilities to let users specify override values for the functions in the adaptation model. That can adapt fewer, more concrete values. The system needs to specify though which functions can be overridden, and which not. In certain cases inverse functions could be used to propagate a default value for a high level function to lower level functions.

- *Supportability.* The problems of adaptability arise also on the point of supportability. The undo possibilities though can be a very big help for support departments.
- *Control.* Control depends most on the domain model, but the possibility to exclude data from certain time periods helps the users to be in control of the system. The system could also employ time series analysis tools to identify different periods of system use.
- *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.

At the point that events have been executed, the pull model can be faster as only the functions involved in a certain concept need to be executed.

- *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. Further, the user model format does not need to change at the moment the domain model changes.
- *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 that 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 push model. If necessary a Turing-complete language could be used to write the functions in the adaptation model. This does include problems with termination though.
- *Concurrency.* Pull adaptation systems are rather well suited for concurrency. As a (pure) functional system does not update data, every transaction needs to know which data item is the top item at the start of the transaction. During the transaction only items earlier than this item are considered which guarantees consistency when new items are added in the mean time.

In case of maintenance by data agregation we again have the advantage of the fact that normal operation does not update data items, but only adds. This means the only time locking is necessary is when the aggregated data needs to replace the original data items. These replacements might even be scheduled for later processing at a time when the store is not being used.

6.2 Examples of pull models

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 [19]. 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 [1]. Other pull systems can be found in: [8, 9],[22],[17]

7 Hybrid adaptation models

Both adaptation models have their advantages and disadvantages. The push model for example might need workarounds for dates, or age (date dependent) of a person. The pull 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 he model use phase is especially suited for a pull approach. The modeling phase is more directed towards a push approach. As a result of this there is the possibility to have a mixed adaptation model. Such a mixed model can combine the advantages of both pure models.

7.1 Point by point evaluation of hybrid adaptation models

- *Predictability.* As hybrid system the system inherits the predictability of the push and pull systems. Both have good predictability, so a hybrid adaptation system should also have good predictability.
- *Adaptability.* By storing system independent user properties the hybrid system can offer the user clear high-level properties the user can change. This could mean that the adaptability of a hybrid system is better than both the rule based and functional approaches.

This better adaptability 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.

- *Supportability.* The supportability of a hybrid adaptation model is probably less than that of the rule based model, but certainly better than the functional model. Also a good hybrid model can retain undo capabilities.

- *Control.* Depending on the exact adaptation model there are still possibilities to do time based analysis and time period exclusion.
- *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.
- *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 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 analysis offered by a function-based approach is not necessary.
- *Concurrency.* Hybrid adaptation models need a concurrency framework that is as capable as that of rule-based adaptation models. The hybrid nature, however, of the models can lead to more localised locking needs than the needs of rule-based models. As such locking should be less of a problem in hybrid models.

7.2 Examples of hybrid adaptation models

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. Those user models are then used at time query time to make a rank of different recommendations. Examples of recommender systems can be found in: [21],[20]

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] Attentive user interfaces. *Special Issue of Communications of the ACM*, 46(3):30–72, March 2003.
- [2] 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.

- [3] The adaptive web. *Special Issue of Communications of the ACM*, 45(5), May 2002.
- [4] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998. ISBN 0-201-57168-4.
- [5] P. De Bra, A. Aerts, G. Houben, and H. Wu. Making generalpurpose adaptive hypermedia work. In *Proceedings of the WebNet Conference*, pages 117–123, 2000.
- [6] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.
- [7] Peter Brusilovsky and David W. Cooper. Domain, task, and user models for an adaptive hypermedia performance support system, January 2002.
- [8] Susan Bull, Jim Greer, Gordon McCalla, Lori Kettel, and Jeff Bowes. User modelling in i-help: What, why, when and how. In M. Bauer, P.J. Gmytrasiewicz, and J. Vassileva, editors, *User Modeling: Proceedings of the Eight International Conference, UM01*, pages 117–126. Springer, 2001.
- [9] Susan Bull and Gord McCalla. Modelling cognitive style in a peer help network. *Instructional science*, 30(6):497–528, November 2002.
- [10] 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.
- [11] Dave Cartwright. Diy user profiling. http://www.webdevelopersjournal.com/articles/user_profiling_diy.html, April 2000.
- [12] Darren Charters. Electronic monitoring and privacy issues in business-marketing : the ethics of the doubleclick experience. *Journal of business ethics*, 35(4):243–254, February 2002.
- [13] David N. Chin. Strategies for expressing concise, helpful answers. *Artificial intelligence review*, 14(4-5):333–350, October 2000.
- [14] L. Miguel Encarnação and Stanislav L. Stoev. An application-independent intelligent user support system exploiting action-sequence based user modelling. In Judy Kay, editor, *User Modeling: Proceedings of the Seventh International Conference, UM99*, pages 245–254. Springer, 1999.
- [15] Josef Fink. Transactional consistency in user modeling systems. In Judy Kay, editor, *User Modeling: Proceedings of the Seventh International Conference, UM99*. Springer, 1999.
- [16] Josef Fink and Alfred Kobsa. User modeling for personalized city tours. *Artificial intelligence review*, 18(1):33–74, September 2002.
- [17] Michael Fleming and Robin Cohen. User modeling in the design of interactive interface agents. In *Proceedings of the seventh international conference on User modeling*, pages 67–76. Springer-Verlag New York, Inc., 1999.
- [18] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufman, San Mateo, CA, 1993.
- [19] F. Linton, D. Joy, and H. Schafer. Building user and expert models by longterm observation of application usage. In *Proceedings of the seventh international conference on User modeling*, pages 129–138. Springer-Verlag New York, Inc., 1999.
- [20] Paul P. Maglio and Rob Barrett. How to build modeling agents to support web searchers. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*. Springer, 1997.

- [21] Miquel Montaner and Beatriz Lopez. A taxonomy of recommender agents on the internet. *Artificial intelligence review*, 19(19):285–330, June 2003.
- [22] Maria Virvou, John Jones, and Mark Millington. Virtues and problems of an active help system for unix. *Artificial intelligence review*, 14(1-2):23–42, April 2000.
- [23] Hongjing Wu. *A reference Architecture for Adaptive Hypermedia Applications*. PhD thesis, Technical University of Eindhoven, November 2002. isbn: 90-386-0572-2.