# The development of User Modeling over three decades

Katharina Schwarz - Matrikelnummer 539733
Media System Design - FH Darmstadt

14th January 2003

## 1 Introduction

Have you ever tried to find a specific piece of information on the web, and ended up with an endless list of links that related in some way to your query? And were you frustrated to think that each document probably contained some information that interested you, but it was too much trouble to go find it? Can you imagine how much worse this gets in a multimedia environment, where the retrieval results contain not only text, but also audio, video and graphics? Now wouldn't it be nice if the result of your query were a single presentation that showed you everything you wanted to know, as though it had been made just for you?

For this wishful thinking to become reality, there has to be a method to give the computer the intelligence to answer the question "what does the user want, and how can he[1] get it?". One approach to achieve this is called user modeling. Although user modeling hasn't yet reached the sophistication needed to answer the question correctly, it is moving in this direction.

As described in [6], user modeling was first explored in the late seventies by the researchers P. R. Cohen, C. R. Perrault, J. F. Allen and E. Rich. It has developed into a large field in computer science that plays a role in many applications, e.g. computer based training, e-commerce, natural language dialogue and information retrieval. Not all domains in computer science need to have a user modeling component though. In some cases it would even be foolish to incorporate the user, for example in a system which ensures that a nuclear plant runs smoothly. The risk of human error is kept low here by giving the human as little influence as possible.

[1] The pronouns *he*, *him* and *his* are used generically in this paper to refer to both men and women

In this paper, I want to give a general overview of what user modeling is and where it is applied. I do not have personal experience with most of the applications described, so many statements are based on secondary knowledge. In section 2, I will explain what user modeling is. Section 3 deals with techniques to gather user data and explains how the data is shaped to form a user model. Section 4 shows how a system is adapted to the user. In section 5 I will address security and privacy issues, and section 6 provides the conclusion.

# 2 User Modeling basics

## 2.1 Why

User modeling is an attempt to close the gap between the human and the machine. The goal is to make applications and information easily accessible to as many people as possible, ideally regardless of their limitations and disabilities, and to let the system intelligently support the user.

The user modeling approach is to make the machine understand and adapt to the user instead of forcing the user to understand the machine in order to work with it ([4], pp 16,17).

## 2.2 What

Understanding something requires establishing a model of it. In user modeling the following models are distinguished [14]:

- the designer's model of the user,
- the user's model of the system, and
- the system's implementation of the designer's model of the user.

### 2.2.1 The designer's model of the user

The designer has a model of the user who will work with the application he is developing. This is the so-called target group. The better the designer knows the goals and characteristics of his target group, the better the system can be designed to fit their requirements. The depth of the knowledge about the user that is necessary depends on the complexity of the system. Consider for example an ATM machine. The target group of users of ATM machines is very large, yet easy to define. A typical user has a bank account, he can read and he can have only 2 goals: viewing the bank account balance or

withdrawing a certain amount of money, or both. A more complex system with much functionality provides a means of fulfilling various goals. On one hand it is quite practical for a user to be able to perform many tasks in one and the same application. On the other hand, with rising complexity it becomes more difficult to navigate through the system and find and use a specific function. It is an extremely difficult task for the designer to structure the application and devise the navigation with the expectations of a "typical" user in mind, because there is no "typical" user. Users differ widely in their goals and their knowledge.

One solution is to downscale the system by reducing its functionality. This makes it easier to structure and to use. Yet if the system is to retain its level of complexity, another solution is needed. A common way is to make the system adaptive to a specific user. The designer can split the target group into various user models, develop a mechanism to recognize which model a user belongs to and decide how the application should adapt to them. All the knowledge is implemented into the system at design-time. A system that is equipped with this kind of knowledge can react flexibly to the input it gets from a user at run-time and adapt the interface to his goal [2]. The focus of the paper is on this form of adaptation.

### 2.2.2 The user's model of the system

The user has a mental model of the functionality of the system, based on self-exploration, instruction, training and experience with it. The model changes with time, it becomes either more detailed, or more blurred, depending on how much the user works with the system. Once he has understood the underlying logic, the user expects the system to remain consistent, especially in navigation, structure and design. Inconsistency can confuse a user's model of the system and lead to dissatisfaction and frustration. When a system adapts to the user, it should remain consistent with the model the user has of it.

### 2.2.3 The system's implementation of the designer's model of the user

Adaptive systems are designed to classify the user within the designer's model, gathering data about them from their interaction with the system. Before there is enough individual user data to go by a default model holds the place. This is called the canonical model. As the user interacts with the system, a profile is created and filled up with individual values pertaining

to the particular user. Which attributes represent the user in the profile depends on which part of the system will be altered to suit the user.This representation of the user is never correct, it is just an approximation.The question is, how vague can the system's user model be to still make helpful adaptations to the user? In the following chapters solutions to this problem are discussed.

## 2.3 How

Different methods exist to gather information about a user. User data can be explicitly supplied by the user or implicitly observed by the system. There is even a form of gathering implicit information explicitly with the help of task solving. These methods will be discussed now [14].

### 2.3.1 Explicit modeling

The user is asked to answer some straightforward questions, such as age, gender, occupation, preferences, disabilities. Although this appears to be a simple approach, problems with it are numerous. First, people may not answer the questions or give wrong answers deliberately, out of impatience or for privacy reasons. An interesting trait has been observed in users; most people are subject to the "paradox of the active user". This means that they are unwilling to do anything that appears unrelated to the task they set out to do, such as answering a questionnaire. They usually don't like to be deterred by reading introductions or explanations either. The paradox is that doing these things would support their cause and help them succeed sooner and better [4]. Another problem is the suspicion of the motivation behind the questions. Especially in environments like the Internet, people want to leave as little personal information as possible, because they have no control over what it is used for. A substantial percentage of users would actually refuse to make a use of the services offered by a site that forced them to give personal information (see section 5). Knowing this, explicit user modeling questions should not be compulsory. In any case, the questions should be short, simple and anonymous.

Second, even if they enter the data completely and truthfully, it has become clear in psychological literature that people are unreliable sources of information about themselves ([9], p.326).

Third, what is the value of answers to explicit questions? The information that a system seeks is of a complicated nature, namely a user's goal and their background knowledge. The answers will not be found with the

help of a few simple questions.

This shows that what appears to be the easiest method, simply asking a user what he wants explicitly, is in fact the least reliable one. It is best used for questions about objective facts such as a user's preferences or disabilities (e.g. red/green colour blindness), so that these can be taken into consideration when adapting to a user.

### 2.3.2 Implicit modeling

Implicit modeling serves mainly to find out what the user knows about the domain of the system and how familiar he is with its functionality. The information is gathered by tracking his actions. This leads to assumptions about what he wants to know and to do. For example, if the user clicks on an explanation of a term it is assumed that he didn't know what the term meant. On the other hand, it is possible that he just clicked on it out of curiosity or to reassure himself of his knowledge. The action might even have been random. The problem with implicit data is that there are many explanations for a single action, only one explanation is correct, and the system has no way of knowing which one it is. It will decide on the most likely explanation, but the decision is based on a guess. If another action of the user contradicts that assumption, the system must have a mechanism for resolving the conflict. This is a common feature in user modeling systems. Examples of applications where conflict resolution is implemented are described in chapter 3.

In conclusion, implicit data only becomes a reliable source once there is enough information available to detect a pattern. Thus, implicit modeling is most useful when applied to frequent users of a system.

### 2.3.3 Task solving

Task solving is done mainly in teaching applications. It is a method to find out about a user's knowledge of the domain explicitly. At strategic points in the course, for example in the beginning of the interaction or at the end of each chapter, it is necessary for the application to find out what the level of the students' knowledge is and where he has problems. This is easily deduced from the way he solves tasks of varying levels of complexity.

### 2.3.4 Long-term vs short-term

The data that is gathered about a user reflects different aspects of him. One important classification is to divide the data into *long-term* facts, like

his level of expertise on a specific subject, and *short-term* facts, such as his current interest. The choice of whether to create a short-term or a long-term user model, or a mixture, depends on the nature of the intended application. Take for example a flight booking website. At first sight it is an application that processes short term goals, as a user's current goal is likely to have changed the next time he accesses the website. On the other hand, every time he books a flight, he has to enter the same personal details such as name, billing address, credit card number, etc. The system might as well save him the hassle and store the data in a personal profile. This solution is practical yet controversial, as the user has no control over the data that is in his user model, over who gets to see it and how it is used. As mentioned earlier, many people find this a scary thought. Therefore laws have been instituted that protect user data and user rights from misuse, which apply especially to long-term user models. Chapter 5 deals with the legal side of user modeling.

It has become clear that even the first step to classifying a user in a model, getting data about the user, is a difficult task. Not only is the data sparse and diffuse, the question arises whether it is relevant at all. Still, it is the data that must be used. The following section describes which methods have been devised to make the most out of the available data.

# 3 From user data to user model

How does the system process the information it gains about a user to turn it into a suitable user model? Whether a model is suitable or not depends on the application it is used in. There are different domains in which different kinds of data are gathered and which use different user modeling techniques. In the domain of high functionality applications for instance, one aim is to help the user perform his task. The user model will contain data about the regular tasks of a user and serve to provide shortcuts or give recommendations on working more efficiently. In contrast, an online news provider will concentrate on what the personal interests of a user are on which time of the day and which day of the week, in order to customize his news.

In the following, I will describe some of the most eminent techniques.

## 3.1 Stereotypes

One definition of a stereotype, taken from the website www.dictionary.com, is

> a conventional, formulaic, and oversimplified conception, opinion, or image.

It is assumed that a person whose obvious traits coincide with those of a certain stereotype also has all the other traits that belong to that stereotype. An example of a stereotype is "a woman with an asymmetrical haircut, a single dangling earring and alternative clothes is a feminist. She has an academic background, has strong political opinions and buys groceries at organic food stores". Stereotypes help people to classify those around them. Stereotypes have a negative connotation because many people take these assumptions for facts and are sometimes unwilling to revise them.

In user modeling, a stereotype is a mere collection of traits that often occur together in the world around us. When a user triggers a stereotype, all the traits of this stereotype are assigned to him. These traits, or *attributes*, come with a *value* that indicates to which degree the system assumes that the trait is part of the user's characteristics. A second value, called the *rating*, indicates how sure the system is that this assumption is correct. Each rating comes with a *list of reasons* that explain why this value is believed.

All stereotypes are structured in a hierarchy, with the canonical user model being the topmost one, from which the others descend and become more specific. A user can activate many different stereotypes. All their traits are assigned to his user model.

One important problem in user modeling in general is how to resolve conflicts that occur when inferences based on a user's actions result in contradicting traits in the user model. In stereotyping, the *rating* that shows how sure the system is of the assumption and the *list of reasons* that explain why a particular trait was activated are employed to reason about the validity of either trait in conflict.

Besides implicit assumptions other sources of information influence the user model, such as direct statements from the user or his answers to questions generated by the system. These will override assumptions based on inferences. [8]

The success of the stereotype method depends on how representative the stereotypes really are, on how accurately a user is mapped to a stereotype and on the veracity of the assumptions made about the user ([4], pp 19,20).

For an illustration of how stereotyping works in a real application we now provide a description of GRUNDY.

### 3.1.1 GRUNDY

One of the first applications based on stereotypes is the system GRUNDY developed by E. Rich in 1979 [9], which models users in order to make personalized book recommendations.

In the first session, a user is asked to give a short self-description. The terms used act as triggers for the activation of certain stereotypes. All of the stereotype's generalizations are also activated, so each user model definitely contains the traits of the canonical stereotype called ANY-PERSON in GRUNDY. A user model is like a table with four columns, the headers of which are "attribute", "value", "rating" and "justification". Justification is analogous to the list of reasons mentioned above, which explain why a particular attribute was assigned to the user model. There can be several justifications for a single attribute.

The types of conflicts that can occur in GRUNDY are

- contradictions between two stereotypes or
- contradictions between a stereotype and direct or indirect statements of the user.

In the first case, there are two distinctions:

1. The attributes in conflict pertain to stereotypes that are in the same chain in the hierarchy. In this case, the value of the attribute of the more specific stereotype takes priority.

2. The attributes in conflict belong to stereotypes that are in different groups. This is a more difficult case to solve. One possibility is to trace the branches that the stereotypes belong to in the hierarchy upwards until they meet, and use the value of this more general stereotype. Another way is to compare the kind of attribute. The stereotypes of GRUNDY contain two kinds of attributes: necessary attributes and default attributes. The necessary attributes are those that define the stereotype, e.g. the stereotype of a poet will have a high skill for writing per definition. A default attribute could be that the poet owns and knows how to use a typewriter. If there is a conflict between necessary and default attributes, the former will win.

The second case is easy to resolve. A user's direct or indirect statement always overrides an assumption based on a stereotype.

Once GRUNDY has enough information about the user, it will start giving book recommendations. It selects books from its database that match

the preferences of the user model, and suggests them one by one to the user. He is asked whether he has read the book already, and if so, whether he liked it. If not, he is given a short description of the book and asked whether he would like it. If the user didn't like the book or would not like the book, GRUNDY tries to find out which assumption is to blame for the wrong recommendation by getting feedback from the user on why not, and modifies the user model.

Not only the user model is constantly updated, GRUNDY also adapts the stereotypes themselves. The stereotypes learn from users whether they generally represent them correctly. If a prediction based on a stereotype often gets positive feedback, this stereotype is reinforced, if the feedback is often negative, the stereotype is changed. The changes apply to both the value of the attribute and the rating.

In an experiment on the usefulness of GRUNDY, it gave its users random recommendations and recommendations based on their user model. Statistically, 72% of the controlled recommendations were considered good, as opposed to 47% of the random recommendations.

### 3.2 Machine Learning

As we have seen, the stereotype approach to user modeling focuses mainly on modeling a user's characteristics. Machine learning is rarely employed for this view on user modeling, because its results are based on calculations, which have no relation to characteristics. Instead, it is used to model aspects such as the cognitive processes that underlie a user's actions, the comparison of a user's skills to those of an expert or the user's behavioral patterns or preferences. Historically, focus was on the former two aspects, but with the advent of the World Wide Web more attention was paid to user's behavioral patterns and preferences in web applications, especially in the area of e-commerce.

Machine learning uses learning algorithms to evaluate data gathered from the user. Generally, the more data the algorithm can process, the more accurately it can model the user's preferences and knowledge.

A simple application of machine learning on the web is Syskill & Webert, a software agent developed by M. Pazzani and D. Billsus in 1997 [13]. Similar to the GRUNDY system, its aim is to make recommendations, but its domain is websites instead of books and its user model is based on a Bayesian classifier[2] instead of stereotypes. A user is presented with a man-

[2]The Bayesian classifier is a probabilistic method for classification. It can be used to determine the probability that an example j belongs to class Ci given values of attributes

ually created index page containing many links to different sources on a certain topic. When the user requests a page, it is intercepted by Syskill & Webert and edited so that it contains 3 extra buttons for rating the page as "hot", "lukewarm" or "cold". With a couple of these ratings, the software agent calculates an approximate model of the user's interests and rates all the other links accordingly as interesting or not-interesting. This rating is shown with a little icon placed in front of the link, followed by a number between 0 and 1, which states the probability that the user would like the page. The more ratings Syskill & Webert gets, the more accurate are its recommendations. One major problem that occurred in a real-world setting was that most users didn't bother to give a rating.

The basic problems in user modeling with machine learning in web applications are similar to problems discussed in section 2.3:

- There is a need for a large amount of data, since the result of the algorithm is better the more input it gets.
- The data needs to contain an explicit message, such as the three distinctions hot, lukewarm and cold in the Syskill & Webert system. As mentioned above, there wasn't a lot of feedback from the users. This goes to show once more that users are not willing to do anything more than what they intend to do.
- There has to be a way to monitor changes in the user and transfer these to the model to keep it up to date.

A technical problem of machine learning in the web is that resources such as time and cpu-usage are restrictive factors. An algorithm with a lower percentage of accuracy is often preferable to one that is better at calculating but takes longer [16].

### 3.2.1 DOPPELGAENGER

A fairly large user modeling system that tries to take into account all these problems is DOPPELGAENGER, a system developed in 1995 by J. Orwant at the MIT [7]. Its original application was in personalized text-news. There are several dimensions to personalizing news; favourite topics, time of the day when news is usually read, level of importance of different topics and depth of the story, depending on how much time the user wants to spend.

---

of the example

DOPPELGAENGER collects all sorts of user data via various sensors. All user actions on a computer are tracked. These provide information on timely aspects, such as when the user logs in and how much time he spends in an application, on his actions, such as which tasks he performs or which queries he issues, and perhaps even on the schedule of the user, if he uses a computer-based agenda. Further there are sensors called Olivetti Active Badges that are physically attached to a user to monitor his locomotion, sensors in the user's chair to note when he sits in it and sensors that track eye-movement on the screen. These hardware sensors seem a bit futuristic and are only imaginable in an experimental setting such as a lab, not in real life. Nevertheless, DOPPELGAENGER has already included them in its concept.

In the case of modeling a user's news preferences and habits, the following algorithms are employed:

- Cluster analysis:
  For quickly making assumptions about a new user there is the technique of *cluster analysis*. In the whole user community the system looks for similar interests and clusters these in subsets. A new user is presented with a generic mixture of news. Based on his selections, the user is assigned to a certain community and the system suggests topics that other users in the community like.

- Beta distribution:
  To find out which topics the user really wants to read about, the system needs to extract meaning from text and identify the user's interests. The former is done by making use of commercial services that categorize articles. The latter is calculated with the *Beta distribution*. It depends on simple feedback of the user (like/ don't like) on an article he has read. With this data, the Beta Distribution determines how much the user likes a certain topic and how sure the system is of this value. The result is dodgy because there is no way of knowing why a user liked or disliked a text. Still, it is fast and simple. These requirements are more important in customizing news on the fly than high accuracy.

- Markov model:
  The *Markov model* is used to detect patterns in the user's locomotion when he wears sensors like the Olivetti Badges. By predicting when and where the user is likely to go next, DOPPELGAENGER can

deduce how much time he has and adapt the amount of content to that timeframe.

- Linear prediction:
  With *linear prediction* the time is calculated when the user will next read the newspaper. Based on a user's regular cycle, the formula predicts the future values. This information is used so that the news-providing application can have done all the above-mentioned calculations just before the user is expected to request the news.

The main focus in DOPPELGAENGER is on unobtrusive modeling. Ideally, the user wouldn't even notice that he is continually being observed. Of course this fact may be hard to ignore with an Olivetti Active Badge pinned on, but that might be just a question of advancement in technology and how common it is in everyday life.

More important is the issue of trust in DOPPELGAENGER. The problem is not only getting people to accept being monitored non-stop, but also to convince them that their sensitive data will not be misused. DOPPELGAENGER's approach is to provide a high level of data protection with the Kerberos authentication system and to use PGP to protect potentially sensitive user data in network transactions. Also, the user is informed on a weekly basis about the development of his model. He has the right to label the data as private, semi-private, semi-public and public. Still, this extent of user modeling may be beyond the level of acceptance of many people [7].

### 3.3 Generic user modeling systems

The gist of what has been said so far is that user modeling is a difficult task that requires a lot of thought and coding. Starting with GRUNDY in the late seventies, in the following ten years many user modeling applications were developed for different domains. Not until the early nineties were there efforts to separate the user modeling component from the rest of an application and make it reusable. These new systems were called user modeling shell systems, a term coined by Alfred Kobsa [5]. The user modeling component was like an empty shell that had to be filled up with relevant information about the target group, its characteristics and the assumptions about its members.

The DOPPELGAENGER system is not just a good example for machine learning, it is also a user modeling shell system. It has a server/client architecture. User profiles are stored centrally on the server, so that many different applications can share the data. The above mentioned algorithms

provide something like a toolbox of modeling methods. All applications can access the toolbox and process the data to model the part of the user that they need.

Other, more typical shell systems were built as components that could be incorporated into a single application. One important characteristic of all shell systems was that they were domain independent. Typical services provided by these shells include

- using stereotypes to model users,
- recording and analyzing a user's behaviour,
- clustering user communities based on the similarity of their interaction history,
- consistency maintenance of the user model and,
- a rating mechanism.

Few of the user modeling shell systems that were developed in the early nineties were actually used. In the late nineties e-Commerce boomed on the Internet. Soon the need for personalized web-interfaces became apparent. This approach allowed a transition from anonymous mass-marketing to much more effective 'one-to-one' marketing. Shell systems were developed specifically for modeling users in the web, especially in e-Commerce.

The modern versions of user modeling shell systems are quite different from their predecessors. The old shell systems were developed in an academic environment and related to AI problems and natural language dialogue. User modeling systems for the Internet were faced with completely different requirements. So as not to confuse the two, Alfred Kobsa coined a new term for them: generic user modeling systems [6].

Generic user modeling systems are composed as client/server architectures that serve various applications. A few of their most prominent features are:

- Quick adaptation:
  Visitors to a site must be intrigued and captured in the short period of their first visit. Commercial user modeling systems often offer different modeling methods that are suitable for a varying degree of available knowledge of the user.
- Extensibility:
  The user modeling system should be extensible so that new modeling methods can be added to it.

- Load balancing:
  User model servers must be able to handle changes in their average load. To avoid high response delays or even denials of service, they can distribute the load or employ less thorough user model analyses.

- Failover strategies:
  In case of a total breakdown a fallback mechanism is needed.

- Transactional consistency:
  A mechanism must be provided to prevent inconsistencies in the user model in case of parallel read/write actions on the user model and of abnormal process termination.

Future developments of today's generic user modeling systems might be in the field of mobile computing and smart appliances [6].

# 4 From user model to user adaptation

Although a lot has been said so far about techniques on modeling a user, the benefit for the user has not yet been made evident. The benefit for the user is that an ideal personalized system will seem to every user like it has been made just for him. It is pleasant and easy to work with, because it adapts to the user's preferences and abilities.

There are two kinds of adaptation: Adaptive vs adaptable. The difference between an adaptive system and an adaptable system is that the former learns from the user and adapts to him automatically, whereas the latter can be modified by the user himself. Both approaches have advantages and disadvantages for the user. A user working with an adaptive system requires no special knowledge, since the system will help him and guide him. On the other hand, the user will have difficulty developing a coherent mental model of the system because he does not understand the logic behind its changes. This problem was mentioned earlier in the description of a user's model of a system (see section 2.2.2). In fact, hardly any successful models exist of a purely adaptive system. The adaptable system however puts the user completely in control of its functionality. It is based on the assumption that only the user knows exactly what he wants, not the designer who develops the system. The disadvantage here is that the user has to put a lot of effort into learning to work with and to adapt the system ([4], p.3).

In general, there is never 100% certainty whether the assumptions made about a user are true or not. Any changes that the system makes should

therefore be transparent to the user. He should be in control of the degree to which it adapts to him.

The ideal personalized system mentioned above does not exist, but some systems come close to it. On the other hand, some are still miles away, as you may have experienced yourself. A few common forms of adaptation are described in the following, along with good and bad examples of their realization.

## 4.1 Plan recognition and recommendation

An unobtrusive way of helping a user and keeping him in control of the system is by recognizing what his goal is. Help can then be offered in the form of recommendations or shortcuts. This is usually done by keeping track of how most people solve a common problem and then explaining the pattern to new users.

An example of an agent who offers advice is the MS Office Assistant Clippy, a paperclip with eyes and eyebrows. It provides context sensitive help, giving tips and suggestions how to perform the task it assumes the user wants to do. When it was introduced in 1997, a news bulletin read the following:

> The availability of the new Office Assistant help tool from the American software giant Microsoft Corp. constitutes a significant commercial breakthrough for user modeling technology. For the first time, advanced user modeling techniques have been included within a standard software package targeting at the mass market. It can be expected that this important development will lead to further commercial applications involving latest user modeling techniques. Given the fact that Microsoft is a global player with a strong market position throughout the world, it is very likely that a global trend will emerge towards more adaptive interfaces which build upon sophisticated user modeling technology [15].

As it turned out, the assistant was very unpopular with users, who considered it more of a nuisance than a help. Clippy is therefore scrapped in following versions of MS Office [10].

An application that gives recommendations on the web is the e-commerce site www.amazon.com. Every customer is represented by a long-term user model that keeps track of all his purchases. Similarities in purchases of many

customers are detected. Customers are then clustered in shared-interest-communities (compare with algorithm "cluster analysis" in chapter 3.2.1). When a user looks for a certain item, he is informed about the purchases of other users who bought the same item. In my experience this feature is quite useful for getting to know different sources for a topic or discovering new authors or bands. Amazon also makes personal recommendations to users by assuming what their interests are, based on the collection of all items they have purchased. This view can be distorted, for example when the purchase is a present for someone else. For improvement and correction of their model the user can ask not to include a certain purchase in their profile, and he can rate the recommendations given on a scale of 5 as good or bad.

Amazon's user modeling doesn't force the user to do anything. He can choose whether he wants to rely on it or not. This is in my opinion a successful application of user modeling in e-commerce.

## 4.2 Content, layout and structure

Adaptation can also take place on the level of content, layout and structure. A large project in this field is the AVANTI system [1]. The aim of AVANTI was to adapt web information to the needs of individual users, especially taking elderly and disabled users into account. Tourist information systems were developed for three cities, Vienna, Rome and Kuusamo, to test the concept. They were accessible from anywhere on the net, and additional kiosks were set up where the user interface included special Input/Output devices such as a Braille display, a macro mouse, a speech synthesizer and visual and non-visual interface objects.

On the level of the hypermedia pages, there was adaptation of content, download time, structure and layout. The system made assumptions about the following characteristics of a user when building the user model:

- Interests and preferences concerning the content, like tourist attractions of the town, and concerning the presentation of information.
- Physical and sensorial abilities.
- Domain knowledge about the city, so that known locations could be used as reference points.
- Competence in handling computers and in handling the AVANTI system.

The system gathered both implicit and explicit information about the user and worked with stereotypes to assign an initial set of assumptions to a user.

The AVANTI system ran for 3 years in each of the above mentioned cities. The aim of the system was to find out whether the adaptability and adaptivity of the system was actually useful to the end users. It was evaluated by 180 people. At large, they were well satisfied with its adaptive functionality. Especially disabled users felt the system was designed to meet their needs.

For a better understanding of how the AVANTI system worked, take a look at the following example. It shows what kind of user data was gathered, which assumptions were made, and what the consequences were for the user interface.

Situation: user accessing AVANTI tourist web site of Siena
Location: EasyInternet shop in Siena
Screen size: 640 x 480
Bandwidth: 65K modem
explicit data:

| *Question* | *Answer* |
|---|---|
| Have you ever used a computer before? | Yes |
| Have you ever used the AVANTI system before? | No |
| Have you ever visited Siena before? | Yes |
| Are you interested in information for disabled people: | |
| Information for people with difficult deambulation | No |
| Information for blind people | No |
| Information for wheelchair-bound people | Yes |

[4]

Implicit data:

| *Action* | *Assumption* |
|---|---|
| Takes a longer than average time to click on links | Not familiar with navigation concept/ computers |
| Clicked on link "History of Siena" | Is interested in history of Siena |

With this information, the user interface can be adapted in the following way:

| *Element* | *Influencing Variable* | *Example of Adaptation* |
|---|---|---|
| Content | Knowledge of domain, interest in specific subject in domain | Give more detailed information to history of Siena and of main tourist attractions, add wheelchair accessibility |
| Structure/ number of choices on screen | Familiarity with navigation concept | Maximum of 3 links on main screen |
| Media items | Bandwidth, file type preferences | Small pictures, mainly text |
| Layout | Size of screen | Space text evenly in narrow columns for better readability |

Up to now this paper has given a thorough impression of how user modeling works. What it hasn't talked about is how the protagonist in this domain, the user, feels about it. Many users are apprehensive about the whole idea. Privacy and security play a major role in user modeling. The next section addresses some of these issues.

## 5 Critical issues: privacy and security

Although the initial goal of user modeling is to be of help to the user, the personal data gathered in the process can be misused. Especially in web-applications, many users feel uncomfortable with the fact that their actions are being tracked and their data stored. Surveys as described in [11] have shown that 77% of users are (extremely) concerned about being tracked online (Forrester 1999) and 74% are concerned about divulging personal information online (AARP 2000). Sites that require registration information are left immediately by 41% of users (Boston Consulting 1997) and 32% enter fake information (Forrester 1999).

There are privacy regulations in many countries that apply when a user is or can be identified. The EU has a general directive for all its members, who can further enhance it if desired, which has been done for example in Germany.

Different regulations apply in the USA. So as not to inhibit intercontinental trade or risk prosecution of American businesses by European authorities, the Safe Harbor Framework was developed [12]. American businesses that have joined the Safe Harbor comply with European privacy protection standards.

The EU directive decrees that personal data may only be processed for legitimate purposes. The following qualify as legitimate purposes:

- The fulfillment of a contract
- Taking steps at the request of the data subject prior to entering into a contract
- A legal obligation
- The protection of the vital interests of the data subject
- Public interest

Furthermore, only data necessary for the indicated purposes may be collected, and it must be deleted as soon as it has served those purposes. The data may not be processed further. Additional restrictions apply for very sensitive data (e.g., racial or ethnic origin, political opinions, religious or philosophical beliefs, trade-union membership), and the processing of data concerning health or sex life.

Further prohibitions especially affect the task of user modeling. It is not allowed to let a machine process user data and automatically make a decision about that person which might significantly affect him, e.g. when evaluating personal aspects such as performance at work, creditworthiness, reliability. In Germany, usage logs must be deleted after each session, usage logs of different services may not be combined and user profiles are permissible only if pseudonyms are used. Usage logs are considered critical because they offer a complete history of every step a user has taken. Also, a site must offer the user the possibility to get access and make payments anonymously or under a pseudonym, if it is technically possible and reasonable.

Users, on the other hand, have the right to be informed to which purpose their data is being collected. They have the right to get insight into what is stored about them. If this data is incomplete, incorrect or obsolete, they may demand blocking, rectification or erasure of it.

Some of the prohibitions mentioned above can be lifted if a user gives his unambiguous consent.

As for security, whoever controls the user data is legally obliged in the EU to make sure the technical and organizational environment protects this data from misuse.

If a web application wants to offer personalized services based on user modeling, it should comply with the following guidelines to stay legal.

- Make personalization an explicit purpose of the site.

- Obtain the informed and voluntary consent of the user.
- Provide organizational and technical means for the users to inspect, block, rectify and erase their data.
- Provide adequate security mechanisms.
- Allow anonymous or pseudonymous access if this is technically possible and reasonable. [11]

# 6 Conclusion

This paper provided an overview over the domain of user modeling. It described the goals of user modeling and the benefits for the user, but also the danger and the problems. Basic techniques such as stereotyping and machine learning were introduced along with examples of how they have been applied in different systems.

The Internet had a big impact on classical user modeling research and solutions. The goals and the problems have changed in this environment. The success of a website is measured in numbers of page hits and amount of time spent on the page. Websites need to spark a user's interest at first contact, and make him feel special and understood. Yet nothing is known about the user anymore. Internet access is everywhere and anyone can use it.

The growth of the internet has also lead to the problem of information overload. Finding that one piece of information is like searching for the proverbial pin in a haystack. Much research has been dedicated to making information retrieval and information presentation adaptive to the user.

One of these solutions is currently the subject of a research team at the CWI. They are developing a system that can generate multimedia presentations on the web at runtime, which are adapted to the individual user [3]. User modeling is employed to find out about a user's goal, knowledge, background and preferences.

Extracting all this information from a user with the help of the techniques described above is a challenging task. Whether user modeling is sufficiently advanced to provide the answer remains to be seen.

## References


[1] Josef Fink, Alfred Kobsa, and Andreas Nill. Towards a user-adapted information environment on the Web. In *Multimedia and Standardization 98*, 1998.

[2] Gerhard Fischer. User Modeling in Human-Computer Interaction. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 11:65–86, 2001.

[3] Lynda Hardman. Hypermedia Presentation Generation on the Semantic Web. *Ercim News*, 51:36, 2002.

[4] A. Kobsa, J. Koenemann, and W. Pohl. Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. In *The Knowledge Engineering Review*, volume 16, pages 111–155, 2001.

[5] Alfred Kobsa. Supporting User Interfaces for All Through User Modeling. In *Proceedings HCI International*, pages 155–157, Yokohama, Japan, 1995.

[6] Alfred Kobsa. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 11:49–63, 2001.

[7] Jon Orwant. For want of a bit the user was lost: Cheap user modeling. *IBM Systems Journal*, 35(3&4):398–416, 1996.

[8] Elaine Rich. Stereotypes and User Modeling. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, pages 35–51. Springer-Verlag, Berlin, Heidelberg, 1989.

[9] Elaine Rich. Users are Individuals: individulizing user models. *Int. J. Human-Computer Studies*, 51:323–338, 1999.

[10] MS Office's Clippy Gets Bent Out of Shape. http://www.osopinion.com/perl/story/8902.html.

[11] Privacy and Security in UM. http://www.ics.uci.edu/~kobsa/talks/privsecum/ppframe.htm.

[12] Safe Harbor explanation. http://www.export.gov/safeharbor/.

[13] Syskill & Webert.
http://www.ics.uci.edu/~pazzani/RTF/AAAI.html.

[14] UM in visualization supporting systems.
http://www.uni-paderborn.de/fachbereich/AG/agdomik/
arbeitsschwerpunkte/ucmm/idias/um/introduction.html.

[15] UM Research and Market Trends in Japan and USA.
http://www.dfki.uni-
sb.de/fluids/User_Modeling_Research_and_Market_Trends_in_Japan_and.
html#SECTION00414000000000000000.

[16] Geoffrey I. Webb, Michael J. Pazzani, and Daniel Billsus. Machine Learning for User Modeling. *User Modeling and User-Adapted Interaction*, 11:19–29, 2001.