PUC

Series: Monografias em Ciência da Computação,
No. 8/90

NEW APPROACH TO USER MODELLING BASED ON DOUBLE STEREOTYPES

Paula Y. Guaranys
Carlos J.P. Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453
RIO DE JANEIRO - BRASIL

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC RIO, Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
BRASIL

Tel.:(021)529-9386 TELEX:31078 FAX:(021)274-4546
BITNET:userrtlc@lncc.bitnet

PUC Rio - Departamento de Informática

Series: Monografias em Ciência da Computação, Nº 8/90

Editor: Paulo A. S. Veloso August, 1990

A NEW APPROACH TO USER MODELLING BASED ON DOUBLE STEREOTYPES

Paula Y. Guaranys

Carlos J. P. Lucena

This work has been sponsored in part by FINEP

New Approach to User Modelling Based on Double Stereotypes

Paula Y. Guaranys
PUC-RJ/Computer Science Dept.
Rua Marques de S. Vicente, 225
22453 - Rio de Janeiro - RJ
BRAZIL

Carlos José Pereira de Lucena
PUC-RJ/Computer Science Dept.
Rua Marques de S. Vicente, 225
22453 - Rio de Janeiro - RJ
BRAZIL

20 July 1990

Abstract

The quality of interactive computer systems can be evaluated on the basis of the system's ability to adapt itself to specific users. The system must be able to respond or react based on the knowledge it is capable of acquiring about a given user. A representation of the user's knowledge expressed in a way which is appropriate to the systems use is called a user model.

This paper presents a new method for the construction of a user model which can be used as part of a generic interactive system by a generic user. According to the method, the user model is the result of dynamic transformations of a static double stereotype. A first class of stereotypes represents the estimated user's knowledge about concepts implemented by the system. A second class structures the knowledge about system's concepts in such a way that it is possible to make inferences about whether the user knows a given concept given that he/she knows another concept. The level of knowledge used by both classes is determined by means of an experiment. As the software gets to be used the model for a specific user departs from the stereotypes and gets closer to a good representation of the current user with whom it wants to carry on an adequate interaction.

Key-words: user model, interfaces, stereotypes, interactive systems.

Resumo

A qualidade de sistemas de computação interativos pode ser avaliada com base na capacidade do sistema de se adaptar a usuários específicos. O sistema deve ser capaz de responder ou reagir em função do conhecimento que pode ser adquirido a respeito de um determinado usuário. Uma representação do conhecimento do usuário expressa de forma apropriada para uso de sistema é chamada um modelo do usuário.

Este trabalho apresenta um método novo para construção do modelo do usuário que pode ser utilizado em um sistema interativo e por um usuário genérico. Segundo o método o modelo do usuário é o resultado de transformações dinâmicas de um duplo estereótipo estático. Uma primeira classe de estereótipos representa o nível presumível de conhecimento do usuário sobre conceitos implementados pelo sistema. A segunda classe relaciona o conhecimento sobre conceitos do sistema, de tal forma que se possa fazer inferências sobre o se o usuário conhece um conceito dado que ele/ela conhece um outro. O nível de conhecimento adotado por ambas as classes de estereótipos e determinado através de um experimento. A medida que o software vai sendo utilizado o modelo de usuário específico vai se distanciando dos estereótipos e se aproximando de um boa representação do usuário corrente com o qual ele pretende interagir adequadamente.

Palavras chaves: Modelo de usuário, interfaces, estereótipos, interativo.

1 INTRODUCTION

The number of users that have access to interactive systems has been increasing considerably. The degree of familiarity of each individual user with these systems varies a lot. As a consequence, the quality of interactive computer systems can be evaluated on the basis of its ability to adapt itself to specific users.

For a system to be able to appropriately assist a given user it is necessary that it be capable of knowing that user. The system must be able to respond or react on the basis of the knowledge it can capture about that user. A representation of the user's knowledge expressed in a way which is appropriate to the system's use is called a user model.

The quantity and quality of information available in a user models may vary considerably. Different authors try to justify, in the recent literature, their criteria for formulating different kinds of user models.

Morik (1989) models the user in the context of a conversational environment entitled HAM-ANS (Hamburg Application-Oriented Natural-language System). The user model is built from various stereotypes which are combined to compose the profile of a particular user. The emphasis is placed in the natural language dialog which is generated on the basis of criteria formed through the use of the stereotypes.

In the GUMS system (A General User Modelling Shell), Finin (1989) has used three kinds of default reasoning: stereotyped reasoning, explicit default rules and negation by failure. The system constructs a tree of stereotypes. At each level the user is associated to some facts he/she is supposed to know. When a contradiction occurs, that is, the user knows something that does not belong either to his/her level or to the corresponding sublevels, he/she gets "promoted" in the tree. One difficulty associated with this method has to do with the possibility of overestimating the user's knowledge.

The method proposed in this paper parallels the ideas proposed by Rich (1989) and Chin (1989) about user modelling. In both cases stereotypes are also used. Chin (1989), in particular, also makes use of double stereotypes. For sake of comparison with our proposal, the main characteristics of the approaches used in Chin (1989) and Rich (1989) will be described at the end of this section.

The user modelling method proposed in this paper has been developed for use in a generic interactive system by a generic user. Initially the system interacts with the user assuming he/she has a level of knowledge equivalent to that of a selected class of users investigated by the system's designers.

The information supplied by the system designer, obtained through an experiment, is the primary knowledge base and constitutes the first stereotype. In this proposal we will refer to two kinds of stereotypes. Both are static for a given system and are the starting point for the creation of an individual user model.

The first class represents the user's presumable level of knowledge about the system as perceived by the system's designer. The other structures the concepts used by the system, so that the information about the user's knowledge for a given concept allows the system to infer his/her knowledge about another concept.

In each interaction, during the actual utilization of the system, the model for a specific user departs from the stereotypes and starts to portrait the current user. The user model is therefore established dynamically from two static stereotypes.

The method has been incorporated in the PUC's system (Planned User Communication System) described in Guaranys and Lucena (1989). PUC is a knowledge-based environment for the construction of software interfaces.

Rich's proposal (1989) developed as part of the GRUNDY system also consists of generating a user model from stereotypes. GRUNDY is aimed at advising users on how to choose books that correspond to his/her interests. The systems task involves querying a data base to recommend a book that is compatible with the user's profile, that is, that will provide him/her with a pleasant reading.

The user profile is captured by means of stereotypes that define classes of users. One instance of class is "sportsperson" which describes all the characteristics of a person who enjoys sports in general. This approach to modelling often forces conflicts to the system since, in general, the user needs to be described by more than one stereotype (he/she may belong to more than one class).

To handle possible conflicts Rich (1989) stores assertions about the user associated to justifications about each stored assertion.

The use of stereotypes allows a system to reference a large number of concepts based on a small number of them. This may also lead to the production of undesired results. A problem may occur when the system is not able to classify a specific group of users in a class. Besides, observation gathered during the interaction about an individual cannot efficiently be used to review the initial stereotypes.

The update of the stereotypes is achieved through the change of the values of their attributes. A stereotype is a collection of attributes that co-occur in one individual. The alteration is propagated throughout all the related attributes. For some cases some complex calculations may be involved.

Chin's proposal (1989) incorporated in the KNOME system influenced the approach used in this work since it suggests the utilization of a double stereotype. KNOME uses a set of stereotypes that classify the user's knowledge and the nature of the information that he/she is going to use. As far as the knowledge is concerned, the user may be classified as novice, beginner, intermediate or expert. The information may, in turn, be classified as simple, mondaine or complex.

KNOME is the first system that uses a modelling scheme based on double stereotypes. The justification for that is mainly efficiency, since this process allows the system to store for each fact only its level of difficulty. That is, it does not have to record the relationships between the fact and all the user's stereotypes. Consequently the main problem in KNOME becomes the determination of the user's level of expertise. This is established during the systems interaction with the user which in KNOME needs to be very short.

In the next sections a new method for user modelling via double stereotypes will be presented. The method circumvents some of the difficulties found in GRUNDY and KNOME. As the method is presented it will be simultaneously compared with the two above mentioned research efforts ((Chin,1989) and (Rich,1989)).

2 CENTRAL CONCEPTS USED BY THE METHOD

The method is aimed at generating and maintaining a user model. The model is the result of dynamic transformations applied to a static double stereotype.

Many systems model the users by asking a number of initial questions (Macmillan,1983), others ask the user to talk a little about themselves (Rich,1989) while others yet ask the users to classify themselves (say, as beginners or intermediates). These approaches may lead to good results in the case of very special purpose pieces of software. Nevertheless, for general purpose interactive systems, such as text editors and spreadsheets, all of the above seems to fail. In most cases the user is not capable of analyzing himself/herself for classification purposes. If he/she knows little about the software he/she will not even be able to react. Besides, it may happen that the user misclassifies himself/herself. Users may think, for instance, they belongs to the intermediate level and the system may expect more expertise from members of that group than they actually have. As a result of the above consideration the method here described allows a system to use the metaphor of "looking over the user's shoulders" to be able to generate and/or update the model of the current user.

For the system to be able to "observe" the user during his/her interaction with a particular piece of software it needs some a priori knowledge on which to base itself. That is the reason for the use of stereotypes.

Double stereotypes are even more adequate than simple stereotypes for the construction of individual user models.

Figure 1 shows the stereotype represented by the function "probable user's level of knowledge" (working hypothesis of the system's builder). Level of knowledge is a function which relates the concepts implemented by the software to values which represent confidence factors (CF's) attributed by the designer to the user's knowledge about that particular concept. In figure 2 the method introduces a new idea.

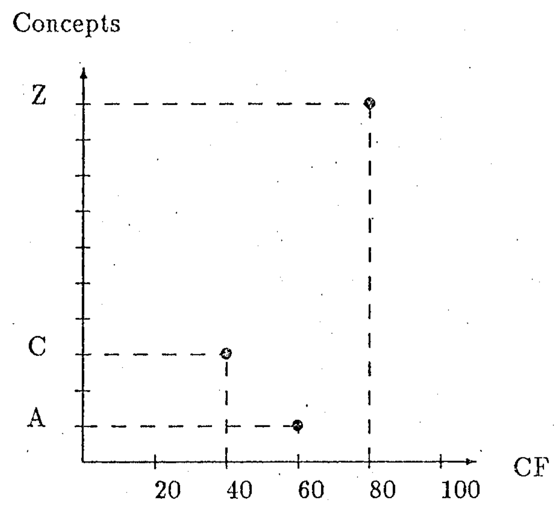

Figure 1:

The stereotype inter-relates two concepts implemented by the software and associates them to a confidence factor. In the vertical axis all the relevant concepts supported by the software are represented. Each concept is cut by a plane parallel to the other two axes. Each point in this plane when projected in the plane defined by the other two axes establishes a relationship between another concept and a confidence factor. Each plane is a stereotype of the level of knowledge that exists about a concept with respect to another concept.The double stereotypes in both this proposal and in Chin's (1989) represent the user's level of knowledge. In KNOME the other part of the stereotype represents the level of difficulty of the information handled by the user. In both cases double stereotypes are used with the same general goal: to analyze the possibility of a user knowing a given concept.

The duplicity allows the reduction of the amount of information stored about a given concept with respect to a user. On the other hand it increases the amount of processing required to get information about a concept with respect to a user.

In the present method the other part of the stereotype is used to get various new pieces of information about the same concept to improve the quality of the user model. In figure 2 the central concept is located in the vertical axis and has a plane associated to it. The new information is the relations obtained with the projection of the points in this plane over the plane formed by the horizontal axis. If no inconsistency is generated each relationship established is added to the user model. In other words, it is incorporated to figure 1.

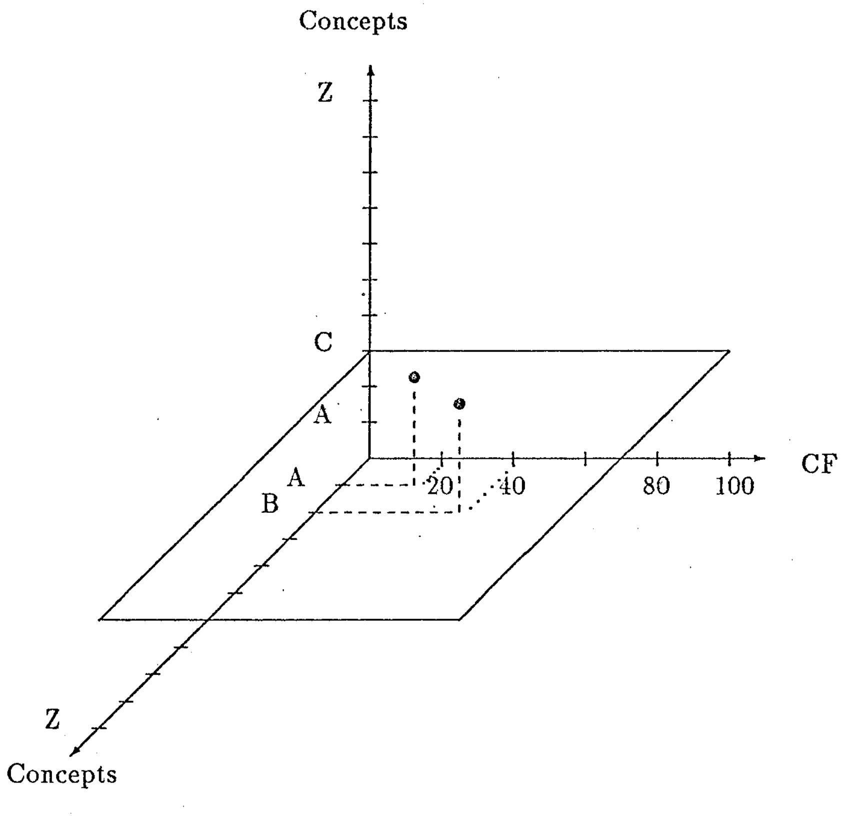

Figure 2:

As new interactions take place the method produces a less "stereotyped" and more "individualized" user model. The amount of information in the knowledge base is increased, but the steps taken towards achieving a particular user model are decreased.

The double stereotype in the way it is used in the proposed method also solves the need for complex computations used by Rich (1989) as a way of altering the values of the confidence factors (CF's) associated to the concepts. In fact, the stereotype of the level of knowledge of one concept with respect to another concept directly provides the new value of a CF of another concept.

The use of the double stereotype also eliminates the need for propagation of an alteration in the values of the attributes throughout all the concepts pertaining to different levels (as it happens in Rich's work (1989)). The possibility of placing two concepts in the vertical axis of figure 2 has also been explored. For this purpose, the relationship between pairs of concepts with all other concepts was evaluated through an experiment. The result was a stereotype without much significance which could only be useful in very specific cases.

It is important at this point to describe how the double stereotype will be formed, that is, how to get the values of the attributes that constitute the stereotypes. There are two possibilities. The first is to determine the values solely on the basis of the judgement of the software designer based on his experience in developing systems to certain classes of users. This approach is widely used.

The experimental approach was preferred in the case of this method. The simplest reason for deciding for this approach stems from the very nature of stereotypes: they constitute a form of default reasoning. Therefore they are better when they result from an experiment and even better when the experiment takes place with initial users. The need for the users to be at the same level is related to the way the experimental data are interpreted to form the double stereotype.

To end this section it is necessary to describe the generation of the user model from the double stereotype. The detailed description takes place in the next section. For now it is sufficient to ascertain that the model is developed during the interaction of the user with the selected piece of software. In what follows a comparison between the principles used in this method and the ones used by Rich (1989) and Chin (1989) is presented.

Rich groups the stereotypes into classes of users. There are several possible classes in a given system and the user may belong to more than one of them. That produces conflicts whose solutions are not very simple. The present proposal is very different in that respect. Here only three classes are used. Depending on the number of concepts known by the user and the level of difficulty of each one of these concepts (determined by the stereotype) the user's class is specified.

According to Rich (1989) the user is "this" or "that". In this approach the user "knows this" or "knows that". Rich's system (1989) was designed to respond to a query. In the present case the interest is to adapt the interface of interactive software to the characteristics of its users.

Chin (1989) in a way similar to this approach needs to check whether a user knows a given fact. An individual user model is also developed from a static stereotype and whenever an information is required the individual model is checked first. Only in the absence of information the appropriate stereotype is used. But in Chin (1989) not all information gathered is also stored. The evaluation of the user continues to depend on the stereotype. In this proposal the stereotype is slowly discarded until we finally get only the individual model. The difference is also a consequence of the goal of KNOME. It is used as part of the UC system (Unix Consultant) (Wilensky and Arens and Chin,1984) which helps user access features of the Unix operating system. Each consultation section is, in general, very short. Therefore it is not practical in KNOME to keep a complete user model. As indicated before the method here described is to be used as part of a generic interactive systems which may produce as a result of its computations a tutorial designed by an authoring system or a text produced by a text editor. In particular, the method is being used as part of an environment which supports the design and implementation of user interfaces.

3 SPECIFICATION OF THE PROPOSED METHOD

The last section has covered the motivations for the method presented in this paper. We are now going to describe in detail the attributes that form the double stereotype and the necessary steps to determine the values of the attributes via an experiment. The generation of the user model from the double stereotype is also described.

3.1 The Double Stereotype

The double stereotype is formed by two sets of information. The first is called the user stereotype and the second is called the knowledge stereotype. Both are specified as sets of facts. The validity of each fact is expressed through a confidence factor (CF).

The user stereotype represents the system's belief about the user's knowledge with respect to a plan. A plan is the necessary strategy to activate a concept provided by the system. The knowledge stereotype represents the connection between two arbitrary plans, given that the user knows one of them.

The attributes that define the user stereotype are all the plans contained in a given piece of software and the CF's associated to them. Consequently, the number of facts in the stereotype is equal to the number of plans in the software. The attributes of the knowledge stereotype are pairs of plans, say PlanX and PlanY and a CF, where CF is the value associated to PlanY given that the user knows how to follow PlanX.

The values of the CF's are provided by the software designer for the user's stereotype (the value ranges from 0 to 100). The definition of the attributes to both the user and the knowledge stereotypes involve the development of an experiment.

3.2 Description of the Experiment

The validation of the double stereotype method required experimentation with a relatively complex and highly interactive software which does not use an on-line help system. The experiment to be described was carried out with the Talisman system (Staa,1988) and involved thirteen subjects.

Talisman is a software engineering environment composed of several integrated tools that support all the phases of a software development process. For the experiment, a specific tool called TALIS-DFD was selected (its on-line help features were turned off). The tool supports the creation, maintenance and display of functional specifications of agents and processes, represented by data flow diagrams, transition diagrams and data dictionaries.

Initially, all the subjects were exposed to a two hour presentation about the tool's structure and utilization. All thirteen subjects were computer science students with no knowledge about Talisman and some knowledge about its underlying methodology. They all have been informed about the necessary actions for the execution of the plans associated to the software. Each subject took notes about the expert's description of the tool.

TALIS-DFD operates in two modes: a graphic and a logic mode. In the graphic mode it is possible to execute operations on a number of objects: processes, external entities, data storages and labels. The operations in the logic mode deal with data flows. This allows for the establishment of connections between the various objects. The available operations to deal with both the objects and the data flows are the following: create, remove, move and change (specifications). In the first case the only possible alteration is the change of names. As for the operations with the data flows it is possible to alter their direction, label and format.

After the collective introduction to the system, the subjects started to work with it individually. They were given a state of a design expressed in TALIS-DFD which described a given status of a functional specification and asked to develop it into an evolved version of the same specification (another state). To execute the task at hand each user needed to know all the plans involved in the software (the software features). They were requested to accomplish the task in one hour. Each individual section, carried by the various users, was taped recorded (subjects were asked to give explanations about why a plan has been chosen). Talisman (1988) also recorded the history of interactions with the system.

As a preparation for the compilation of the data all admissible plans in the software were classified. A plan is syntactically expressed as a predicate with three arguments. Semantically speaking the plan is a relation between the objective to be accomplished (its final effect), the way to accomplish it (the necessary action) and the state in which the application needs to be in order for its execution to be possible (the pre-condition).

As an example suppose a user wants to remove the object "external entity" from the diagram. The plan would look as follows: the end result is "remove external entity", the required action is to press the DEL key and the pre-condition is "cursor pointing to an external entity".

Not all plans are as simple as the one in the above example. In many cases several actions have to take place so that the end result is achieved. That happens, for instance, when the desired effect is the connection between two objects. It is necessary to select the object in one end, the object in the other end, and then specify the direction and the format of the connection. Each action corresponds to a plan that, in turn, has a pre-condition, an action and an effect. The complete plan is obtained by transforming the current pre-condition into an effect to be produced and looking for the corresponding action and pre-condition. The sequence bellow illustrates how to get the actions. In each plan the arguments refers to the pre-condition, action and effect respectively.

plan("establish the connection format", <ENTER>,"create connection")
plan("the selection menu is active", arrows,"establish the connection format")
plan("select the destination object", <ENTER>,"the selection menu is active")
plan("select the origin object",arrows, "select the destination object")
plan("cursor pointing to object", L and INS keys,"select the origin object")

In the preceding paragraphs two examples illustrating plans with different complexities have been presented. It is important to note that the data base stores plans as opposed to nested plans. The system is responsible for generating and recognizing nests of plans.

Knowing what a plan is and the number of users that were capable of using it is the requirement to generate the user's stereotype. It contains the relation between each plan and the CF determined by the experiment. A final aspect to comment about the experiment is the fact that one user was able to perform one hundred percent of the given task. That means that the information provided by the expert was sufficient (complete) for the task to be accomplished.

3.3 Determination of the Confidence Factors

The relationship between each plan and the corresponding CF's form the facts that compose the user stereotype. The number of facts is equal to the number of plans that the designer supplied to the system.

The CF, therefore represents the proportion of users that know a given plan, that is:

$$CF = \frac{\text{number of users that have executed the plan}}{\text{total number of users in the experiment}} \times 100$$

The value of the CF is the integer $0 <= i <= 100$ closer to the result in hand. This approximation applies also to the value of the CF's used in the knowledge stereotype.

The knowledge stereotype is formed by relating a given PlanX to a PlanY and establishing the proportion of users that having executed PlanX also executed PlanY. That is,

$$CF = \frac{\text{number of users that know both PlanX and PlanY}}{\text{number of users that know PlanX}} \times 100$$

Since the stereotype is the inference base for the development of the individual user model, the values of the CF's associated to the facts in the stereotypes are extremely important. To make sure that the stereotype will not overestimate the individual user it is necessary to underestimate the CF's. This is the reason why the result of the experiment will be more reliable if the experiment is conducted with beginners.

The knowledge stereotype is only viable if the values of its three attributes, PlanX, PlanY and CF result from an experiment with homogeneous users. If the level of knowledge differs considerably among the users that participate in the experiment the idea of inferring the CF of a PlanY from the level of knowledge of a PlanX looses its value.

Another advantage of specifying the double stereotype according to the approach described, that is, an experiment with beginners, is that the stereotypes, in particular the user stereotype becomes user independent. Up to now we did not describe how to establish a relationship between stereotypes and types of users. The association is established through the use of rules.

At the beginning of the interaction of the user with the system, while the user's knowledge is represented by the user's stereotype, he/she is considered a beginner. As the interaction takes place and the model begins to develop according to its new CF values the classification may change allowing the user to be classified as intermediate or expert. This will be further discussed in the next section.

3.4 The User Model

The user model is also represented by a set of facts. At the beginning of the interaction of the user with the system the set is empty. During the interaction the facts that constitute the user model are inferred.

Independently of the method used for its generation, all facts of the user model have a structure which is identical to the one used by the user's stereotype, that is, one which relates one plan in the software to a CF.

The need to establish the consistency of the information that are embedded in the facts that represent the user's knowledge calls for an inference engine which is able to operate in two directions, that is, incrementing or decrementing the values of the CF's.

The increment is possible due to the existence of the knowledge stereotype. The decrement is guaranteed by the existence of the information generated and stored in the facts' base during the increment. It allows for the recovery of any previous state as well as the generation of the history of the interaction.

The system maintains a list with all the plans that it believes to be known by the user, that is, the ones with a $CF = 100$ in the individual model. The user is classified as an expert if all his/her plans with $CF <= 10$ in the user's stereotype belong to this list; intermediate if more than half of the plans such that $50 < CF < 70$ belong to such a list. If none of the two criteria apply the user is classified as beginner.

According to the above criterion the variation of the interface during the interaction with the application is slow, leading to smooth changes in the software behavior.

One may inquire why not to classify as an expert a user that knows everything that a beginner knows plus all that an intermediate user knows plus all that only a few know. The answer is: the probability of a user knowing all the plans that only a few know and not knowing a plan that the majority knows is so small that this situation was disconsidered. The same rationality was used for the classification of a user as intermediate.

The approach used for classification is also considered a contribution since it enables a straightforward and efficient procedure for the analysis of user types. It is simple because it is reduced to a decision among three possible states. It is efficient because the conditions for each one of the states are very generic and have their assumptions validated on the basis of an initial prototype that considers the users that participated in the experiment. Another advantage of the approach is the physical independence between the stereotype and the type of user since it is established by means of the described rule. This allows the designer to make adjustments in the interval used for the CF's according to each new user classification. This independence is necessary since software with different functionalities may require the use of different CF intervals.

In what follows the generation and maintenance of the user model are described. In each interaction of the user with the software the system infers whether he/she knows a given plan. When the system infers that the user knows one it checks whether it already has this information. In the affirmative case nothing else needs to be done. Otherwise, this information has to be updated and propagated throughout all plans which are directly related to the executed plan. Later the system checks whether the current user classification remains unchanged according to the above mentioned procedure. When necessary, the current information is replaced by the correct information. The decision procedure bellow summarizes what has been described.

```
IF      not(user_knows(100,Plan))
THEN
        update user's knowledge (Plan) and
        update user classification
END
```

The update of the user's knowledge involves three actions. The CF associated to the executed plan is incremented to 100 and this alteration is recorded. The next action, which is the central part of this rule, involves the propagation itself. The knowledge stereotype is responsible for the propagation. The plan executed is the one designated as PlanX in the specification of the knowledge stereotype. The directly related plan is called PlanX. The CF is the value associated to PlanY and becomes part of the user's individual model, provided that no contradiction is generated. Each alteration of the CF value associated to PlanY as a consequence of PlanX is also recorded. This information will be used later in two situations. It will allow the user to access a history of the system's inferences and allows the system to recover a given state. The set of steps described as follows summarize the actions above.

```
update user's knowledge (PlanX) $\longleftrightarrow$
    record fact: user_knows(100,PlanX) and
    record fact: change CF value associated to PlanX
    FOR EACH fact in knowledge stereotype related to PlanX:
        instantiate: knowledge_stereotype(CF,PlanX,PlanY) and
        instantiate: user_knows(Current_CF,PlanY) and
        get_highestvalue(Current_CF,CF,PlanX,Highest_CF) and
        update:      user_knows(Highest_CF,PlanY) and
        record fact: change the value of CF associated to PlanY
                     as a consequence of PlanX
```

If the system concludes that the user does not know a given plan, inverse actions, that correspond exactly to the previous ones, are executed. The difference lies on the actions corresponding to the update of the user's knowledge. The following sequence of steps summarizes what has been said.

```
update user's knowledge (PlanX) $\longleftrightarrow$
    eliminate fact: user_knows(100,PlanX) and
    eliminate fact: changes CF value associated to PlanX
    FOR EACH fact in knowledge stereotype related to PlanX
        instantiate:     knowledge_stereotype(CF,PlanX,PlanY) and
        eliminate fact:  change the CF value associated to PlanY
                         as a consequence of PlanX and
        analyse and decide NewCF value associated to PlanY and
        record fact:     user_knows(NewCF, PlanY)
```

The user model for a given piece of software is represented by the facts in the individual model, by the double stereotype, by the information recorded as a consequence of changes in the CF's associated to the plans and by the user's classification. At a given moment, the system's belief about the knowledge of the user being modelled is given by the information that exists in the individual model and only when it is not available the stereotype is used.

4 CONCLUSIONS

User modelling is specially important for interactive software systems that are used by heterogeneous groups of users which require some flexibility in the way they interact with the computer. This is the case of complex software environments such as PUC (Guaranys and Lucena,1989) where the final user is expected to be able to compose the interface for his/her application. This has provided the initial motivation for this work.

Nevertheless, the proposed method for user modelling is independent of the application domain and of the software functionality. Therefore, the method may be incorporated to interactive software systems in general.

In the method the model evolves from a stereotype which is used as a form of default reasoning. The form of reasoning adopted in the method is non-monotonic since this is the only possible way of establishing the consistency of the facts' base.

The method utilizes a user stereotype and a knowledge stereotype which are obtained from an experiment conducted with initial users. This approach was used because it allows a substantial reduction in the errors that may be introduced when a software designer estimates, independently, the level of the user's knowledge. The experiment used to validate the method has been conducted with beginners so that we work with the worst case, that is, a conservative double stereotype. Practice has demonstrated that this approach leads to a higher satisfaction of the end user.

In this method the formation process of the user model is slow at the beginning. As the system "gets to know" the user the number of updates in the database is reduced and, therefore, the response time increases. This does not constitute a problem because a user during the first interactions with a new software expects slow reactions because, otherwise, he/she may get inhibited (Shneiderman,1988). As the user gets familiarized with the software the response time may increase.

The independence between the type of user and his stereotype, as described in this paper, has the additional advantage of modularization. The separation allows for adjustments, both in the stereotypes and in the intervals used for the classification of the CF's. This is compatible with the proposal since the double stereotype remains static. The analysis of evaluation mistakes made during the testing phase determines the alteration in the CF values.

The modelling method proposed in this paper, when incorporated to a system with help characteristics gives way to a software which is capable of operating according to the "looking over the shoulder" metaphor.

The method has been implemented in a SUN Workstation as a module of the PUC (Guaranys and Lucena, 1989) system.

5 REFERENCES

CHIN,D.N.,(1989). "KNOME: Modeling What the User Knows in UC";In Alfred Kobsa & Wolfgang Wahlster, editors, User Models in Dialog Systems;Springer-Verlag.

FININ,T.W.,(1989). "GUMS - A General User Modeling Shell";In Alfred Kobsa & Wolfgang Wahlster, editors, User Models in Dialog Systems,Springer-Verlag.

GUARANYS,P.Y. & LUCENA,C.J.P.,(1989). "PUC: A Knowledge Based Environment for Planned User Communication",In: COMPSAC 89 - Computer Software & Application Conference, Orlando, Florida.

MACMILLAN,S.A.,(1983). "User Models to Personalize an Intelligent Agent"; Ph.D. Thesis; School of Education, Stanford Univ.;Stanford,CA.

MORIK,K.,(1989). "User Models and Conversational Settings: Modeling the User's Wants"; In Alfred Kobsa & Wolfgang Wahlsters,editors,User Models in Dialog Systems,Springer Verlag.

RICH,E.,(1989). "Stereotypes and User Modeling";In Alfred Kobsa & Wolfgang Wahlsters, editors, User Models in Dialog Systems,Springer-Verlag.

SHNEIDERMAN,B.,(1988). "Designing the User Interface Strategies for Effective Human-Computer Interaction",Addison-Wesley Publishing Company.

STAA,A.v.,(1988). "Talisman: Reference Manual"; Staa Informatica,(in portuguese).

WILENSKY,R. & ARENS,Y. & CHIN,D.N.,(1984). "Talking to UNIX in English: An Overview of UC";In: Communications of the ACM 27,574-593.