Building adaptive applications on widely-used platforms with BGP-MS *

Wolfgang Pohl and Jörg Höhle
Dept. of Mathematics and Computer Science, University of Essen
{Wolfgang.Pohl,Joerg.Hoehle}@gmd.de

Josef Fink
Institute for Applied Information Technology
German National Research Center for Information Technology
Josef.Fink@gmd.de

Do-Wan Kim
Information Science, University of Regensburg
KIM@alf4.ngate.uni-regensburg.de

Abstract

Today, standard software on widely-used platforms is employed by a large number of people. This user population is often quite heterogeneous, because people tend to have different preferences, knowledge, goals, etc. In order to satisfy individual needs, software systems can employ user modeling techniques to adapt their behaviour to each user. User modeling shell systems provide these techniques for application developers, but are rarely available on widely-used platforms. In this paper, we present BGP-MS-Jr., a limited version of the user modeling shell BGP-MS, which runs under MS-Windows. BGP-MS-Jr. is being used as the basis for a user modeling system that serves as an intelligent component of WING-MIT, a tutorial help system. We describe how WING-MIT employs this user modeling system to realize adaptive behaviour. However, in order to bring full BGP-MS functionality to all platforms and to satisfy the demands of distributed applications, a new system architecture is needed. We suggest a generic and flexible architecture that will make BGP-MS platform-independent, applicable in distributed contexts, and configurable according to the needs of a wide range of applications.

1 Introduction

Users of interactive software systems have individual preferences, knowledge, goals, interests etc. But they do not only differ with respect to such mental categories, they may also have very specific needs caused by physical disabilities. Designers of software systems have long tried to cope with the heterogeneity of the user population by offering facilities, which allow the user to change the system's behaviour to a certain degree, according to his preferences.

Oppermann [Oppermann, 1991] mentions that users rarely take advantage of such features, which often demand a lot of experience and knowledge to be appropriately employed. Alternatively,

*The research described in this paper has been partly supported by the German Society for the Advancement of Scientific Research under Grant No. Ko 1044/4-3

software systems may be enabled to adapt their behaviour with respect to the current user in a self-controlled way. Adequate self-adaptation must be based on assumptions about the user; these assumptions must be formed and then stored and maintained in a *user model*. In order to perform these tasks, mechanisms are needed that "draw assumptions about the user based on his interaction with the application system, represent and store these assumptions, draw additional assumptions based on initial assumptions, take appropriate actions when inconsistencies betweeen assumptions are detected, and supply the application with current assumptions about the user" [Kobsa and Pohl, 1995]. A user modeling system must be developed that comprises the required user modeling techniques and puts them at the disposal of the interactive application.

Until recently, developers of adaptive applications had to program their user modeling systems from scratch. In the last years, however, there have been several attempts to bring the most important, user modeling techniques into application-independent user modeling shell systems [Finin, 1989; Brajnik and Tasso, 1994; Vergara, 1994; Orwant, 1995; Kobsa and Pohl, 1995; Kay, 1995; Paiva and Self, 1995]. An application developer can select a suitable shell system and fill it with user modeling knowledge of the application domain. Thus, the shell is the basis for a user modeling system that cooperates with the application in order to realize adaptivity.

Like most shell systems, also the BGP-MS user modeling shell [Kobsa and Pohl, 1995] has originally been developed on high-end workstations. However, if adaptive features shall be provided in interfaces for all, user modeling techniques must be available also on standard and widely-used application platforms. In the BGP-MS project, we have made a first step towards this direction. This paper presents BGP-MS-Jr., a limited version of the user modeling shell system BGP-MS, which runs on PCs under MS-Windows. The first application to employ this system was KN-AHS, an adaptive hypertext system [Kobsa *et al.*, 1994]. In this contribution, we show how BGP-MS-Jr. has been used in developing a user modeling system for the intelligent tutorial help system WING-MIT [Kim, 1995]. For future applications, the limitations of BGP-MS-Jr. have to be overcome. Therefore, we suggest a new and flexible architecture that helps to bring every desired BGP-MS functionality to widely-used platforms, and additionally copes with the demands of distributed applications. The paper ends with a summary.

2 BGP-MS-Jr., a user modeling shell for MS-Windows-PCs

2.1 BGP-MS-Jr. is BGP-MS for PCs

The current version of the user modeling shell system BGP-MS that is available for use on MS-Windows PCs is limited in comparison to the full workstation version. The limitations concern the power of the user model representation component and the graphical interfaces for application developers. There were mainly technical reasons why BGP-MS has not been ported fully onto the PC platform until now. First, its graphical interfaces are based on GINA (a Lisp- and Motif-based tool for generating interfaces [Spenke *et al.*, 1992]), which is only available for UNIX workstations. Since there was an urgent need for making BGP-MS work with its first PC-based applications, we decided not to port the interfaces using a PC graphics package. Second, for each of its inferences in predicate logic, BGP-MS employs several instances of the logical reasoning engine OTTER [McCune, 1994] which have to be executed in parallel. This could not be adequately realized in MS-Windows, due to its cooperative scheduling strategy.

The resulting system is called BGP-MS-Jr. It has no graphical interfaces, and there is no support for logical representation and inferences. The first limitation reduces the comfort of BGP-MS for the developer, but does not constrain the use of any BGP-MS user modeling feature. The second

limitation is more problematic. Not only is predicate logic a mighty representation and inference formalism for use within BGP-MS. But also the logic-based "view-connecting inferences" that allow the representation of both negative assumptions about the user (e.g. "the system believes that the user does *not* believe something") and relationships between different aspects of the user model (e.g. "if the system believes that the user believes that one of his goals has a consequence, this consequence is also a goal") are prohibited. The first applications of BGP-MS-Jr. did not need predicate logic but were designed to make use of negative assumptions. So, we had to develop a new approach to the representation of this assumption type.

In the following section we will give a brief overview of the facilities BGP-MS-Jr. offers for user model representation, among them the new mechanism for negative assumptions. User model acquisition can be supported by application-specific inferences, which we mention in a short section. However, the main tool for user model acquisition is the stereotype management component, which is presented afterwards. Finally, we summarize how BGP-MS-Jr. is applied in the development of a user modeling system.

2.2 Representing user models in BGP-MS-Jr.

The mechanisms available for user model representation in BGP-MS-Jr. form a subset of the facilities of full BGP-MS, with one exception: there is a different technique for representing negative assumptions about the user. This section describes these mechanisms in brief. For more details, see [Kobsa and Pohl, 1995].

Partitions structure the user model The partition mechanism, which forms the outer representation layer in BGP-MS, allows one to represent different types of assumptions about the user as well as the system's domain knowledge in separate partitions. Partition hierarchies can be built by introducing inheritance links between partitions. Thus, it is possible to avoid redundancy in the user model by putting common contents of a set of partitions P_1,...,P_n into a common superordinate partition P_s, from which P_1,...,P_n will inherit at the time of retrieval. It is important to note that the partition mechanism performs this procedure automatically in order to keep the user model free from redundancy. Leaf partitions of a hierarchy are also called *views* if we refer to all their (possibly inherited) contents.

Names of standard partitions (for positive assumptions) are of the form {<actor><modality>}+, where <actor> may be S (for System), U (for User), or M (denoting Mutuality), and <modality> may be B (for Believes) or W (for Wants). Examples of typical partitions are (see also [Kobsa and Pohl, 1995]):

- SB (System Believes) is used to store domain knowledge which BGP-MS possibly needs for carrying out its user modeling tasks.
- SBUB (System Believes User Believes) is supposed to contain the "privately" held assumptions of the system about the user's beliefs about the application domain.
- SBMBUW will contain the mutual beliefs of system and user about the user's goals (with respect to the application).

Partitions are also employed for the representation of stereotypes. Stereotypes are models of expected subgroups of the potential user population. Currently, there may be a belief partition for each stereotype. For example, if an application developer wants to model the group of users that are novices with respect to the application domain, he can use a partition NOVICEB (for Novice Believes) to represent the potential knowledge of members of this group.

Terminological knowledge fills partitions Within partitions, assumptions can be represented with the conceptual knowledge representation language SB-ONE. SB-ONE is a member of the KL-ONE family of knowledge representation systems which has been succeeded by terminological or description logics. It can be used to represent domain knowledge in partition SB or assumptions about the familiarity of the user or a user group with concepts of the application domain. But also other representation purposes can be achieved with SB-ONE.

The application can make entries into the user model using the 'Belief and Goal Description Language' BGDL. This language allows the formulation of restricted expressions of modal logic in a Lisp-like fashion. The main structure of BGDL expressions is $\mathcal{M}p$, where $\mathcal{M}$ is a sequence of modal operators (with negations), and p is an expression of first order predicate calculus (FOPC). BGDL expressions refer to knowledge in partitions as follows: $\mathcal{M}$ refers to a partition, and if p denotes terminological knowledge, it refers to a corresponding SB-ONE structure in the given partition. Two examples for BGDL expressions are:

- `(B S (:concept whale))`
  The concept 'whale' is part of the system's domain knowledge; this expression refers to an SB-ONE concept with name 'whale' in partition SB.
- `(B S (B U (all x (-> (whale x) (fish x)))))`
  The user is assumed to believe that whales are fish; this expression refers to an ISA relation between the SB-ONE concepts named 'whale' and 'fish' in partition SBUB.

BGDL expressions are also used for queries to the user model. Such a query has a BGDL expression as parameter; the answer is positive, if and only if the expression refers to a valid user model entry.

Negation partitions contain negative assumptions In order to allow the representation of negative assumptions (e.g., assumptions about what the user does not believe or know) without the mechanisms of predicate logic and view-connecting inferences, negation partitions have been introduced. A typical negation partition is SB$\sim$UB (System Believes not User Believes), which is supposed to contain explicit assumptions about what is unknown to the user with respect to the application domain. Contents of negation partitions can also be referred to by BGDL expressions:

- `(B S (not (B U (:concept mammal))))`
  The user is assumed not to know the concept 'mammal'; this expression refers to the SB-ONE concept with name 'mammal' in partition SB$\sim$UB.

It is important to note the difference between contents of SB$\sim$UB and knowledge that is not contained in SBUB.

Negation partitions must be treated differently from standard partitions, because no inferences are allowed within negation partitions. We will give an example to illustrate this: If the user is assumed to know that dolphins are whales (represented by an SB-ONE subsumption relation between the concepts 'dolphin' and 'whale') and that whales are animals, he is assumed to implicitly know that dolphins are animals. However, if the user is assumed to not know both ISA relations, it cannot be inferred logically that he is assumed to not know the transitive relation between the concepts 'dolphin' and 'animal'.

Negation partitions play a special part in checking the consistency of BGP-MS-Jr. user models. Without negation partitions, consistency must only be checked within views. Now, for each partition P, there can be several partitions P' such that it is an inconsistency if an SB-ONE structure is contained in both P and P'. A procedure is needed that computes such pairs of

conflicting partitions. We have developed a first approach to such an algorithm, which is based on the possible worlds semantics of modal logic, but has not yet proven to solve all cases of the problem and therefore is not described here. In BGP-MS-Jr., a simple version of it works in special cases. A typical example for a pair of conflicting partitions are SBUB and SB∼UB.

2.3 Supporting user model acquisition with application-specific inferences

Besides representation, user model acquisition is the main task of user modeling systems. The dialogue act component of BGP-MS that allows the definition of simple rules for inferring assumptions about the user from user actions [Pohl *et al.*, 1995] is also available in BGP-MS-Jr. The developer can take advantage of the dialogue act type libraries of BGP-MS (e.g. one with acquisition rules suitable for adaptive hypertext systems), or can define application specific dialogue act types. Other application specific inferences can be coded by the developer into arbitrary Lisp functions. They will be executed when entries into the user model are made by the application.

2.4 Stereotype Management

The stereotype management facility of BGP-MS activates and retracts a stereotype by adding and deleting inheritance links between the partition SBUB and the stereotype partition. This requires special treatment of SBUB in consistency handling. In BGP-MS-Jr., if there are conflicting user model contents, newer contents are preferred to older ones. When a new entry into SBUB contradicts a former one in SB∼UB, it is simply removed from SB∼UB and inserted into SBUB. In the reverse case, BGP-MS-Jr. inserts the entry into SB∼UB and tries to remove it from SBUB. It cannot be removed when it is inherited from a stereotype partition. Therefore, the rule that an entry cannot be contained in conflicting partitions is relaxed in that it is accepted that an entry can be contained in SB∼UB and inherited into SBUB from a stereotype partition. This case is interpreted as meaning that the user does not know that knowledge item, hereby giving the explicit knowledge in SB∼UB priority over the stereotype content.

In the current implementation, it is not possible to model stereotypical knowledge of what the user does not know, e.g. to use SB∼UB in a symmetrical way to SBUB.

For every stereotype, the application developer must define activation and retraction conditions. For this purpose, a set of predefined condition schemes can be used, among them:

- IFKNOWN *list*: is satisfied if all knowledge items contained in *list* are known to the user (i.e., are all contained in view SBUB).
- IFUNKNOWN%OF *n list*: is satisfied if *n* percent of the knowledge items contained in *list* are unknown to the user (i.e., are not contained in view SBUB).
- IFNOTKNOWNCONCEPTS% *n*: is satisfied if *n* percent of the concepts contained in the stereotype are not known to the user (i.e., are all contained in view SB∼UB).

BGP-MS periodically checks the activation conditions of inactive and the retraction conditions of active stereotypes. Moreover, BGP-MS records source information for the contents of the user model, so that it can tell whether an assumption about the user was observed by the application (a primary assumption) or results from a dialogue act, a stereotype activation or an application-defined inference (a secondary assumption). When evaluating the conditions, only primary assumptions are considered.

2.5 Communication between BGP-MS-Jr. and an application

The communication between BGP-MS-Jr. and an application is based on a message-oriented paradigm, i.e. it is carried out by sending and receiving messages. The following messages can be sent by the application, eventually leading to an answer message:

bgp-ms-tell is used to inform BGP-MS-Jr. about new assumptions about the user.

bgp-ms-ask is used to query the user model.

d-act may be used to inform BGP-MS-Jr. about the occurrence of dialogue acts.

This kind of communication has been realized with KN-IPCMS[1], an inter-process communication management system. In KN-IPCMS, the sender or a receiver of a message are identified by unique names (so-called ports), which are supervised by KN-IPCMS and known to all potential communication partners. KN-IPCMS offers selective and unselective send and receive functions. For example, the sender of a message specifies the recipient (selective send), or the receiver of a message wants to receive the next message, regardless of the sender (unselective receive).

KN-IPCMS is today available on two platforms, on SUN workstations running SunOS 4.1.x and on the PC running MS-DOS and MS-Windows 3.1. KN-IPCMS was designed for high performance and low resource usage. It therefore takes advantage of shared memory and semaphores on the workstation and global memory and messages on the Windows platform. For the sake of language independence, KN-IPCMS was implemented under MS Windows in ANSI C in the form of a dynamic link library. Therefore it could be easily integrated in nearly every program that runs under MS-Windows 3.1. It was already successfully used from Asymetrix Toolbook, Microsoft Word and Golden Common Lisp simultaneously.

2.6 Developing an application with BGP-MS-Jr.

BGP-MS-Jr. must be filled with application dependent knowledge. This comprises the following steps:

1. Relevant partitions must be defined (usually there are at least SB, SBUB, SB∼UB and stereotype partitions).
2. Stereotype partitions must be filled. The SB partition may be filled with domain knowledge.
3. Stereotype activation and retraction conditions must be defined.
4. Dialogue acts may be defined or selected from predefined libraries.
5. Application-specific inferences may be defined.

[1] KN-IPCMS stands for KoNstanz Inter-Process Communication Management System

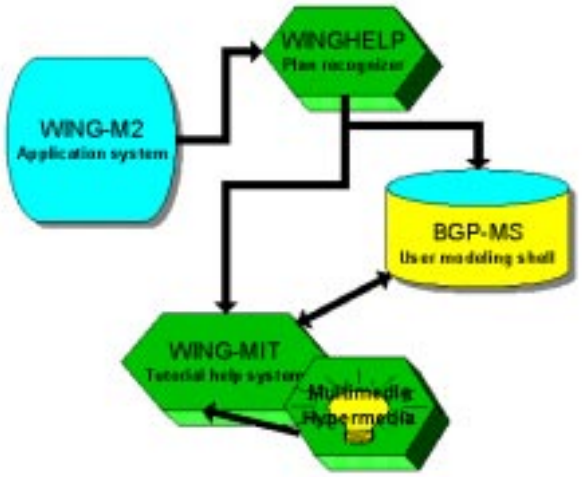

Figure 1: Structure of WING-MIT

3 WING-MIT: user modeling with BGP-MS-Jr.

3.1 Introduction

The tutorial help system WING-MIT (Multimedial and Intelligent Tutorial Help System for WING-M2, a material information system, developed in the project WING-IIR[2]) is a hybrid between a help system and a tutoring system. It therefore has two goals: to provide necessary help without interfering with the user's goals and to produce a high learning effect. These objectives are achieved in WING-MIT by integrating intelligent components like a plan recognition system and a user modeling system. Thus it imitates a human tutor, who also combines help and tutoring.

WING-MIT is designed in such a way that in its environment the user will recognize the structure of the application system and also the problematic situation which led him into WING-MIT. The user is explained what he has done (dialogue history), why an error occurred (context-sensitive diagnosis) and how he can solve the problem (context-sensitive help). Help information is presented in a multimedial, adaptive and context-sensitive way. Characteristics of WING-MIT are the teaching-learning process, possibilities for exploration, heuristic choice of training sessions and the symbiotic overall design of the complex system.

3.2 Intelligent components of WING-MIT and multimedial explanation

WING-MIT closely works together with two intelligent components, the plan recognition and the user modeling system (see Figure 1). The plan recognition system for WING-MIT is WINGHELP [Kim and Maurer, 1994; Kim, 1995]. Plan recognition is done by identifying the user's actions through associating them with the elements of a plan library. This library contains all possible sequences of user actions and plans according to the domain of the underlying application system WING-M2. WINGHELP offers (active) context-sensitive minimal help, either on request of the user or if a system state is observed in which the user needs help. The user's need for assistance is determined by plan recognition and identification of plan categories in the plan library. A further task of WINGHELP is to supply information about the user to the user modeling system.

[2] This project is situated at the department of linguistic information science at the University of Regensburg, and granted by the german ministry for economics.

BGP-MS-Jr. is the basis for the user modeling system of WING-MIT. WINGHELP tracks the user's action sequences. Recognized plans are passed to the user modeling system which employs them for drawing assumptions about the user's knowledge concerning the application domain. On activation, WING-MIT offers adaptive, context-sensitive tutorial help via multimedial explanation based on the individual user model.

Multimedia is realized in WING-MIT in a prototypical manner and refers to text, color-coding, graphics, animation, natural language output, video clips, combinations thereof and to hypertext and hypermedia. The objectives which are to be achieved by the use of multimedia within WING-MIT can be discussed on two levels: the level of successful transfer of information to the user and the effectiveness of the user's learning, and the level of suppliance of information.

3.3 Using BGP-MS-Jr. with WING-MIT

WING-MIT employs a user model on the following three levels:

- for the generation of help information which is adapted to the user's knowledge,
- for suppliance of help information and information sources
- and for treatment of suboptimal user interaction.

This section describes how BGP-MS-Jr. has been used to build a user modeling system for WING-MIT, and gives examples of user-model-based adaptive behaviour of the system.

3.3.1 Partition hierarchy and domain knowledge

WING-MIT uses the partition mechanism to represent four different types of knowledge. The partition SB includes terminological domain knowledge about WING-M2 as well as plans and sequences of actions of the plan recognition system WINGHELP. Besides, the partitions SBUB and SB$\sim$UB are used to model the current individual plan knowledge of the user. Furthermore, stereotype partitions are used to model different potential knowledge levels of users. For example, the relation between the plans and sequences of actions contained in the plan library of WINGHELP is part of the WING-M2 domain knowledge; it can be modeled as a concept hierarchy in partition SB (see figure 2).

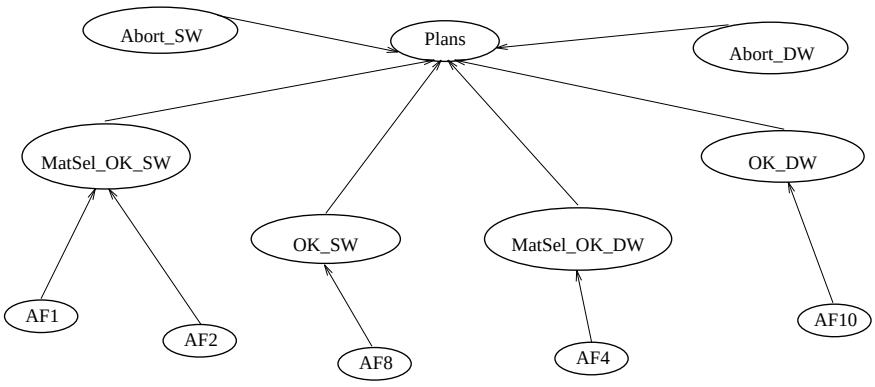

Figure 2: Part of plan hierarchy of domain WING-M2

In figure 2, AFn denotes a sequence of actions which is in the plan basis of the plan recognition system and which is mapped onto a plan. AF1 (string inserted, characteristic selected and 'OK' button pressed in search window) is, for example, an optimal sequence for the plan 'MatSel_OK_SW' (material selection with OK in search window) during a search for characteristics of a material, whereas AF8 ('OK' button pressed without input in search window) is a wrong sequence, which is mapped onto the plan 'OK_SW' (OK in search window).

The plan and action sequence concepts are the potential content of the individual user model. For example, if WINGHELP recognizes that the user has performed an optimal action sequence, the corresponding concept is entered into SBUB. Another example is: If the last user actions corresponds to several wrong action sequences, the user will be queried which of the related goals he is pursuing. The plan concept that fits the user's answer will be entered into SB$\sim$UB, thus reflecting the fact that the user did not know the plan to achieve his last goal. Since stereotypes are pre-defined models of potential individual users, they also contain plan and action sequence concepts. Moreover, the hierarchy of plan and action sequence concepts can be used for inferences about the user; this will be discussed in the next section.

3.3.2 Application-specific inferences

Assumptions about the plan knowledge of the user are primarily made based on observations by the plan recognition system WINGHELP. For WING-MIT, also two rules for inferring additional assumptions about the user have been implemented. The rules exploit the domain knowledge represented as a hierarchy of plan and action sequence concepts in partition SB together with a relevance relation between action sequences. One action sequence is relevant to another one if both belong to the same plan category, i.e. both have a common direct superconcept in the hierarchy. For the description of the rules, assume – without loss of generality – that there are three action sequence concepts C1, C2, and C3 with a common direct plan superconcept P:

1. If concept C1 is considered to be known to the user, its relevant concepts C2 and C3 are checked whether they are not known to the user. If they are not found in partition SB$\sim$UB, it is assumed that the user knows the relevant concepts C2 and C3 (i.e. they are inserted in SBUB).

2. If the parent concept P is considered to be not known to the user, it is assumed that also its children concepts C1, C2 and C3 are not known to the user (i.e. they are inserted in SB$\sim$UB).

Apart from these two inference rules, direct observations by WINGHELP and the activation (or retraction) of stereotypes are the means of forming the individual user model for WING-MIT.

3.3.3 Stereotypes and control of stereotype evaluation

WING-MIT employs three stereotypes, which model the plan knowledge of expert users, advanced users and beginners. In addition, there is a stereotype 'AnyPerson' that contains general knowledge that every user is supposed to have and which is always active. The overall structure of the WING-MIT stereotype model is an "onion model", i.e. the stereotypes can be ordered by a subset relation:

$$\text{AnyPerson} \subset \text{Beginner} \subset \text{Advanced} \subset \text{Expert}$$

In the definition of activation and retraction conditions, WING-MIT uses the predefined condition schemes IFKNOWNCONCEPTS% and IFNOTKNOWNCONCEPTS%, respectively, with thresholds between 25 and 80 percent.

3.3.4 Example of user-model-based adaptation

Learning takes place when help information is transformed into knowledge. “Not every piece of information will automatically become a new part of knowledge. Information often stands at the beginning of a learning process.” [Kuhlen, 1990]. To increase the learning effect, help information should be adjusted to the user’s needs and should be intelligible. In one example we will show how WING-MIT adapts help information according to the current user model. Adaptations are mainly based on the current classification of the user, i.e. which stereotype is currently activated. Each help information is available in different pre-defined versions, corresponding to the different stereotypes. In a help situation, the text that is associated to the currently active stereotype is presented to the user.

Consider the following example of an erroneous situation: A user types a material name in the edit field of WING-M2 and clicks ‘OK’. This sequence of actions is wrong. The user should use the ‘TAB’ key to enforce dynamic adjustment of a displayed list of material characteristics according to the given material and then choose a characteristic before finally clicking ‘OK’. There can be various reasons for this mistake: for example the user does not know the meaning of the ‘OK’ button (lack of knowledge, problems with the used metaphor), he does not know what ‘characteristic’ means (lack of knowledge), he does not know the sequence of actions for that kind of search (lack of knowledge, deviation of the mental model), or he knows it and has forgotten about it (mistake by forgetting), and so forth. WING-MIT should try to eliminate those reasons by offering the proper information to the user.

With the help of WINGHELP, WING-MIT displays a short description of the user’s action or actions in an explanation field. Furthermore, it shows a description of the user’s goals that the plan recognition system has found and a description of the optimal sequence of actions to pursue these goals:

> You have clicked the ‘OK’ button after inserting the name of a material. You probably were looking for data for that material. You can pursue that goal by pressing ‘TAB’ after inserting the name. Then the list of characteristics is adjusted for your material and will contain only characteristics where there is data for your material. In that list you can then choose the characteristics of interest for you by clicking on them. Then press the ‘OK’ button to start the search.

Underlined terms in the text symbolize hypertext links. They lead to nodes, the contents of which can be adjusted adaptively to the knowledge level of the user as modeled within BGP-MS-Jr. (Beginner, Advanced and Expert):

Beginner: The list of characteristics contains the so-called material characteristics, which represent the features of the materials. The database contains 19 characteristics within 6 groups. The first number of each characteristic shows the group of that specific characteristic.

Advanced: The database contains 19 characteristics within 6 groups.The first number of each characteristic shows the group of that specific characteristic.

Expert: The database contains 19 characteristics within 6 groups.

An additional text field tells the user how to find more information. This field is supposed to stimulate the user’s motivation and activity when looking for information:

Beginner: Please try to get more information. Use the mouse to point on any object on the screen, for example the ‘OK’ button. Then you can see an animation or a video-clip that shows the optimal usage. And then you can also do an exercise in the tutor.

Advanced: You can get more information by pointing to any object with the mouse. In the exercise field you can also find out more about your mistake and do a tutor exercise.

Expert: You can get more information by pointing to any object with the mouse.

The help information, which is displayed for an object in that mode, is also adapted according to the user model. For example, the following text is displayed to explain the 'OK' button:

Beginner: WING-M2 has an 'OK' and a 'Cancel' button. 'OK' means 'Yes, go on!'. Now you want that your insertions will come into effect. 'OK' will start the search in the database.

Advanced: You can start or continue your search by clicking on the 'OK' Button.

Expert: 'OK' means 'Yes, go on!'.

By being provided context-sensitive help, the user gets more motivated to look for information. The motivation and the adaptivity during a material search will increase the learning effect for the user. The adaptive help information will be comprehended faster and more efficiently, which will have a positive effect not only on the search but also on the learning.

4 Perspectives of user modeling in widely-used applications

4.1 Introduction

When thinking about user modeling for all and about employing BGP-MS in widely-used applications, some questions come to mind. They address current deficiencies of BGP-MS since for now their answers are not satisfying.

- Can the whole functionality of BGP-MS, which provides necessary services for an adaptive system, be brought to platforms that are not meant to fit high-end computational needs? Such platforms (i.e. standard personal computers) are the environments that are and will be available to most computer users.
- Can BGP-MS be used in a real usage scenario with an adaptive standard application like a text processor, without annoying or hindering the user? Applications may be highly interactive and most of the computations, whether in the application or in the user modeling system, often have to be performed between two closely subsequent interactions of the user with the system.
- Is it possible to configure BGP-MS with respect to the functionality required for a specific user modeling task? This is especially important for systems running on less powerful platforms and which only make use of parts of the functionality, because unnecessary workload can then be avoided.
- Is it possible to achieve a greater flexibility concerning the architectural scenario for BGP-MS? Can the application and the user modeling shell system be physically separated? Today, the application and BGP-MS have to run on the same computer. However, some of today's widely-used and increasingly popular applications are not based on a local, but rather on a distributed scenario (e.g. the World Wide Web). So far, BGP-MS can't be employed in such a wide area context.

Having examples like World Wide Web in mind, interesting user modeling scenarios can be anticipated, where synergetic effects concerning the usage of long-term and dynamic (i.e., updated from session to session) user models can be expected. Interesting scenarios include the following:

1. Several instances of one application (e.g., several instances of a text processor) or several applications, running on one computer, have access to the same user model.

2. Applications running on different computers or even on different platforms have access to the same user model via a network.

3. One or more users have access to the same user model containing assumptions about the knowledge, beliefs, preferences, etc. of a user group as a whole. Examples for such user groups can be project teams or working groups in a Computer Supported Cooperative Work (CSCW) context.

The common characteristic of the last two scenarios are distributed application instances that take advantage of a common user model, e.g. to improve the coherence of the user's working environment. Distributed user modeling scenarios call for a network-oriented BGP-MS. In addition, a multi-user functionality is required, which should also take care of the privacy of the users being modeled. Examples for such a multi-user functionality are identification/authentication, access control, integrity control, and perhaps transactional behaviour [Gray and Reuter, 1993]. Some of these features are already implemented in the user modeling server 'Doppelgänger' [Orwant, 1995], and all of them are already established in other areas (e.g., distributed operating systems or (distributed) data base systems).

4.2 Architectures of today's user modeling shell systems

In this section we identify three different architectures for the adaptive application and the user modeling shell system and point out their main limitations. Finally we briefly describe a new architecture for BGP-MS that avoids nearly all disadvantages, and at the same time allows BGP-MS to be used in all scenarios envisaged in the previous section.

4.2.1 The local and monolithic architecture

The user modeling shell systems GUMS [Finin, 1989], UMT [Brajnik and Tasso, 1994], um [Kay, 1995] and PROTUM [Vergara, 1994] are all based on the assumption that the application and the user modeling shell system form one process. Every instance of an application must have its own user modeling system. Dynamic and long term user models can hardly be realized due to possible inconsistencies (e.g., when different instances of an application want to write their versions of the user model to persistent storage). Problems arise at development time if the programming languages of the application and the user modeling system differ. The overall adaptive system is executed sequentially at run time, i.e., no form of parallelism concerning the computations can be achieved.

None of the above-mentioned scenarios can be addressed by such a user modeling shell system.

4.2.2 The local and separated architecture

BGP-MS (and also BGP-MS-Jr.) runs concurrently to one or more applications on the same computer. Different instances of an application need not necessarily have a user modeling system

of their own, in contrast to the above architecture. One central user modeling system carries out the user modeling tasks for all instances of an application and, in general, for all applications that run on the same computer. At run time, all processes are competing for the resources on the computer. This competition is supervised by the processes themselves (cooperative multitasking like in MS-Windows) or by the central scheduler of the operating system (preemptive multitasking like in UNIX). Computations in the user modeling system can be done asynchronously without directly affecting the user's interaction with the application system. However, in this setting application and user modeling system run concurrently, but not in parallel; one computer still has to do all the work.

User modeling shell systems within such a setting are able to address the first of the scenarios mentioned in section 4.1.

4.2.3 The distributed architecture

The user model server 'Doppelgänger' [Orwant, 1995] and its applications form separate processes that may run on different computers. This client-server scenario requires communication through local or wide area network transport protocols. In this architecture the computer running the application can become entirely relieved of user modeling tasks. The response times resulting from queries to the user modeling system do not longer depend on the power of the computer running the adaptive application. Therefore also less powerful platforms (e.g. low-end PCs or office machines like a photocopy machine) are also able to run adaptive applications. The computations in the user modeling system that are inherently asynchronous to the application (e.g., processing of observations, evaluation of stereotypes or forward inferences) can be carried out in parallel.

This architecture is well suited for a user model server, which can host user models for several applications and/or instances of an application. Such a user model server is able to address all the scenarios mentioned above, but cannot be used when the computer running an adaptive application is not connected to the server machine.

4.3 A generic architecture for the user modeling shell system BGP-MS

Current user modeling shell systems are all able to handle a local scenario, either in a monolithic or in a separated setting. Distributed user modeling scenarios can however only be handled by a distributed architecture with respect to the user modeling shell system and the application. But is this distinction between local and distributed user modeling shell systems necessary? Is it possible to develop *one* generic architecture for a user modeling shell system which can easily emulate all mentioned architectures and therefore can be used in all described scenarios? And will it also be possible to overcome the disadvantages pointed out at the beginning of this chapter?

Recent results in the field of interoperable objects (or sometimes called component objects) are quite promising with respect to such a universal architecture [Dobbs, 1992; Janssen and Spreitzer, 1994]. The basis of many of these approaches is the Object Management Architecture (OMA), as promoted by the Object Management Group (OMG) [OMG, 1991]. Within this approach, objects are the basic components which form a software system. Each object has a well-defined description of its interface in a generic interface description language, which allows for the definition of operations and exceptions, types and subtypes. The Object Request Broker (ORB) takes care of anything that is needed to complete the invocation of an operation (e.g., locate the corresponding object on a local or remote computer, translate the arguments, invoke the operation and translate the data returned).

According to the OMA, BGP-MS could be modeled as a set of objects, each of which represents a certain functionality requested by the user model developer (e.g., BGDL interpretation, dialogue act management, representation and inferences in first order logic, view-connecting inferences, stereotype management, etc.). A properly chosen subset of the top-level functions of these subsystems in the current version of BGP-MS could be established as operations on the objects. The implementation of the object operations is mainly determined by the current BGP-MS functionality. The existing software structure (i.e. the division into subsystems) can therefore be viewed as being manifested to some extent in the object hierarchy. Corrections in the program code will nevertheless be necessary where in the existing program the encapsulation of an object is violated, e.g. through the use of global variables for communication between objects. Additional changes beyond that, like a redesign of operations according to an object-oriented approach, are not necessary.

Additional objects can then be planned for the near future. An example is a configurator, which is able to support the user model developer in configuring BGP-MS for a particular user modeling task and in setting up such a configuration in a specific hardware and software environment. A second new object could be the reimplementation of the BGP-MS inter-process communication system KN-IPCMS (cf. section 2.5). Then both communication paradigms, the action-oriented one (i.e. remote procedure calls supported by the ORB) and the message-oriented one (i.e. messages sent via KN-IPCMS), are at the disposal of the developer.

The evaluation of suitable ORB implementations for BGP-MS is nearly finished. One of the most promising candidates is the system ILU (Inter-Language Unification system), which is currently under development at Xerox PARC [Janssen *et al.*, 1995]. ILU is an ORB implementation that currently provides support for the programming languages ANSI C, C++, Modula-3, Python and Common Lisp. Therefore ILU seems to be a well-suited platform for the current reengineering tasks in BGP-MS. At the same time it offers interesting facilities like authentication, transaction and thread support for the further developments of BGP-MS in the near future.

5 Summary

The importance of user-adapted interaction along with user modeling as a base technique is growing, particularly since interactive systems and user interfaces are more and more demanded to be suitable for everyone. In this paper, we tried to show current efforts in bringing user modeling to standard computer platforms and to widely-used applications. We presented BGP-MS-Jr., a limited version of the user modeling shell system BGP-MS that can be employed in the development of adaptive applications on MS-Windows PCs. To demonstrate its usefulness, the intelligent tutorial help system WING-MIT and how it uses BGP-MS-Jr. as the basis for its user modeling system was described. In addition, we discussed a flexible and open architecture for BGP-MS that will make the system usable on all kinds of platforms, help to overcome current limitations and perhaps make any further distinction between different versions of BGP-MS superfluous. Implementing this architecture will render BGP-MS more suitable for a wide range of present and future applications, thus bringing adaptivity to user interfaces for all.

References

[Brajnik and Tasso, 1994] G. Brajnik and C. Tasso. A Shell for Developing Non-monotonic User Modeling Systems. *Int. J. Human-Computer Studies*, 40:31–62, 1994.

[Dobbs, 1992] Dr. Dobbs Special Report, The Interoperable Objects Revolution. Vol. 19 (16), 1992.

[Finin, 1989] T. W. Finin. GUMS: A General User Modeling Shell. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 411–430. Springer, Berlin, Heidelberg, 1989.

[Gray and Reuter, 1993] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, San Mateo, CA, 1993.

[Janssen and Spreitzer, 1994] B. Janssen and M. Spreitzer. ILU: Inter-Language Unification via Object Modules. In *OOPSLA '94 Workshop on Multi-Language Object Models*, 1994.

[Janssen *et al.*, 1995] B. Janssen, D. Severson, and M. Spreitzer. *ILU Reference Manual*. Xerox Corporation, 1995.

[Kay, 1995] J. Kay. The um toolkit for reusable, long term user models. *User Modeling and User-Adapted Interaction*, 4(3), 1995.

[Kim and Maurer, 1994] D.-W. Kim and H. Maurer. Entwurf eines intelligenten Hilfesystems (WINGHELP) für WING-M2. WING-IIR Arbeitsbericht 52, Universität Regensburg, 1994.

[Kim, 1995] D.-W. Kim. WING-MIT: Das auf einer multimedialen und intelligenten Benutzerschnittstelle basierende tutorielle Hilfesystem für das Werkstoffinformationssystem WING-M2. In U. Malinowski, editor, *Proc. 3rd Workshop 'Adaptivität und Benutzermodellierung in Interaktiven Softwaresystemen'*, 1995.

[Kobsa and Pohl, 1995] A. Kobsa and W. Pohl. The User Modeling Shell System BGP-MS. *User Modeling and User-Adapted Interaction*, 4(2):59–106, 1995.

[Kobsa *et al.*, 1994] A. Kobsa, D. Müller, and A. Nill. KN-AHS: An Adaptive Hypertext Client of the User Modeling System BGP-MS. In *Proc. of the Fourth International Conference on User Modeling*, pages 99–105, Hyannis, MA, 1994.

[Kuhlen, 1990] R. Kuhlen. Zum Stand pragmatischer Forschung in der Informationswissenschaft. In J. Herget and R. Kuhlen, editors, *Pragmatische Aspekte beim Entwurf und Betrieb von Informationssystemen. Proceedings des 1. internationalen Symposiums für Informationswissenschaft*, pages 13–18, Konstanz, 1990.

[McCune, 1994] W. W. McCune. OTTER 3.0 Reference Manual and Guide. Technical Report ANL-94/6, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, 1994.

[OMG, 1991] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, Revision 1.1, OMG Document 91.12.1, 1991.

[Oppermann, 1991] R. Oppermann. Experiences with Evaluation Methods for Human-Computer Interaction. Arbeitspapiere der GMD 540, GMD, St. Augustin, 1991.

[Orwant, 1995] J. Orwant. Heterogeneous Learning in the Doppelgänger User Modeling System. *User Modeling and User-Adapted Interaction*, 4(2):107–130, 1995.

[Paiva and Self, 1995] A. Paiva and J. Self. TAGUS – A User and Learner Modeling Workbench. *User Modeling and User-Adapted Interaction*, 4(3), 1995.

[Pohl *et al.*, 1995] W. Pohl, A. Kobsa, and O. Kutter. User Model Acquisition Heuristics Based on Dialogue Acts. In *International Workshop on the Design of Cooperative Systems*, pages 471–486, Antibes-Juan-les-Pins, France, 1995.

[Spenke *et al.*, 1992] M. Spenke, C. Beilken, T. Berlange, A. Bäcker, and A. Genau. GINA User Manual Version 2.1 for Common Lisp. Arbeitsbericht 614, GMD, St. Augustin, 1992.

[Vergara, 1994] H. Vergara. PROTUM: A Prolog Based Tool for User Modeling. WIS Memo 10, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.