User Modeling for Modular Adaptive Hypermedia

Vadim Chepegin[1], Lora Aroyo[1], Paul De Bra[1], Dominik Heckmann[2]

[1] Eindhoven University of Technology, Computer Science Department,
Den Dolech 2, P.O. Box 513, 5600 MB
Eindhoven, the Netherlands
[2] Universität des Saarlandes
Postfach 151150, D-66041
Saarbrücken, Germany
{vchepegi, laroyo, debra}@win.tue.nl, heckmann@dfki.de
http://wwwis.win.tue.nl/, http://www.5cm.de/dominik/

Abstract. More and more users work simultaneously with multiple applications, to perform various tasks. This situation puts high demands on the user adaptive systems (UAS), which traditionally support users' work in a single isolated domain, and which now shift towards personalization in multi-task and cross-application contexts. In their attempt to meet the increasing demands, UAS grow in complexity, but they do little about their compatibility and interoperability. We propose a Component-based Architecture for UAS: (*CompAS*). The key feature of *CompAS* is modularization of knowledge models and as a consequence allowing existing UAS to share their user models by means of a centralized user model service. As a proof of concept we show how two existing applications, AHA! [1] and UserModelService [2] can achieve interaction based on the extracted user model.

1 Introduction

The current boost in the field of wireless and ubiquitous computing has a great impact on many aspects of system engineering, and especially on the design and development of User Adaptive Systems (UAS) [3]. The provision of personalized and user-centered information services becomes a multifaceted task in an open and distributed world of mobile users and smart systems. It becomes a formidable challenge for UAS to cope with this rising complexity of interactions and to provide the appropriate adaptation to the numerous user goals. A critical factor for the increase in the complexity is the fact that in this context tasks of various nature are carried out in parallel and/or sequentially.

Consider the following scenario. Ann is a trainee in a program for art critics. In her training she uses an Adaptive Hypermedia (AH) Art Critics Course and an Intelligent Tutoring System (ITS) for Artifacts Classification simultaneously. The following collaboration acts between the applications can appear in this scenario in order to support Ann in achieving her learning goals.

1. Ann starts her work with the AH course. At some point the AH application does not have enough information about Ann's knowledge on a particular topic related

to Artifacts Classification and delegates the following steps to the ITS in Artifacts Classification in order to collect more assessment information about Ann during her interaction with the ITS.
2. Ann works with the ITS for Artifacts Classification and perform various training tasks. When she has achieved the level of knowledge indicated as needed by the AH course the control is turned over to the Art Critics course again.

The educational application is only one example of possible areas, where we can observe collaboration among systems and multi-task environments for the users. The systems in the given scenario as a rule do not share the same understanding of the domain, user modeling, adaptation technologies, and they also articulate tasks in different terms. Thus, the UAS illustrated here are not compatible with each other, which causes problems for their collaboration. As a consequence they cannot adaptively support the user's flow of activities as a seamless sequence of interrelated tasks within an overall process.

In Section 2 we propose a Component-based Architecture for User Adaptive Systems (*CompAS*) to serve as a reference model for UAS and in this way to facilitate the process of the creation, communication and integration of such systems. In Section 3 we illustrate some of the implementation issues involved in the realization of such an architecture. We show how two existing applications (AHA! [1] and UserModelService [2]), built independently, can cooperate within the context of common user tasks. Conclusions and discussion for further development are finally presented in Section 4.

2 *CompAS* Architecture

In this section we briefly describe features of the *CompAS* architecture, which was initially introduced in [8], [9]. The architecture follows earlier hypertext / hypermedia models [4], [5], knowledge-based systems architectures [6] and intelligent tutoring systems architectures [7] by having a Domain (Expert) model, User (Student, Learner) model and Adaptation (Teaching, Tutoring) model. Additionally, it is extended with a separate Application model that extracts all the application dependent knowledge from the Domain, User and Adaptation models in order to allow for a strict separation of concerns between general adaptation techniques and application-specific strategies. This way it also facilitates the definition of the adaptation in a task-oriented manner. The *CompAS* architecture also emphasizes on the role of an external User model, which becomes a central point for communication among user adaptive systems since all their decisions are based on the state of the user and her environment.

Separation of Knowledge Models. One of the key aspects of *CompAS* is modularization of knowledge models. The main stress in this architecture is given by the fact that the developers have to put various different types of knowledge into specific models and then encapsulated them into single software components. Here we provide an informal definition of architectural building blocks with their functionality.

1. The Domain Model is a conceptual representation of the real world; it consists of concepts and relations between them. It is, in principle, application independent. In particular, it is not concerned with problems, tasks or their solutions.

2. The Application Model contains a generic description of the user tasks in the context of a particular application. It contains a description of the application in terms of Role-Goals-Tasks-Methods hierarchies, and represents user long-term goals in order to support dynamic, knowledge-based application selection, which can facilitate the user in solving her current task [6].
3. The User Model in *CompAS* consists of two parts: a User Modeling Service (keeps track of user's interaction with various applications, stores stable user characteristics and various environmental aspects) and a User Agent (that is a reflection of the user in the system and can move together with the user from one device and application to another).
4. The Adaptation Model is responsible for the application of special procedures to plan and perform the adaptation. The adaptation model realizes adaptation based on the users tasks, environmental conditions, preferences, etc. and relationships between concepts of the Domain Model, like prerequisites.

In *CompAS* architecture we separate the knowledge about the user from the knowledge about the user's tasks and subject domains. Thus, we introduce four types of ontologies, which support the population and sharing of various types of knowledge: User Model ontology, Application (Task) ontology , Adaptation ontology, and Domain ontology.

Centralized User Modeling. User models are an essential part of every adaptive system. In order to adaptively to support the user flow of activities as a sequence of interrelated continuous processes through multiple applications, UAS cannot be isolated entities any longer. Instead they would operate better if they exchanged their knowledge about the user. There are two ways to approach this problem and to establish *quid pro quo* communications among systems: (1) peer-to-peer communication [10], and (2) a centralized model [11]. The former is more concrete in the sense that when you decide to share knowledge with a particular application you can create a specific bridge between them. The second approach, the centralized UM, means that several applications establish communications among themselves via a single user model service. This method has well known benefits, e.g.,

1. Sharing knowledge among applications leads to a synergetic effect, when an application takes advantage from the user interaction with several applications;
2. A User Modeling Service can support important functions such as: scrutability, privacy, history of changes, multiple views and resolution of conflicts;
3. And last, but not least, is the decreasing number of bridges between every two applications from order of N^2 to order of N, between each application and a User Model Service

We have chosen the centralized user model approach, but there are a number of obstacles that must be overcome when doing so. We have divided the problems into several facets as follows: (1) Communication between applications; (2) Collaboration between applications; (3) Sharing structures and components; (4) Collaboration between users.

We distinguish between the syntactic level and the semantic level of integration. To solve the communication problem between applications entails applications speaking

the same language and using the same format and protocol when communicating. The solution can be found in the recently established query and mark up languages for user modeling: UserQL and UserML respectively [12].

Turning to the second problem of the semantic integration of applications, this requires the first problem to be solved first. It brings with it an additional requirement in terms of sharing the same view to the user modeling process and problem domain. The area of ontological engineering provides us with ideas for further research when we come to this point. In the broader context the integration of ontologies and problem solving methods can support knowledge model interaction within the existing architecture [6], [13].

We have already discussed (in Section 2) the problem of sharing components within the proposed *CompAS* architecture. Due to limited space, we will not be addressing the fourth problem in this particular paper. In the next section, we present some considerations for implementation and supply examples for solving the above problems. We further illustrate communication realization for an application and a UserModelService.

3 Implementation Issues

The general-purpose adaptive engine AHA! [1] developed at the Eindhoven University of Technology, aims at bringing adaptivity to a wide variety of applications such as on-line information systems, on-line help systems, museum and shopping websites, in addition to the area of on-line textbooks. We would like to isolate the internal User Model of AHA! and extract it in order to achieve a sharable user model.

The "u2m.org" UserModelService (UMS), developed at Saarland University, is a system that supports user modeling processes by storing user models (and context models), providing access to them, and offering strategies for conflict resolution. As a first step, we are using this UserModelService to extract the internal AHA! user model by using the exchange language UserML [12]. As a next step we target the semantic level of integration of several independent applications.

AHA! information model. The AHA! information model consists of values of all user-specific attributes associated with every concept of the Domain Model. In other words, every concept has its own reflection in the User Model. The current realization of AHA! does not have a Domain Model in a traditional meaning. All attributes associated with concepts serve user modeling purposes only. In a user profile these attributes usually have meaning of knowledge, interest, etc. This is a realization of the so called 'overlay user model'.

Concepts are described by attribute/value pairs, which are represented in the system by the 'attribute' tables. For each attribute of each concept from the domain the system keeps a value. This information is stored in the 'profrec' table.

An action can be associated with each attribute. An action is fired when the special precondition of the rule (IF part) is true. A table with the name 'action' is in charge of keeping this sort of data. The action part of the rule (THEN or ELSE part) is defined in the table 'assignment'. By assigning 'true' or 'false' value to the 'truestat' attribute of that table we determine whether this particular table contains the THEN part or ELSE

part of a rule. For user modeling issues only 3 tables are relevant: ‘profrec’, ‘concept’ and ‘attribute’ since they define the User Model.

Communication between AHA! and UserModelService. This subsection illustrates the communication between AHA! and UMS. The active part of AHA! is its adaptation engine, which generates the best next page for the user to visit. The tailoring process is based on the user model. The actual adaptation is performed by the rule-based system. Rules, using by the adaptation engine, utilize knowledge about user performance and characteristics, which the engine extracts from an internal user model (database). When we externalise or extract the user modeling functionality from AHA!, the engine has to request knowledge about a user from a separate service. This can be done by formulating HTTP requests to a special URI of a PHP script, which is the frontend of the UserModelService, responsible for answering questions coming from applications:

```
http://www.u2m.org/service.php?subject=Alex&auxiliary=
knowledge&predicate=aha.tutorial&range=aha.statement
```

This UserQL/URI represents the query “Tell me all about Alex’s knowledge on the topic ‘aha.tutorial’ of which the range of values should be ‘aha.statement’”. AHA! will receive an answer in UserML. It applies its adaptation strategy in the light of new received data and shows the new content to the user. AHA! interprets the next user action and decides how this new knowledge influences the internal state of the user. At the end of the interaction loop AHA! updates the user model (through UserModelService) with new values by means of one of its interfaces called ‘adder.php’.

```
http://www.u2m.org/adder.php?subject=Alex&auxiliary=
knowledge&predicate=aha.tutorial.desinger&range=
aha.statement&object=75
```

What we have described above is the communication between a typical AHA! application and the UserModelService, as it is being implemented in a first prototype. It is important to note that the separation of domain, adaptation and user model is advocated by the AHAM [4] reference model, but thus far not realized in AHA!. The current work on using AHA! with UserModelService brings AHA! one step closer to being an implementation of AHAM.

4 Conclusions and Further Work

This paper aimed to demonstrate the first steps towards a generic architecture for User Adaptive Systems (*CompAS*), which will support the integration of existing adaptive software and will facilitate their further development. We extracted the internal user model of AHA! and transfered it to a shared UserModelService. The great challenge is the higher level integration, which we call integration on a semantical level, and is the next step on the way towards cross-application adaptation.

Acknowledgements. The research presented in this paper has been performed within the context of the CHIME Token2000 project. It also serves the goals of the Prolearn Network of Excellence on Professional Learning.

References

1. De Bra, P., Calvi, L.: AHA: a Generic Adaptive Hypermedia System, Proc. of the 2nd Workshop on Adaptive Hypertext and Hypermedia, pp. 5-12, Pittsburgh, 1998.
   http://wwwis.win.tue.nl/ah98/DeBra.html
2. Dominik Heckmann: Reference Manual to the U2M User Model Server Dominik Heckmann (January 2004) Technical Report, Saarland University
3. Anthony Jameson: Modelling both the Context and the User, Personal and Ubiquitous Computing, **5**, 2001, 29–33.
4. De Bra, P., Houben, G.J., Wu, H.: AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147–156, Darmstadt, Germany, 1999.
   http://wwwis.win.tue.nl/ debra/ht99/ht99.ps
5. Koch, N. and Wirsing, M.: The Munich Reference Model for Adaptive Hypermedia Applications, in Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, 2002, pp. 213–222.
   http://www.pst.informatik.uni-muenchen.de/personen/kochn/munich-koch-wirsing-final.pdf
6. Motta, E., Domingue, J., Cabral, L., and Gaspari, M.: IRS-II: A framework and Infrastructure for Semantic Web Services. In 2nd International Semantic Web Conference 2003 (ISWC 2003), 20-23 October 2003, Sundial Resort, Sanibel Island, Florida, USA.
   http://kmi.open.ac.uk/people/domingue/papers/irs-iswc-03.pdf
7. Wenger, E.: Artificial Intelligence and Tutoring Systems. Computational approaches to the communication of knowledge. - Los Altos: Morgan Kaufmann, 1987. – 486p.
8. Chepegin, V., Aroyo, L., De Bra, P., Houben, G.: CHIME: Adaptive Framework for Semantic Web Service Integration. In InfWet'03 Conference, pp. 29-36, 2003.
   http://wwwis.win.tue.nl:8080/infwet03/proceedings/3.pdf
9. De Bra, P., Aroyo, L., Chepegin, V.: "The Next Big Thing: Adaptive Web-Based Systems" // Journal of Digital Information (in press)
10. W. Nejdl, B. Wolf, Ch. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr and T. Risch. EDUTELLA: A P2P Networking Infrastructure Based on RDF , Edutella White Paper, November 2001
    http://edutella.jxta.org/reports/edutella-whitepaper.pdf
11. Tomaz Klobucar, Vanja Senicar, Borka Jerman Blazic: Privacy Issues of a Smart Space for Learning, In: Proceedings of ICALT 2004 (The 4th IEEE International Conference on Advanced Learning Technologies), August 2004.
12. Heckmann, D., Krüger, A.: A User Modeling Markup Language (UserML) for Ubiquitous Computing. In Proceedings of User Modeling 2003, 9th International Conference, UM 2003, Johnstown, PA, USA, June 22-26, 2003, 393–397.
    http://www.sigmod.org/sigmod/dblp/db/conf/um/um2003.html
13. Bylander T, Chandrasekaran B.: Generic tasks for knowledge-based reasoning: The "right" level of abstraction for knowledge acquisition. In: Gains B, Boose J (eds). Knowledge acquisition for knowledge-based systems. London: Academic Press, 1988; p.65–77.