

Functional Agent Systems for User Modeling

Andreas Lorenz

Fraunhofer Institute for Applied Information Technology
Schloß Birlinghoven
53754 Sankt Augustin, Germany
Andreas.Lorenz@fit.fraunhofer.de

Jörg Denzinger

University of Calgary
Department of Computer Science
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
denzinge@cpsc.ucalgary.ca

Abstract

This paper introduces an approach to employ multi-agent technologies for user modeling purposes. It illustrates how several teams of agents are defined separated by their functionality. Within each of the functional layers, the particular team represents a flexible implementation to employ highly specialized entities for sub-tasks. In the overall solution, these teams cooperate to fulfill the requirements of user modeling in a more appropriate way.

1 Introduction

A growing number of application fields, such as location-based services, eLearning applications, and exhibition guides, need to adapt to the user's behavior, preferences, technical capabilities, and so forth in order to be accepted by users. In the development of user-adaptive systems, several problems need to be solved. Recent projects and research work at the Fraunhofer Institute have revealed two major aspects to be addressed: How do we implement adaptivity in an efficient way that is compatible with several domains, and how can we bridge the lack of standards.

Common generic user modeling systems, such as [Kobsa, 2001], adapt the system behavior starting from the definition of the user's context, which is composed of four dimensions as introduced by Gross and Specht [2001]: *Identity*, *Location*, *Time*, and *Environment*. In addition, users establish relationships to several entities within the domain they are acting in. For example, they communicate with other people, they form groups of users trying to attain a certain goal, they provide data to computer systems, and so forth. Strictly speaking, an adequate representation of a human being as the user of a computer system has to take the human's *activity* (in different *roles*) into consideration.

This contribution introduces a different view on an adequate representation of a computer system's user as a human being. It takes up common user and context modeling methodologies and illustrates the idea of applying agent technology [Jennings and Wooldridge, 1998] to implement them in a generalized way.

2 Users and Agents compared

At the current stage, the state-of-the-art system development and programming approach is the object-oriented approach [Odell, 2002]. Applying this approach for user modeling purposes allows defining users as single entities. One of the drawbacks is that objects are considered passive since their methods need to be invoked by any other entity, therefore the agent approach for representing users sounds more likely:

“Software agents have their own thread of control, localizing not only code and state but their invocation as well. Such agents can also have individual rules and goals, making them appear like ‘active objects with initiative’. In other words, when and how an agent acts is determined by the agent.”[Odell, 2002]

As active entities, users do willfully behave following their own decisions in order to fulfill desires. They plan their decisions based on their knowledge, goals, beliefs, memories, or emotions. The freedom of users with regard to behavior is often limited by internal (like time, credibility) and/or external (e.g. laws, contracts) restrictions. The desires to be fulfilled emerge from their interests, intentions, goals, and preferences. All of these attributes are part of the above-mentioned *Identity* dimension. Analyzing the variety of agent modeling approaches (intention-based, rule-based, role-based, to mention just a few) shows, that behavior and identity are central issues in modeling agents. For example, modeling emotional agents with personality is described in Padgham and Taylor [1997]. In addition, users cooperate with other users (including being deliberately unwilling to cooperate). For this purpose, they build (and release) communities, for instance to share information, or to get the help of an expert. In translation to multi-agent terminology, cooperating agents need the support of organization and communication, whereby the latter includes transferring/receiving information as well as negotiation, teaching, and even deliberately hiding information or attempting misinforming others.

Furthermore, users improve their own behavior by learning directly from external entities (teachers) or by observing their environment and its entities. Additionally, both learn by analyzing the outcome of their past (re-)actions in similar situations. In Wooldridge [1996] is shown that agents can also learn and improve routine interactions.

2.1 Recent Approaches to apply Agent Technology

In several application fields the attempt to combine user modeling with agent technologies has already been carried out. For the sake of brevity we will not be able to present an extended review about all applications.

Driven by the boom of web-applications in the late 1990s, the value of personalization was increasingly recognized in the fields of eCommerce, information access, and eLearning. In the field of eCommerce, Pazzani and Billsus [2002] have introduced adaptive web site agents that recommend relevant documents to the user in an Amazon.com-like manner. They argued that the information is best used to change the behavior of an animated agent (avatar) to assist the user. In Billsus and Pazzani [2000] an intelligent information agent is considered to be a personal assistant that gradually learns about users' interests. Therefore, the specific design of the agent's user model is motivated by a number of observations and requirements. Like the adaptive web agents presented in Menczer and Belew [2000], agent technology is either used for personalized information acquisition or for individual information presentation.

In the domain of eLearning, Vassileva *et al.* [1999] base the adaptation within the I-Help system on models of human users maintained by personal agents:

“Each personal agent manages a user model containing information about the *user's goals* (help requests, current goal), about *knowledge resources / competencies* on certain topics or tasks, and about the *relationships* existing between the user and other users.”

The Baghera project [Webber *et al.*, 2001] has implemented personal interface agents for students and teachers, and tutor agents that base whose didactical decisions on a student model. Since the learner's knowings consist of a diversity of conceptions, the attempt to model conceptions as sets of (problem-, operator-, language-, and control-) agents in Webber *et al.* [2002] can be seen as the next step taken to not only base on a student model but actually use agent technologies for modeling purposes. Some more projects implementing learning systems based on multi-agent architectures are presented in [Webber *et al.*, 2002, Section 2].

Furthermore, agent technologies have been applied for personalizing location-based services like city- and tourism-guides. The Deep Map Agents introduced in Fink and Kobsa [2002] provide tour recommendations, analyze spoken text, generate speech output etc. These agents, which loosely adhere to the FIPA agent specification [FIPA], communicate to a User Modeling Server (UMS) about the user's interaction with the system and query the UMS for user characteristics. In the EU-founded CRUMPET-project [Poslad *et al.*, 2001], FIPA compliant user agents are hosted on the end user terminal devices and provide the user with the service GUI. These user agents adapt the information presentation to the platform evaluating the usage profile of the user. As Kobsa has anticipated in [Kobsa, 2001], mobile user models seem to be worth considering here. By employing methodologies of true mobile agents, the user model could stay with the user at any device.

In recent attempts to apply agent technology, agents are either connected with an external user model or users are modeled as single agents working together in a multi-agent system. In terms of a general definition of an agent:

An Agent is a triple $\{Dat, Act, Sit\}$, whereby *Dat* represents the possible values of the internal knowledge base of the agent, *Act* is the set of possible actions, and the set *Sit* consists of the situations the agent acts in,

the approach “user = agent” maps the internal knowledge base of the agent to the user model. The possible user actions within the domain determine the set *Act* of agent actions, and *Sit* consists of the user's contexts.

On the one hand, this approach provides a number of benefits: regarding to some currently underrepresented attributes of human beings (such as individual information acquisition, individual decision finding) the approach provides a more adequate representation of active users. Additionally, the user modeling component becomes more able to react to changes (for example if the user's interest decreases over time), and we have the possibility to apply several artificial intelligence methodologies (like Case-Based Reasoning).

On the other hand, the drawbacks are manifold - in this paper, two of the main drawbacks are addressed. Firstly, the implemented agent systems are strongly connected with one certain application. As the result, the user models cannot be applied to any other domain without totally replacing the agent's internal knowledge base and the agent's set of actions. In this case, applying agent-technology does not lead to any benefit in that point. Secondly, the implemented individual behavior of each user is defined by the agent's developer and cannot be changed at runtime. For example, when the user is tracked by a local tracking system, the positioning process cannot switch to another wide-area sensor if the user leaves the local range.

3 Functional Agent Systems for User Modeling

To overcome the drawbacks mentioned above, we propose an approach that generally can be described as “user = set of multi-agent systems”. We base this approach on the following chain of human interaction within a domain:

information acquisition => understanding => decision finding => acting.

Obviously, this chain could be implemented based on the above-mentioned approach as well. The user is then represented by an agent, which encloses sensors to receive data from the domain, knowledge to enrich the collected information, self-controlling possibilities based on the user-model, and actuators influencing the domain. This section introduces a general architecture to implement more flexibility. To illustrate these ideas, a system for augmenting everyday environments with audio information (the LISTEN system, which was developed in an European founded project of the Fraunhofer Institute and its partners [Eckel, 2001]) will be given as an example.

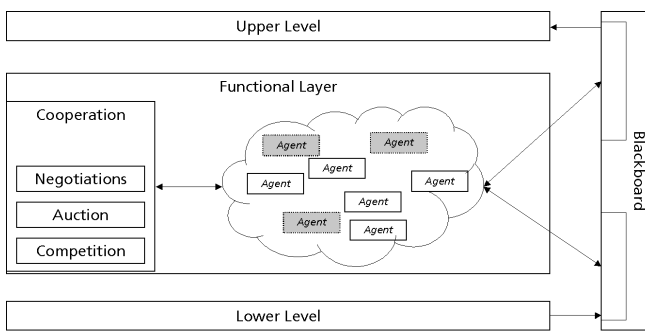


Figure 1: A Layer encapsulating a Set of Agents

3.1 Architecture

Depending on context, and environment, users act in real life in different roles. In the role of a curator of an exhibition, for example, the user will concentrate on aspects like how to order the exhibits, how to illuminate them, or whether to play a sound and what kind of sound, if any. In the role of the visitor of an exhibition, the same user will be more relaxed, feels the atmosphere of the lights and lets the underlying music sink in. Without much doubt, such a user will deliberately alternate between these roles at daily work.

In the above-discussed recent approaches the frequent change of roles would entail completely replacing the user representing agent. In our approach, for each of these roles should exist one highly specialized agent in an agent system in order to implement real individual behavior. At the functional layer of decision making, actually a whole multi-agent system can react better to changes in the user's context than a single agent. If the mentioned user acts in the role of the curator, s/he will decide to alter the order of two paintings. In the role of a visitor, s/he will never come up with such an idea. Using a concept of cooperation between the decision finding agents, an appropriate decision will be found. Furthermore, concepts to resolve conflicts between the agents can be implemented. For example, the agents representing the different roles could compete with each other for several control tasks; if the user's context changed, then other agents could come to the fore.

In Figure 1 the content of such a layer is illustrated. The set of agents communicates using a blackboard (see later). To implement cooperation concepts, negotiation-, auction-, and competition- platforms are provided. Some of the agents (in dotted boxes) could be inactive, e.g. because of their information demand is not fulfilled yet, or another agent is more specialized in processing the same task (and thus, the agent lost in the competition for this task).

Similar concepts can be applied for the other layers as illustrated in Figure 2. The user's ideas, preferences, relations with other entities, and so forth are then represented by sets of agents accommodated at several functional layers. By this means, the user is represented by cooperating sets of agent systems. The next subsections describe these functional layers in more detail.

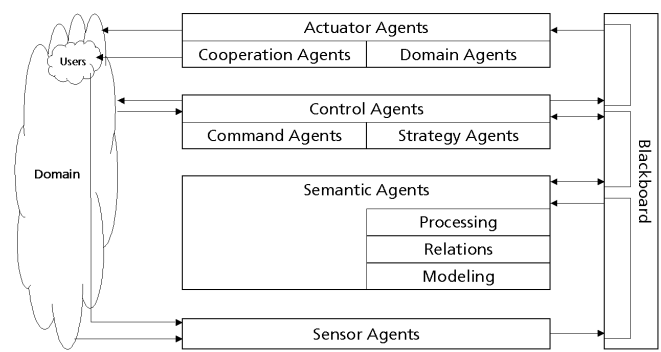


Figure 2: The Layered Architecture

Domain The types of domains for adaptive systems are manifold. Possible applications are eLearning systems, hypermedia, mobile services, etc., each introducing a research field with conferences and workshops on its own. Obviously, all of these application domains share a certain aspect: they are interacting with a user or groups of users. If we treat users of a domain as one special kind of a domain entity, then we can list the components a domain is composed of as follows: domain entities (domain objects and users), attributes of entities, relations between entities, and messages/events exchanged among entities. Every adaptive system needs to expressly map parts of these components to an internal model. Based on this description, context-frames are formulated in order to describe entities of the domain.

Example: In a museums guide application the domain has to cover art objects representing different contents (e.g. artists and epochs), as well as visitors with personal interests and preferences in arts. Both entities possess a spatial position.

Sensor Agents The very first functional layer serves as a pure information collector. A set of sensor agents is placed within the environment and connected to variable parameters of the domain. These sensors are used for recognizing every change within the environment and especially for the perception of the user's interaction with this environment, and therefore the agents need to be configured domain dependent. The content of the evolving data vectors are written to one partition of the blackboard and read by appropriate agents on the semantic layer.

Example: In order to receive tracking data, cameras are installed as sensors that continuously transmit a stream of float numbers representing the position and orientation of visitors. The sensor agents deliver this data in certain time intervals. In addition, visitors can provide information about their interests, preferences etc. via questionnaires.

Semantic Agents The set of semantic agents of our approach semantically enriches the data vectors delivered by the sensor agents to retrieve knowledge that can be further processed. The summary of agents on this layer defines the context model of an adaptive system capturing the current situation the users act in, their preferences, interests, their

social dependencies, and so forth. In addition, the context model contains information about the user's environment. On its way through the sub-layers of the semantic layer, the information delivered by the sensor agents successively gain a meaning, starting with arranging the sensor information into context-frames. The extracted knowledge is the basis of the decision making process.

Entity Modeling A set of contextual agents assigns the information bits received from the sensor agents to attributes of defined contexts-frames and from this, corresponding entities are instantiated. As the result, the contextual agents keep the information about the user's context up to date.

Relations The entities of the domain may relate to each other in certain ways. In order to express associations, aggregations, compositions and additional dependencies, a set of agents observe such relations and display this information on the corresponding partition of the blackboard.

Processing The set of agents on this layer, such as history agents, statistic agents, or other intelligent agents (e.g. to employ CBR methodologies), observe the evolution or process of contexts (or parts of contexts) over time. Since several inference mechanisms may derive different assumptions about the context, the agent system needs to be able to break up contradictory evidences or inconsistencies if necessary. As the result, the processing agents provide different views on the data captured about context to the control agents. In addition, plans of users can be recognized and learned on this level.

Example: On this layer, x- and y-coordinates arise from the stream of float numbers, and from adjacent values points in space are assembled. By determining the position and orientation of the visitor, relationships like *inFrontOf* and *isLookingAt* obtain a meaning. Additionally, the visitor's past behavior is processed, e.g. in order to find out his/her most interesting objects in future.

Control Agents This set of agents serves as the control center for the domain's adaptive behavior. The agents base their decisions on the knowledge provided by the semantic agents. In order to control the actuator agents for realizing the decision, control agents generate commands, or sequences of commands. Depending on the level of integration with the domain the contextualization service provided can either be based on adaptive methods implemented in the external application or on high-level requests to render adaptive components in the domain.

Strategy Agents The strategy agents define strategies on the highest level of how the overall system should behave in the interaction. These decisions are based on the interaction history and can be modeled using a variety of agents having different interests for influencing the user, rule-based approaches, or CBR approaches that try to find the best match of previous user interactions. In addition, the strategy agents

have to define the cooperation strategy with other users within the domain.

Command Agents The command agents use the knowledge about the current context of a user to control the behavior of the actuator agents, or to answer simple requests from external systems or the domain as well. Using the results of the strategy agents, the agreed-upon strategies are transformed into sequences of commands.

Example: When the visitor focuses on a certain art object (indicated by the *isLookingAt*-relationship), the decision to play an underlying sound (e.g. a spoken text about the object's background) will be taken. In order to control the behavior of the actuator agents, this decision is translated into a command like *playSound(soundID)*. The choice of *soundID* depends on the history of the user's visit, the visitor's preferences, and his/her knowledge in arts. To embark on a strategy, *prompting* (i.e. drawing the visitor's attention to a special object) is used to guide a visitor on a specific tour to art objects that have not been visited so far. For this purpose, a command *prompt(objectID)* is sent when the current object loses the visitor's focus.

Actuator Agents The set of actuator agents handle the connection back to the domain in order to transform the commands of the control agents into domain actions. One sub-set of these agents implements domain-dependent methods to directly change variable parameters of the domain. A second sub-set, the cooperation agents, create and send the corresponding messages when one of the control agents has decided to cooperate with another entity of the domain.

Example: In case of *playSound(soundID)*, the file associated with *soundID* is loaded into the sound player. In the *prompt(objectID)* case, an attractor sound is emitted from the physical position of the object associated with *objectID*.

3.2 Communication between the Agents

To enable the several agents and agent systems to cooperate with each other, a blackboard approach for sharing information is applied. Basically, the blackboard represents a very common platform [Ash and Hayes-Roth, 1990] where the agents can write information down and read the results written by others. In order to implement this approach in an efficient way for a huge number of agents, we decided to partition the blackboard by the type of the shared information. By this means, the agents of two adjacent layers share access to one partition. As the result, the agents of one level read the information delivered by the sub-layer agents from one partition, and write their own results back to another partition. For example, all semantic agents can read and manipulate the lowest partition of the blackboard (cf. Figure 2) that only the control agents can read.

In order to enable information transfer between a number of users, cooperating agents are defined to send messages/events to external receivers. These agents behave in the same way as the other domain-influencing agents.

4 Benefits

Besides the advantages as described for the recent approaches (as stated before), this approach enables the use of highly specialized agents to fulfill any aspect of the functional layers. Additionally, the overall system becomes more flexible to react to changed user contexts. Getting back to the example of a user whose position is delivered by a local-area tracking system: If the user leaves the range of the local system, a wide-area sensor agent (like a GPS-agent) can now deliver the user's position (probably in less granularity). On the upper levels, the agents can still work as they did before - or they can also change in order to react to the less granular information, if necessary.

In addition to the possibilities of changing the configuration at runtime, by implementing several roles, and several ways of thinking, differences between individuals can be represented more appropriately. Users who behave individually can be modeled by different configurations of the agent systems for the several layers. For employing the user representation in another domain, the domain-independent agents of the framework are retained unchanged. For example, the *prompt*-command agent can also be used in an eLearning system to guide the pupil to the next lesson. Of course, the actuator agent who transfers the command into domain actions (like blinking hyperlinks) must be replaced.

In addition, conflicts inside users themselves can be explicitly recognized. Regarding for example the limited credibility of a person, taking one decision can interfere with the possibilities of taking another one. In this case, such conflicts between the decision making agents on the control layer can be solved by negotiations - without any effect on the agents placed on the other layers except those actuator agents who need to be (re-)configured to realize the decision.

5 Conclusion and Future Work

In this paper, we have presented a generalized architecture to separate the functionalities of user modeling on several layers. Each of these layers will accommodate a multi-agent system to provide a flexible and highly specialized implementation. To transfer knowledge between the internal agent systems, we have introduced a partitioned blackboard-approach. In order to cooperate with external systems, a special set of agent has been defined.

The approach of separating the user modeling component into functional layers has already been defined, and implemented, for the mentioned LISTEN-project conducted by the Fraunhofer Institute and partners. The next steps are the ongoing definition of functional-agent sets and the mechanisms for cooperation and communication, and their implementation and evaluation of the profitableness for several domains.

References

- [Ash and Hayes-Roth, 1990] D. Ash and B. Hayes-Roth. Temporal representations in blackboard architectures. *Knowledge Systems Laboratory*, 1990.
- [Billsus and Pazzani, 2000] D. Billsus and M.J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2):147–180, 2000.
- [Eckel, 2001] G. Eckel. Listen - augmenting everyday environments with interactive soundscapes. In *13 Spring Days Workshop "Moving between the physical and the digital: exploring and developing new forms of mixed reality user experience"*, Porto, Portugal, 2001.
- [Fink and Kobsa, 2002] J. Fink and A. Kobsa. User modeling for personalized city tours. *Artificial Intelligence Review*, 18(1):33–74, 2002.
- [FIPA] FIPA Foundation for Intelligent Physical Agents. <http://www.fipa.org>.
- [Gross and Specht, 2001] T. Gross and M. Specht. Awareness in context-aware information systems. In Oberquelle, Oppermann, and Krause, eds., *Mensch und Computer - 1. Fachuebergreifende Konferenz*, pages 173–182, Bad Honnef, Germany, 2001. Teubner-Verlag.
- [Jennings and Wooldridge, 1998] N.R. Jennings and M.J. Wooldridge, eds. *Agent Technology*. Springer, 1998.
- [Kobsa, 2001] A. Kobsa. Generic user modeling systems. *Journal of User Modeling and User-Adaptive Interaction*, 11(1):49–63, 2001.
- [Menczer and Belew, 2000] F. Menczer and R.K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the web. *Machine Learning*, 39(2/3):203–242, 2000.
- [Odell, 2002] J. Odell. Objects and agents compared. *Journal of Object Technology*, 1(1):41–53, 2002.
- [Padgham and Taylor, 1997] L. Padgham and G. Taylor. A system for modeling agents having emotion and personality. In L. Cavedon, A. Rao, and W. Wobcke, eds., *Intelligent Agent Systems*, pages 59–71. Springer-Verlag, 1997.
- [Pazzani and Billsus, 2002] M.J. Pazzani and D. Billsus. Adaptive web site agents. *Autonomous Agents and Multi-Agent Systems*, 5(2):205–218, 2002.
- [Poslad *et al.*, 2001] S. Poslad, H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf. CRUMPET: Creation of user-friendly mobile services personalised for tourism. In *2nd International Conference on 3G Mobile Communication Technologies*, pages 26–29, London, UK, 2001.
- [Vassileva *et al.*, 1999] J.I. Vassileva, J.E. Greer, and G.I. McCalla. Openness and disclosure in multi-agent learner models. In *Workshop on Open, Interactive, and Other Overt Approaches to Learner Modelling*, Le Mans, France, 1999.
- [Webber *et al.*, 2001] C. Webber, L. Bergia, S. Pesty, and N. Balacheff. The baghera project: a multi-agent architecture for human learning. In J.I. Vassileva, ed., *Workshop on Multi-Agent Architectures for Distributed Learning Environments*, pages 12–17, San Antonio, TX, USA, 2001.
- [Webber *et al.*, 2002] C. Webber, S. Pesty, and N. Balacheff. A multi-agent and emergent approach to learner modelling. In F. van Harmelen, ed., *Proceedings of ECAI 2002*, Amsterdam, Netherlands, 2002. IOS Press.
- [Wooldridge, 1996] M.J. Wooldridge. *Intelligent Agents II*, chapter II, pages 97–110. Springer-Verlag, 1996.