Zebra: A New User Modeling System for Triangular Model of Learners' Characteristics

Loc NGUYEN[a,1]
[a] *University of Science, Ho Chi Minh city, Vietnam*

Abstract. The core of adaptive system is the user model that is the representation of information about an individual. User model is necessary for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users. The system that collects user information to build up user model and reasons out new assumptions about user is called user modeling system (UMS). There are two main tendencies towards implementing UMS: domain-independent UMS and domain-dependent UMS. The latter is called generic UMS known widely but our approach focuses on the domain-dependent UMS applied into adaptive e-learning especially. The reason is that domain-independent UMS is too generic to "cover" all learners' characteristics in e-learning, which may cause unpredictable bad consequences in adaptation process. Note that user is considered as learner in e-learning context. Many users' characteristics can be modeled but each characteristic is in accordance with respective modeling method. It is impossible to model all learners' characteristics because of such reason "there is no modeling method fit all characteristics". To overcome these obstacles and difficulties, we propose the new model of learner "Triangular Learner Model (TLM)" composed by three main learners' characteristics: knowledge, learning style and learning history. TLM with such three underlying characteristics will cover the whole of learner's information required by learning adaptation process. The UMS which builds up and manipulates TLM is also described in detail and named Zebra. We also propose the new architecture of an adaptive application and the interaction between such application and Zebra.

Keywords. User modeling system, adaptive learning

Introduction

User modeling system (UMS) is defined as the system that collects information about to build up user model and reason out new assumptions about user. UMS (s) have a long evolutionary process from early user modeling systems embedded into specific application to user modeling shells and user modeling servers which are separated from adaptive system and communicate with adaptive system according to client-server architecture. Note that the term "UMS" indicates both user modeling shell and user modeling server in this paper.

Before discussing main topic, we should glance over existing UMS in section 1. Section 2 described the Zebra – our modeling system in detailed. Section 3 is the conclusion.

[1] PhD student at University of Natural Science, Vietnam.

1. Existing user modeling systems

1.1. Early user modeling systems

Early UMS (s) concentrate on question-answer (dialog) system and human-computer interaction. They are components embedded in concrete application. An example for such dialog system is GRUNDY [1] developed Rich in his PhD thesis. GRUNDY play the role of book recommender in the library when it calculates recommendation of books, based on assumptions about users' personal traits. Such traits which are educational and intellectual level, preference for thrill, fast-moving plots or romance, tolerance for descriptions of sexuality, violence and suffering… are represented as user model. GRUNDY use stereotype method [1] to build up user model, based on users' answers to questions during their first usage of system. For example, if user has a mail first name, GRUNDY infers a high sex tolerance and a low one for romance.

1.2. User modeling shells

There is a need for developing UMS (s) as separated components whose functionality is not dependent on any adaptive application. Such UMS (s) are called user modeling shell. The term "shell" is borrowed from the field of expert system; thereby, the purpose of shells is to separate user modeling functionality from adaptive application.

User modeling shell goes towards generic purpose but it is not totally independent on application and often integrated into application when it is deployed. Examples of user modeling shell are GUMS, UMT, PROTUM, TAGUS, um.

GUMS

GUMS [2] is the abbreviation of "General User Modeling System" developed by Tim Finin in his PhD thesis. It is a first modeling shell that abstracts information about user by allowing defining the simple stereotype [1] hierarchies in form a tree of structure. Each stereotype is associated facts and rules describing system's reasoning about it. Moreover GUMS interacts with specific application by storing facts that application provides and answering any queries of application concerning assumptions about user.

UMT: UMT (User Modeling Tool) developed by Brajnik and Tasso [3] models information about users as stereotypes which contain their assumptions in form of attribute-value pairs. UMT allows developers/specialists to define hierarchical user stereotypes, triggers, rules for user model inferences and contradiction conditions.

PROTUM: PROTUM (PROlog based Tool for User Modeling) [4] is written by the programming language PROLOG. It is more powerful than UMT although it uses stereotype method to model user like UMT did because its hierarchy of stereotypes is not limited to tree structure and assumptions about users are not based on attribute-value pairs like UMT.

TAGUS: TAGUS developed by Paiva and Self [5] also applies stereotype method into modeling user. Thus it allows defining the hierarchical stereotype but each assumption about user in stereotype is represented in first-order formulas with meta-operators expressing the type of assumption such as: belief, goal…

um: um developed by Kay [6] aims to provide the library of user modeling functionalities in which assumptions about users' knowledge, goal, background… are represented in attribute-value pairs

1.3. User modeling servers

User modeling server has the same purpose with user modeling shell when both of them aim to separate user modeling functionality from adaptive system. However user modeling server is totally independent on application. It is not integrated into applications and interacts with applications through inter-process communication. It can reside on the different site from application's site and serve more than one instance of application at the same time. The communication between user modeling server and adaptive application is based on client-server architecture in which modeling server is responsible for answering application's requests. Examples of user modeling server are BGP-MS, Doppelgänger, CUMMULATE, Personis.

BGP-MS: BGP-MS developed by Kobsa and Pohl [7] is the user modeling server taking interest in user's knowledge, belief and goal. It receives user's observations provided by adaptive application and processes internal operations of classification and calculation based on these observations. BGP-MS use stereotype method, natural language dialogs and questionnaires to build up user model. BGP-MS has four essential components:

- *Individual user model* contains assumptions about user.
- *Stereotype* component manages the hierarchy of stereotypes.
- *Automatic stereotype management* is responsible for the activation and deactivation of assigned stereotypes of an individual user model

These main components communicate together through the *functional interface*. Developer interacts with BGB-MS by the *graphic interface.*

Doppelgänger: Doppelgänger [9] developed by Orwant is the server that monitors users' actions and discovers patterns from these actions. Basing on such patterns, Doppelgänger aims to deliver user a personalized daily newspaper; it provides news in which user can be interested. The architecture of Doppelgänger is split into two levels: sensor level and sever level

- *Sensor level*. There are sensors having responsibility for gathering information about user. Sensors can be either software or hardware.
- *Server level*: Make inferences on information provided by sensors.

CUMMULATE: CUMMULATE [10] is developed as generic student-modeling server in which information about user is represented on two levels: *event storage* and *inferenced user model*. Student actions being monitored are sent to event storage by a standard http-based event-reporting protocol. Such actions are considered events. CUMMULATE adds a timestamp to each event and stores it permanently in event storage. The event storage allows several *inference agents* that process events in different ways and convert these events into the inferenced user model, for example, some agents monitor user's knowledge and others predict user's interests. The architecture of CUMMULATE is open to a variety of *external inference agents* that receive requests from event storage and manipulate user model.

Personis: Personis [11] is the modeling server whose considerable feature is to allow user to control and scrutinize his/her model. Such user model is called scrutable user model. Personis is based on um toolkit but more complicated than um toolkit. The architecture of Personis is divided into four parts:

- The *server itself* is responsible for managing user model.
- A set of *generic scrutiny tools* allow users to see and control their own user models.
- A set of *Adaptive Hypermedia Application* (AHA) are denoted AHA_1, AHA_2,…, AHA_n.
- A set of *views* in which each view of user model available to each AHA is responsible for defining the components used by such AHA.

LDAP-UMS: LDAP-UMS [12] is the user modeling severs based on Lightweight Directory Access Protocol (LDAP). It focuses on data handling and enhances user modeling functionality by enabling user model to achieve "pluggable" feature. UMS uses a directory structure of LDAP to manage user's information spreading across a network. The architecture of LDAP-UMS inherits the LDAP directory server; so, UMS is composed of several pluggable user modeling components and can be accessed by external clients. The core of architecture is the *Directory Component* attached its three sub-components: *Communication*, *Representation* and *Scheduler*.

2. Zebra: A User Modeling System for Triangular Learner Model

Exist user modeling systems develop fast in recent years; they are trending towards servers that give support to adaptive applications with fully response to queries about user information available in user model. However we recognize that generic UMS (s) are too generic to describe all fine characteristics of user when she/he is learner in e-learning context. Especially in situation that our research focuses on domain of e-learning, such UMS (s) prove to be less effective in providing assumptions about user to adaptive learning applications. In learning environment, users who play role of learners must be modeled by special method. The content of learner model can be divided into two categories: domain specific information and domain independent information. Domain specific information is knowledge that learner achieved in certain subjects. Otherwise domain independent information includes personal traits not related to domain knowledge such as: interests, learning styles, demographic information… Each kind of information is in accordance with respective modeling method. For example, knowledge model is often created by overlay method, which called overlay model. So it is impossible to model all learners' characteristics because of such reason "there is no modeling method fit all characteristics".

To overcome these obstacles and difficulties, we propose the new learner model that contains three most important characteristics of user: knowledge (**K**), learning styles (**LS**) and learning history (**LH**). Such three characteristics form a triangle; so our model is called Triangular Learner Model (TLM). TLM with three underlying characteristics will cover the whole of user's information required by learning adaptation process. The reasons for such assertion are:

- Knowledge, learning styles and learning history are prerequisite for modeling learner.

- While learning history and knowledge change themselves frequently, learning styles are relatively stable. The combination of them ensures the integrity of information about learner.
- User's knowledge is domain specific information and learning styles are personal traits. The combination of them supports UMS to take full advantages of both domain specific information and domain independent information in user model.

We also introduce the architecture of UMS which builds up TLM; it is named Zebra. The name "Zebra" implicates that our UMS will run fast and be powerful like African zebra.

2.1. Triangular Learner Model

TLM is constituted of three basic features of user: knowledge, learning styles and learning history which are considered as three apexes of a triangle (see figure 1). Hence TLM has three sub-models: *knowledge sub-model*, *learning style sub-model* and *learning history sub-model*.

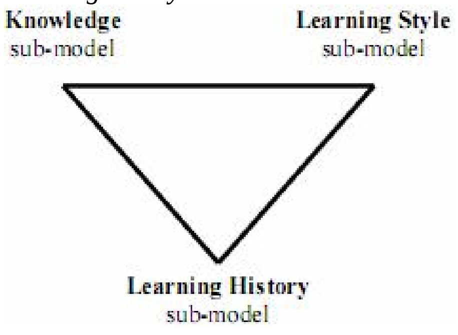


Figure 1: Triangular Learner Model

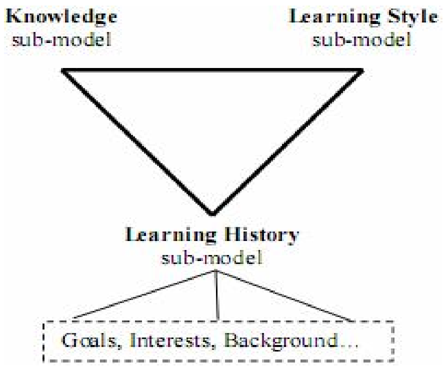


Figure 2: extended Triangular Learner Model

We propose the method that combines overlay method and Bayesian network [13] to build up knowledge. In overlay method, the domain is decomposed into a set of knowledge elements and the overlay model (namely, user model) is simply a set of masteries over those elements. The Bayesian network is the directed acyclic graph (DAG) in which the nodes are linked together by arcs; each arc expresses the dependence relationships between nodes. The strengths of dependences are quantified by Conditional Probability Table (CPT). The combination between overlay model and BN is done through following steps:

- The structure of overlay model is translated into Bayesian network, each user knowledge element becomes an node in Bayesian network
- Each prerequisite relationship between domain elements in overlay model becomes a conditional dependence assertion signified by CPT of each node in Bayesian network

So **knowledge** sub-model is called as Bayesian overlay sub-model.

Learning styles are defined as the composite of characteristic cognitive, affective and psychological factors that serve as relatively stable indicators of how a learner perceives, interacts with and responds to the learning environment. There are many models of learning styles in theory of psychology such as: Dunn and Dunn, Witkin, Riding, Myers-Briggs, Kolb, Honey-Mumford, Felder-Silverman… We choose Honey-

Mumford [14] and Felder-Silverman model [15] as principal models which are presented by Hidden Markov Model (HMM) [16]. According to Honey-Mumford and Felder-Silverman model, learning styles are classified into following dimensions:

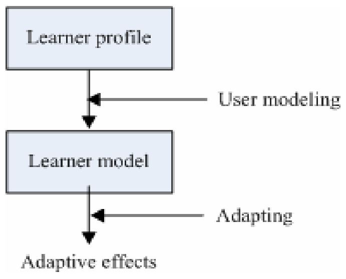
- *Verbal/Visual*. Verbal students like learning materials in text form. Otherwise visual student prefer to images, pictures…
- *Active/Reflective*. Active students understand information only if they discussed it, applied it. Reflective students think thoroughly about things before doing any practice
- *Theorist/ Pragmatist*. Theorists think things through in logical steps, assimilate different facts into coherent theory. Pragmatists have practical mind, prefer to try and test techniques relevant to problems

For modeling learning style using HMM we must define states, observations and the relationship between states and observations in context of learning style. So each learning style is now considered as a state. The essence of state transition in HMM is the change of user's learning style, thus, it is necessary to recognize the learning styles which are most suitable to user. After monitoring users' learning process, we collect observations about them and then discover their styles by using inference mechanism in HMM. So learning style sub-model is modeled as HMM.

The last sub-model stores and manipulates learner's **learning history** in form of XML data files. All learners' actions: learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates… are logged in this sub-model. School reports also recorded in this sub-model. We consider this sub-model as a feature of learners because every student has individual learning process in her/his life and the data about such learning process are recorded as pieces of information in learning history sub-model. Information in this sub-model is necessary for data mining in e-learning to discover not only knowledge and learning styles but also other learners' characteristics such as interests, background, goals… The mining engine in the core of Zebra often uses this sub-model for many mining tasks. For this reason, this sub-model is drawn as the apex at the bottom of triangle in architecture of TLM. This implicates that learning history sub-model is the most important sub-model in TLM when it is considered as the basic of two other sub-models. Figure 2 shows the extended TLM in which learning history sub-model is the root for attaching more learners' characteristics such as interests, background, goals… to TLM.

2.2. The architecture of Zebra

The essence of user modeling systems is mining user's profile to discover valuable patterns in form of user's features. These features which are personal traits or characteristics in learning context navigate adaptive applications to give support to user in her/his learning path. The purpose of Zebra is to mine user's learning profile to build up her/his TLM. Hence Zebra has the inside *mining engine*.



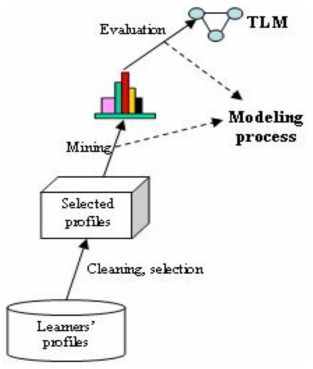


Figure 3: Modeling tasks are similar to profile mining tasks **Figure 4**: The mining process in Zebra

Moreover Zebra must implement the powerful inference mechanism to reason learners' new assumptions (or characteristics) out TLM. In section 3.1, we propose two methods: Bayesian network combined overlay model and hidden Markov model to infer learners' knowledge and styles. Both Bayesian network and Markov model are special cases of belief network. In general belief network is directed acyclic graphs in which nodes represent variables, arcs signify direct dependencies between the linked variables, and the strengths of these dependencies are quantified by conditional probabilities. Belief network is the robust mathematical tools appropriate to reasoning based on evidences. Zebra must have another inside engine – the *belief network engine*.

The core of Zebra is the composition of two engines: *mining engine* (**ME**) and *belief network engine* (**BNE**).

- Mining engine is responsible for collecting learners' data, monitoring their actions, structuring and updating TLM. Mining engine also provides important information to belief network engine; it is considered as input for belief network engine. In short, mining engine creates TLM by applying mining algorithms, for example, it is possible to modeling user's learning path by using sequential pattern mining. Mining engine has another important functionality that is to discover some other characteristics (beyond knowledge and learning styles) such as: interests, goals, learning context… This functionality is the extension of Zebra in future.
- Belief network engine is responsible for inferring new personal traits from TLM by using deduction mechanism available in belief network.

Zebra provides *communication interfaces* (**CI**) that allow users to see or modify restrictedly their TLM. Adaptive applications also interact with Zebra by these interfaces. *Communication interfaces* are implemented as web services used widely on internet. According to World Wide Web Consortium, a web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (especially WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. When complying with web service standard, it is possible to publish *CI* (s) on internet for third-parties to communicate with Zebra more effectively.

There is external program so-called *observer* having responsibility for tracking learners' actions. Observer catches and delivers user observations to Zebra. Observer interacts with Zebra through CI.

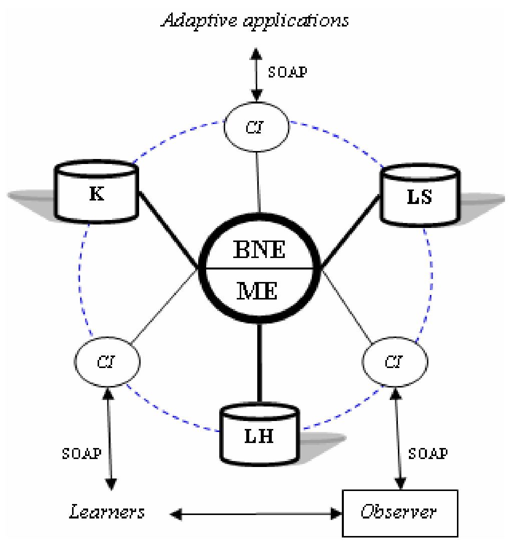


Figure 5: The architecture of Zebra.

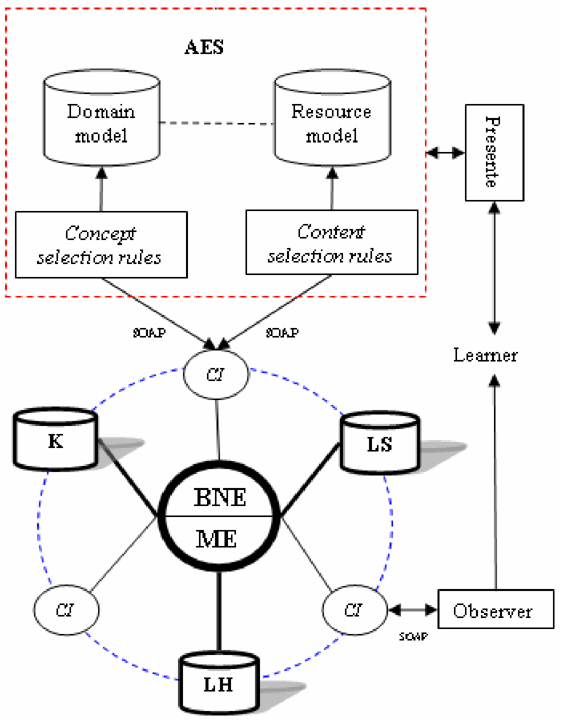


Figure 6: The new architecture of AES and the interaction between AES and Zebra

ME and BNE denote mining engine and belief network engine respectively. LH, K and LS denote Learning History sub-model, Knowledge sub-model and Learning Styles respectively. CI denotes communication interface.

2.3. The interaction between Zebra and adaptive applications

Zebra aims to support adaptive learning applications; so in this section we should glance over what adaptive applications are and discuss about how Zebra interacts with such applications. The most popular adaptive learning system supporting personalized learning environment is **AES** (**A**daptive **E**ducation **S**ystem). Here, AES is regarded as an example for illustrating prominent traits of adaptive learning system. The origin architecture of AES has four components:

- *Resource component*: contains learning resources (lecture, test, example, exercise…) and associated descriptive information (metadata).
- *Domain component*: constitutes the structure of domain knowledge so-called domain model. Domain model was often shown in the form of graph.
- *Adaptation component*: is the centric component which gives effect to adaptation. It contains *content selection rules* and *concept selection rules*. We apply content selection rules to choosing suitable educational resources from resource domain (component). On the other hand, concept selection rules are used to choose appropriate concept from domain. These rules must obey user model so that the selection gets correct.
- *User Model*: information and data about user.

We propose the new approach in which the *user model* is removed from AES and becomes the TLM managed by the user modeling system Zebra. Now AES owns only three components: resource component, domain component and adaptation component. The reason is that learner model (TLM) becomes too complex to be maintained by AES and AES should only focus on improving adaptation process and the performance of system will get enhanced when AES takes full advantage of functionalities of Zebra. All operations relating TLM are executed by Zebra instead of AES. AES interacts with Zebra via *communication interfaces* according to SOAP protocol. Figure 6 shows the new architecture of AES and the interaction between AES and Zebra. There are only three instances of communication interfaces but the number of them is not limited in practice.

The adaptation process performed by adaptation components includes two main sub-processes: concept selection process and content selection process.

- In *concept selection process*, concept selection rules are used to match learners' knowledge to concepts in domain model. In other words, these concepts are filtered to find ones which is necessary for learners to learn in their course.
- In *content selection process*, learning resources are selected from resource component based on content selection rules that match learners' learning styles to attributes of resources in resource space. In other words, this process finds the resources that learner preferred or suitable to learner.

The adaptation process includes following steps:

- *Step 1*: The projection of domain model onto knowledge sub-model by using concept selection rules results in a set of domain knowledge called A that student has to learn. This is concept selection process.
- *Step 2*: A is used as filter to choose a set of learning resources called B that relating to A.
- *Step 3*: The projection of B onto learning styles sub-model by using content selection rules results in a sub-set of learning resources (lectures, exercises, test…) called C tailoring to learner's preferences. This is content selection process. C is considered as a set of recommendation resources.
- *Step 4*: C is show in content presenter. Presenter can be human-machine interfaces, web sites, learning management system (LMS), teaching support applications…
- *Step 5*: Learner study C by interacting with content presenter.
- *Step 6*: Observer monitors learners in order to catch and delivers learner's observations to Zebra. Zebra uses such observations to update TLM.

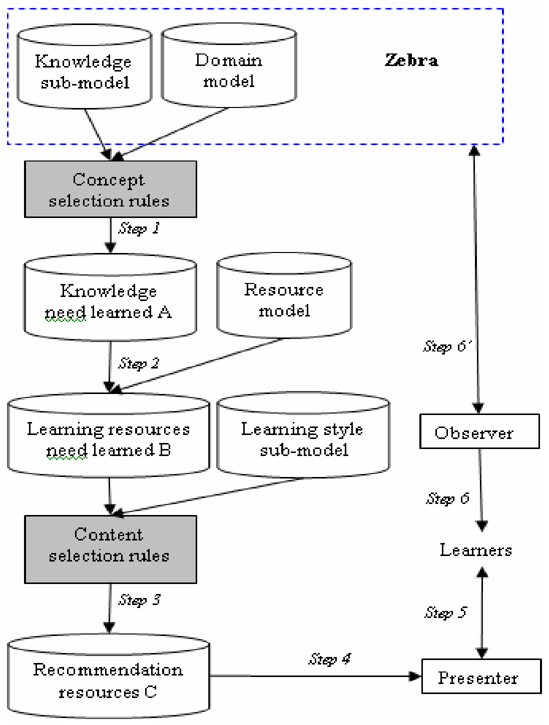


Figure 7: Steps in adaptation process

3. Conclusion

In this paper, the “Triangular Learner Model (TLM)” composed by three underlying characteristics: knowledge, learning style and learning history aims to cover the whole of learner’s information required by learning adaptation process. Hence TLM has three sub-models: knowledge sub-model, learning style sub-model and learning history sub-model which are considered as three apexes of a triangle.

UMS which builds up and manipulates TLM so-called Zebra includes the core engine and a set of communication interfaces. The core engine is the composition of two sub-engines: mining engine and belief network engine. Mining engine applies data mining algorithms into discovering and structuring TLM. Belief network engine is responsible for inferring new personal traits from TLM by using deduction mechanism available in belief network. Communication interfaces allow users to see or modify restrictedly their TLM and adaptive applications also interact with Zebra through them. We also propose the new architecture of AES (Adaptive Education System) that interacts with Zebra.

References

[1] Elaine Rich. User Modeling via Stereotypes. COGNITIVE SCIENCE 3, 329-354 (1979).

[2] Tim Finin, David Drager. GUMS: A General User Modeling System. Proceedings of the Canadian Society for Computational Studies of Intelligence 1986 (CSCSI-86).

[3] Brajnik, G. and Tasso, C. A Shell for Developing Non-monotonic User Modeling Systems. International Journal of Human-Computer Studies 40, (1994) 31-62, DOI: 10.1006/ijhc.1994.1003

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

[5] Paiva, A. and Self, J. TAGUS: A User and Learner Modeling System. Proceedings of the Fourth International Conference on User Modeling. Hyannis, MA (1994) 43-49.

[6] Judy Kay. The um Toolkit for Cooperative User Modelling. User Modeling and User-Adapted Interaction, vol. 4, no. 3 p.p. 149-196, 1995 (UMUA 95)

[7] Alfred Kobsa, Wolfgang Pohl. The User Modeling Shell System BGP-MS. User Modeling and User-Adapted Interaction 4(2), 59-106, 1995 (UMUAI-1995)

[8] Alfred Kobsa. First experiences with the SB-ONE knowledge representation workbench in natural-language applications. ACM SIGART Bulletin, vol. 2, no. 3 p.p. 70-76, 1991

[9] Jonathan L. Orwant. Doppelgänger: A User Modeling System. Bachelor thesis. Masschusetts institute of Technology June 1991

[10] Peter Brusilovsky. KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. Thirteen International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), New York, NY (2004) 104-113, 10.1145/1013367.1013386

[11] Judy Kay, Bob Kummerfeld, Piers Lauder. Personis: A server for user models. Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2002), P.p. 201-212, 2002

[12] Alfred Kobsa, Josef Fink. An LDAP-Based User Modeling Server and its Evaluation. User Modeling and User-Adapted Interaction 2006 (UMUAI-2006)

[13] Eugene Charniak. Bayesian Network without Tears. AI magazine. 1991

[14] Peter Honey, Alan Mumford. The Manual of Learning Styles. Maidenhead: Peter Honey Publications, 1992.

[15] R. M. Felder, L. K. Silverman. Learning and Teaching Styles in Engineering Education. Journal of Engineering Education, 1988.

[16] R. Dugad, U. B. Desai. A tutorial on Hidden Markov models. Signal Processing and Artificial Neural Networks Laboratory, Dept of Electrical Engineering, Indian Institute of Technology, Bombay Technical Report No.: SPANN-96.1, 1996