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

Loc Nguyen

University of Science, Ho Chi Minh city, Vietnam

Email: ng\_phloc@yahoo.com

Bich-Thuy T. Dong

University of Science, Ho Chi Minh city, Vietnam

Email: thuy@hcmus.edu.vn

## Abstract

The core of adaptive system is the user model that is 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, user model, learner model, Triangular Learner Model, TLM, Zebra.

## 1. Introduction

The core of adaptive system, as aforementioned in (Nguyen & Do, Learner Model in Adaptive Learning, 2008), 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). Based on the survey of user model and adaptive learning mentioned in (Nguyen & Do, Learner Model in Adaptive Learning, 2008) and (Nguyen, State of the Art of Adaptive Learning, 2009), there are two main tendencies towards implementing UMS: domain-dependent UMS and domain-independent UMS. The latter is called generic UMS known widely but my 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 in the same UMS because of such reason “there is no modeling method fit all characteristics”.

To overcome these obstacles and difficulties, I 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. I also propose the new architecture of an adaptive application and the interaction between such application and Zebra.

Before discussing main topic, we should glance over existing UMS (s) in section [2](#_II.1._Existing_user). Section [3](#_II.2._Zebra:_A) described the Zebra – my modeling system in detailed. Section [4](#_V.2._Towards_ubiquitous) concerning future trend of TLM and Zebra is an example of TLM extension, in which TLM will be integrated into ubiquitous service. This means that Zebra will support ubiquitous user modeling. Section [5](#_II.3._Conclusion) is the conclusion.

## 2. Existing user modeling systems

Recall that user modeling system (UMS) is defined as the system that collects user information 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 section [2](#_II.1._Existing_user). The main content of this section [2](#_II.1._Existing_user) is mainly extracted from the master thesis “*User Modeling and User Proﬁling in Adaptive E-learning Systems*” of the author (Fröschl, 2005) and the article “*Generic User Modeling Systems*” of the author (Kobsa, 2007). I express my deep gratitude to the authors Fröschl and Kobsa for providing their great researches.

### 2.1. Early user modeling systems

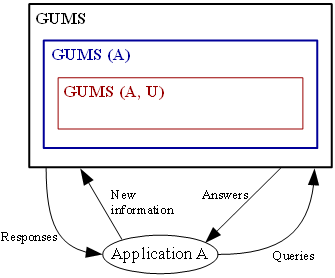
Early UMS (s) concentrate on question-answer (dialog) system and human-computer interaction. They are components embedded in concrete application. An example for such dialog system is GRUNDY developed by author (Rich, 1979) in her PhD thesis. GRUNDY plays the role of book recommender in the library when 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, etc. are represented as user model (Fröschl, 2005, p. 59). GRUNDY uses stereotype method (Rich, 1979) to build up user model, based on users’ answers to questions during their first usage of system. For example, if user has a male first name, GRUNDY infers a high sex tolerance and a low one for romance.

### 2.2. 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** is the abbreviation of “General User Modeling System” developed by authors (Finin & Drager, 1986). It is a first modeling shell that abstracts information about user by allowing defining the simple stereotype (Rich, 1979) hierarchies in form a tree of structure. Each stereotype is associated facts and rules describing system’s reasoning about it (Fröschl, 2005, p. 60). Users are classified into such stereotypes. The initial stereotype of user is assigned by system and can be altered later according to observations about her/him. Both GUMS and GRUNDY apply stereotype method into modeling user. Moreover GUMS interacts with specific application by storing facts that application provides and answering any queries of application concerning assumptions about user. GUMS ensures the consistency of new facts with old assumptions about user in user model. If any inconsistency takes place, GUMS will inform to application by a response (see figure [2.2.1](#_Figure_II.1._The)) (Finin & Drager, 1986, p. 225).



**Figure 2.2.1.** The architecture of GUMS

Where A, GUMS(A) and GUMS(A, U) denote application A, modeling system for application A, and model for user U in application A, respectively (Finin & Drager, 1986, p. 225).

**UMT** (User Modeling Tool) developed by authors (Brajnik & Tasso, 1994) models information about users as stereotypes which contain their assumptions in form of attribute-value pairs (Brajnik & Tasso, 1994, p. 41). UMT allows developers/specialists to define hierarchical user stereotypes, triggers, rules for user model inferences and contradiction conditions (Kobsa, 2001, p. 50). At the first time user interacts with system, trigger is activated to assign her/his to an initial stereotype. The assumptions of initial stereotype are added into user model by applying inference rules. Some of these assumptions can be retracted whenever contradictions occur.

**PROTUM** (PROlog based Tool for User Modeling) developed by author (Vergara, 1994) is written by the functional language PROLOG, please see (Wikipedia, Prolog, 2014) for surveying language PROLOG. PROTUM 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 (Fröschl, 2005, p. 63). Moreover PROTUM determines the activation rates of triggers in order to activate or deactivate stereotypes which have been already assigned to user. These rates are used to resolve conflicts between inconsistent assumptions of two activated stereotypes (Fröschl, 2005, p. 63).

**TAGUS** developed by authors (Paiva & Self, 1995) 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 and goal (Kobsa, 2001, p. 51). TAGUS also has inference mechanism and truth maintenance discovering contradictions of assumptions and diagnosing unexpected user’s behaviors (Kobsa, 2001, p. 51).

**um** developed by author (Kay, 1995) aims to provide the library of user modeling functionalities in which assumptions about users’ knowledge, goal, background, etc. are represented in attribute-value pairs. So um is often considered as *um toolkit*. User model is composed of pieces of information called as components accompanied by the set of evidences for user verification. There are four types of components: preferences, knowledge, beliefs, and attributes. Each evidence consists of four parts: reliability class, source identifier, part identifier, and timestamp (Kay, 1995, pp. 166-168). Source identifier refers to the computer program that produces evidence. Part identifier is the sub-part or sub-function of such program. Timestamp tells us when evidence is produced or collected. Reliability class is used to classify evidences and there are five types of reliability class such as observation, stereotype activation, rule invocation, user input and told to the user (Kobsa, 2001, p. 51). So um combines stereotype method and rule-based method in implementation.

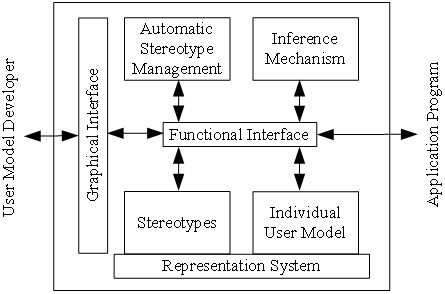
### 2.3. User modeling servers

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 from application. It is not integrated into applications and interacts with applications through inter-process communication (Kobsa, 2007, p. 3). 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, CUMULATE, Personis.

**BGP-MS** developed by authors (Kobsa & Pohl, 1995) is the user modeling server taking interest in user’s knowledge, belief and goal. Note that the name BGP-MS is abbreviation of Belief, Goal and Plan Maintenance System (Kobsa & Pohl, 1995, p. 4). Although BGP-MS is really a user modeling shell, it can be considered as user modeling server because communication protocol inside BGP-MS allows it to be deployed as network server which responses many adaptive applications (Kobsa, 2007, p. 5). BGP-MS receives user’s observations provided by adaptive application and processes internal operations of classification and calculation based on these observations (Fröschl, 2005, p. 63). BGP-MS uses stereotype method, natural language dialogs and questionnaires to build up user model. BGP-MS has four essential components (Fröschl, 2005, p. 64):

* *Individual user model* contains assumptions about user.
* *Stereotype* component manages the hierarchy of stereotypes. Both *stereotypes* and *individual user model* are based on the *representation system* which is the integrated suite of knowledge representation mechanism for representing assumptions about user. Representation system is based on the conceptual knowledge representation language SB-ONE (Kobsa & Pohl, 1995, p. 16). SB-ONE has two levels such as general level and individualized level. In the general level, representational elements are general concepts and general attribute descriptions. In the individualized level, assumptions about user are represented based on using individualized concepts and individualized attribute descriptions which are linked to respective general concepts and general attribute descriptions specified in the general level (Kobsa, 1991, p. 71).
* *Automatic stereotype management* is responsible for the activation and deactivation of assigned stereotypes of an individual user model (Fröschl, 2005, p. 64).

These main components communicate together through the *functional interface*. Developer interacts with BGB-MS by the *graphic interface*. The architecture of BGB-MS is represented in figure [2.3.1](#_Figure_II.2._The) (Fröschl, 2005, p. 64).



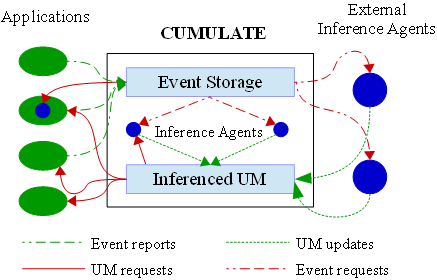
**Figure 2.3.1.** The architecture of BGP-MS

As seen in figure [2.3.1](#_Figure_II.1.3.1._The), the adaptive system communicates with main components through functional interface. BGP-MS defines the new communication protocol called KN-IPCMS where KN-IPCMS is abbreviation of KoNstanz Inter-Process Communication Management System (Kobsa & Pohl, 1995, p. 12).

**Doppelgänger** developed by author (Orwant, 1991) 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 (Fröschl, 2005, pp. 66-68).

* *Sensor level*. There are sensors having responsibility for gathering information about user. Sensors can be either software or hardware. There are specific techniques inside sensors in order to extract valuable information from users’ actions. Each sensor has its own specific purpose, for example, one gathers amount of time of computer use and another tracks user’s physical location (Fröschl, 2005, p. 66). User model is stored as LISP-like list, for example, a piece of user model expressing “it is likely that user Orwant prefers to read the news topic *Olympics*” is described as follows: “(*object orwant news* (*object preferences* (*assertion* (*likes topic olympics* (*strength* 0.8)(*probability* 0.9))(*confidence* 0.8)(*technique beta*))…)…)” (Orwant, 1995, p. 6). Please see (Wikipedia, Lisp (programming language), 2014) for surveying programming language LISP.
* *Server level*. Doppelgänger makes inferences on information provided by sensors. For example, when sensors provide events such as “user often cares about stock market index in the evening and he usually reads sport news after lunch”, Doppelgänger can predict user’s habit and tell the news recommendation system such habit. The author (Orwant, 1995, pp. 10-23) uses three learning techniques for inference tasks such as predicting user’s preference by beta distribution, modeling time event by linear prediction, and predicting physical location by Markov model. Markov model is introduced in (Fosler-Lussier, 1998) and (Schmolze, 2001).

**CUMULATE** developed by authors (Brusilovsky, Sosnovsky, & Shcherbinina, 2005) is a 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 with note that HTTP protocol is introduced in (Wikipedia, Hypertext Transfer Protocol, 2014). Such actions are considered events. CUMULATE adds a timestamp to each event and stores it permanently in event storage (Brusilovsky, Sosnovsky, & Shcherbinina, 2005, p. 2). 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 CUMULATE is open to a variety of *external* *inference agents* that receive requests from event storage and manipulate user model. CUMULATE is really a combination of event-driven approach and agent-based approach, which is prominent in distributed e-learning environment. Figure [2.3.2](#_Figure_II.3._The) shows its architecture (Brusilovsky, Sosnovsky, & Shcherbinina, 2005, p. 3) (Brusilovsky, 2004, p. 5).

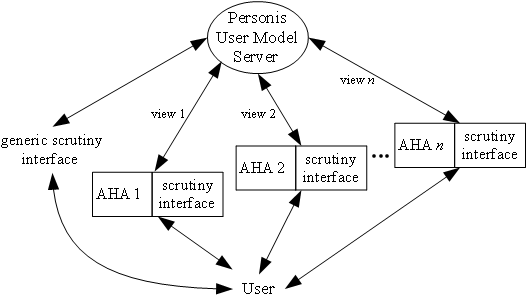


**Figure 2.3.2.** The architecture of CUMULATE

**Personis** developed by authors (Kay, Kummerfeld, & Lauder, 2002) is the modeling server whose considerable feature is to allow user to control and scrutinize her/his model (Kay, Kummerfeld, & Lauder, 2002, p. 203) (Fröschl, 2005, p. 68). Such user model is called scrutable user model. Personis is based on um toolkit but more complicated than um toolkit. The high-level architecture of Personis is divided into four parts such as *the* *server itself*, *generic scrutiny tools*, *adaptive hypermedia applications,* and *views* (Kay, Kummerfeld, & Lauder, 2002, pp. 205-207).

* The *server itself* is responsible for managing user model.
* A set of *generic scrutiny tools* allow users to see and manipulate their own user models. Users interact with these tools through a generic scrutiny interface. This interface is application-independent.
* A set of *adaptive hypermedia applications* (AHA) are denoted AHA1, AHA2,…, AHAn. Each AHA includes two parts: first, the core enables user to do some learning tasks and second, the scrutiny interface associated with the core enables the core to interact with scrutiny tools. This interface is similar to generic scrutiny interface except that it belongs to the context of AHA.
* There is a set of *views* of user model; each view is available to each AHA and responsible for defining the components used by such AHA with note that components are parts of user model (Kay, Kummerfeld, & Lauder, 2002, p. 206). Applications can use either the same or different views.

Figure [2.3.3](#_Figure_II.4._The) shows the high-level architecture of Personis (Kay, Kummerfeld, & Lauder, 2002, p. 205).

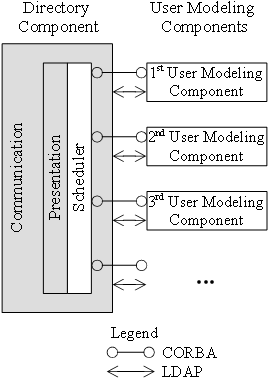


**Figure 2.3.3.** The architecture of Personis

**LDAP-UMS** is the user modeling severs based on Lightweight Directory Access Protocol (LDAP), published by (Fink, 2004) in his PhD thesis. 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. Please see (Wikipedia, Lightweight Directory Access Protocol, 2014) for surveying LDAP. 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* (Fink, 2004, pp. 75-76).

* *Communication* sub-component handles the communication between external clients and *Directory Component*, between *Directory Component* and *User Modeling Components*. Each *User Modeling Component* is dedicated to perform user modeling tasks such as collecting user information and inferring new assumptions about user. The *Directory Component* and *User Modeling Components* interact together via CORBA (Wikipedia, Common Object Request Broker Architecture, 2014) and LDAP.
* *Representation* sub-component is responsible for managing the directory contents.
* *Scheduler* sub-component is responsible for wrapping the underlying LDAP server with a component interface and harmonizing different sub-systems with *User Modeling Components* (Fink, 2004, p. 75).

This architecture allows developer to add more self-developed *User Modeling Components* to LDAP-UMS. So LDAP-UMS is the open and flexible user modeling server, which becomes powerful one among modern user modeling servers. Figure [2.3.4](#_Figure_II.5._The) shows the architecture of LDAP-UMS (Fink, 2004, p. 75) (Fröschl, 2005, p. 71).



**Figure 2.3.4.** The architecture of LDAP-UMS

Readers are recommended to read the PhD thesis “User Modeling Servers – Requirements, Design, and Evaluation” of the author (Fink, 2004) in order to understand comprehensively the powerful system LDAP-UMS.

Now existing user modeling systems were introduced briefly in this section [2](#_II.1._Existing_user) and most of them aim to serve requests for generic user information. The next section [3](#_II.2._Zebra:_A) is to propose a user modeling system which is designed and implemented for specific situation in which users (main modeled objects) are learners in learning environment like e-learning web sites in order to gain high performance and precise reasoning for new information from learner model.

## 3. Zebra: A User Modeling System for Triangular Learner Model

Existing user modeling systems (UMS) 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 I recognize that generic UMS (s) described in previous section [2](#_II.1._Existing_user) 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) are proved 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 (Fröschl, 2005, p. 27) is knowledge that learner achieved in certain subjects. Otherwise, domain independent information (Fröschl, 2005, pp. 29-31) includes personal traits not related to domain knowledge such as interests, learning styles, and demographic information. Each kind of information is in accordance with respective modeling method. For example, knowledge model can be created by overlay method, which is called overlay model (Mayo, 2001, pp. 56-58). So it is impossible to model all learners’ characteristics because of the reason “there is no modeling method fit all characteristics”. Moreover, most UMS (s) require effective inference techniques in their modeling tasks but this is impossible if we cannot recognize which individual characteristics are important.

To overcome these obstacles and difficulties, I 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 my 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.

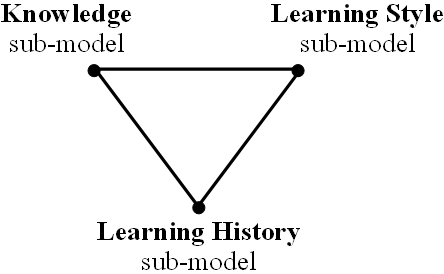
I also introduce the architecture of UMS which builds up TLM; it is named **Zebra**. The name “Zebra” implicates that my UMS will run fast and be powerful like African zebra (Wikipedia, Zebra, 2014).

This section [3](#_II.2._Zebra:_A) includes four sub-sections as follows:

* Sub-section [3.1](#_II.2.1._Triangular_Learner) describes TLM in detailed.
* Sub-section [3.2](#_II.2.2._The_architecture) describes architecture of Zebra.
* Sub-section [3.3](#_II.2.3._Interaction_between) tells us interaction between Zebra and adaptive learning systems like AHS and AEHS.
* Because Zebra is implemented as computer software, sub-section [3.4](#_II.2.4._Implementation_of) focuses on the implementation of Zebra.

### 3.1. Triangular Learner Model

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 [3.1.1](#_Figure_II.6._Triangular)). Hence TLM has three sub-models: *knowledge sub-mode*l, *learning style sub-model* and *learning history sub-model*.



**Figure 3.1.1.** Triangular Learner Model

I propose the method that combines overlay method and Bayesian network to build up **knowledge sub-model**. 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. Bayesian network (Neapolitan, 2003, p. 40) is the directed acyclic graph (DAG) in which nodes are linked together by arcs; each arc expresses the relationship between two node. The strength of relationship is quantified by Conditional Probability Table (CPT). The combination between overlay model and Bayesian network is done through following steps:

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

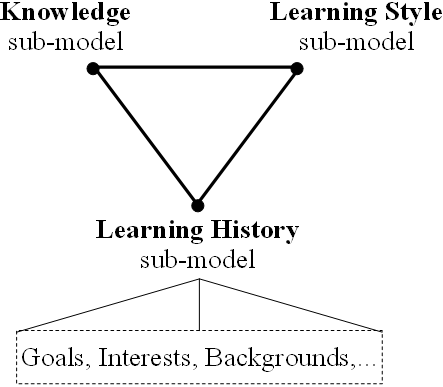
So knowledge sub-model is called as Bayesian overlay sub-model, which is described particularly in (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009). The proposed approach that combines overlay model and Bayesian network so as to build up Bayesian overlay sub-model is also described particularly in (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009).

**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” (Stash, 2007, p. 93). There are many models of learning styles in theory of psychology such as Dunn and Dunn (Dunn & Dunn, 1996), Witkin (Witkin, Moore, Goodenough, & Cox, 1977), Riding (Riding & Rayner, 1998), Myers-Briggs (Wikipedia, Myers-Briggs Type Indicator, 2014), Kolb (Kolb & Kolb, 2005), Honey-Mumford (Honey & Mumford, 2000), and Felder-Silverman (Felder & Silverman, 1988). I choose Honey-Mumford model and Felder-Silverman model as principal models and construct them by hidden Markov model. According to Honey-Mumford model and Felder-Silverman model, learning styles are classified into following dimensions:

* *Verbal/Visual*. Verbal students like learning materials in text form. Otherwise visual students prefer to images, pictures, video, etc.
* *Active/Reflective*. Active students understand information only if they discussed it and applied it. Reflective students think thoroughly about things before doing any practice.
* *Theorist/Pragmatist.* Theorists think things through in logical steps, understand different facts into coherent theory (Stash, 2007, p. 106). Pragmatists have practical mind, prefer to try and test techniques relevant to problems (Stash, 2007, p. 106).

For modeling learning style by using Hidden Markov Model (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 which learning styles 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 (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008). *All learners’ actions:* *learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates, etc. are logged in learning history 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, backgrounds, and 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 [3.1.2](#_Figure_II.7._extended) shows the extended TLM whose learning history sub-model is the root to which more learners’ characteristics such as interests, backgrounds, and goals to TLM are attached.

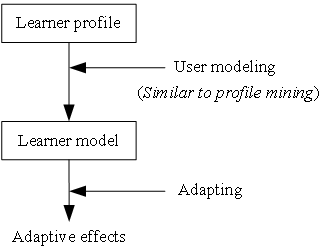


**Figure 3.1.2.** extended Triangular Learner Model

The successive sub-section [3.2](#_II.2.2._The_architecture) describes the architecture of Zebra.

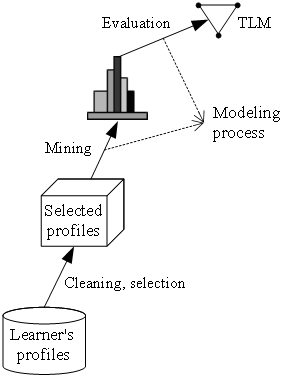
### 3.2. The architecture of Zebra

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. As aforementioned, the user modeling system that manipulates TLM is called Zebra. 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 [3.2.1](#_Figure_II.8._Modeling) indicates that user modeling (Brusilovsky, 1996, p. 2) is similar to profile mining.



**Figure 3.2.1.** Modeling task is similar to profile mining task

As aforementioned in (Nguyen & Do, Learner Model in Adaptive Learning, 2008), terms such as user model, student model and learner model have the same meaning because users are considered as learners or students in this research.



**Figure 3.2.2.** The modeling process in Zebra

As seen in figure [3.2.2](#_Figure_II.2.2.2._The) the goal of modeling process in Zebra is to mine users’ profiles but this process gets higher level than traditional data mining when it results out TML which is the most structured and valuable knowledge about users.

Moreover Zebra must implement the powerful inference mechanism to reason learners’ new assumptions (or characteristics) out TLM. I propose two methods: Bayesian network combined overlay model (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009)and hidden Markov model to infer learners’ knowledge and learning styles. Both Bayesian and Markov model are special cases of belief network. In general, belief network (Murphy, 1998) 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*.

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

* **Mining engine** (**ME**) 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 model user’s learning path by using sequential pattern mining. Mining engine has three other important functionalities that are *to* *discover some other characteristics* beyond knowledge and learning styles (such as interests, goals, learning context) and *to support learning concept recommendation* and *to support collaborative learning*. The last one which supports collaborative learning based on user group construction is the advanced functionality of Zebra.
* **Belief network engine** (**BNE**) is responsible for inferring new user information from TLM by using deduction mechanism available in belief network. New information mentions users’ new knowledge and learning styles. This engine applies both Bayesian network and hidden Markov model into its tasks. Two sub-models: knowledge and learning style are managed by this engine.

Zebra also provides *communication interfaces* (**CI**) that allow users and adaptive systems to see or modify restrictedly their TLM. Adaptive applications also interact with Zebra by these interfaces. CI (s) can be implemented as web services used widely on internet. According to World Wide Web Consortium ([http://www.w3.org](http://www.w3.org/)), a web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network (W3C, Web Services Architecture, 2004). It has an interface described in a machine-processable format such as WSDL (W3C, Web Services Description Language (WSDL) 1.1, 2001). Other systems interact with the web service in a manner prescribed by its description using SOAP messages (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000), typically conveyed using HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) with an XML (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) 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.

SOAP

SOAP

**BNE**

**ME**

*Learners*

*Observer*

*Adaptive applications*

**LS**

**K**

**LH**

SOAP

**Figure 3.2.3.** The architecture of Zebra

Figure [3.2.3](#_Figure_II.2.2.3._The) depicts the architecture of Zebra in which ME and BNE denote mining engine and belief network engine, respectively. K, LS and LH denote knowledge sub-model, learning styles sub-model and learning history sub-model, respectively. Finally, CI denotes communication interface.

Zebra is implemented as computer software available at internet link <http://www.locnguyen.net/st/dissertations/zebra>. The next sub-section [3.3](#_II.2.3._Interaction_between) mentions the interaction between Zebra and adaptive applications.

### 3.3. Interaction between Zebra and adaptive applications

Zebra aims to support adaptive learning applications; so in this sub-section [3.3](#_II.2.3._Interaction_between) we should glance over what adaptive applications are and discuss about how Zebra interacts with such applications. As aforementioned (Nguyen, State of the Art of Adaptive Learning, 2009), the most popular adaptive learning system supporting personalized learning environment is adaptive education hypermedia system (AEHS). Remind that the storage layer of AEHS has four models (Karampiperis & Sampson, 2005):

* *Media space* or *resource model* contains learning resources (lectures, tests, examples, exercises, etc.) and is associated with descriptive information (metadata).
* *Domain model* constitutes the structure of domain knowledge, which was often represented in form of graph. Knowledge items stored in domain model are concepts, subjects, topics, etc.
* *Adaptation model* is the centric component which gives effect to adaptation. It contains *concept selection rules* and *content selection rules*. Concept selection rules are used to choose appropriate concepts from domain model. On the other hand, we apply content selection rules into choosing suitable educational resources from media space. These rules must be in accordance with user model so that the selection gets correct.
* *User model* represents information and data about user.

Here, AEHS is regarded as an example for illustrating prominent traits of adaptive learning system. I propose the new approach in which the *user model* is removed from AEHS and becomes the TLM managed by the user modeling system Zebra. Now AEHS owns only three components: resource model, domain model and adaptation model. The reason is that learner model (TLM) becomes too complex to be maintained by AEHS and AEHS should only focus on improving adaptation process and the performance of system will get enhanced when AEHS takes full advantage of functionalities of Zebra. All operations relating TLM are executed by Zebra instead of AEHS. AEHS interacts with Zebra via communication interfaces (CI) according to SOAP protocol (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000). Figure [3.3.1](#_Figure_II.11._The) shows the new architecture of AEHS and the interaction between AEHS and Zebra. There are only three instances of CI but the number of them is not limited in practice. In the next sub-section [3.4](#_II.2.4._Implementation_of), the adaptive learning system [WOW](#_Figure_II.30._Typical) based on AHA! system (De Bra & Calvi, 1998) is introduced as a sample AEHS.

**BNE**

**ME**

**LS**

**K**

**LH**

Presenter

Learner

Observer

SOAP

*Concept selection rules*

SOAP

*Content selection rules*

SOAP

Domain model

Media space

**AEHS**

**Figure 3.3.1.** The new architecture of AEHS and the interaction between AEHS and Zebra

In figure [3.3.1](#_Figure_II.2.3.1._The), *observer* is external program that tracks learners’ actions and *presenter* is any application or web site that introduces or presents learning resources to learners. Given the adaptive learning system WOW mentioned in successive sub-section [3.4](#_II.2.4._Implementation_of), the sample presenter is e-learning web site of WOW shown in figure [3.4.18](#_Figure_II.30._Typical). When WOW is associated with Zebra, the observer is itself, WOW.

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 learner’s knowledge to concepts in domain model. In other words, these concepts are filtered to find ones that are necessary for learner to study in her/his course. Knowledge sub-model supports the concept selection process.
* In *content selection process*, learning resources are selected from resource model based on content selection rules that match learner’s learning styles to attributes of resources in resource space. In other words, this process finds the adaptive resources that are suitable to learner. It is most likely that learner prefers such adaptive resources. Learning style sub-model and learning history sub-model support the content selection process.

The adaptation process shown in figure [3.3.2](#_Figure_II.12._Steps) includes following steps:

1. *Step* 1: The projection of domain model onto knowledge sub-model by using concept selection rules results in a set of domain knowledge items called *A* that student has to learn. This is concept selection process.
2. *Step* 2: *A* is used as filter to choose a set of learning resources called *B* that relates to *A*.
3. *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, tests, etc.) called *C* tailoring to learner’s preferences. This is content selection process. *C* is considered as a set of recommendation resources.
4. *Step* 4: *C* is shown in content presenter. Presenter can be human-machine interfaces, web sites, learning management system (LMS), teaching support applications, etc. Please see (Wikipedia, Learning management system, 2014) for more details about LMS.
5. *Step* 5: Learner studies *C* by interacting with content presenter.
6. *Step* 6: Observer monitors learner in order to catch and deliver learner’s observations to Zebra. Zebra uses such observations to update TLM.

Knowledge sub-model

Domain model

Concept selection rules

Knowledge

need learned *A*

Learning resources

need learned *B*

Resource model

Content selection rules

Recommendation resources *C*

need learned

Presenter

Learners

**Zebra**

Observer

*Step* 1

*Step* 2

*Step* 3

*Step* 4

*Step* 5

*Step* 6

*Step* 6

Learning style sub-model

**Figure 3.3.2.** Steps in adaptation process with support of Zebra

Sub-section [3.4](#_II.2.4._Implementation_of) describes the implementation of Zebra because Zebra is implemented as computer software.

### 3.4. Implementation of Zebra

Zebra is implemented as a computer software working as a server. Zebra server is written by [Java language (https://www.oracle.com/java)](https://www.oracle.com/java) which is an object-oriented programming language supported by (Oracle). The version of Java language when Zebra is built up is [1.6 updated 3](http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html#jdk-6u3-oth-JPR). There is convention that the term “*Zebra server*” indicates the software server that implements Zebra. [Triangular Learner Model](#_II.2.1._Triangular_Learner) (TLM), [belief network engine](#_Figure_II.10._The) (BNE), and [mining engine](#_Figure_II.10._The) (ME) are implemented as services or *daemons* inside Zebra server. Daemon (Wikipedia, Daemon (computing), 2014) is defined as a computer program running implicitly in an operating system such as Windows, Unix, and Linux. There are 9 main daemons: knowledge static Bayesian network daemon, knowledge dynamic Bayesian network daemon, learning style daemon, learning history data daemon, learning concept recommendation daemon, learning path daemon, discovering user interests daemon (document classification + user searching), constructing user communities daemon, and mailing list daemon. Essentially, TLM and two engines (BNE and ME) are decomposed into such 9 daemons. Table [3.4.1](#_Table_II.2.4.1._Relationships) shows relationships of two engines (including BNE and ME), TLM (including knowledge sub-model, learning style sub-model, learning history sub-model), and 9 daemons.

|  |  |  |
| --- | --- | --- |
| **Engine** | **TLM** | **Daemon** |
| BNE | Knowledge sub-model | Knowledge static Bayesian network daemon |
| Knowledge dynamic Bayesian network daemon |
| Learning style sub-model | Learning style daemon |
| ME | Learning history sub-model | Learning history data daemon |
| Learning concept recommendation daemon |
| Learning path daemon |
| Discovering user interests daemon  (document classification + user searching) |
| Constructing user communities daemon |
|  | | Mailing list daemon |

###### **Table 3.4.1.** Relationships of two engines, TLM, and 9 daemons

According to table [3.4.1](#_Table_II.2.4.1._Relationships), two daemons: knowledge static Bayesian network daemon and knowledge dynamic Bayesian network daemon belong to knowledge sub-model which, in turn, is managed by BNE. Learning style sub-model belongs to learning style sub-model which, in turn, is managed by BNE. Five daemons: learning history data daemon, learning concept recommendation daemon, learning path daemon, discovering user interests daemon, and constructing user communities daemon belong to learning history sub-model which, in turn, is managed by ME. The mailing list daemon is utility daemon and so it does not belong to any sub-model. Table [3.4.2](#_Table_II.2.4.2._Responsibilities) summarizes responsibilities of such 9 daemons. Of course, these responsibilities also belong to respective TLM sub-models.

|  |  |  |
| --- | --- | --- |
| *N*0 | *Daemon* | *Responsibility* |
| 1 | Knowledge static Bayesian network daemon | * Building up Bayesian model by combination of overlay model and Bayesian network; please see (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) for more details. * Evaluating user knowledge (see figure [3.4.11](#_Figure_II.19._Evaluation)), based on inference mechanism inside Bayesian network. |
| 2 | Knowledge dynamic Bayesian network daemon | Building dynamic Bayesian network to model user knowledge in chronological process. |
| 3 | Learning style daemon | * Applying hidden Markov model into building up learning style sub-model. * Inferring user’s learning styles from observations of her/his study. |
| 4 | Learning history data daemon | This daemon inherits all excellent aspects of log file management function of the generic adaptive system AHA! (De Bra & Calvi, 1998) in order to organize, store, and manage learning history data (learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates, etc.) as structured data like XML files and relation database. Please see (Nguyen, State of the Art of Adaptive Learning, 2009) for more details about AHA! system. |
| 5 | Learning concept recommendation daemon | This daemon mines learning history data to extract sequential patterns. After that, it breaks such patterns into concepts/learning materials which are recommended to users. This responsibility is the first extended function of learning history sub-model. Please see (Nguyen & Do, Learning Concept Recommendation based on Sequential Pattern Mining, 2009) for more details about learning concept recommendation based on mining learning history. |
| 6 | Learning path daemon | This daemon take advantage of sequential patterns resulted from learning concept recommendation daemon in order to recommend user optimal learning path (learning route) so that she/he is successful in her/his studying process. |
| 7 | Discovering user interests daemon  (document classification + user searching) | * This daemon discovers user interest based on document classification. The basic idea is to consider user interests as classes of documents. The process of classifying documents is also the process of discovering user interests. This responsibility is the second extended function of learning history sub-model. Please see (Nguyen, A Proposal Discovering User Interests by Support Vector Machine and Decision Tree on Document Classification, 2009) for more details about discovering user interests by document classification. * This daemon also supports users to search documents relevant to their course. |
| 8 | Constructing user communities daemon | This daemon is responsible for constructing user groups or user communities, which supports group adaptation and collaborative learning. This responsibility is the third extended function of learning history sub-model. |
| 9 | Mailing list daemon | This daemon is responsible for managing mailing list of users and sending e-mails to users (see figure [3.4.15](#_Figure_II.27._Mailing)). |

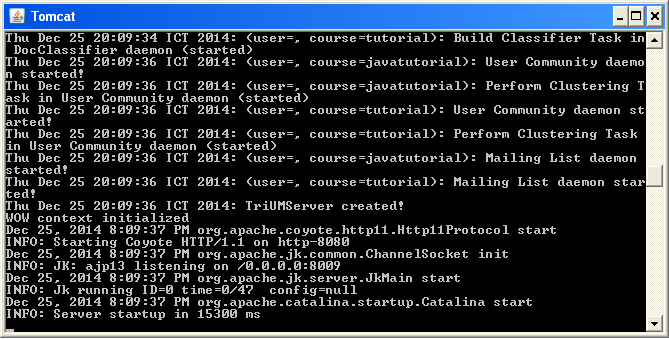
###### **Table 3.4.2.** Responsibilities of 9 daemons

These responsibilities will be mentioned below. Zebra server, TLM, BNE, and ME have synonyms: TriUMServer, TriUM, ZebraNetworker, and ZebraMiner, respectively when you look up source code of Zebra implementation. Communication interface (CI) is implemented as so-called *Delegator* which dispatches users/applications requests to Zebra server and sends back server responses to users/applications.

Besides implementing Zebra architecture and TLM, Zebra server provides many utility tools for managing learners and supporting adaptive learning. Zebra server is available at internet link as follows:

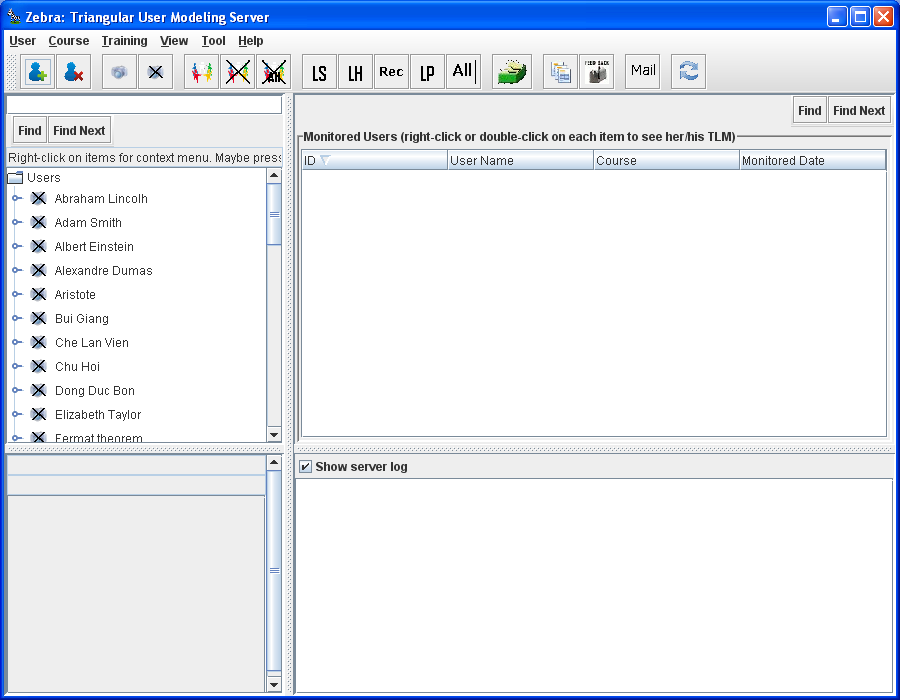
<http://www.locnguyen.net/st/dissertations/zebra>

When you startup Zebra server, it runs as underlying service as shown in figure [3.4.1](#_Figure_II.13._Zebra). Note that Zebra server runs inside Apache Tomcat web server (Apache, 1999).



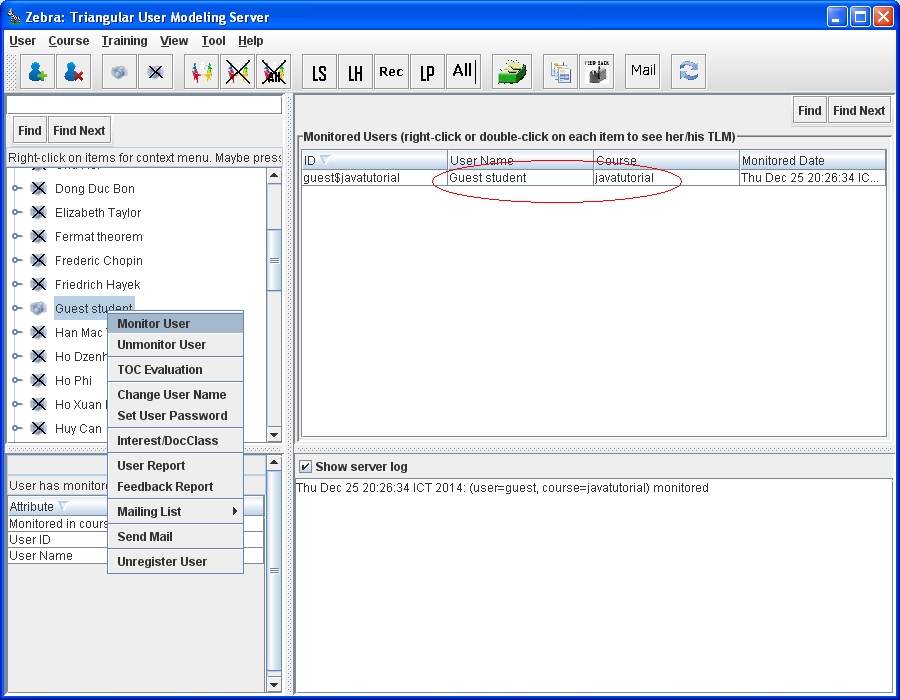
**Figure 3.4.1.** Zebra is running

Zebra server is often manipulated via *Zebra control panel* shown in figure [3.4.2](#_Figure_II.14._Zebra).



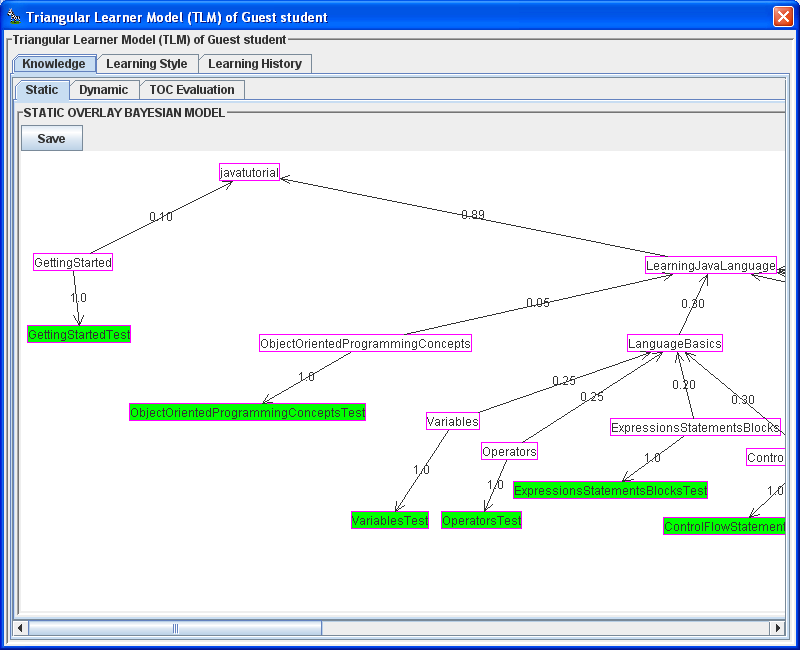
**Figure 3.4.2.** Zebra control panel

The main purpose of Zebra server is to manage learners. So, you can choose a learner appearing in the left column of control panel and monitor her/him. For example, learner “*Guest student*” is monitored in the “Java course” as shown in figure [3.4.3](#_Figure_II.15._Monitoring). Java course is designed and written as “Java tutorial” (Oracle, The Java™ Tutorials, 2009) which is web-based learning course teaching Java programming language to learners. It contains knowledge items, concepts, HTML lectures and online tests. Please read (W3Schools, 1999) and (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010) for more details about HTML and XHTML. The Java tutorial is available at [http://docs.oracle.com/javase/tutorial](http://docs.oracle.com/javase/tutorial/) (Oracle, The Java™ Tutorials, 2009). There are many courses like “Java course” built in Zebra server. Teachers and experts can create their own courses. Each learner is monitored according to the course she/he studies because her/his knowledge model is constructed by combination of overlay model and Bayesian network; please see section (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) for more details about knowledge sub-model. It is conventional that default user monitored by Zebra server is Guest student and users are learners in learning context. Guest student has identification “*guest*” and her/his TLM inside Java course is named “*guest$javatutoral*” when the identification of Java course (Java tutorial) is “*javatutoral*”. If there are many courses, each learner owns many TLM (s) too. For example, if Zebra server manages two courses such as Java course identified by “javatutorial” and Oracle course identified by “oracletutorial”, learner John indentified by “john” will have two TLM (s) such as “john$javatutorial” and “john$oracletutorial”. It is concluded that each TLM always associates with a course.



**Figure 3.4.3.** Monitoring a learner

The Guest student is appeared in the center of the control panel when she/he is monitored. If you double-click on the “Guest student”, her/his TLM managed by two engines BNE and ME is opened as shown in figure [3.4.4](#_Figure_II.16._TLM).

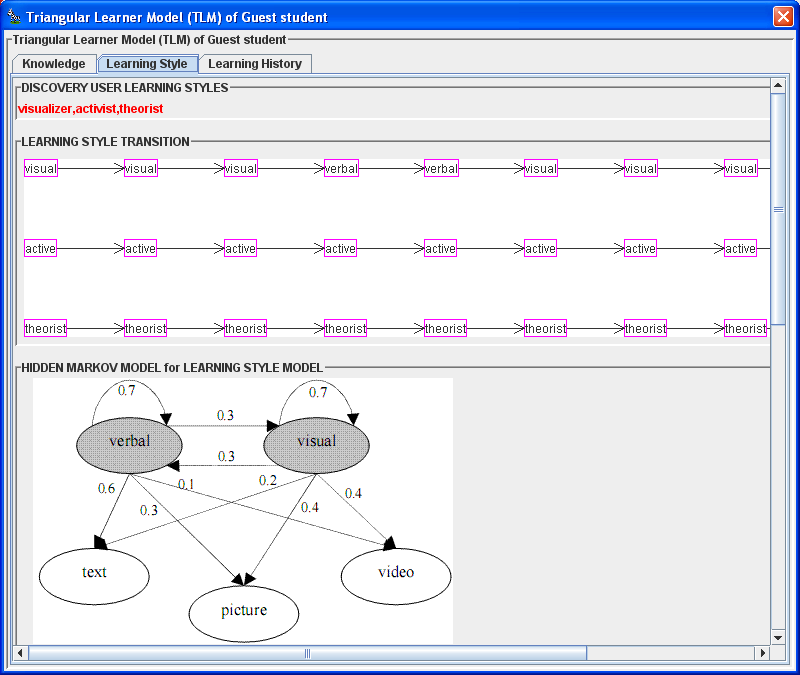


**Figure 3.4.4.** TLM of given learner

There are three tabs “Knowledge”, “Learning Style”, and “Learning History” shown in figure [3.4.4](#_Figure_II.16._TLM), visualizing three sub-models of TLM such as knowledge sub-model, learning style sub-model, and learning history sub-model. Knowledge sub-model is constructed by Bayesian overlay model which is combination of overlay model and Bayesian network. Software [JavaBayes (http://www.cs.cmu.edu/~javabayes/)](http://www.cs.cmu.edu/~javabayes/), version 0.346, year accessed: 2007, developed by author [Fabio Gagliardi Cozman](http://www.cs.cmu.edu/~fgcozman/home.html) – Escola Politécnica – University of São Paulo, is used to build up Bayesian network. I express my deep gratitude to the author Fabio Gagliardi Cozman for providing the great Bayesian network software.

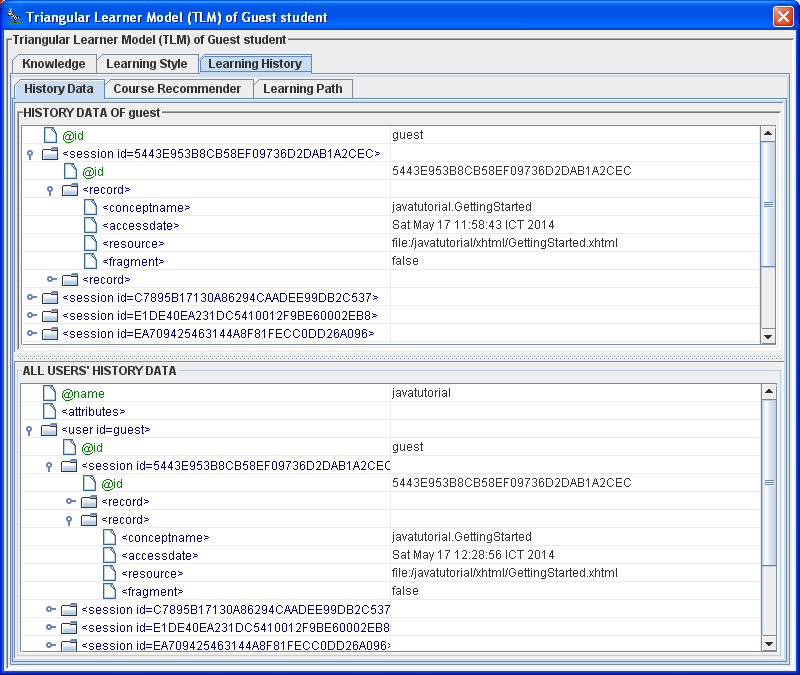
Figure [3.4.4](#_Figure_II.16._TLM) expresses static Bayesian overlay model in which nodes represent concepts, knowledge items and evidences. Green-shading nodes represent evidences like tests and exercises; please see (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) for more details about Bayesian overlay model. Software [JGraph (https://www.jgraph.com/)](https://www.jgraph.com/) developed by [JGraph Ltd Company](https://www.jgraph.com/about-jgraph-company.html), year accessed: 2008 is used to draw Bayesian network as seen in figure [3.4.4](#_Figure_II.16._TLM). I express my deep gratitude to the JGraph Ltd Company for providing the great graph software.

Learning style sub-model is constructed by hidden Markov model shown in figure [3.4.5](#_Figure_II.17._Learning). Hidden Markov model inside learning style sub-model is developed by authors [Kanav Kahol and Troy L. McDaniel](http://read.pudn.com/downloads85/sourcecode/math/328917/toneRec/HMM/HMM.java__.htm), year accessed: 2008. I express my deep gratitude to the authors Kanav Kahol and Troy L. McDaniel for providing their great software. BNE is responsible for manipulating knowledge sub-model and learning style sub-model. BNE uses software [Java Scientific Library (http://www.ee.ucl.ac.uk/~mflanaga/java/)](http://www.ee.ucl.ac.uk/~mflanaga/java/) developed by author [Michael Thomas Flanagan](http://www.ee.ucl.ac.uk/~mflanaga/), year accessed: 2008 for some mathematical tasks. I express my deep gratitude to the author Michael Thomas Flanagan for providing his great software.



**Figure 3.4.5.** Learning style sub-model constructed by hidden Markov model

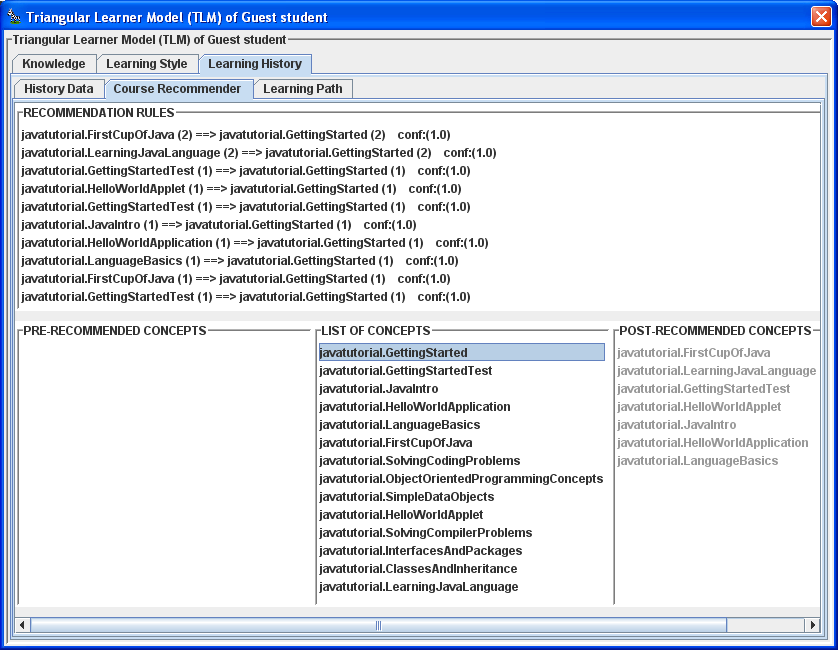
Recall that learning history sub-model, the most important sub-model among three sub-models of TLM, structures and stores coarse information like learning history data in XML files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) or relation database (Ramakrishnan & Gehrke, 2003, pp. 25-94). Figure [3.4.6](#_Figure_II.18._Learning) shows a piece of learning history data that learning history sub-model stores and manages. ME is responsible for manipulating learning history sub-model.



**Figure 3.4.6.** Learning history sub-model stores coarse information in XML files

ME uses open source software [Weka (http:/www.cs.waikato.ac.nz/ml/weka/)](http://www.cs.waikato.ac.nz/ml/weka/) developed by [Machine Learning Group](http://www.cs.waikato.ac.nz/ml/) at the [University of Waikato](http://www.cs.waikato.ac.nz/) for mining tasks such as mining recommendation rules, mining sequential patterns, and clustering user groups, year accessed: 2008. I express my deep gratitude to the Machine Learning Group for providing their great software.

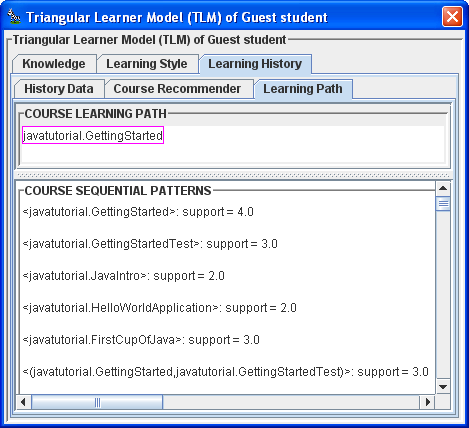
The first extended function of learning history sub-model is learning concept and learning path recommendation based on sequential pattern mining (Nguyen & Do, Learning Concept Recommendation based on Sequential Pattern Mining, 2009), which is illustrated in figures [3.4.7](#_Figure_II.19._Learning) and [3.4.8](#_Figure_II.20._Learning).



**Figure 3.4.7.** Learning concept recommendation

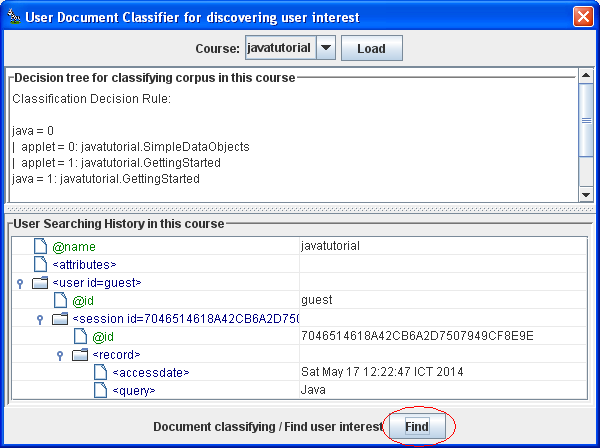
The upper part of figure [3.4.7](#_Figure_II.19._Learning) lists recommendation rules or sequential rules which are extracted from sequential patterns. The lower part of figure [3.4.7](#_Figure_II.19._Learning) shows recommended concepts derived from recommendation rules. For example, given concept “javatutorial.GettingStarted”, learners are recommended to study successive concepts (post-recommended concepts): “javatutorial.FirstCupOfJava”, “javatutorial.LearningJavaLanguage”, etc. These recommended concepts are shown on e-learning website of the adaptive learning system WOW (see figure [3.4.18](#_Figure_II.30._Typical)).

The lower part of figure [3.4.8](#_Figure_II.20._Learning) lists sequential patterns mined from learning history data. The upper part of figure [3.4.8](#_Figure_II.20._Learning) shows course learning paths (learning routes); for example, learning path “javatutorial.GettingStarted” rather than learning path “javatutorial.GettingStarted→javatutorial.GettingStartedTest” is provided to learners because the support of sequential pattern <javatutorial.GettingStarted> (4.0) is higher than the support of sequential pattern <javatutorial.GettingStarted→javatutorial.GettingStartedTest> (3.0). This learning path is recommended on e-learning website of the adaptive learning system WOW (see figure [3.4.18](#_Figure_II.30._Typical)). Please see (Nguyen & Do, Learning Concept Recommendation based on Sequential Pattern Mining, 2009) for more details about sequential pattern mining and learning concept recommendation.



**Figure 3.4.8.** Learning path recommendation

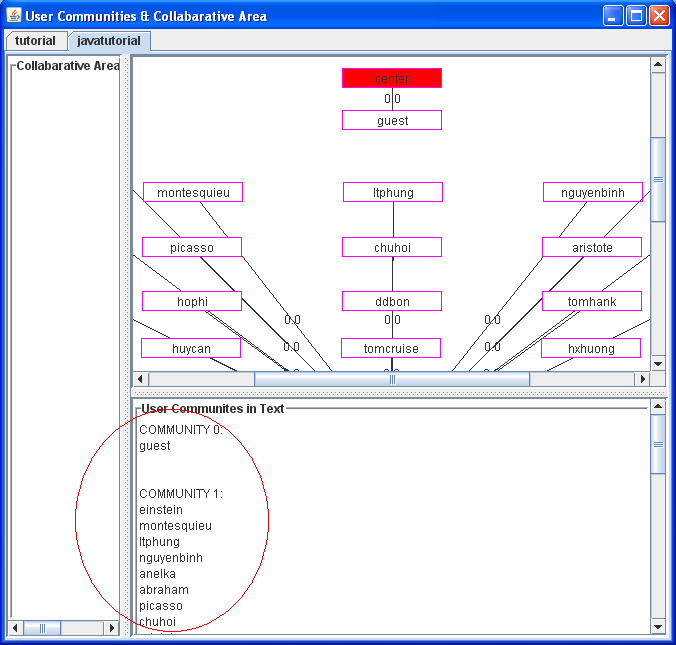
The second extended function of learning history sub-model is to discover user interests based on document classification. Such function can be performed via Zebra control panel, as shown in figure [3.4.9](#_Figure_II.21._Discovering).



**Figure 3.4.9.** Discovering user interests based on document classification

The lower part of figure [3.4.9](#_Figure_II.21._Discovering) displays learners’ searching history and the upper part lists classification rules constructed from such searching history. When button “Find” is clicked, classification rules are applied into discovering interests of a learner.

Figure [3.4.10](#_Figure_II.22._Constructing) shows the third extended function of learning history sub-model which constructs user groups or user communities.

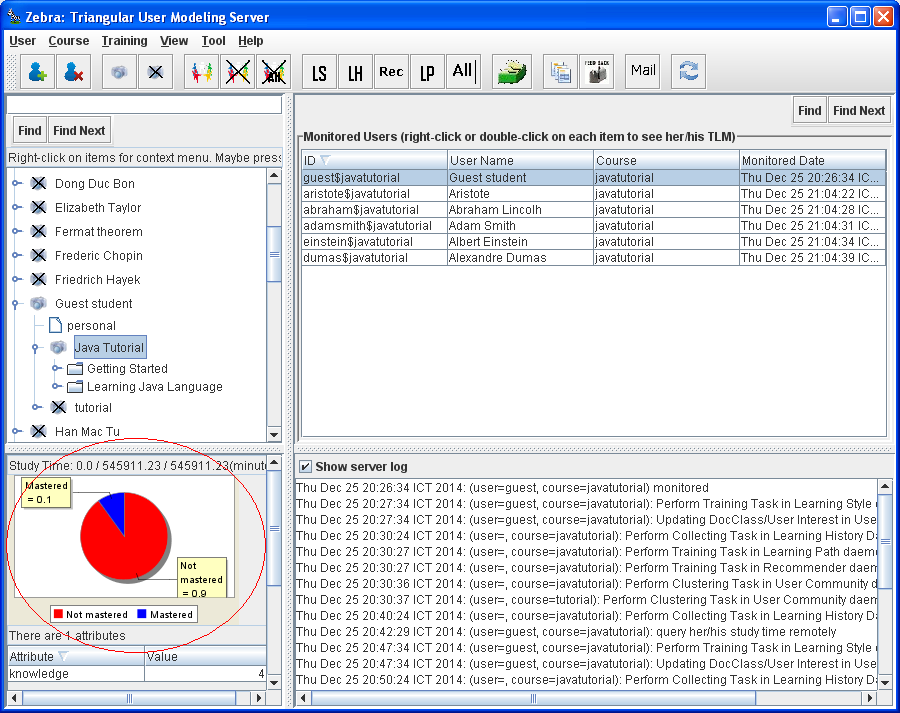


**Figure 3.4.10.** Constructing user groups

As seen figure [3.4.10](#_Figure_II.22._Constructing), there are two communities 0 and 1. Community 0 has only user “Guest student”. These communities provide collaborative area for the adaptive learning system WOW (see figure [3.4.25](#_Figure_II.37._Collaborative)).

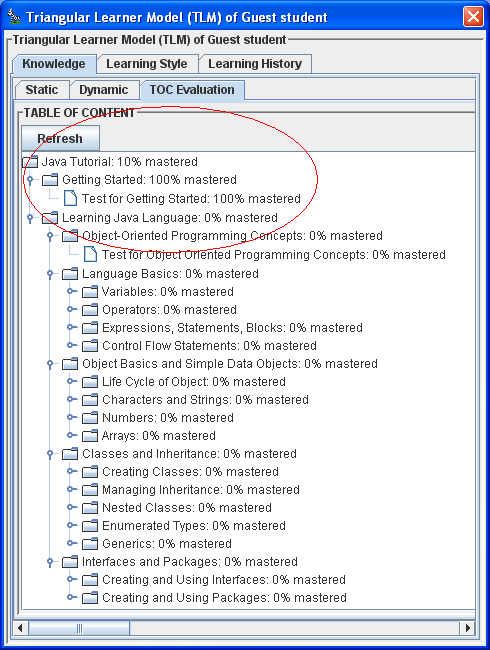
Recall that there are three tabs “Knowledge”, “Learning Style”, and “Learning History” shown in figure [3.4.4](#_Figure_II.16._TLM), visualizing three sub-models of TLM such as knowledge sub-model, learning style sub-model, and learning history sub-model. You can explore these tabs in order to comprehend TLM.

According to knowledge sub-model, it is easy to evaluate how much knowledge the Guest student gains. Figure [3.4.11](#_Figure_II.19._Evaluation) expresses evaluation of knowledge of Guest student. According to pie chart in figure [3.4.11](#_Figure_II.19._Evaluation), blue area represents the percentage of mastery over given concept or course. It is easy to recognize that the Guest student masters the Java course about 10% (the indicator *mastered*=0.1). Software [JFreeChart (http://www.jfree.org/jfreechart/)](http://www.jfree.org/jfreechart/) developed by [Object Refinery Limited Company](http://www.jfree.org/), year accessed: 2008 is used to draw evaluation chart as seen in figure [3.4.11](#_Figure_II.19._Evaluation). I express my deep gratitude to the Object Refinery Limited Company for providing the great chart software.



**Figure 3.4.11.** Evaluation of learner’s knowledge

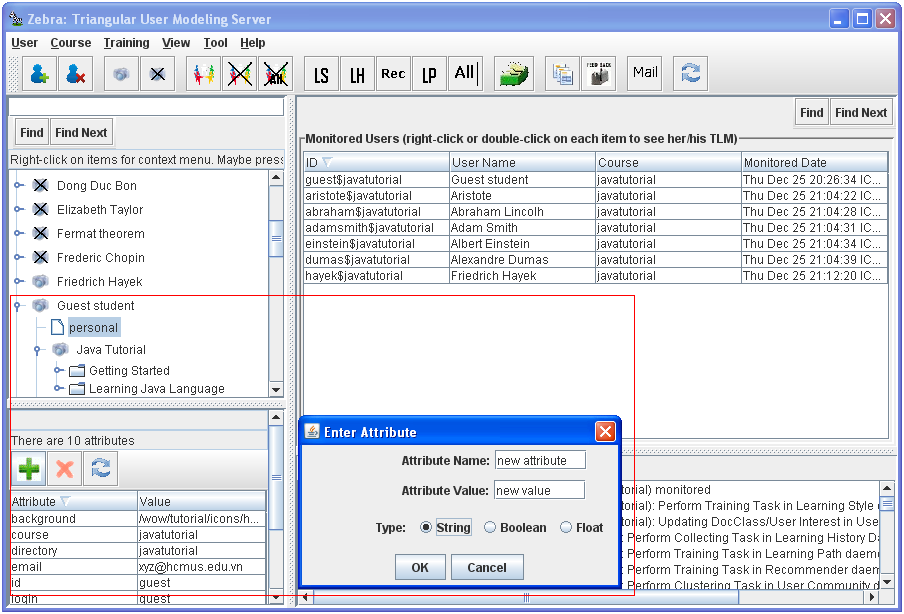
Learner’s knowledge can be evaluated in detailed as shown in figure [3.4.12](#_Figure_II.24._Evaluation).



**Figure 3.4.12.** Evaluation of learner’s knowledge in detailed

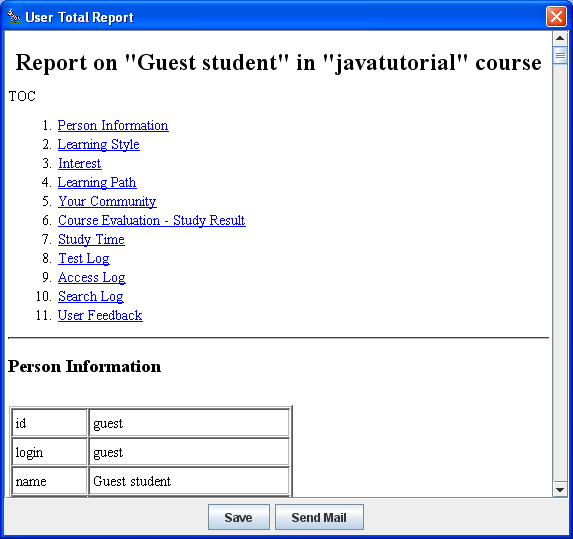
Knowledge evaluation is also a main aspect of the adaptive learning system WOW (see figure [3.4.21](#_Figure_II.33._Knowledge)). In fact, WOW retrieves evaluation information from Zebra server.

Personal information can be added, deleted, and updated via the control panel, as shown in figure [3.4.13](#_Figure_II.20._Updating).



**Figure 3.4.13.** Updating personal information

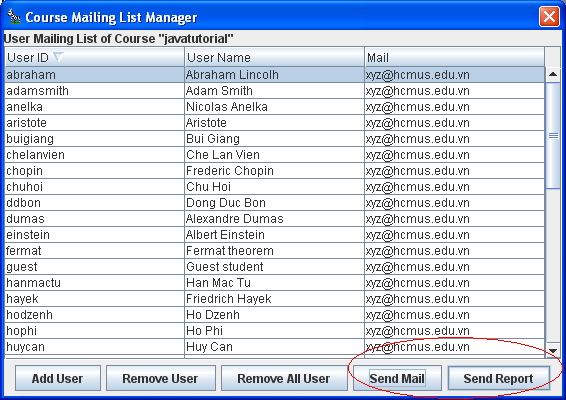
Learner’s studying process and studying results are structured in so-called *learning report* shown in figure [3.4.14](#_Figure_II.26._Learner).



**Figure 3.4.14.** Learning report

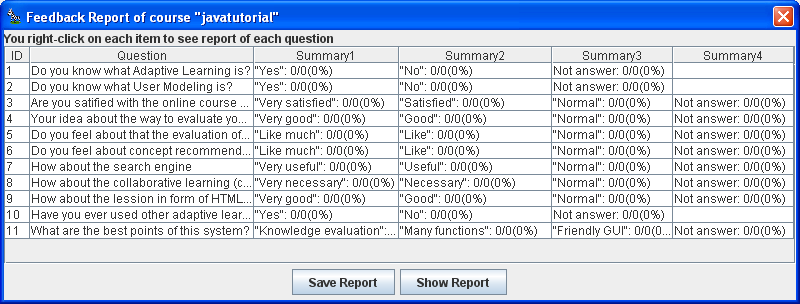
As seen in figure [3.4.14](#_Figure_II.26._Learning), learning report contains 11 items such as Personal Information, Learning Style, Interest, Learning Path, Your Community, Course Evaluation – Study Result, Study Time, Test Log, Access Log, Search Log, and User Feedback, which describes comprehensively all information about learner.

Zebra server provides a so-called mailing list tool that allows us to send learning report and mails to users. You can also add/remove users to/from mailing list via mailing list tool. Figure [3.4.15](#_Figure_II.27._Mailing) shows the mailing list tool.



**Figure 3.4.15.** Mailing list tool

Zebra server also provides a useful utility tool that collects users’ feedbacks so as to evaluate itself and adaptive learning system. Concretely, Zebra server issues a questionnaire consisting of a number of questions. These questions help Zebra server to survey users’ opinions and feelings about Zebra. Users answer these questions as online feedbacks. Zebra server collects such feedbacks and analyzes them so as to evaluate itself and adaptive learning system according to pre-defined measures. Figure [3.4.16](#_Figure_II.28._Feedback) gives an example of feedback report after Zebra server collects such feedbacks.



**Figure 3.4.16.** Feedback report

Now main aspects of the user modeling system Zebra were described. It is necessary to describe the adaptive learning system which interacts with Zebra according to figures [3.3.1](#_Figure_II.11._The) and [3.3.2](#_Figure_II.12._Steps). In other words, the adaptive learning system takes advantages of users’ information provided by Zebra. Please see (Nguyen, State of the Art of Adaptive Learning, 2009) for more details about adaptive learning systems.

The adaptive learning system interacting with Zebra server is WOW which is the modified version of AHA! system. The AHA! system is developed by the author (De Bra & Calvi, 1998) and so, the WOW system shows my deep gratitude to the author [Paul De Bra](http://wwwis.win.tue.nl/~debra/). Moreover, I also thank [Eindhoven University of Technology](http://www.tue.nl/) where the AHA! system was developed. The AHA! system is available at internet link <http://aha.win.tue.nl/>. The AHA! 3.0 tutorial is available at <http://aha.win.tue.nl:18080/aha/tutorial/>.

Both AHA! and WOW are adaptive educational hypermedia system (AEHS); please see (Nguyen, State of the Art of Adaptive Learning, 2009) for more details about AEHS. Recall that WOW interacts with Zebra via communication interfaces (CI). CI supports many protocols such as HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) and SOAP (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000). WOW packed with Zebra server is available at internet link as follows:

<http://www.locnguyen.net/st/dissertations/zebra>

The main aspect of WOW is to support evaluation of user knowledge based on Bayesian overlay model. In (Nguyen, Incorporating Bayesian Inference into Adaptation Rules in AHA architecture, 2009), that dummy attribute is added into each knowledge item allows WOW to execute adaptation rule following inference mechanism inside Bayesian overlay model built in Zebra. The corporation between WOW and Bayesian inference is strong point of WOW. After running Zebra server as shown in figure [3.4.1](#_Figure_II.13._Zebra), you execute WOW by opening a browser pointing to WOW location, for example, <http://localhost:8080/wow/javatutorial> because WOW is web application written by Java Servlet (Wikipedia, Java Servlet, 2014). Figure [3.4.17](#_Figure_II.29._WOW) shows WOW login web page (Wikipedia, Web page, 2014) in which Guest student studies.



**Figure 3.4.17.** WOW login web page

Figure [3.4.18](#_Figure_II.30._Typical) is a typical adaptive learning web page that WOW gives to learners. Left column of the web page shows three important parts supporting adaptive navigation (Brusilovsky, Adaptive Hypermedia, 2001) as follows:

* *Links to online HTML lectures* (*web pages*) *associated with concepts* that learners should master. Please read (W3Schools, 1999) and (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010) for more details about HTML and XHTML.
* *Knowledge evaluation for current concept* according to Bayesian inference. The online HTML lectures are adaptive to the knowledge evaluation, for example, if learner does not enough knowledge to study advanced concepts, such advanced concepts are shown as gray or disabled items. As seen in figure [3.4.18](#_Figure_II.30._Typical), because Guest student mastered absolutely the concept “Getting Started” (mastered = 1), it is sufficient for her/him to study next concept “Learning Java language” and so, the concept “Learning Java language” is enabled.
* *Learning path and learning concept recommendation*. When Guest student mastered absolutely the concept “Getting Started”, she/he is recommended to study concepts: “First Cup of Java”, “Learning Java Language”, etc.

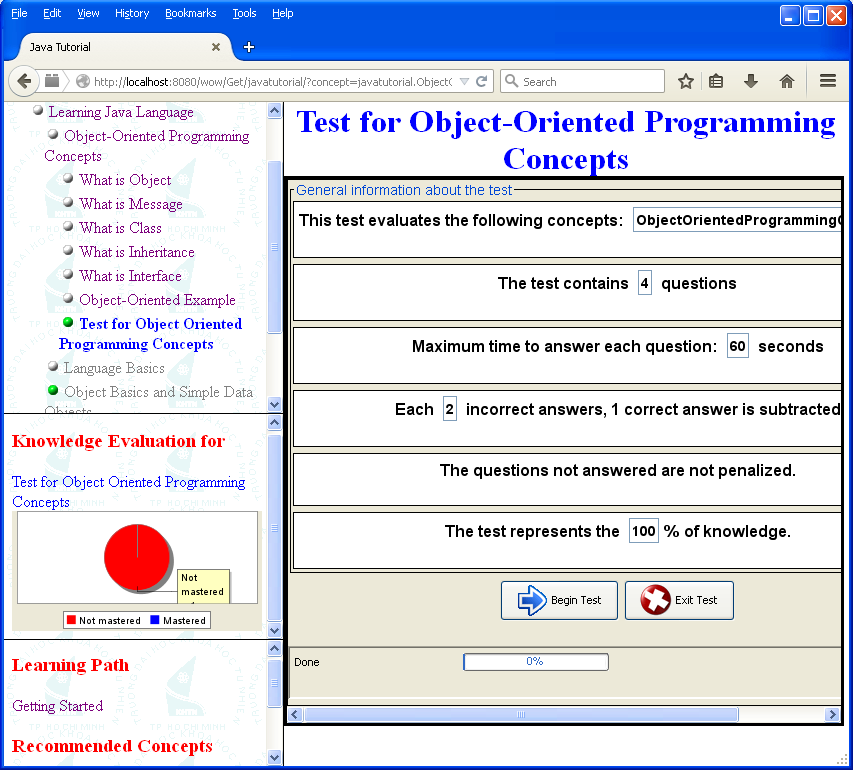
Right column of the web page shows two parts:

* The upper part lists utility tools such as “TOC”, “Glossary”, “Search”, “Feedback”, “User Query”, and “Collaborative” which help learners to take full advantage of WOW.
* The lower part isthe current online HTML lecture (web page) associated with the concept that learners study. This part supports adaptive presentation. Please see (Brusilovsky, Adaptive Hypermedia, 2001) for more details about adaptive presentation and adaptive navigation.



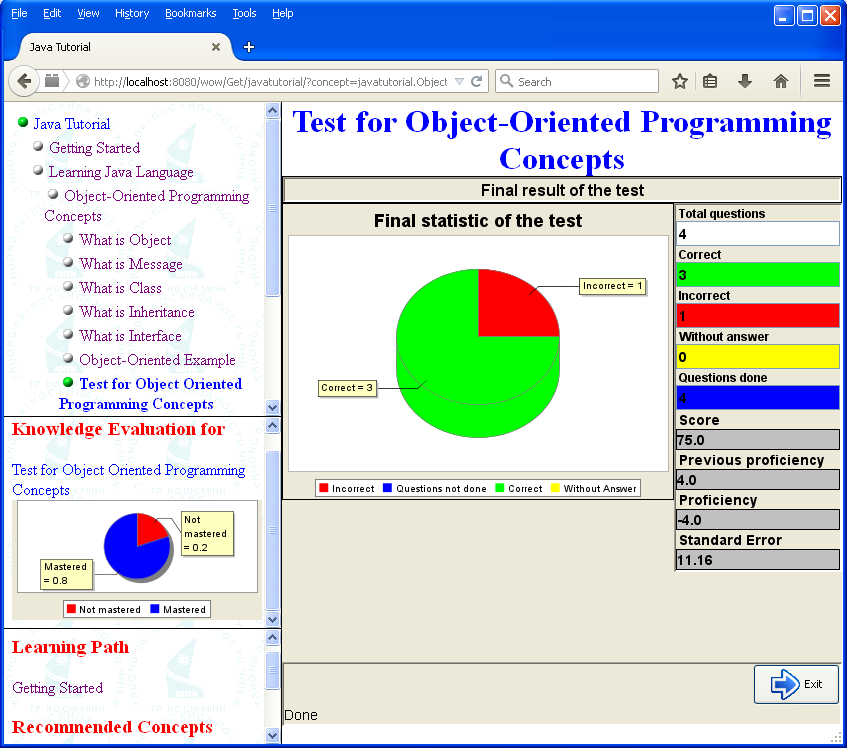
**Figure 3.4.18.** Typical adaptive learning web page supported by WOW

As seen in figure [3.4.18](#_Figure_II.30._Typical), the knowledge evaluation value represented by indicator “mastered” is 1 with regard to concept “Getting Started”, which means that Guest student comprehends concept “Getting Started”. Essentially, indicator “mastered” is posterior probability of concept “Getting Started” inside the Bayesian overlay model. The relationship between knowledge evaluation and inference mechanism inside Bayesian overlay model implies the corporation between WOW and Zebra server. Suppose Guest student does the online test “Test for Object Oriented Programming Concepts” after she/he studies concept “Object Oriented Programming Concepts” under the concept “Learning Java Language”. Figure [3.4.19](#_Figure_II.31._Leaner) shows that Guest student does such online test with note that the indicator “mastered” is 0. It means that Guest student does not master the concept “Object Oriented Programming Concepts” yet. The online test module inside AHA! is developed by Universidad de Córdoba (<http://www.uco.es>), Spain, 2008. I express my deep gratitude to the Universidad de Córdoba for providing the great online test module.



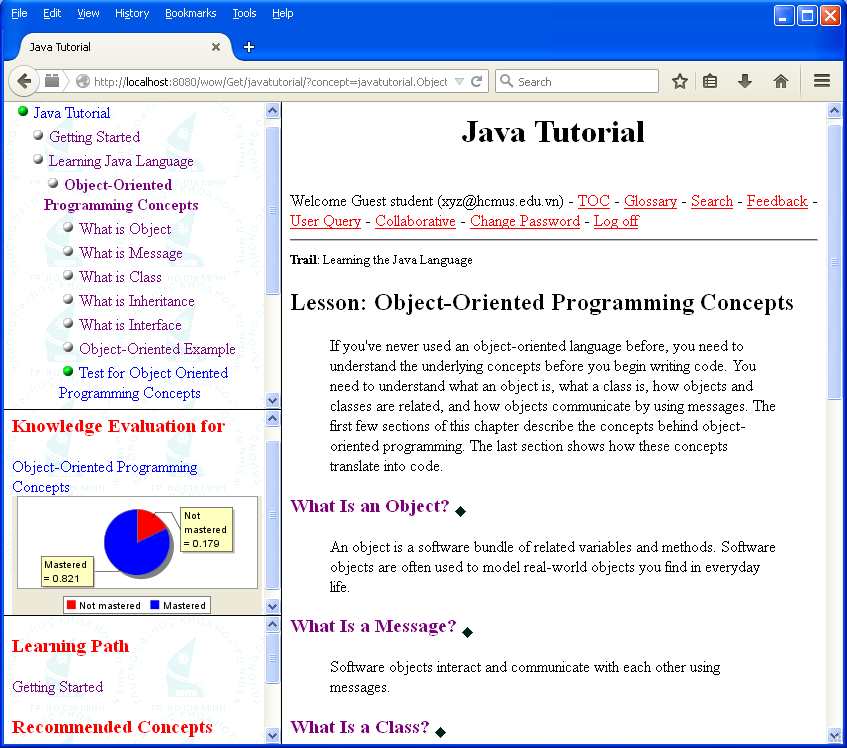
**Figure 3.4.19.** Leaner does online test via WOW

After doing the online test, Guest student’s knowledge is enhanced and the indicator “mastered” for the test “Test for Object Oriented Programming Concepts” is 0.8, as shown in figure [3.4.20](#_Figure_II.32._Result).



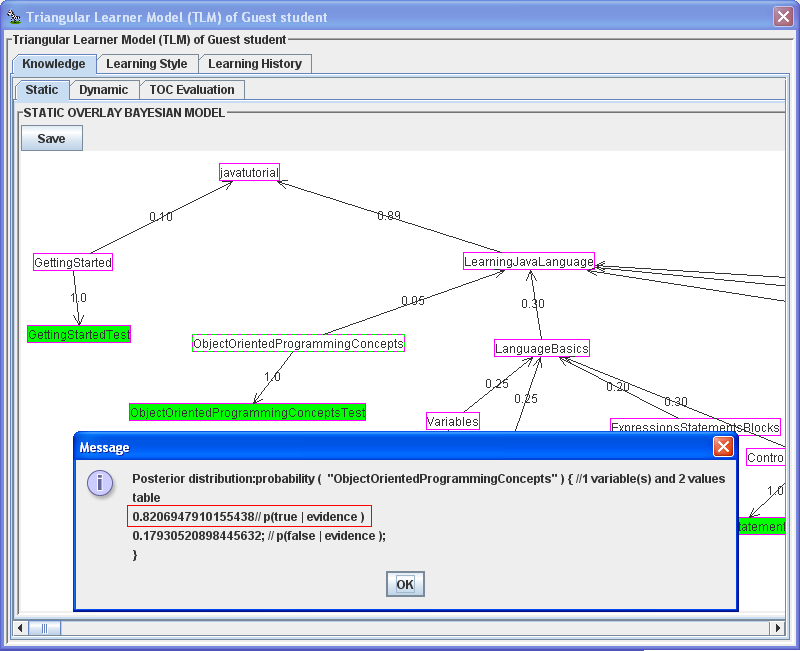
**Figure 3.4.20.** Result of online test

The indicator “mastered” for the concept “Object Oriented Programming Concepts” becomes 0.821. It means that Guest student masters the concept “Object Oriented Programming Concepts” about 82.1% after she/he finishes successfully the test “Test for Object Oriented Programming Concepts”. Therefore, figure [3.4.21](#_Figure_II.33._Knowledge) shows such her/his improvement of knowledge.



**Figure 3.4.21.** Knowledge evaluation is increased after learner finishes test successfully

There is a question “How to determine that Guest student achieves 82.1% knowledge of concept “Object Oriented Programming Concepts”; in other words, how to determined indicator “mastered” = 1”. The answer of this question implicates the corporation between WOW and Zebra server. Shortly, the test “Test for Object Oriented Programming Concepts” is evidence inside Bayesian overlay model. Based on the test result (see figure [3.4.20](#_Figure_II.32._Result)), inference mechanism inside Bayesian overlay model is executed and the posterior probability represented by indicator “mastered” gets 0.821. Figure [3.4.22](#_Figure_II.34._Posterior) expresses the posterior probability of concept “Object Oriented Programming Concepts” and it is 0.8206947910.821.

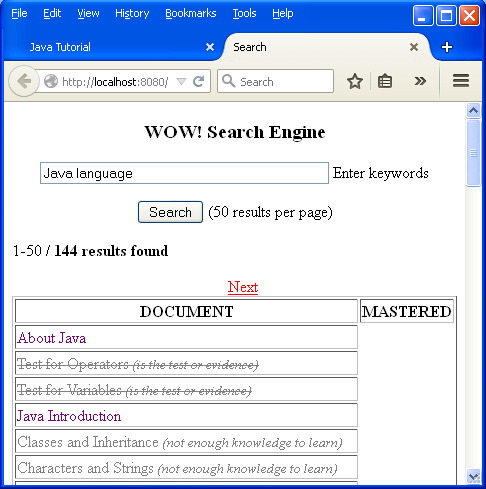


**Figure 3.4.22.** Posterior probability of given concept

Note that Bayesian overlay model is built in Zebra server and so, it is easy to recognize that figure [3.4.22](#_Figure_II.34._Posterior) shows a partial function of Zebra control panel. Therefore, it is possible to conclude that there is a corporation between Zebra server and WOW. Note that figures [3.4.4](#_Figure_II.16._TLM) and [3.4.22](#_Figure_II.34._Posterior) show the same function of Zebra server which manipulates the TLM.

Additionally, WOW provides a search engine that allows learners to search information relevant to Java course. Figure [3.4.23](#_Figure_II.35._Search) shows this search engine.

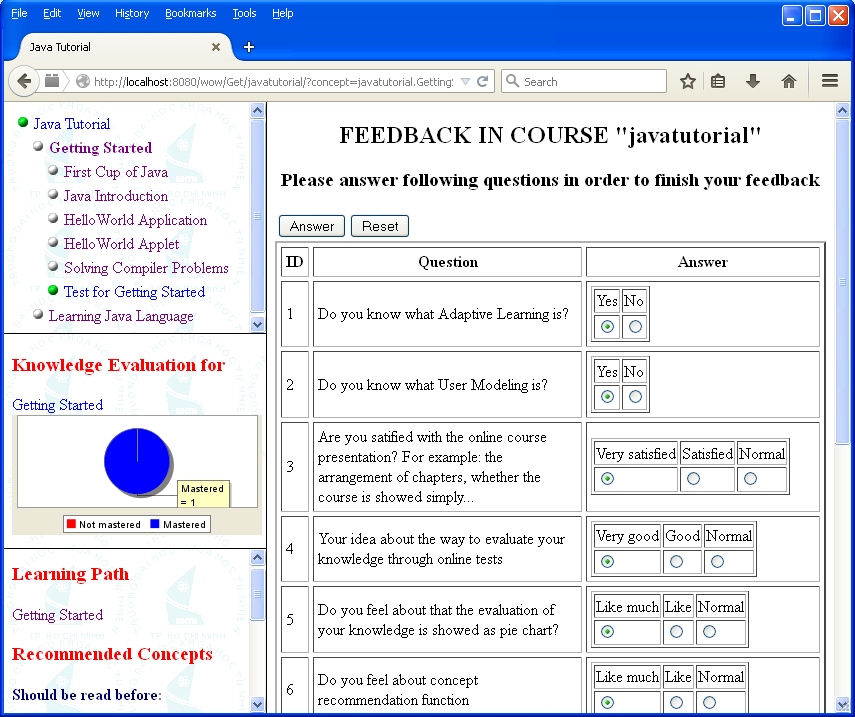
WOW uses software [Lucene Core (https:/lucene.apache.org/)](https://lucene.apache.org/) developed by LuceneTM/SolrTM Committers (<https://lucene.apache.org/whoweare.html>) at [The Apache Software Foundation](http://apache.org/) as default search engine, year accessed: 2008. I express my deep gratitude to the LuceneTM/SolrTM Committers for providing their great search engine.



**Figure 3.4.23.** Search engine in WOW

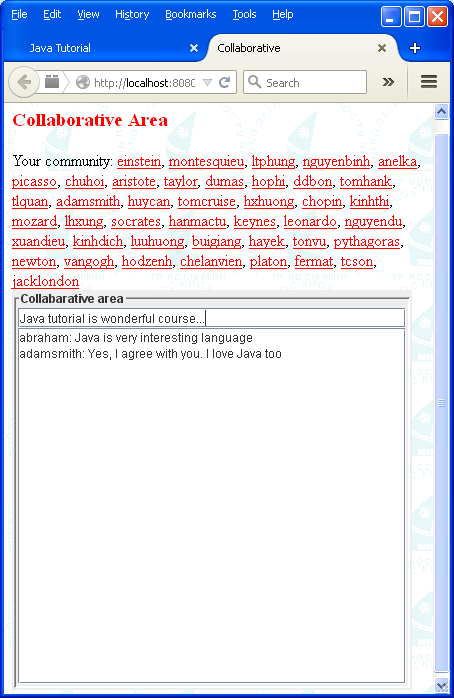
Based on aforementioned knowledge evaluation (see figure [3.4.22](#_Figure_II.34._Posterior)), the search engine also supports adaptive recommendation when concepts resulted from a search task will be disabled or stroke out if learner has not enough knowledge to study these concepts. As seen in figure [3.4.23](#_Figure_II.35._Search), concepts that learner has not enough knowledge to study are “Classes and Inheritance” and “Characters and Strings”.

Recall that Zebra server collects users’ feedbacks in order to evaluate itself and WOW (see figure [3.4.16](#_Figure_II.28._Feedback)). Of course, users must send feedbacks to Zebra. WOW provides the utility tool that help learners to send their feedbacks to Zebra server. Figure [3.4.24](#_Figure_II.36._WOW) shows the WOW feedback tool.



**Figure 3.4.24.** WOW feedback tool

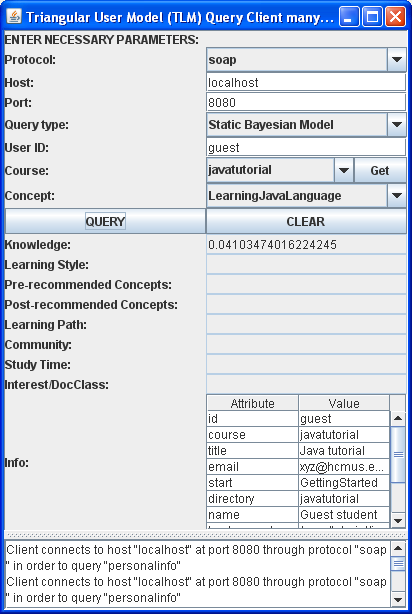
Recall that the third extended function of learning history sub-model is constructing user groups or user communities (see figure [3.4.10](#_Figure_II.22._Constructing)). The WOW system helps learners to interact with other learners in the same community via *collaborative area* shown in figure [3.4.25](#_Figure_II.37._Collaborative).



**Figure 3.4.25.** Collaborative area for collaborative learning

Now functions of adaptive learning system WOW were described and WOW can be considered as the thick client of Zebra server. WOW interacts with Zebra server and takes advantages of users’ information received from Zebra server so as to change its action to provide both learning contents and pedagogic environment/methods appropriate to each learner. There is an advancement that WOW is wrapped as a portlet (Abdelnur & Hepper, 2003) when WOW is deployed in portal environment (Kofman, 2009). Some typical portal environments are [Liferay (http://www.liferay.com/)](http://www.liferay.com/) and [eXo Platform (http://www.exoplatform.com/)](http://www.exoplatform.com/). WOW portlet uses the Ajax library [Dojo Toolkit (http://dojotoolkit.org/)](http://dojotoolkit.org/) developed by [The Dojo Foundation](http://dojofoundation.org/), year accessed: 2008 for speeding up response of WOW. Readers should refer to (W3Schools, AJAX Tutorial) in order to comprehend Ajax technology.

Another client that is called *thin client* also queries users’ information from Zebra server. Note that this thin client interacts with Zebra server via communication interfaces (CI). CI supports many protocols such as HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) and SOAP (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000). Figure [3.4.26](#_Figure_II.38._Thin) shows the thin client connecting to Zebra server.



**Figure 3.4.26.** Thin client connecting Zebra server

The thin client packed with Zebra server is available at internet link as follows:

<http://www.locnguyen.net/st/dissertations/zebra>

Zebra server uses web services engine [Axis (https://axis.apache.org/)](https://axis.apache.org/) developed by [The Apache Software Foundation](http://apache.org/) to publish CI. Other softwares used by Zebra server and WOW include [JComponentPack (http://zfqjava.com/)](http://zfqjava.com/) 2006 developed by [Extreme Component Company](http://zfqjava.com/), Java Mail 2009 developed by [Sun Microsystems](http://www.oracle.com/us/sun/), [SwingX](https://swingx.java.net/) 2007 developed by [SwingLabs](https://java.net/projects/swinglabs/), Sortable Table by David Gilbert & Nobuo Tamemasa, JavaScript lightweight graph [Mochikit (http://mochi.github.io/mochikit/)](http://mochi.github.io/mochikit/) developed by [Mochi Media, Inc.](http://mochimedia.com/), and some tools supported by Java communities over the world. I express my deep gratitude to these individuals, organizations, communities, and companies for providing their great softwares.

In general, Zebra server is open source software whose first release version 3.3 launched in November 21, 2009 from which there is no significant change until the final release version 3.10 launched in February 20, 2010. Therefore, I encourage software developers to improve and extend Zebra server. This is my biggest honor and happiness. In December 5, 2014 Zebra server is honorably demonstrated at [The 17th International Conference on Interactive Computer aided Learning (ICL2014), The 2014 World Engineering Education Forum](http://weef2014.org/), Dubai, UAE. Recall that Zebra server is available at internet link as follows:

<http://www.locnguyen.net/st/dissertations/zebra>

This sub-section [3.4](#_II.2.4._Implementation_of) ends up the main section [3](#_II.2._Zebra:_A) that proposes Zebra – the user modeling system for [Triangular Learner Model (TLM)](#_II.2.1._Triangular_Learner).

The limitation of this research is lack of privacy in learner model although external applications only access TLM via communication interface (see sub-section [3.2](#_II.2.2._The_architecture)) but user information is not encrypted now. The solution is to plug additional encrypted/decrypted module into communication interface but this will decrease system performance. Currently, Zebra does not support ubiquitous computing yet but ubiquitous user modeling is the future trend of Zebra as discussed in next section [4](#_V.2._Towards_ubiquitous).

## 4. Towards ubiquitous user modeling

Because [Triangular Learner Model](#_II.2.1._Triangular_Learner) (TLM) and the proposed user modeling system [Zebra](#_II.2._Zebra:_A) trend to support ubiquitous user modeling, it is necessary to survey ubiquitous user modeling. Nowadays there is a need for users to interact with many information technology (IT) systems at anywhere, for examples, users withdraw cash by ATM card or book airplane ticket online or play games on mobile phones (Heckmann D. , 2005, p. 3). This tends to develop ubiquitous user modeling system which performs ongoing modeling, ongoing sharing and ongoing exploitation of user models in ubiquitous computing environment that shifts from desktop interaction to mobile interaction. Ongoing modeling, ongoing sharing and ongoing exploitation of user models are three most important aspects of ubiquitous user modeling. Ubiquitous user model is defined as the user model which is monitored at any time, at any location and in any interaction context. Ubiquitous user model can be shared or integrated when necessary.

According to (Heckmann D. , 2005, p. 6), ubiquitous user modeling describes ongoing modeling and exploitation of user behavior with a variety of systems that share their user models. Ubiquitous user modeling is considered as the intersection of three domains: user modeling, ubiquitous computing, and semantic web, as shown figure [4.1](#_Figure_V.2.1._Ubiquitous) (Heckmann D. , 2005, p. 6).

**Figure 4.1.** Ubiquitous user modeling

All contents relevant to ubiquitous user modeling are extracted from the PhD research “Ubiquitous User Modeling” of the author (Heckmann D. , 2005). For convenience, some reference links are ignored and so readers should read this research available at <http://books.google.com.vn/books?id=e5adLEi4gYgC> for more details about ubiquitous user modeling. I express my deep gratitude to the author Dominikus Heckmann for providing her/his great research. This section [4](#_V.2._Towards_ubiquitous) includes following sub-sections:

* Sub-section [4.1](#_V.2.1._Overview_of) is an overview of ubiquitous user modeling.
* Ubiquitous user modeling system models all things (including learners) in real world as situation models. Hence, sub-section [4.2](#_V.2.2._Knowledge_presentation) mentions knowledge representation of situation model.
* Based on situation model, ubiquitous user modeling system simulates the real world as virtual world that is called ubiquitous world. Sub-section [4.3](#_V.2.3._Ubiquitous_World) sketches such ubiquitous world.
* Ubiquitous user modeling system is implemented as ubiquitous user model service. Sub-section [4.4](#_V.2.4._Ubiquitous_user) describes architecture and main actions of ubiquitous user model service. Author Dominikus Heckmann (Heckmann D. , 2005, pp. 155-169) proposed and implemented this ubiquitous user model service.
* Sub-section [4.5](#_V.2.5._Incorporating_Zebra) focuses on future trend of TLM and Zebra in which Zebra is integrated into ubiquitous user model service. In other words, Zebra will be developed towards ubiquitous user modeling.

### 4.1. Overview of ubiquitous user modeling

As discussed ubiquitous user modeling is the integration of user modeling, ubiquitous computing and semantic web. It is necessary to glance over these areas. Partial sub-sections [4.1.1](#_V.2.1.1._User_modeling), [4.1.2](#_V.2.1.2._Ubiquitous_computing), and [4.1.3](#_V.2.1.3._Semantic_web) browse context-aware user modeling, ubiquitous computing, and semantic web, respectively.

#### 4.1.1. User modeling and context-awareness

User modeling function is not only responsible for building up user model, maintaining the consistence of the model but also integrated with context-awareness function. It must monitor user’s behaviors and changes in context and can take out new assumptions about user and context. The schema of user modeling with integrated context-awareness is shown in following figure [4.1.1.1](#_Figure_V.2.2._User) (Heckmann D. , 2005, p. 15). Because user modeling is surveyed in (Nguyen & Do, Learner Model in Adaptive Learning, 2008), this sub-section [4.1.1](#_V.2.1.1._User_modeling) only mentions context-aware user modeling when ubiquitous system always focuses on environment and context.

**User Model**

(Aspects of User)

**Personal Journal**

(with contextual parts)

**Context Model**

(Aspects of Context)

**Model**

**Upward Inference**

(Model Acquisition)

**Downward Inference**

(Model Application)

**Information**

(about user)

**Information**

(about context)

**Input**

**New Information**

(about user)

**New Information**

(about context)

**Output**

**Figure 4.1.1.1.** User modeling with integrated context-awareness

Input data about user and context are processed in upward inference direction. This is essentially process of collecting data that adds aspects about user to user model or personal journal. The downward inference direction is responsible for taking out new assumptions about user or context (Heckmann D. , 2005, p. 15). It can use artificial intelligence techniques to improve deduction ability.

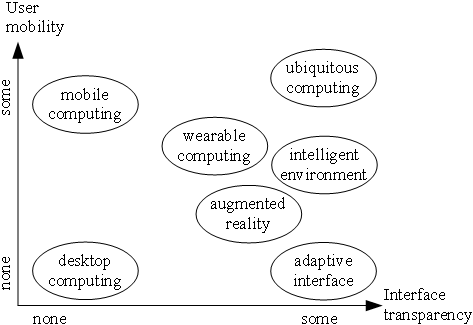
As aforementioned in figure [4.1](#_Figure_V.2.1._Ubiquitous), ubiquitous user modeling is the intersection of user modeling, ubiquitous computing, and semantic web. The successive sub-section [4.1.2](#_V.2.1.2._Ubiquitous_computing) mentions ubiquitous computing.

#### 4.1.2. Ubiquitous computing

According to (Heckmann D. , 2005, p. 19), ubiquitous computing system is defined as a heterogeneous set of computing devices, a set of supported tasks and some infrastructure on which the devices rely on in order to carry out their tasks. There are two basic properties of ubiquitous computing system: ubiquity and transparency (Heckmann D. , 2005, p. 19).

* Ubiquity: User can interact with system at anywhere.
* Transparency: The system is non-intrusive and is integrated into everyday environment.

There are some research fields which are similar to ubiquitous computing such as mobile computing, wearable computing, intelligent environment, etc. Author (Maffioletti, 2001, p. 2) gives two dimensions: *user mobility* and *interface transparency* in order to distinguish these fields. User mobility (ubiquity) reflects the degree of freedom that user can move around, when interacting with system. Interface transparency reflects the attention that system requires of user. The boundary between these fields (or the spatial arrangement) is represented in following figure [4.1.2.1](#_Figure_V.2.3._Spatial) (Heckmann D. , 2005, p. 20).



**Figure 4.1.2.1.** Spatial arrangement of ubiquitous computing similar fields according to two dimensions: user mobility and interface transparency

As shown in figure [4.1.2.1](#_Figure_V.2.3._Spatial), ubiquitous computing gains the highest level of user mobility of interface transparent. Note that intelligent environment is the environment (around user) enriched by computers embedded in everyday objects like computer, mobile devices, blackboard, tables, chairs… and enriched by sensors to catch information from context (Heckmann D. , 2005, p. 19). Ubiquitous computing system can be considered as advanced intelligent environment.

**Situation model and situated interaction**

Authors (Barwise & Perry, 1981) suggests the *Situation Semantics* which describes a framework to conceptualize everyday situations. Situations are constituted of objects owning properties and standing in relationships. The real world consists of many situations. Each situation is established at a concrete time and place. Situations are categorized into various situation-types; it means that situation-types are partial functions that specify situations (Heckmann D. , 2005, p. 21).

Situations constitute the *situation model* which consists of three main sub-models: user model, context model and resource model:

* *User model* comprises user-related information: knowledge, goals, personal traits, etc. User model was mentioned in (Fröschl, 2005).
* *Context model* comprises user perception and actions which are independent of individual user when they affect equally all users in the same environment (Heckmann D. , 2005, p. 23). In other words, perception and actions are determined by environment.
* *Resource model* gives support to resource-adaptive process that is aware of what resources are available to (ubiquitous computing) system at any time and that employs a predefined adaptation strategy to perform well when faced with such a situation of varying resource availability (Heckmann D. , 2005, p. 23). Resource is defined as available means to solve a task.

Figure [4.1.2.2](#_Figure_V.2.4._Situation) depicts situation model comprising of user model, context model, and resource model.

User model

Context model

Resource model

**Situation model**

**Figure 4.1.2.2.** Situation model

However it can contains more models such as usage model, location model, personal journal, low-level sensor data and event log (Heckmann D. , 2005, p. 63). The following figure [4.1.2.3](#_Figure_V.2.5._Extended) is the extended situation model (Heckmann D. , 2005, p. 63).

User Model

Context Model

Usage Model

Location Model

Personal Journal

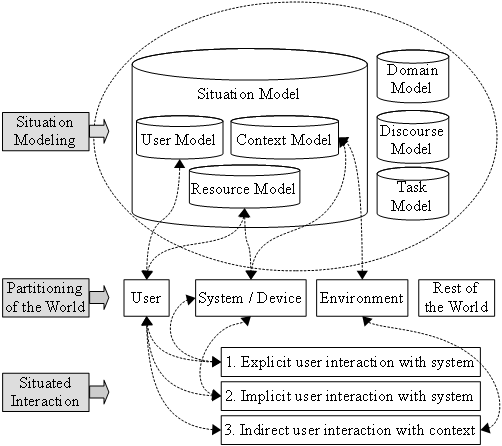
Low Level Sensor Data

Resource Model

Event Model

**Figure 4.1.2.3.** Extendedsituation model

The process so-called *situation modeling* process is responsible for constructing situation model. The following figure [4.1.2.4](#_Figure_V.2.6._Situated) (Heckmann D. , 2005, pp. 23-24) gives the overview of situated interaction for ubiquitous computing where the world is divided into four parts: *user*, *system/mobile devices*, *environment* and *rest of the world*.



**Figure 4.1.2.4.** Situated interaction and situation modeling for ubiquitous computing

So *situated interaction* denotes the explicit, implicit and indirect interaction between a human and an intelligent environment (such as desktop, laptop, and mobile devices), that take situation-awareness into account (Heckmann D. , 2005, p. 25). Note that intelligent environment is everything around user consisting of information technology (IT) devices, tables, chairs, etc. Implicit interaction is defined as the interaction of human with environment, which is aimed to achieve a goal. Within this process the system acquires implicit inputs from the user and may present implicit output to the user. Implicit input are user actions which have not chief purpose of interaction with IT devices.

As aforementioned in figure [4.1](#_Figure_V.2.1._Ubiquitous), ubiquitous user modeling is the intersection of user modeling, ubiquitous computing, and semantic web. The successive sub-section [4.1.3](#_V.2.1.3._Semantic_web) mentions semantic web.

#### 4.1.3. Semantic web

According to W3Consortium <http://www.w3c.org>, semantic web is a web of data. It is about common formats for integration and combination of data drawn from diverse sources and about language for recording how the data relates to real world objects. The semantic web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries (W3C, W3C semantic web activity, 2001). The basic idea of semantic web is the use of ontology to annotate resource in the World Wide Web (Heckmann D. , 2005, p. 27). Semantic web is based on the Resource Description Framework (RDF). According to W3Consortium, RDF is integrated into a variety of applications from library catalogs and world-wide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events; using XML (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) as an interchange syntax (Daly, Forgue, & Hirakawa, 2004). In brief, RDF is the means to represent the meta-data about resources on internet such as title, author, modification date, copyright, and license information about web document (Heckmann D. , 2005, p. 29). The resource is defined as a network data object (web document, ontology entity, etc.) or service that can be identified by a URI where URI is string-format identifier used to refer to web resource on internet (Heckmann D. , 2005, p. 28).

So RDF in form of XML file is used as the means to represent semantic web. However there is another language that is more powerful for expressing semantic web than RDF; this is Ontology Web Language (OWL) giving support to interpretability. OWL adds more vocabulary for describing properties and classes of web resources and among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes (Heckmann D. , 2005, p. 29). The General User Model Ontology (GUMO) is also based on OWL. Please read (Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005) for more details about GUMO. Because ontology is very important to semantic web, please study some basic concepts of ontology (Wikipedia, Ontology (information science), 2014).

Ubiquitous user modeling system models all things (including learners) in real world as situation models. Hence, successive sub-section [4.2](#_IV.2.2._Knowledge_presentation) mentions knowledge representation of situation model.

### 4.2. Knowledge presentation of situation model

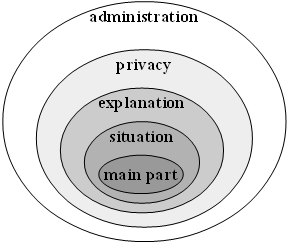
Situation model consisting of user model, context model and resource model is the core of ubiquitous user modeling system. Situation model has many situations which are represented as situational statements. Each *situational statement* contains information about user model entries, context data, sensor data, data on mobile devices, environment, etc. (Heckmann D. , 2005, p. 53). The concept “situational statement” derived from the integration of three domains: user modeling, ubiquitous computing and semantic web gives us the way to represent knowledge base of ubiquitous user modeling. Now we should survey this concept.

The representation of situational statement is based on RDF and FrameNet approach. FrameNet approach aims to create an online lexical resource for English, based on frame semantic and supported by corpus evidence (Heckmann D. , 2005, p. 46). Information is organized as frames. A frame is an intuitive construction that makes it possible to formalize the links between semantics and syntax in the results of lexical analysis. Each frame identifies a set of frame elements which are frame-specific semantic roles such as participants, props, phases of a state of affairs (Heckmann D. , 2005, p. 47).

RDF is based on the triples of *subject*, *predicate* and *object* (Heckmann D. , 2005, p. 57)*.* These elements (*subject*, *predicate*, *object*) are resources which can refer to different ontologies such as OWL and GUMO (Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005). The *subject* is linked to the *object* denoting the value through an arc labeled with the *predicate*. It means that the *subject* has a property *predicate*, valued by *object*. For example, the triples “*subject* = Susan; *predicate* = blood pressure; *object* = high” denotes the statement “Susan has high blood pressure”. Note that “situational statement” is called in brief “statement” and **statement model refers to situation model**. Figure [4.2.1](#_Figure_V.2.7._RDF) expresses RDF triple that represents a statement.

**Figure 4.2.1.** RDF triple

Each situational statement is represented as the onion model with five layers (Heckmann D. , 2005, pp. 58-59), shown in figure [4.2.2](#_Figure_V.2.8._Onion). Each layer containing attributes is considered a collection or box. There are five boxes: *main part*, *situation*, *explanation*, *privacy* and *administration* (Heckmann D. , 2005, p. 59). The organization is the hierarchy of boxes (layers) wrapped around the main part box. The attributes in each box are arranged in form of RDF triple: *subject* – p*redicate* – *object*. It is easy to recognize that the organization of situation statement is derived from FrameNet. Consequently each box is considered as a frame in FrameNet approach. Note that attributes can be linked to ontologies such as OWL and GUMO.



**Figure 4.2.2.** Onion model of statement

* *Main part box* contains main information about statement in form of attribute-value pairs (Heckmann D. , 2005, p. 59). Besides basic attributes derived directly from RDF triple such as *subject*, *predicate* and *object*, there are two optional attributes: *auxiliary* and *range.* For example, the main part box of statement “*Susan’s mark of math test is* 5” is expressed as: “*subject* = Susan”, “*predicate* = mark of math test” and “*range* = 5”. The attribute *predicate* in original RDF triple is now divided into three attributes: *auxiliary*, *predicate* and *range* in order to extend the ability of representing statement in real world. Table [4.2.1](#_Table_V.2.1._Main) (Heckmann D. , 2005, p. 59) shows the main part box.

|  |
| --- |
| **Main part** |
| Subject |
| Auxiliary |
| Predicate |
| Range |
| Object |

###### **Table 4.2.1.** Main part box

* *Situation box* contains information about the temporal and spatial embedding of situational statement (Heckmann D. , 2005, p. 61). This is really important in ubiquitous computing when it is necessary to observe user at any time and anywhere. Situation box has five attributes: *start*, *end*, *durability*, *location* and *position*. All of them are optional. The attribute *start* expresses the time when the statement becomes valid. The attribute *end* expresses the time when the statement ends. The attribute *durability* expresses the time span that the statement is still valid. The attribute *location* denotes the spatial location that the statement occurs. It is slightly different from the attribute *position* which denotes exactly the coordinates. Table [4.2.2](#_Table_V.2.2._Situation) (Heckmann D. , 2005, p. 61) shows the situation box.

|  |
| --- |
| **Situation** |
| Start |
| End |
| Durability |
| Location |
| Position |

###### **Table 4.2.2.** Situation box

* *Explanation box* consists of attributes which are required by the user’s right for explanation and by the conflict resolution mechanism for inferred user modeling data (Heckmann D. , 2005, p. 62). There are five attributes: *source*, *creator*, *method*, *evidence* and *confidence*. All of them are optional. The attribute *source* declares where the statement was stored or from which sensor measurement it came from (Heckmann D. , 2005, p. 62). The attribute *creator* gives the name of system or person who created statement. The attribute *method* tells us how the main part box was inferred or measured. The attribute *evidence* provides the list of evidences that support statement. The attribute *confidence* provides the place to store the creator’s expected level of confidence in statement. Table [4.2.3](#_Table_V.2.3._Explanation) (Heckmann D. , 2005, p. 62) shows the explanation box.

|  |
| --- |
| **Explanation** |
| Source |
| Creator |
| Method |
| Evidence |
| Confidence |

###### **Table 4.2.3.** Explanation box

* *Privacy box* provides critical information about privacy of user model data, especially in case of distributing sensitive data (Heckmann D. , 2005, p. 62). There are five optional attributes: *key*, *owner*, *access*, *purpose* and *retention*. The *key* attribute forms an encrypted security key. The *owner* attribute provides the person or system that is responsible for managing the distribution of user model data and editing three remaining attributes: *access*, *purpose* and *retention*. The *access* attribute which specifies access right of system or person has possible values: public, friend and private. The *purpose* attribute which puts restrictions on the intention for which statement may be used has possible values: commercial, research and minimal. The *retention* attribute which expresses how long statement is kept or used has possible values: long, middle and short. Privacy is especially important in distributed environment like ubiquitous computing system. Table [4.2.4](#_Table_V.2.4._Privacy) shows the privacy box.

|  |
| --- |
| **Privacy** |
| Key |
| Owner |
| Access |
| Purpose |
| Retention |

###### **Table 4.2.4.** Privacy box

* *Administration box* is the outermost layer in the onion model of statement (Heckmann D. , 2005, p. 63). It fastens organizational issues in huge sets of situation statements. Administration box has five attributes: *id*, *unique*, *replaces*, *group* and *notes*. The *id* attribute is the locally unique identifier of statement but the *unique* attribute is the global unique identifier of statement all over the world. *Unique* attribute is often defined as URI. Maybe the creator who creates a statement has no right to delete or replace it. So the *replaces* attribute specifies which statement can be replaced by a newer one. As discussed, the situation model consists of three sub-models: user model, context model and resource model. However it can contains more model such as usage model, location model, personal journal, low-level sensor data, event log. These models are denoted by attribute *group*. The least important attribute *notes* contains additional unstructured meta information about statement. Table [4.2.5](#_Table_V.2.5._Administrator) shows the administration box.

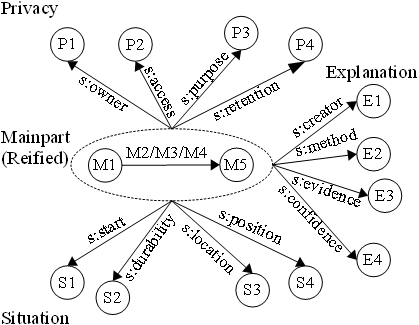
|  |
| --- |
| **Administration** |
| Id |
| Unique |
| Replaces |
| Group |
| Notes |

###### **Table 4.2.5.** Administrator box

That the representation of situation model is derived from RDF triple and OWL makes the integration of three fields: user modeling, ubiquitous computing and semantic web. The model of situational statement (onion model) is in form of XML file (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008); all attributes are translated into XML-elements. The syntax of XML-format to represent situational statement is called *SituationML* or *UserML* (Heckmann D. , 2005, pp. 67-70). The onion model with five boxes is represented by XML format as below (Heckmann D. , 2005, p. 69):

|  |
| --- |
| *<statement>*  *<mainpart>*  *<subject>a*1*</subject>*  *<auxiliary>a*2*</auxiliary>*  *<predicate>a*3*</predicate>*  *<range>a*4*</range>*  *<object>a*5*</object>*  *</mainpart>*  *<situation>*  *<start>a*6*</start>*  *<end>a*7*</end>*  *<durability>a*8*</durability>*  *<location>a*9*</location>*  *<position>a*10*</position>*  *</situation>*  *<explanation>*  *<source>a*11*</source>*  *<creator>a*12*</creator>*  *<method>a*13*</method>*  *<evidence>a*14*</evidence>*  *<confidence>a*15*</confidence>*  *</explanation>*  *<privacy>*  *<key>a*16*</key>*  *<owner>a*17*</owner>*  *<access>a*18*</access>*  *<purpose>a*19*</purpose>*  *<retention>a*20*</retention>*  *</privacy>*  *<administrator>*  *<id>a*21*</id>*  *<unique>a*22*</unique>*  *<replaces>a*23*</replaces>*  *<group>a*24*</group>*  *<notes>a*25*</notes>*  *</administrator>*  *<statement>* |

Following figure [4.2.3](#_Figure_V.2.9._The) shows the schema of statement model in RDF graph notation in the variation with reification (Heckmann D. , 2005, pp. 70-72). The reified main part is interpreted more with situation attributes, privacy attributes and explanation attributes. Each oval refers to a resource which may be an ontology entity. The attributes are reified as RDF predicates such as *s:start*, *s:location*, and *s:owner* while their values can be defined as RDF objects.



**Figure 4.2.3.** The schema of statement model in RDF graph notation

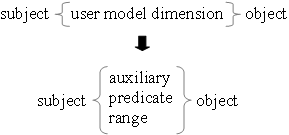
Based on situation model, ubiquitous user modeling system simulates the real world as virtual world that is called ubiquitous world. Successive sub-section [4.3](#_IV.2.3._Ubiquitous_World) sketches such ubiquitous world.

### 4.3. Ubiquitous World

Ubiquitous user modeling system cannot reflect totally real world; so it must perform modeling tasks in a uniform virtual world model in order to simulate the real world. Such virtual world is so-called Ubiquitous World. Moreover statements in situation model refer to various ontologies. So we should glance over user model ontologies (sub-section [4.3.1](#_V.2.3.1._User_model)) and ubiquitous world (sub-section [4.3.2](#_V.2.3.2._Ubiquitous_World)).

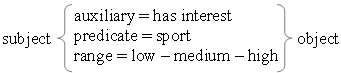
#### 4.3.1. User model ontologies

As discussed, RDF and ontologies are main means of integrating user modeling and ubiquitous computing. The attributes in situational statement in form of RDF triple can refer to other ontology namely GUMO (Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005). Thus the user model dimension in RDF triple is divided into three parts: *auxiliary*, *predicate*, *range* (Heckmann D. , 2005, p. 86). Figure [4.3.1.1](#_Figure_V.2.10._Situational) shows the situational statement represented by RDF triple.



**Figure 4.3.1.1.** Situational statement represented by RDF triple

So user model dimensions such as auxiliary, predicate, range or any other information about user are entities in GUMO. The GUMO gives the clear separation in the modeling between user model auxiliaries, predicate classes and special ranges (Heckmann D. , 2005, p. 86). For example the statement “*Peter likes sport very much*” is translated into the form of RDF triple, shown in figure [4.3.1.2](#_Figure_V.2.11._An) as follows:



**Figure 4.3.1.2.** Anexample of statement represented by RDF triple

The auxiliaries give support to specify the predicates which refer to ontology entities. There are some following auxiliaries (Heckmann D. , 2005, p. 87):

* The auxiliary “*has property”* leads to *basic user dimensions* such as personality, contact information, demographics, etc.
* The auxiliaries “*has interest”* and “*has preference”* lead to *user model interest categories* such as music categories or film genres.
* The auxiliaries “*has regularity”* and “*has done”* lead to usage data and low-level sensor data.
* The auxiliary “*has location*” leads spatial ontology.

According to GUMO, *basic user dimensions* that are basic information about user are classified into contact information, demographics, personality, characteristics, etc. and *user model interest categories* include film, music, sports, etc. Table [4.3.1.1](#_Table_V.2.6:_Basic) (Heckmann D. , 2005, p. 88) shows basic user dimensions in GUMO.

|  |
| --- |
| **Basic User Dimension** |
| Contact Information |
| Demographics |
| Ability and Proficiency |
| Personality |
| Characteristics |
| Emotional State |
| Physiological State |
| Mental State |
| Motion |
| Role |
| Nutrition |
| Facial Nutrition |

###### **Table 4.3.1.1.** Basic user dimensions in GUMO

Table [4.3.1.2](#_Table_V.2.7:_User) (Heckmann D. , 2005, p. 98) shows user model interest categories in GUMO.

|  |
| --- |
| **Interest Category** |
| Film |
| Music |
| Sports |
| PC-Games |
| Recreation |
| Environmental Topics |
| Science |
| Museum |

###### **Table 4.3.1.2.** User model interest categories in GUMO

Note that there are not only GUMO but also other ontologies to be applied in ubiquitous user modeling. Besides GUMO – the ontology supporting situation model (situational statement) about user, we have some ontologies such as physical ontology, spatial ontology, temporal ontology, activity ontology, and inference ontology.

When user model ontologies are defined according to GUMO, it is easy to describe ubiquitous world in successive sub-section [4.3.2](#_V.2.3.2._Ubiquitous_World).

#### 4.3.2. Ubiquitous World

Ubiquitous World (UbisWorld) is considered as the partial copy of real world. It represents some parts of real world such as office, school, and airport. UbisWorld includes persons, objects, locations as well as times, events and their properties and features in real world (Heckmann D. , 2005, p. 100). So UbisWorld is also defined as the uniform virtual world used for simulation, inspection and control (Heckmann D. , 2005, p. 100). All tasks in ubiquitous user modeling are assumed to take place in UbisWorld. The correlation between UbisWorld and real world is shown in following figure [4.3.2.1](#_Figure_V.2.12._The) (Heckmann D. , 2005, p. 100).

**Real World**

**Modeling Standards**

(ontologies, xml…)

Reduction &

Abstraction

Modeling

**Ubiquitous Tools**

Stimulation

Monitor & Control

Representation & Use

**Figure 4.3.2.1.** The correlation between UbisWorld and real world

There are three main processes relating to UbisWorld:

* UbisWorld is reduced and abstracted from real world.
* UbisWorld is modeled by modeling standards like ontologies and markup languages.
* UbisWorld is simulated and used by ubiquitous tools.

The author (Heckmann D. , 2005, p. 101) gave an example of UbisWorld in which there is a room in real world having a light switch and table. Such room is abstracted as UbisWorld room. The UbisTools can simulate the light-on, light-off behavior of the real room; thus if the virtual switch gets pressed, the virtual light in UbisWorld room will shine. The UbisWorld room can be used to monitor the real room; for example every time the real switch in real room gets pressed, the virtual light in UbisWorld room will shine. Moreover the UbisWorld room can control real room; for example, that the virtual switch gets pressed causes the real light to shine.

UbisWorld can use many ontologies such as physical ontology, spatial ontology, temporal ontology, activity ontology, situation ontology and inference ontology (Heckmann D. , 2005, p. 102). The semantic web ontologies and languages like RDF and OWL support the integration of such UbisWorld ontologies. The attributes of situational statement in user model are specified in UbisWorld. Figure [4.3.2.2](#_Figure_V.2.13._UbisWorld) (Heckmann D. , 2005, p. 102) gives a combination of UbisWorld and ontologies.

User/Group

Device/Object

Location

Time

Event

Situation

(GUMO)

Inference

**Predicates**

**Individuals**

**Figure 4.3.2.2.** UbisWorld and ontologies

Ubiquitous user modeling system is implemented as ubiquitous user model service. Successive sub-section [4.4](#_V.2.4._Ubiquitous_user) describes architecture and main actions of ubiquitous user model service.

### 4.4. Ubiquitous user model service

*Ubiquitous user model service* is responsible for manipulating ubiquitous user model. It is essentially an implementation of ubiquitous user modeling system. According to (Heckmann D. , 2005, p. 155), ubiquitous user model service is an application independent server with a distributed approach for accessing and storing user information, the possibility to exchange and understand between different applications as well as adding privacy and transparency to the statements about user. The semantics of user information (situational statement) is mapped to the ontology GUMO. In brief, we call ubiquitous user model service as ubiquitous service in user modeling context.

The general architecture and work flow of ubiquitous user model service are mentioned in successive sub-sections [4.4.1](#_V.2.4.1._Architecture_of) and [4.4.2](#_V.2.4.2._Main_work), respectively. Author Dominikus Heckmann (Heckmann D. , 2005, pp. 155-169) proposed and implemented this architecture.

#### 4.4.1. Architecture of ubiquitous user model service

The architecture of ubiquitous user model service consists of five boxes: *Distributed Services box*, *Application box*, *Distributed Statements box*, *Distributed Ontologies box*, *Interfaces & Exchange box*. Figure [4.4.1.1](#_Figure_V.2.14._Architecture) (Heckmann D. , 2005, p. 156) depicts this architecture together such five boxes.

E-learning

Shopping

Sensor

Speech

**Applications**

**Distributed Ontologies**

Ontology Editor

UserModel Editor

Context Editor

Location Manager

**Interfaces**

**Exchange**

Query (*SituationQL*)

Add (SituationML)

**…**

Others

DB

XML

RDF

**Distributed Statements**

Inference Engine

Ontology Reasoning

Retrieval Filter

Situation Server

Situation Adder

Interface Manager

Confliction Resolver Reasoning

**Distributed Service**

**Figure 4.4.1.1.** Architecture of ubiquitous user model service

**Distributed Services** **box** is the main box enriched by other boxes. It is constituted of internal modules (or services) such as *Situation Server* or *User Model Server*, *User Model Adder* or *Situation Adder*, *Retrieval Filter*, *Conflict Resolution*, *Inference Engine*, *Interface Engine* and *Ontology Reasoning* which are very necessary to perform ubiquitous user modeling tasks (Heckmann D. , 2005, pp. 155-156).

* Situation Server is the web server that manages the storage of statements about user.
* Situation Adder is the parser which analyzes incoming new statements and writes them to repositories.
* Retrieval Filter is responsible for controlling the retrieval of situational statements.
* Conflict Resolution is responsible for detecting and resolving possible conflicts.
* Inference Engine is the proactive engine that applies meta rules, writes new statements and triggers events.
* Interface Manager has a control mechanism that integrates user interfaces.
* Ontology Reasoning is responsible for applying knowledge from various ontologies.

**Application box** lists applications that possibly cooperate with distributed services such as e-learning, sensors, airport service, and mobile phone (Heckmann D. , 2005, p. 156).

**Distributed Statements box** (Heckmann D. , 2005, pp. 156-157) plays the role of distributed database management system. It is responsible for storing and managing situational statements about user. This module separates data from software. So the situation model is stored at here. It is considered that distributed statements box contains repositories of statements. These repositories are independent from the services, which allow various services to operate independently on the same knowledge bases. Please read the book “Principles of Distributed Database Systems” (Özsu & Valduriez, 2011, pp. 1-38) for more details about distributed database management system.

**Distributed ontologies box** (Heckmann D. , 2005, p. 157) is responsible for manipulating and integrating various ontologies into modeling service. These ontologies are used for the interpretation of statements, for the detection of conflicts and for the definition of expiry defaults and privacy defaults. This module separates the syntax of ontologies from semantics of ontologies.

**Interfaces & Exchange box** (Heckmann D. , 2005, p. 157) separates the user model service from the user interfaces and development tools. Each interface and tool can operate with different repositories, different ontologies and different services in distributed services box. It is very important for ubiquitous computing in the manner of computation at any time and at anywhere. There are some interfaces and tools such as *ontology editor*, *user model editor*, *context editor*, *UserML viewer*, *XForms viewer*, *location manager*, *strategy visualizer* are very necessary to assist user/administrator in manipulating or surveying ubiquitous user model (see figure [4.4.1.1](#_Figure_V.2.14._Architecture)). The communication between user/administrator and distributed services is established through exchange tool. Thus the language *SituationQL* (or *UserQL*) is used to issue queries about user information (namely situational statements) and the language *SituationML* (or *UserML*) is used to answer such queries. Please read (Heckmann D. , 2005, pp. 124-128) for more details about *SituationQL* or *UserQL* and read (Heckmann D. , 2005, pp. 67-74) for more details about *SituationML* or *UserML*.

Derived from the general architecture shown in figure [4.4.1.1](#_Figure_V.2.14._Architecture), the work flow of ubiquitous user model service is mentioned in successive sub-section [4.4.2](#_V.2.4.2._Main_work).

#### 4.4.2. Main work flow of ubiquitous user model service

Ubiquitous user model service supports three operations: *Add*, *Request* and *Report*. The work flow of them is shown in following figure [4.4.2.1](#_Figure_V.2.15._The):

**Distributed Ontologies**

**Situation Server**

**Conflict Resolution Filter & Ranking**

**Distributed Situational Statements**

XML

DB

**Situation Adder**

**2**

**4.1**

**4.2**

**4.3**

**Users**

**Systems**

**Devices**

**1**

Add

**Users**

**Systems**

**Devices**

**3**

Request

**Users**

**Systems**

**Devices**

**5**

Report

**Figure 4.4.2.1.** The work flow of ubiquitous user model service

The work flow includes steps below (Heckmann D. , 2005, pp. 157-158):

* The system, users or sensors add statements in form of *SituationML* **(1)**
* The statements are sent to module *Situation Adder* that analyzes the incoming data and distributes them to repositories in *Distributed Statements box* **(2)**.
* The system, users or sensors request statements in form of *SituationQL* **(3)**.
* Some repositories are chosen from *Distributed Statements box* in order to retrieve appropriate situational statements **(4.1)**.
* The conflict resolution strategies are applied into such statements **(4.2)**.
* Such statements are mapped to ontologies **(4.3)**.
* Final statements are sent to users/system in form of *SituationML* **(5)**.

Successive sub-section [4.5](#_V.2.5._Incorporating_Zebra) focuses on future trend of [Triangular Learner Model](#_II.2.1._Triangular_Learner) (TLM) and the proposed user modeling system [Zebra](#_II.2._Zebra:_A) in which Zebra is integrated into this ubiquitous user model service.

### 4.5. Incorporating Zebra into ubiquitous user model service

When surveying the architecture of ubiquitous service, I recognize that such service lacks of the robust inference mechanism. The *Inference Engine* in *Distributed Services box* only writes statements and triggers events. On the contrary, [Zebra](#_II.2._Zebra:_A) has two robust inference engines: *mining engine* and *belief network engine* with ability to infer new assumptions about user and to support personal recommendation. But Zebra don’t support ubiquitous computing and user model in distributed environment. If the combination of Zebra and ubiquitous user model service is successful, there is a new user modeling system that takes advantage of both Zebra and ubiquitous computing. Thus it is possible to predict situational statement in ubiquitous environment.

So my idea is to incorporate Zebra into ubiquitous service by replacing *Inference Engine* (in *Distributed Services box*) with Zebra; figure [4.5.1](#_Figure_V.2.16._Incorporating) expresses such incorporation. But this raises some obstacles that must be overcome.

* *Firstly*, this is how Zebra interacts with *Distributed Services box* of ubiquitous service when the database and data structures in Zebra such as Bayesian network, hidden Markov model, and sequential pattern are very different from statement repositories in ubiquitous service. Please see (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) and (Nguyen & Do, Learning Concept Recommendation based on Sequential Pattern Mining, 2009) for more details about Bayesian network and sequential pattern, respectively.
* *Secondly*, most techniques in Zebra like data mining, artificial intelligence, and machine learning are unfamiliar to ubiquitous service.
* *Finally*, the most serious obstacle is that the output of inference process in Zebra such as new information about user and new personal recommendations is represented in the form which is incompatible with knowledge representation of situation model in ubiquitous service such as ontologies and RDF.

Ontology Reasoning

Retrieval Filter

Situation Server

Situation Adder

Interface Manager

Confliction Resolver Reasoning

**Distributed Service**

**BNE**

**ME**

**Zebra**

**Distributed Ontologies**

**Applications**

**Interfaces**

**Distributed Statements**

**Figure 4.5.1.** Incorporating Zebra into ubiquitous service

These problems can be solved by getting the best out of the *communication interfaces* (**CI**) in architecture of Zebra. Note that each CI allows users to see or modify restrictedly their TLM. In addition, all outside applications interact with Zebra via these interfaces. So the ubiquitous service will interact with Zebra by special CI by request-response protocol including five following steps:

1. Ubiquitous service sends one request statement in form of *SituationQL* to CI.
2. CI interprets this statement into the data structure which inference engines of Zebra such as *mining engine* and *belief network engine* are aware of.
3. The request in form of data structure that Zebra knows is sent to Zebra.
4. Inference engines perform some concrete deduction tasks in order to take out some new assumptions, information, personal recommendations about user. This output is sent back CI.
5. CI interprets such output into a XML file (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) in form of *SituationML* which is readily understandable for ubiquitous service and sends it to ubiquitous service.

Figure [4.5.2](#_Figure_V.2.17._Request-Response) interprets such five steps of communication protocol between Zebra and ubiquitous service.

**BNE**

**ME**

**Zebra**

Request Receiver

**1** (SituationQL)

Zebra Interpreter

**2**

**3**

Responder

**4**

**5** (SituationML)

**CI**

**Figure 4.5.2.** Request-Response communication protocol between Zebra and ubiquitous service

There is a big advantage of communication between Zebra and ubiquitous service when CI is implemented in web service standard; so ubiquitous service doesn’t need to know what technologies inside Zebra are. The interoperation between CI and ubiquitous service is done via SOAP protocol (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000). Ubiquitous service interacts with the Zebra in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) with an XML (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) serialization in conjunction with other web-related standards.

## 5. Conclusion

In general, the “[Triangular Learner Model (TLM)](#_II.2.1._Triangular_Learner)” 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.

The user modeling system that builds up and manipulates TLM is called [Zebra](#_II.2._Zebra:_A). Zebra includes the core engine and a set of communication interfaces. The core engine is the composition of two sub-engines: mining engine (ME) and belief network engine (BNE). ME structures TLM and applies data mining techniques into discovering other characteristics beyond knowledge and learning style. ME manages learning history sub-model. BNE is responsible for inferring new information from TLM by using deduction mechanism available in belief network. BNE manages knowledge sub-model and learning style sub-model. Communication interfaces (CI) allow users to see or modify restrictedly TLM. Adaptive applications also interact with Zebra through CI. I also propose the new architecture of adaptive education hypermedia system (AEHS) that interacts with Zebra.

This research is fundamental research, thus, the methodology to build up TLM is specified and proven via mathematic tools. The feasibility of TLM and architecture is authenticated via the user modeling software [Zebra](#_II.2._Zebra:_A) which constructs and manages TLM. Moreover, the adaptive learning system that interacts with Zebra is also implemented as e-learning web-based application [WOW](#_Figure_II.30._Typical). Another strong point of TLM is ability of extension, other user information such as user interests and goals can be discovered or extracted from TLM. Researchers can use the methodology, models and mathematical formulas in this research to build up their own user model and user modeling system. They can also develop TLM and Zebra by extending advanced functions such as discovering user goals, context-aware adaptation, ubiquitous modeling, and mobile service. Section [4](#_V.2._Towards_ubiquitous) discussing the future trend is an example of TLM extension; in which TLM will be integrated into ubiquitous service, which means that Zebra will support ubiquitous user modeling. This research contributes to user modeling and adaptive learning community both a methodology for constructing user model and a whole learner model TLM.

## Acknowledgments

I am very grateful to Prof. Dr. Dong, Bich-Thuy T. for providing me invaluable help, knowledge, experiences and encouragement during the time of research. She is my supervisor and sponsor. If it were not for her help, I would not complete the research in time. I have furthermore to thank the Department of Information System – Faculty of Information Technology – University of Science for conveniences during the time of research.

I express my deepest gratitude for my family and friends who have supported me in difficult times. I also express my warmest gratitude to authors who provided excellent works to which I refer during doing this research. I also express my gratitude to all people who gave me the possibility to complete this research.

## References

Abdelnur, A., & Hepper, S. (2003). *JavaTM Portlet Specification version 1.0.* Java Community Process.

Apache. (1999). *Apache Tomcat*. (The Apache Software Foundation) Retrieved 2008, from Apache Tomcat website: http://tomcat.apache.org

Barwise, J., & Perry, J. (1981). Situations and Attitudes. *The Journal of Philosophy, 78*(11 Seventy-Eighth Annual Meeting of the American Philosophical Association Eastern Division ), 668-691.

Brajnik, G., & Tasso, C. (1994, January). A Shell for Developing Non-monotonic User Modeling Systems. (B. R. Gaines, Ed.) *International Journal of Human-Computer Studies, 40*(1), 31-62.

Brusilovsky, P. (1996). Methods and Techniques of Adaptive Hypermedia. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction, 6*(2-3), 87-129.

Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction, 11*(1-2), 87-110.

Brusilovsky, P. (2004). KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In S. Feldman, M. Uretsky, M. Najork, & C. Wills (Ed.), *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (pp. 104-113). ACM New York, NY, USA.

Brusilovsky, P., Sosnovsky, S., & Shcherbinina, O. (2005). User Modeling in a Distributed E-Learning Architecture. In L. Ardissono, P. Brna, & A. Mitrovic (Ed.), *Proceedings of the 10th international conference on User Modeling* (pp. 387-391). Edinburgh, UK: Springer-Verlag Berlin, Heidelberg.

Daly, J., Forgue, M.-C., & Hirakawa, Y. (2004, February 10). *World Wide Web Consortium Issues RDF and OWL Recommendations - Semantic Web emerges as commercial-grade infrastructure for sharing data on the Web.* Retrieved 2009, from World Wide Web Consortium website: http://www.w3.org/2004/01/sws-pressrelease

De Bra, P., & Calvi, L. (1998). AHA! An open Adaptive Hypermedia Architecture. (D. Tudhope, & D. Cunliffe, Eds.) *The New Review of Hypermedia and Multimedia, 4*(1), 115-139.

Dunn, R., & Dunn, K. (1996). *The Dunn and Dunn Model*. (Online Learning Style Assessments & Community) Retrieved October 8, 2014, from The Official Site of Dunn and Dunn Learning Styles: http://www.learningstyles.net/en/about-us

Felder, R. M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. *Journal of Engineering Education, 78*(7), 674-681.

Finin, T., & Drager, D. (1986, May 1). GUMS: A General User Modeling System. (R. Simpson, Ed.) *Proceedings of the workshop on Strategic computing natural language*, 224-230.

Fink, J. (2004). *User Modeling Servers - Requirements, Design, and Evaluation* (Vol. 277 of Dissertations in Artificial Intelligence). Amsterdam, Netherlands: IOS Press.

Fosler-Lussier, E. (1998). *Markov Models and Hidden Markov Models: A Brief Tutorial.* Technical Report TR-98-041, International Computer Science Institute, USA.

Fröschl, C. (2005). *User Modeling and User Profiling in Adaptive E-learning Systems.* Master Thesis, Graz University of Technology, Austria.

Heckmann, D. (2005). *Ubiquitous User Modeling.* Universität des Saarlandes. Künstlichen, German: Volume 297 Dissertationen zur Künstlichen Intelligenz, IOS Press.

Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., & Wilamowitz-Moellendorff, M. v. (2005). Gumo – The General User Model Ontology. In L. Ardissono, P. Brna, A. Mitrovic, L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *User Modeling 2005, Proceedings of the 10th International Conference, UM 2005* (pp. 428-432). Edinburgh, Scotland, UK: Springer Berlin Heidelberg.

Honey, P., & Mumford, A. (2000). *Learning Styles Helper's Guide.* Maidenhead, Berkshire, England: Peter Honey Publications Limited.

Karampiperis, P., & Sampson, D. (2005). Adaptive Learning Resources Sequencing in Educational Hypermedia Systems. (C.-K. Looi, & Kinshuk, Eds.) *Educational Technology & Society, 8*(4), 128-147.

Kay, J. (1995). The um Toolkit for Cooperative User Modelling. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction, 4*(3), 149-196.

Kay, J., Kummerfeld, B., & Lauder, P. (2002). Personis: A Server for User Models. In P. De Bra, P. Brusilovsky, & R. Conejo (Ed.), *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH '02)* (pp. 201-212). Springer Berlin Heidelberg.

Kobsa, A. (1991, June 1). First experiences with the SB-ONE knowledge representation workbench in natural-language applications. (J. Doyle, Ed.) *ACM SIGART Bulletin - Special issue on implemented knowledge representation and reasoning systems, 2*(3), 70-76.

Kobsa, A. (2001, March 27). Generic User Modeling Systems. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction, 11*(1-2), 49-63.

Kobsa, A. (2007). Generic User Modeling Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: methods and strategies of web personalization* (Vol. I, pp. 136-154). Berlin, Heidelberg, New York: Springer Verlag.

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

Kofman, V. (2009, November 02). *An Introduction to Java Enterprise Portals and Portlet Development.* Retrieved July 03, 2015, from The Flagship of The Developer.com Network: http://www.developer.com/java/web/article.php/3846491/An-Introduction-to-Java-Enterprise-Portals-and-Portlet-Development.htm

Kolb, A. Y., & Kolb, D. A. (2005). *The Kolb learning style inventory–version 3.1 2005 technical specifications.* Boston, MA, USA: Hay Resources Direct 200 (Hay Group).

Maffioletti, S. (2001). Requirements for an Ubiquitous Computing Infrastructure. *Cultura Narodov Prichernomoria journal, 3*, 35-40.

Mayo, M. J. (2001). *Bayesian Student Modelling and Decision-Theoretic Selection of Tutorial Actions in Intelligent Tutoring Systems.* PhD Thesis, University of Canterbury, New Zealand.

Murphy, K. P. (1998). *A Brief Introduction to Graphical Models and Bayesian Networks.* Retrieved 2008, from Kevin P. Murphy's home page: http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html

Neapolitan, R. E. (2003). *Learning Bayesian Networks.* Upper Saddle River, New Jersey, USA: Prentice Hall.

Nguyen, L. (2009). A Proposal Discovering User Interests by Support Vector Machine and Decision Tree on Document Classification. *The International Workshop on Social Networks Mining and Analysis for Business Applications (SNMABA2009) in conjunction with The 2009 IEEE International Conference on Social Computing (SocialCom).* *4*, pp. 809-814. Vancouver, Canada: Computational International Conference on Science and Engineering 2009 (CSE ’09), IEEE Publisher. doi:10.1109/CSE.2009.112

Nguyen, L. (2009). Incorporating Bayesian Inference into Adaptation Rules in AHA architecture. *Proceedings of 12th International Conference on Interactive Computer aided Learning (ICL2009).* Villach, Austria: Kassel University Press, Kassel, Germany. Retrieved from http://www.icl-conference.org/dl/proceedings/2009/archive.htm

Nguyen, L. (2009). State of the Art of Adaptive Learning. In H. R. Arabnia, A. Bahrami, & A. M. Solo (Ed.), *Proceedings of The 2009 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE 2009). The 2009 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP’09)* (pp. 126-133). Las Vegas, Nevada, USA: CSREA Press USA. Retrieved from https://www.researchgate.net/publication/221186474\_State\_of\_the\_Art\_of\_Adaptive\_Learning

Nguyen, L., & Do, P. (2008). Learner Model in Adaptive Learning. *The 2018 World Congress on Science, Engineering and Technology (WCSET2008).* *35*, pp. 396-400. Paris, France: World Congress on Science, Engineering and Technology (WASET). Retrieved from https://www.researchgate.net/publication/282679719\_Learner\_Model\_in\_Adaptive\_Learning

Nguyen, L., & Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. (M. E. Auer, Ed.) *International Journal of Emerging Technologies in Learning (iJET), 4*(4), 41-45. doi:10.3991/ijet.v4i4.684

Nguyen, L., & Do, P. (2009). Learning Concept Recommendation based on Sequential Pattern Mining. In E. Chang, F. Hussain, & E. Kayacan (Ed.), *Proceedings of The 2009 Third International Digital Ecosystems and Technologies Conference (IEEE-DEST 2009)* (pp. 66-71). Istanbul, Turkey: IEEE Publisher. doi:10.1109/DEST.2009.5276694

Oracle. (2009). *The Java™ Tutorials.* Retrieved 2009, from Oracle Help Center website for Java Standard Edition: http://docs.oracle.com/javase/tutorial

Oracle. (n.d.). *Java language*. (Oracle Corporation) Retrieved December 25, 2014, from Java website: https://www.oracle.com/java

Orwant, J. L. (1991). *Doppelgänger: A User Modeling System.* Bachelor Thesis, Massachusetts Institute of Technology, USA.

Orwant, J. L. (1995). Heterogeneous Learning in the Doppelgänger User Modeling System. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction, 4*(3), 107-130.

Özsu, T. M., & Valduriez, P. (2011). *Principles of Distributed Database Systems* (3rd Edition ed.). Springer New York.

Paiva, A., & Self, J. (1995). TAGUS - A User and Learner Modeling Workbench. (A. Kobsa, Ed.) *International Journal of User Modeling and User-Adapted Interaction, 4*(3), 197-226.

Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (3rd Edition ed.). New York, USA: McGraw-Hill Higher Education.

Rich, E. (1979). User Modeling via Stereotypes. *Cognitive Science, 3*(4), 329-354.

Riding, R., & Rayner, S. (1998). *Cognitive Styles and Learning Strategies: Understanding Style Differences in Learning Behaviour* (illustrated, reprint ed.). London, England, UK: David Fulton Publishers.

Schmolze, J. G. (2001). *An Introduction to Hidden Markov Models.* Lecture Notes on course "COMP 232-Knowledge Based Systems", Department of Computer Science, Tufts University.

Stash, N. (2007). *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System.* Eindhoven University of Technology, Netherlands. ACM SIGWEB Newsletter.

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

W3C. (2000, May 8). *Simple Object Access Protocol (SOAP) 1.1*, 1.1. (D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, . . . D. Winer, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: http://www.w3.org/TR/2000/NOTE-SOAP-20000508

W3C. (2001). *W3C semantic web activity.* Retrieved 2009, from W3C website: http://www.w3.org/2001/sw

W3C. (2001, March 15). *Web Services Description Language (WSDL) 1.1*, 1.1. (E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: http://www.w3.org/TR/2001/NOTE-wsdl-20010315

W3C. (2004, February 11). *Web Services Architecture*. (H. Haas, A. Brown, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

W3C. (2008, November 26). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 1.0. (T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: http://www.w3.org/TR/2008/REC-xml-20081126/

W3C. (2010, November 23). *XHTML™ 1.1 - Module-based XHTML - Second Edition*, 1.1. (S. McCarron, M. Ishikawa, M. Altheim, S. McCarron, Editors, & World Wide Web Consortium) Retrieved September 28, 2014, from W3C website: http://www.w3.org/TR/2010/REC-xhtml11-20101123

W3Schools. (1999). *HTML(5) Tutorial.* Retrieved 2014, from W3Schools website: http://www.w3schools.com/html

W3Schools. (n.d.). *AJAX Tutorial.* Retrieved July 03, 2015, from W3Schools website: http://www.w3schools.com/ajax/

Wikipedia. (2014, August 9). *Common Object Request Broker Architecture*. (Wikimedia Foundation) Retrieved September 16, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Common\_Object\_Request\_Broker\_Architecture

Wikipedia. (2014, December 7). *Daemon (computing)*. (Wikimedia Foundation) Retrieved December 25, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Daemon\_(computing)

Wikipedia. (2014, September 14). *Hypertext Transfer Protocol*. (Wikimedia Foundation) Retrieved September 15, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Hypertext\_Transfer\_Protocol

Wikipedia. (2014, September 12). *Java Servlet*. (Wikimedia Foundation) Retrieved September 15, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Java\_Servlet

Wikipedia. (2014, September 11). *Learning management system*. (Wikimedia Foundation) Retrieved September 18, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Learning\_management\_system

Wikipedia. (2014, September 12). *Lightweight Directory Access Protocol*. (Wikimedia Foundation) Retrieved September 16, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Lightweight\_Directory\_Access\_Protocol

Wikipedia. (2014, September 21). *Lisp (programming language)*. (Wikimedia Foundation) Retrieved October 6, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Lisp\_(programming\_language)

Wikipedia. (2014, September 29). *Myers-Briggs Type Indicator*. (Wikimedia Foundation) Retrieved October 8, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Myers-Briggs\_Type\_Indicator

Wikipedia. (2014, August 28). *Ontology (information science)*. (Wikimedia Foundation) Retrieved September 17, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Ontology\_(information\_science)

Wikipedia. (2014, September 23). *Prolog*. (Wikimedia Foundation) Retrieved September 30, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Prolog

Wikipedia. (2014, October 5). *Web page*. (Wikimedia Foundation) Retrieved October 7, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Web\_page

Wikipedia. (2014, October 6). *Zebra*. (Wikimedia Foundation) Retrieved October 7, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Zebra

Witkin, H. A., Moore, C. A., Goodenough, D. R., & Cox, P. W. (1977). Field-dependent and Field-independent Cognitive Styles and Their Educational Implications. (S. Messick, Ed.) *Review of Educational Research, 47*(1), 1-64.