

# Student Modeling in Design Pattern ITS

Zoran Jeremić and Vladan Devedžić

FON – School of Business Administration, University of Belgrade,  
Jove Ilića 154, POB 52, 11000 Belgrade, Serbia and Montenegro  
jeremycod@yahoo.com, devedzic@galeb.etf.bg.ac.yu

**Abstract.** This paper presents Student model implementation of Design Pattern's intelligent tutoring system. To our knowledge there hasn't been realized at least one intelligent tutoring system for learning Design Patterns. Some software tools, like Rational XDE have support for Design Patterns. There are many approaches and technologies for student modeling, but choosing the right one depends on the information that model should contain. In suggested solution, student model is created by using model template which is filled in with new attribute values. The model represents a global solution which can be applied in other domains and also supports simultaneous user exploitation.

## 1 Introduction

Intelligent Tutoring Systems (ITS) are advanced computer-based instructional systems that have separate knowledge bases, for instructional content, which specify what to teach, and for teaching strategies, which specify how to teach [1].

Traditional ITS usually contains four components: (1) the domain module, containing the structure of the domain and educational content, (2) the student model, which has information concerning the user (3) the pedagogical module, which encompasses knowledge regarding the various pedagogical decisions (4) Graphical user interface (GUI) that enables communication between system and student [2].

The purpose of this paper is to present an ITS system for learning Design Patterns. The system gradually introduces student with the concept of design patterns and describes most frequently used classes of patterns. The primary objective of the work presented in this article is to describe student model used in Design Pattern ITS. A student model must contain, maintain and update information such as "description of the student's knowledge, learning style and experience" [3].

The issue imposed in software realization for learning the Design Patterns, is the way of organization of learning process. In the course of realization of this problem, this project should be supported by the "Design Patterns" book [4]. The authors divided this book into 5 chapters, first two of which give an account of the Design Patterns in general, while the further three make the Design Patterns` catalogue grouped per types into creational, structural and behavioral patterns.

During interaction of a user with the system, the system is permanently assessing skill of a user, same resulting in change of student's characteristics in the student's model.

## 2 Design Pattern ITS Architecture

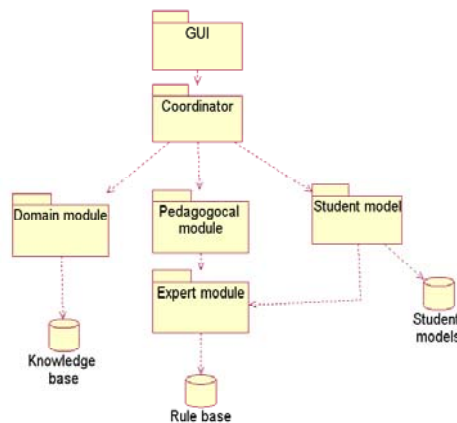
Fig. 1 depicts system's architecture. System basic components are: Pedagogical model, Expert model, Student model, Domain model, Coordinator and GUI.

*Pedagogical model* – The pedagogical model provides the knowledge infrastructure in order to tailor presentation of the teaching material according to the user model. In a specific learning session, the pedagogical model must perform the following tasks: (1) select a concept to teach, (2) select a teaching method, (3) select the course unit to be presented, (4) evaluate the user's performance (5) provide a feedback to the student [1].

*Expert model* – Expert model uses Jess (Java Expert System Shell) as a rule-based inference engine. Jess has capacity to “reason” using knowledge that is supplied in the form of declarative rules. Jess uses the Rete algorithm to process rules. When rules are examined by the inference engine, actions are executed if the information supplied by the pedagogical model satisfies the conditions in the rules. Pedagogical model uses these actions for making decisions in curriculum sequencing and evaluating student model.

*Domain model* - Domain model contains knowledge regarding the subject being taught as well as the actual teaching material. A course is constructed by creating instances of ontology classes and setting relations between instances, building semantic networks of concepts, units, fragments and tests. Ontology classes and instances are described textually in a XML document [5].

*Coordinator* – Coordinator has the role to control the functionality of the whole system. It interacts with the other components of the ITS calling the inference engine of the expert model whenever it is necessary. All communications between components in the system depends on this component. It also provides multiuser session.



**Fig. 1.** Architecture of the system - basic system components and their relations

GUI – Student interact with the system by using Web browser. Course unit presentations are defined by creating a common template for each class of the ontology (concept, unit, fragment, test). Templates are defined using a textual language based on Java Server Pages, that allows insertion of the parts of the chosen course unit into HTML code. Student interaction with the system is implemented by using HTTP protocol. System uses HTML compatible Web browser on the client side, and Tomcat 4.0 Web Server as JSP container on the server side.

### 3 Student Modeling

The student’s model is used to keep data on the user, being essential for system operations which adapt instructions material to student’s characteristics [6]. The student’s model comprises models of system’s users and mechanisms for creating these models [1].

Data on a student are called attributes or characteristics of student. The model may have any number of characteristics depending on requirements of the system. In this paper, three basic categories of student’s characteristics are being used (see figure 2):

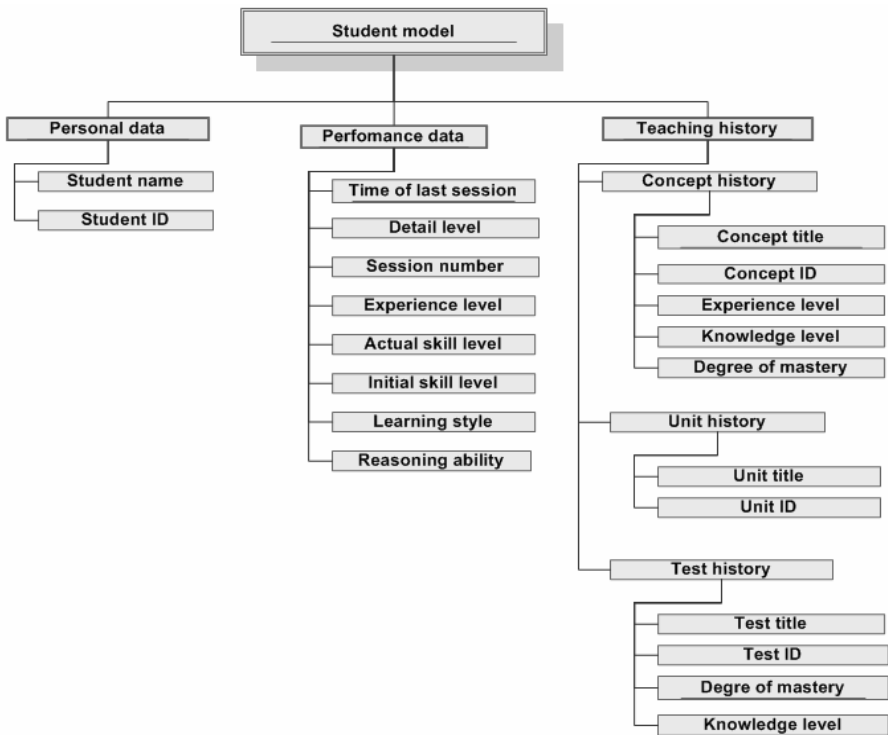


Fig. 2. Student model attributes grouped into categories

- (1) personal data - These data refer to personal characteristics of student (name, ID, e-mail ...).
- (2) performance data - Characteristics representing cognitive and individual characteristics of a student and long-term characteristics generally.
- (3) overlay data - They define knowledge of the field being studied and represent attributes related to the element of learning in domain model. These characteristics are connected to the covered chapter (teaching history), but they are also used to update the overall assessment of a student.

There is a number of techniques for student's modeling in various books, and the following three are the most used ones [7]:

- (1) Stereotype model - A new student is categorized under one of the given stereotypes, and the system gets his performances adapted on the basis of a category attributed to a student.
- (2) Overlay model - The model of student's skill is being changed during student's covering of instructions program. During session the system interacts with a student, and the results of these interactions affect user's performances in the student's model.
- (3) Combination model - This model represents the combination of the previous two models. At the beginning of the session, user chooses one of stereotypes that suits most his actual skill of the field being studied, and during the session the system makes judgments of a user and makes correction of the student's model on the basis of same.

The student's combination model is used in the proposed solution. In the course of registration of a student, the system creates the student's model to be filled up with data placed in an XML document which the system saved during the preceding session, or, if a new user is in question, the XML document is used which represents the model for all users registering for the first time. More detailed description of this method of creation of the student's model is given in [3]. On the basis of the initial interaction of a user with the system, effected at the beginning of the first session, the system classifies the user into one of the following categories: beginner, user of intermediate skill, advanced user, i.e. it assigns him one of stereotypes. Learning session is going on in compliance with the assigned stereotype until the completion of the first concept, when testing of students is made. After testing of student, pedagogical module, which makes assessment of a student, introduces "actual skill level" characteristic obtained on the basis of student's foreknowledge and the assessment of the first concept.

Further course of the session develops on the basis of this student's characteristic, same being permanently changed and updated [8]. Apart from this student's characteristic, other student's characteristics, important for the course of the session, are established: learning style, which refers to a learning style which suits student most, detail level, desirable level of details.

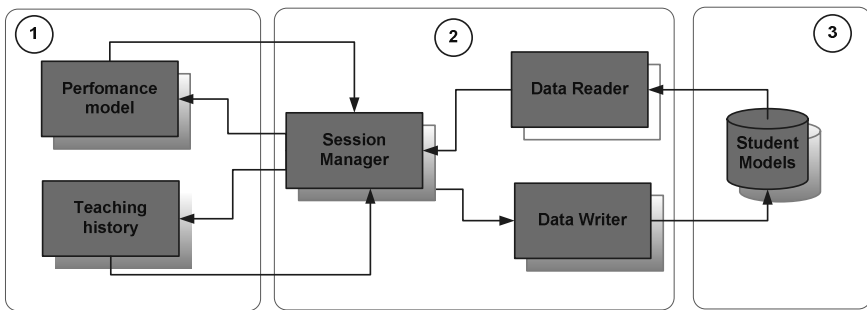
Calculation of values of attributes is made by applying group of rules and simple functions from pedagogical module to the group of parameters which the system gets automatically and updates same in the course of each session. At the end of the

session, the student's model is recorded in XML document and read in again at the beginning of the next session.

## 4 The Components of the Student Model

While classifying a student in components, it is necessary to analyze what kind of decisions are made in the Design Patterns ITS and what kind of information are needed to support these decisions. Tutor must take decisions, such as: selection of appropriate issue for a student, selection of appropriate learning method, making curriculum, etc.

Several components in the pedagogical module have a task to take such decisions, during which they communicate with the student's model in order to get relevant information. Two components of the student's model are in charge of keeping data on a student (see figure 3):



**Fig. 3.** Student model component's: 1-Student model, 2-Student model creation and management components, 3-Student models database

- (1) Performance model - This component keeps data in connection with assessment of the overall student's skill, as well as data concerning student's foreknowledge, learning style, etc.
- (2) Teaching history model - First of all, this component is intended for keeping records on the work of the system during the session, as well as on student's skill of teaching units (overlay data) [9]. This component copies skill domain structure, i.e. a separate instance is created for each concept, teaching unit and test, which keeps information on it.

Apart from the stated components, the student's model comprises another 3 components which take part in creating the student's model (see figure 3):

- (1) Session Manager - The component of the student's model which manages with all other components and makes their coordination possible.
- (2) Data Reader - This component enables reading-in of data on a student, located in XML document. It is based on SAX Parser, and it is called for at the beginning of the session, immediately upon creation of the student's model.

- (3) Data Writer - XML Writer which enables reading-in of data from student's model to XML document at the end of learning session.

### 5 Analysis of the Proposed Solution

Upon analyzing the proposed solution, it can be concluded that the student's model is functionally fully independent from the other components of the system. Applying the Facade project pattern, the interface, through which all communications of the other components with the student's model are done, is realized. Three most important functions which are realized by using these communications are as follows:

- (1) the student's model updating,
- (2) use of previous values of the student's model attribute for assessment of student's characteristic,
- (3) follow-up of the system's work history.

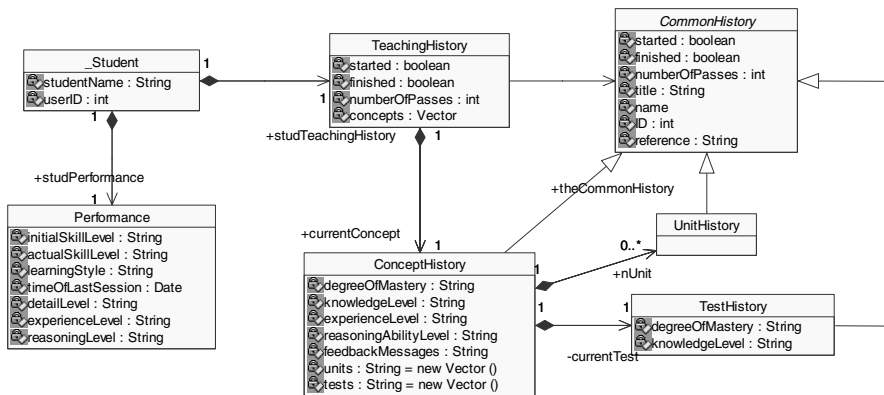


Fig. 4. Simplified class diagram of the student model (the Student model creation components have been left out)

In Figure 4 there is a simplified class diagram of the student's model. It can be seen that the student's model has diversified structure at the root of which there is a *Student* class, the instance of which is created for each student separately. It includes *Performance* class, which keeps long-term parameters of a student and *TeachingHistory* class, which copies structure of skill domain. In other words, a new instance of *ConceptHistory* class is created for each concept within the skill domain and placed into Vector type object which is located in *TeachingHistory* class object. *UnitHistory* and *TestHistory* classes represent teaching units and tests for skill testing within the concept.

Apart from the basic attributes for identification of a certain instructions material, all classes, covered by *TeachingHistory*, also comprise parameters which register the system covering through the given instructions material during interaction with a student. In such a way, the system memorizes that the student mastered specific

instructions material and in the next session it sends him back to the point where the previous session was terminated.

The specific quality, which distinguishes the proposed solution from the other solutions of student's model, is the way in which keeping data on a student is realized. In other words, for each student there is a separate XML document created during registration of a user. Usage of one XML document for all students would be an inefficient solution, since only the data on one user could be accessible in one session.

## 6 Conclusion

A flexible approach to student modeling, using combination of stereotype and overlay techniques, is represented in this paper. Using special XML documents to keep data on each student and SAX parsers to parse data into model's structure, the efficient student's model is created, the main advantage of which, compared to the other similar papers, is an universal approach to creation of a model. The student's model keeps all essential data on a student which are necessary to other components for realization of learning session. It allows for efficient management and updating of these data.

The solution of the student model, proposed in this paper, is not strictly related to the field used in this paper. Such a solution may be applied at any ITS without changes or with a few changes depending on requirements of pedagogical module.

This system opens numerous possibilities for its upgrading. One of the planned upgrading of this system in further versions is creation of Web application, which will make simultaneous work of great number of users possible.

## References

1. Prentzas, J., et.al.: A Web-Based Intelligent Tutoring System Using Hybrid Rules as Its Representational Basis. In Proc. of 6th Intern. Conf., ITS, France (2002) 119-128.
2. Bunt, A., Conati, C.: Probabilistic Student Modeling to Improve Exploratory Behaviour. Jour. of User Mod. and User-Adapted Inter. 13(3) (2003) 269-309.
3. Miskelly, T.: Interactive Student Modeling. In Proc. of the 36th Ann. ACM SE Reg. Conf., USA (1998) 88-94.
4. Gamma, E., et.al.: Design Paterns –Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, USA (1995).
5. Macias, J.A., Castels, P.: Aaptive Hypermedia Presentation Modeling for Domain Ontologies. In Proc. of 10th Int. Conf. on HCI, USA (2001) 710-714.
6. Devedzić, V.: Knowledge Modeling – State of the Art. Integ. CAE, Vol.8, No.3 (2001) 257-281.
7. Conlan, O.: Novel Components for supporting Adaptivity in Education Systems – Model-based Integration Approach. In Proc. of the 8th ACM Int. Conf. on Mult, CA, USA (2000) 519-520.
8. El-Sheikh, E., Sticklen, J.: Leveraging a Task-specific Approach for Intelligent Tutoring System Generation: Comparing the Generic Tasks and KADS Frameworks. In Proc. of 12th Int. Conf. on IEA of AI and ES, Egypt (1999) 809-819.
9. Yuijian, Z., W.Evens, M.: A Practical Student Model in an Intelligent Tutoring System. In Proc. of the 11th IEEE Int. Conf. on Tools with AI, USA (1999) 13-18.