

AN INTELLIGENT TUTORING SYSTEM FOR LEARNING DESIGN PATTERNS

ZORAN JEREMIĆ, VLADAN DEVEDŽIĆ, DRAGAN GAŠEVIĆ
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, gasevic@yahoo.com

The paper describes design of the Design Pattern ITS system, an intelligent tutoring system for learning Design Patterns. The basic idea of this system is a systematic introduction into the concept of most frequently used classes of patterns. An individual course is generated automatically for a given teaching goal and is dynamically adapted at run time to the student's individual progress and preferences according to the teaching expertise. The system provides explicit support for adaptive presentation constructs, and admits external navigation mechanisms and user model update strategies. At this point the system has been implemented and experimented in the domain of integration of elementary

1. Introduction

Design Pattern is an intelligent tutoring system (ITS) for learning design patterns. The Tutor gradually introduces student with the concept of design patterns and describes most frequently used classes of patterns. The issue imposed in course realization of this problem, the "Design Patterns" book [1] should support this course. The course for learning Design Patterns is divided into three main sections, creational patterns, structural patterns and behavioral patterns.

The ITS system for learning the Design Patterns, described in this paper, makes tutorial model of learning the Design Patterns possible, as well as independent study of patterns for more advanced students. The system provides an intelligent representation of educational material adjusted to student performance, such as degree of backward knowledge, desirable detail level, assessments of the system on the level of student's acquaintance with the matter being currently taught, as well as with the entire material.

During interaction of a user with the system, the Tutor monitors the performance of the student, same resulting in change of student' characteristics in the *Student model*. These assessments are provided through interaction of a user with the system by way of tests and solving problems. Assessments obtained through such interactions are compared with the other parameters and assessments in order to get the final judgment of a student.

Figure 1 depicts the system's architecture, composed of the Pedagogical module, Expert module, Student model, Domain model, Coordinator and GUI [2].

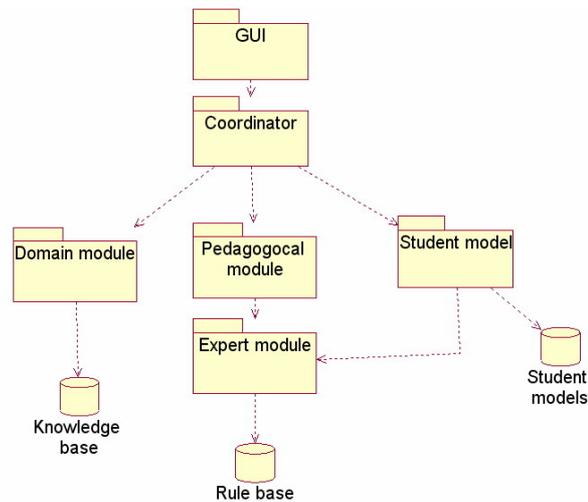


Figure 1. Design Pattern system architecture – main components of the system and relations among them

Pedagogical module provides the knowledge infrastructure necessary to tailor the presentation of the teaching material according to the student model. *Expert module* uses Jess (Java Expert System Shell) as a rule-based inference engine. *Pedagogical module* uses the *Expert module* for making decisions in curriculum sequencing and evaluating the student model. *Domain model* contains the knowledge about design patterns and the actual teaching material. *Coordinator* controls the functionality of the whole system. We use an HTML-based *GUI* on the client side and Tomcat 4.0 Web Server as JSP container on the server side.

2. Student modeling

Student model stores details about the student's current problem-solving state and long-term knowledge progress, essential for adapting the material to the student's characteristics (attributes). In this paper, three categories of student's characteristics are considered [3]:

- 1/ Personal data - the student's personal characteristics (name, ID, e-mail ...).
- 2/ Performance data - the student's cognitive and individual characteristics, as well as other general long-term characteristics.
- 3/ Overlay data - the current level of mastery of design patterns and attributes related to the corresponding elements in the domain model.

There is a number of techniques for modeling student; the most frequently used ones are overlay model, stereotype model, and combination model [4]. We used the combination model in the *Design Pattern* system.

When the student registers to the system for the first time, the newly created student model is initialized with default values from a stereotype. It is the system that selects the stereotype, based on the student's initial interaction with the system. *Design Pattern* gradually introduces other characteristics into the student model based on the estimated student's knowledge, such as degree of mastery, experience level, learning style, detail level, etc.

Attribute values in the student model are calculated by applying a dedicated group of rules and simple functions from *Pedagogical module*. The values are updated throughout the session.

At the end of each session, the system stores the student model as an XML document. The next time the student logs onto the system, the data from the stored XML document are used to initialize the student model.

3. Components of the Student model

Several components in the *Pedagogical module* make decisions about the student model. The decisions are related to the selection of an appropriate topic for the student, selection of an appropriate learning method, curriculum sequencing, and so on. During such processes, the *Pedagogical module* communicates with the student model in order to get relevant information.

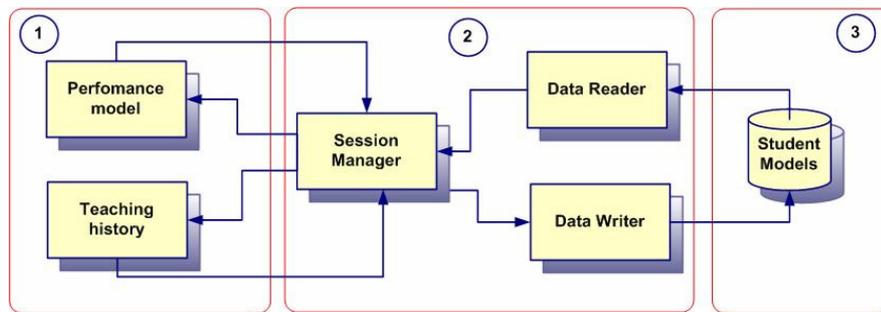


Figure 2. Student model component's: 1-Student model, 2-Student model creation and management components, 3-Student models database

The student model has two components (see figure 2):

- 1/ Performance model stores data related to the assessment of the student's overall skills, as well as data related to the student's previous knowledge, learning style, etc.
- 2/ Teaching history model keeps track of the material presented to the student during the session and the student's mastery of teaching units [5].

Other three components take part in creating the student model as well (see figure 2):

- 3/ Data Reader enables reading-in the student model data from the XML document; it is based on SAX Parser, and is invoked at the beginning of the session.
- 4/ Data Writer stores the student model to the XML document at the end of each learning session.
- 5/ Session Manager coordinates all other components.

4. Curriculum sequencing

Curriculum sequencing is one of the key functions in an intelligent tutoring system. In our approach, curriculum sequencing is realized by *Pedagogical module* component. *Domain model* organize instructional units into a hierarchy of topics, lessons, fragments and tests, which are related by prerequisite, part-of and other relationships. *Pedagogical module* of Design Pattern system is composed of three functional modules:

- Instructional Planner
- Feedback Mechanism
- Assessment Component.

The main purpose of the Instructional Planner is to provide system with instructional plan based on student knowledge of domain matter. Instructional plan is composed of variable sequence of instructional plan items, e.g. a list of topics, a list of lesson, etc. The Instructional Planner is composed of the following components:

- Instructional Plan Generator
- Discourse Planner.

Instructional Plan Generator is responsible for building the instructional plan that will be executed to produce the teaching session; in fact, this component is the real instructional planner.

Discourse Planner is responsible for execution of instructional plan. It also monitors the performance of the student in order to check if the plan is still appropriate. If Discourse Planner decides that the current plan is not valid for the

rest of the session then Instructional Plan Generator should consider the current situation and modify the plan in order to take into account the new condition.

If a student is having a session with Tutor for the first time, system gives him a set of exercises and tests before the tutoring process starts. In this way, the student model estimates the new student's background knowledge and determines initial values. After the initial assessment of the student, knowledge system runs in teaching mode. At the beginning of this mode, system develops an initial plan of teaching actions. The plan is based on the contents of the lesson being presented, the relevant pedagogical rules and the student model. The plan represents a detailed outline of the lesson presentation.

At the end of presented topic, the Tutor runs at examination mode. It generates exercises and tests for the student and assesses his knowledge. During the session, system observes and adapts the student progress with the generated course. If the student answers the test items correctly, he progresses along the course and no changes to the course are necessary. However, if the student fails to answer the test items correctly, the system must modify student performances. If the student's performance does not meet expectations, the course is dynamically re-planned. Through dynamic regeneration, each student is able to get a highly personalized course for his needs [6].

The system must memorize each activity of a user and the system, as well as all assessments of a student, updating the student model. These data may be used to prepare instructional plan, as well as to give advice and recommendations for further work [7].

5. Course material organization and delivery

Design Pattern Tutor dynamically generates course material by taking into account a specific learning goal and the initial level of the student's knowledge. During the session, system observes and adapts the student progress with the generated course. If the student's performance does not meet expectations, the course is dynamically re-planned. Through dynamic regeneration each student is able to get a highly personalized course for his needs.

Design Pattern Tutor provides control over presentation design, by using an explicit presentation separate from contents. Course unit presentation is defined by creating a common template for each class of the ontology (topic, lesson, and test). Templates are defined using a textual language based on Java Server Pages, which allows insertion of the part 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 [8].

During the session, system observes and adapts the student progress with the generated course. Domain model of Design Pattern Tutor is made up of concepts, which correspond to one pattern. Each concept is divided in units – the elementary

pieces of domain knowledge. There are a fixed number of units in particular concept, but the size of unit is not fixed. The system uses unit variants technique, which consists of keeping two or more alternative pages with adapted content, e.g. one for each knowledge level: beginner, intermediate and expert. Each unit has an arbitrary number of fragments – a chunk of information that should be presented to the user. The user model and the concept relationships of the domain model provide the information that allows the system to determine which chunk of information should be presented to the user. The chunk of information may also consist of fragment variants, i.e. fragments related by an “or” relationship.

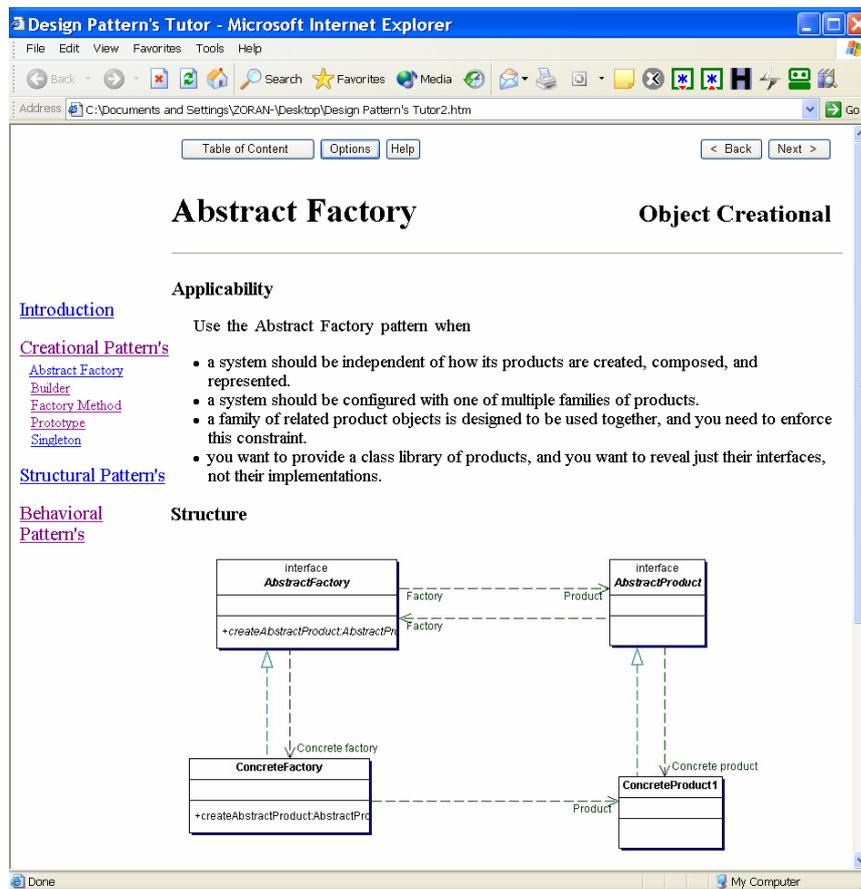


Figure 3. A Web page shows a screenshot of Design Pattern Tutor

The system provides students with two kinds of navigation through the course material (figure 3):

- Direct guidance – The student sees only one options to continue with the browsing activity i.e. just one button to navigate to the “next” page is displayed. The destination of the “best” link is determined by the system.
- Link removing – advanced students could choose which topics to learn by selecting appropriate link from content menu, but links that the system considers inappropriate are removed, i.e. they are not longer available. Anchors of these links are replaced by text.

The student has the option of letting the Tutor choose the next topic or choosing it himself. In both cases, the student must achieve sufficiently ready score for the topic. The topics are represented in a dependency graph, with links representing the relationship between topics, which include prerequisite and related topic. A student is ready to learn a topic only if he has performed sufficiently well on its prerequisites.

At the end of each topic student has to complete test. If the student fails to give correct answers, the Tutor must provide an alternative learning path to the student, such as a hint. If the learner tries to move on to a new topic before the Tutor feels that the student has explored the current topic sufficiently, the Tutor will generate a warning, suggesting better exploration of the current topic. These warnings also remind the student of the availability of hints. The student can choose either to follow the advice or to move on.

6. Conclusion

A prototype of Design Pattern system is implemented using Java so that the system can run on heterogeneous platforms. It consists of Pedagogical model, Expert model, Domain model, Coordinator, GUI.

The student model of the *Design Pattern ITS* is functionally fully decoupled from the other components of the system. This approach to student modeling uses a combination of stereotype and overlay techniques. Its main advantage compared to other similar models is a universal approach to model creation. The approach is not strictly related to design patterns as the domain of teaching/learning. It can be applied to ITS in any domain without changes or with a few changes, depending on the requirements of the pedagogical module.

Curriculum sequencing is based on execution of instructional plan generated based on student knowledge of domain matter. Course units are generated dynamically from domain model by using adaptive presentation templates.

There are several directions in future research and future development of the Design Pattern system:

- development of a case-based generator of new problems for student,
- development of a graphical authoring tool for domain model maintenance.

7. References

1. E. Gamma, et al, Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Company, (USA, 1995).
2. V. Devedžić, Knowledge Modeling – State of the Art, Integ. CAE, Vol.8, No 3, pp. 257-281, (2001).
3. Zoran Jeremić, Vladan Devedžić, Design Pattern ITS: Student Model Implementation, The 4th IEEE Int. Conf. on Advanced Learning Technologies, Finland, forthcoming, (2004).
4. O. Conlan, Novel Components for supporting Adaptivity in Education Systems – Model-based Integration Approach, In Proc. of the 8th ACM Int. Conf. on mult., pp 519-520, (CA, USA, 2000).
5. Z. Yuijian, M.W.Evens, A Practical Student Model in an Intelligent Tutoring System, In Proc. of the 11th IEEE Int. Conf. on Tools with AI, pp. 13-18, (USA, 1999).
6. P. Brusilovsky, J. Vassileva, Course sequencing techniques for large-scale web-based education, Int. J. Cont Engineering Education and Lifelong Learning, Vol. 13, Nos. 1/2, pp. 75-94, (2003).
7. J. Prentzas, I. Hatzilygeroudis, J. Garofalakis, A Web-Based Intelligent Tutoring System Using Hybrid Rules as Its Representational Basis, In Proceedings of the 6th Intern. Conference, ITS, pp. 119-128, (France and Spain, 2002).
8. J.A. Macias, P. Castels, Adaptive Hypermedia Presentation Modeling for Domain Ontologies, In Proc. of 10th Int. Conf. on HCI, pp. 710-714, (USA, 2001).