Modeling and diagnosing of student knowledge in Intelligent tutoring systems

Zoran Jeremić

Military academy, Belgrade, Serbia

jeremycod@yahoo.com

Abstract: This paper is primarily focused on design and implementation of the student model in DEPTHS (Design Pattern Teaching Help System), an intelligent tutoring system for learning software design patterns. There are many approaches and technologies for student modelling, but choosing the right one depends on the intended functionalities of the system the student model is going to be used in. Those functionalities often determine the kinds of information that the student model is going to contain. The student model used in DEPTHS is a result of combining two widely known modelling approaches, namely, stereotype and overlay modelling. The model is domain independent and can be easily applied in any other learning domain. The paper also presents a method based on fuzzy rules (i.e. combination of production rules and fuzzy logics) that we apply to assess students knowledge in DEPTHS.The results of the evaluation of DEPTHS in general and its student model in particular, are presented as well.

Keywords. Intelligent tutoring systems, student model, knowledge diagnosis, adaptive presentation, fuzzy logic.

1 Introduction

Intelligent Tutoring Systems (ITS) belong to an advanced generation of computer-based instructional systems that have separate knowledge bases for instructional content (specifying what to teach) and teaching strategies (specifying how to teach) [Prentzas, 02].

One of the key components of an ITS is the student model, which, among other student relevant data, keeps data about the student's knowledge of the subject domain. Depending on the design of the student model, these data might reflect the system's beliefs of the student's knowledge, the student's beliefs of his/her knowledge level of the domain, or the mixture of the two. Other relevant elements of the student model are, for example, data about the student's learning style, his/her learning behaviour, preferences, etc. In a typical ITS every student has a unique student model. It is the basis for decision making in the system and it evolves along the learning process.

DEPTHS (Design Pattern Teaching Help System) is an ITS for learning software design patterns. Software design patterns are conceptual models that describe successful solutions to common and recurring software design problems and can be applied over and over again when analyzing, designing, and developing software applications regardless of the particular context. The system gradually introduces the concept of design patterns and proceeds to teach a student most frequently used classes of patterns.

In this paper, we describe our work on design, implementation and evaluation of a performance-focused student model in DEPTHS. In particular, we detail the approach taken for modelling a student's knowledge as well as the technique used to diagnose the student's knowledge. We begin with an overview of the related work in Section 2, and then introduce DEPTHS and its architecture in Section 3. Section 4 describes the student model in DEPTHS and is followed by description of the diagnose procedure in Section 5. Section 6 describes the experiment performed during the evaluation of DEPTHS and gives a short description of the findings. The conclusions are presented in the final section.

2 Related works

In order to be able to adapt itself to each individual student, an educational system has to keep track of student's actions, and subsequently analyze them in order to deduce how the student's "state of mind" evolves. Based on this abstraction of the student's state the system is able to decide how to perform the adaptation. The representation of the student's state of mind is called a student model. Student model contains all information that the system knows about the student, such as his[1] learning style, background knowledge, job situation, colours or media preferences, and the like. The process of creating such a kind of representation of the student's characteristics in an ITS is often referred to as student modelling.

A student model is generally initialized either with default values or by asking the student to fill up a questioner. Thereafter, it is maintained by the system, although the student may be able to review and edit it at any time later. A diagnostic engine combines the student model with other models of the system to derive new "facts" about the student. Accordingly, the diagnostic engine can update the student model with the derived facts or initiate an action in the system.

The problem of initializing a student model (typically referred to as a 'cold-start problem' [Denaux, 05]) is often solved by assigning a student to one of the predefined groups of students, as is the case in the stereotypes-based approach. The notion of stereotypes was first introduced by Rich in [Rich, 79] in the system called GRUNDY. Since then, stereotypes have been used in many systems [e.g. Ballim, 91, Huang, 91, Murphy, 97]. An appealing property of stereotypes is that they enable a system to quickly start with a customized interaction with the student [Kay, 00]. However, they do not permit the formation of a student model focused on special individual characteristic(s). Another problem of the stereotype approach to user model acquisition is that it is quite inflexible – stereotypes are constructed in a hand-crafted way before real users have interacted with the system and they are not updated until a human does so explicitly.

A classical approach to student modelling is to first model the subject domain (e.g. using the knowledge of a domain expert), and subsequently use a subset of the domain model as the student model. That is how the well known *overlay model* is built. The advantage of this approach is that the same representation can be used for domain and student modelling. If the student model is a real subset of the domain model (i.e. the incorrect knowledge is not included in the model), we talk about a *strict overlay model*. On the other hand, in an *extended overlay model*, incorrect of faulty knowledge (also known as misconceptions) are part of the student model, as well. Examples of such systems are: AHM [Silva, 98], AHA (Adaptive Hypermedia Architecture) system [De Bra, 98], ISIS-Tutor [Brusilovsky, 94]. The major advantage of the overlay model is its simplicity; the elements of the model can be mapped directly on to the knowledge used to engineer the system. In practice, a combination of two or more methods is frequently applied; especially different methods are used for initializing and maintaining the student model. InterBook [Eklund, 97] is a well known example of such a practice. InterBook uses a concept based overlay model, whereas the student model is initialized using a stereotype model.

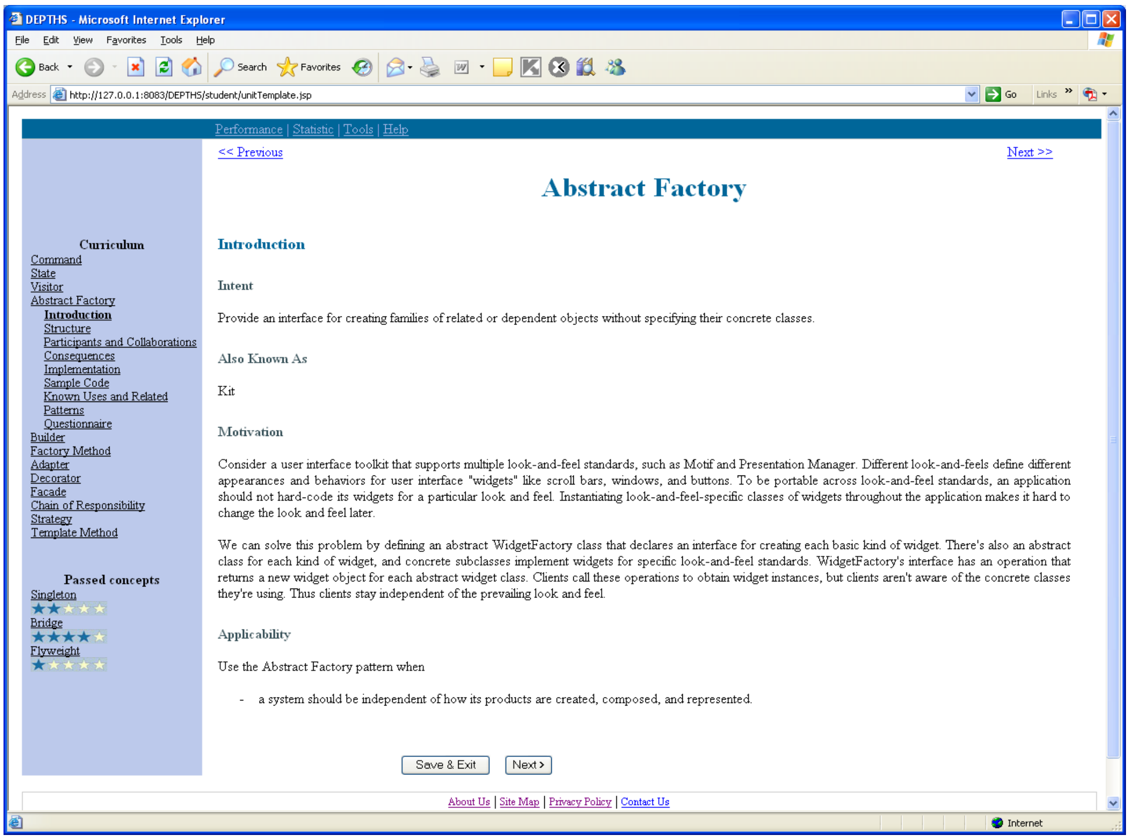
3 DEPTHS as an interactive learning environment

DEPTHS is a Web-based ITS for teaching software design patterns. The system provides a student with education material adjusted to the student's cognitive characteristics, background knowledge and performance in the subject domain. DEPTHS gradually introduces the concept of a design pattern and proceeds with providing tuition in the most frequently used classes of patterns.

A frequently encountered issue in teaching design patterns is related to the organization of the learning process itself. To face this concern, DEPTHS facilitates both tutorial mode and self-paced mode to learning design patterns. The student is free to decide whether to allow the system to guide him through the instructional content (tutorial mode) or to explore the content in his own preferred way (self-paced mode).

[1] Since gender-neutral language tends to be imprecise and/or cumbersome, throughout the paper we arbitrarily decided to use masculine pronoun for each generic reference to a student.

Figure 1 presents a snapshot of a typical content page in DEPTHS. The system utility menu is displayed at the top of the page. Content menu on the left side of the page provides student with navigational links to available units in the course material as well as with information about passed units. The rest of the page is reserved for the course unit content and navigational utilities.


DEPTHS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://127.0.0.1:8083/DEPTHS/student/unitTemplate.jsp
Go Links
Performance | Statistic | Tools | Help
<< Previous
Next >>
Save & Exit
Next >
About Us | Site Map | Privacy Policy | Contact Us
Internet

Abstract Factory

Curriculum
Command
State
Visitor
Abstract Factory
  Introduction
  Structure
  Participants and Collaborations
  Consequences
  Implementation
  Sample Code
  Known Uses and Related Patterns
  Questionnaire
Builder
Factory Method
Adapter
Decorator
Facade
Chain of Responsibility
Strategy
Template Method

Passed concepts
Singleton ★★☆☆☆
Bridge ★★★★☆
Flyweight ★☆☆☆☆

Introduction

Intent

Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

Also Known As

Kit

Motivation

Consider a user interface toolkit that supports multiple look-and-feel standards, such as Motif and Presentation Manager. Different look-and-feels define different appearances and behaviors for user interface "widgets" like scroll bars, windows, and buttons. To be portable across look-and-feel standards, an application should not hard-code its widgets for a particular look and feel. Instantiating look-and-feel-specific classes of widgets throughout the application makes it hard to change the look and feel later.

We can solve this problem by defining an abstract WidgetFactory class that declares an interface for creating each basic kind of widget. There's also an abstract class for each kind of widget, and concrete subclasses implement widgets for specific look-and-feel standards. WidgetFactory's interface has an operation that returns a new widget object for each abstract widget class. Clients call these operations to obtain widget instances, but clients aren't aware of the concrete classes they're using. Thus clients stay independent of the prevailing look and feel.

Applicability

Use the Abstract Factory pattern when

- a system should be independent of how its products are created, composed, and represented.

Figure 1. Web-based GUI of DEPTHS

DEPTHS provides a student with two kinds of navigation through the course material:

- Direct guidance – The student sees only one option to continue the browsing activity, i.e. just one button (next) is displayed to go to the next page (Figure 1). The destination of the 'next' button is determined by the system.
- Link removal – Advanced students can choose the concept to learn by selecting appropriate links from the Content menu. However, the links that the system considers inappropriate are removed, i.e. they are made unavailable.

Regardless the working mode, a student must achieve sufficient score for one concept in order to 'safely' move forward. If the student tries to switch to a new concept before exploring the current concept sufficiently, the system issues a warning, suggesting better exploration of the current concept. The student can choose either to follow the advice or to move on.

DEPTHS keeps track of every activity performed both by the student and the system itself, and stores the details of each observed activity in the student model. Additionally, all assessments related data are stored in the student model. The system uses these data to prepare instructional plans, as well as to provide a student with recommendations for further work [Prentzas, 02]. Furthermore, these data are used to provide an adaptive presentation of the instructional material, i.e. presentation adjusted to the student in terms of his background knowledge, desirable level of content details, and the preferred learning style.

DEPTHS also enables assessment of the student's mastery of design patterns, both in terms of the topic that has just been studied and the topics of the entire curriculum. The assessment methods include tests and solving pre-specified design problems. The student's knowledge is assessed at the end of each concept. Based on the assessment results and other related data, such as the time spent on solving the problem, the test difficulty level, etc., the system assesses the student's knowledge and updates the student model accordingly. If the student performed poorly, the system suggests an alternative learning path to him.

3.1 DEPTHS architecture

Figure 2 depicts the system architecture. *Domain module* contains the knowledge about design patterns and the related teaching material. It is designed as a network of concepts. Each concept corresponds to a single design pattern. The concepts form a dependency graph, where each link represents a relationship between the concepts it connects. For example, the *prerequisite* link signifies that a student is ready to learn the concept the link points to only if he has completed the concept the link originates from. Each concept is decomposed into units – elementary pieces of domain knowledge. Both the number of units and the size of each unit are arbitrary. DEPTHS uses unit variants technique, which essentially means keeping an alternative content unit (i.e. an HTML page) for each recognised knowledge level (e.g. beginner, intermediate and expert). Each unit has an arbitrary number of fragments – chunks of information that can be presented to the student. These chunks may appear in the form of fragment variants, i.e. fragments related by 'OR' relationship. The student model and the concept relationships of the *Domain module* provide the information that allows the system to determine which fragments should be presented to the student.

Pedagogical module provides the knowledge infrastructure needed for tailoring the presentation of the teaching material according to the student model. During a learning session, *Pedagogical module* performs the following tasks:

(1) provides a curriculum consisting of a concepts plan, a lessons plan, and a tests plan;
(2) selects an appropriate teaching method;
(3) decides how to present the teaching material to the student;
(4) evaluates the student's performance, and
(5) provides a feedback to the student [Bunt, 03].

Pedagogical module supervises the operation of the entire system. It interacts with all other components of DEPTHS and acts as an intermediary in communication between the system's components. In other words, all communication between the other system's components depends on this component. Finally, it coordinates simultaneous work of multiple users.

Both *Pedagogical module* and *Student model* use *Expert module* to make decisions related to curriculum sequencing, presentation planning, feedback provision, and evaluating and updating the student model. *Expert module* uses Jess (Java Expert System Shell) as a rule-based inference engine based on the Rete inference algorithm [Friedman-Hill, 03B].

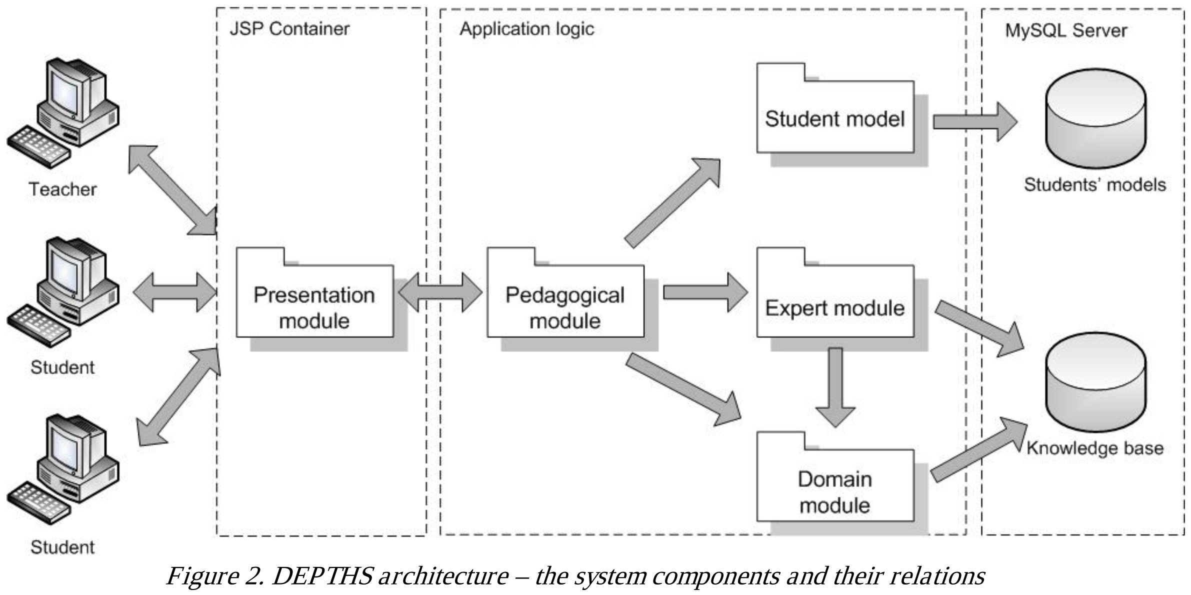


Figure 2. DEPTHS architecture – the system components and their relations

Students interact with the system by using a Web browser. A common presentation template has been defined for each component of the domain module (concept, unit, fragment, and test) and the system uses these templates when rendering the content of course units. The templates are defined using a textual language based on Java Server Pages, which allows insertion of parts of the chosen course unit into the HTML code. Students' interaction with the system is implemented using HTTP protocol. The system uses HTML compatible Web browser on the client side, and Tomcat 5.0 Web Server (http://tomcat.apache.org/) as JSP container on the server side.

4 Student modelling

Information about a student kept in his student model is the basis for providing adaptation of instructional material in any adaptive learning environment (e.g. an ITS) [Devedzić, 01]. A student model may hold any number and kind of student’s characteristics; the selection of characteristics to be considered is determined by the system requirements. Some of those characteristics are domain-independent whereas others are domain-specific; likewise, some of them are static while others are dynamic. Static features are set before the learning process takes place, in most cases using questionnaires, and they remain unchanged throughout the whole learning process. On the other hand, dynamic features are constantly updated during the learning session based on the collected data. These data come directly from the student’s interactions with the system, and includes, for example, the pages the student has visited, the time spent on each page, tests the student has taken, the student’s results on those tests and the like.

2.1 Student Model in DEPTHS

In DEPTHS, three basic categories of the students’ characteristics are considered (Figure 3):

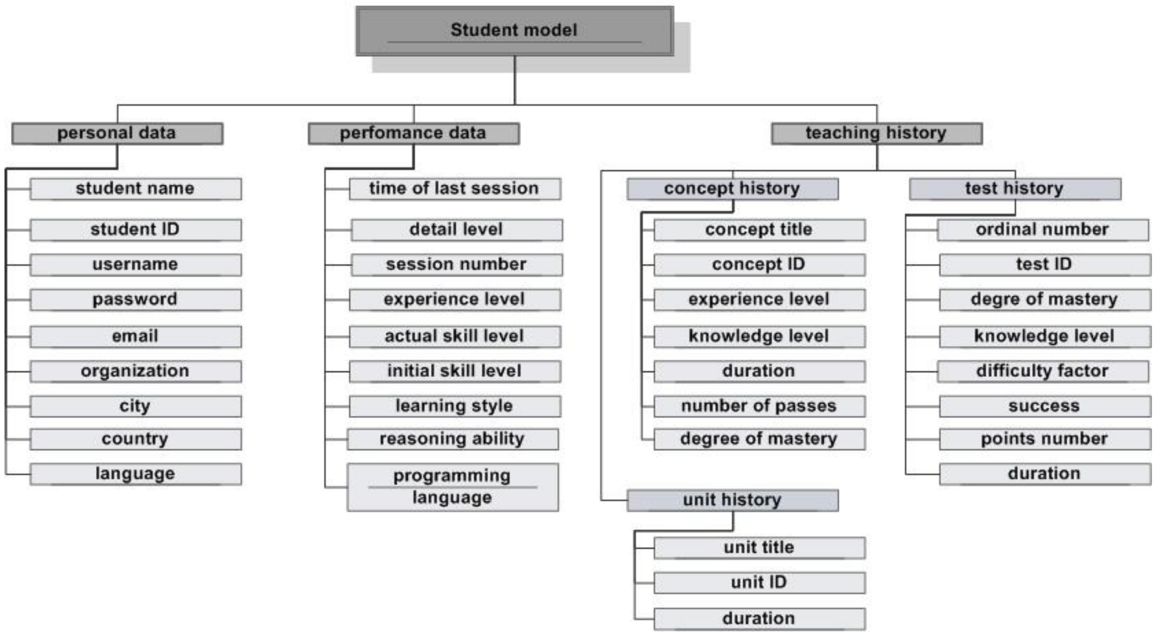


Figure 3. Student model attributes grouped into categories

1. Personal data – personal characteristics of the student (name, ID, e-mail ...). This information represents the static part of the student model, and is collected at the beginning of the learning session through the questionnaire. These data are not essential for the adaptation of the teaching material.

2. Performance data – cognitive and individual characteristics of the student. These data represent a mixture of static and dynamic data. Static data, such as the desired detail level (when presenting content), the experience level or the preferred programming language are collected during the

registration procedure (i.e. through the questionnaire). Dynamic data are derived from the learning sessions and are changed as the student progresses through the course material. Examples of dynamic data include: actual skill level, learning style, reasoning ability, etc. These data, both static and dynamic, are the most important data for one of the system's primary functions – adaptive presentation of the teaching material. These data represent the system's 'believes' of the student's traits. The quality of these believes directly affects the quality of the adaptation of the teaching material. If the system fails to precisely and exactly assess student's traits, it will fail to provide good teaching material, too. Some of the data in this group affect the content and other affect the presentation of the teaching material. Actual skill level, for example, indicates how much the student has learned. This trait is used by the system to provide the student with the lesson which is the most suitable for his knowledge level. On the other hand, programming language trait is used to present students with code examples written in the programming language they are familiar with, for example, if a student is familiar with Java programming language, he will get code examples in Java.

3. Teaching history – data related to the system's actions during the session. These data keeps track about everything that the student has done during the learning process. It also keeps tracks about the student's performance and other specific data related to the learning session, such as the time spent on solving tests, the student's success on a particular test, etc. These data are less important for adaptive presentation than performance data, but they are very important for reflective learning which often plays considerable role in a learning process. The system uses these data in order to provide the student with feedback about what he has done well and where he failed, what should be revised and how to make the learning process more successful. These data are very important for the teacher, too. The teacher can use the statistic data about his/her class success in order to, for example, find out which lessons are too difficult for students to understand, or which questions are overly complex for students.

The DEPTHS' student model belongs to the category of combined models. When a new student registers with the system, he has to select a stereotype according to his self-judgement of his mastery level of the subject domain. The following categories (i.e. stereotypes) are available: beginner, intermediate and advanced. Subsequently, the system creates the student's model with default values for the chosen stereotype. Learning session then proceeds in compliance with the assigned stereotype until the first concept is completed and the first test is conducted. Based on the test results, and previous knowledge about the student, the system determines the values for other attributes required for the overlay student model. The session then develops according to the new values of the student's performance.

4.2 The student model in use

One of the most important features of an ITS is its capability of adapting instruction to the student's needs. To accomplish this task, the ITS must know the student's knowledge state accurately. One of the most common solutions for diagnosing the student's level of knowledge is testing.

In DEPTHS, we use tests to assess students' knowledge of each domain topic. A test is created as a set of questions and exercises dynamically selected depending on the student previous knowledge level. The *Pedagogical module* (Figure 4), i.e. its *Diagnostic engine* to be more precise, is responsible for diagnosing student knowledge level based on the test results. It uses a set of pedagogical rules and domain knowledge in order to assess test results and diagnose the student's knowledge level based on these results. The diagnose procedure is described in more detail in Section 5. If the student fails to answer all the test items correctly, the system provides him with an alternative learning plan, i.e., it suggests lessons that the student needs to re-learn in order to reach better results. The student can choose either to accept the tutor's recommendations or to move on to the next concept. If he chooses to accept the recommended learning plan, he will be tested again at the end, and the *Diagnostic engine* will compare the new results with those previously obtained in order to diagnose the student's knowledge level for the learned concept, as well as the student's overall knowledge level of the domain. It also checks if the student's knowledge state has changed, and sends a message to the Adaptation engine to adapt the teaching material to the student's new performance level.

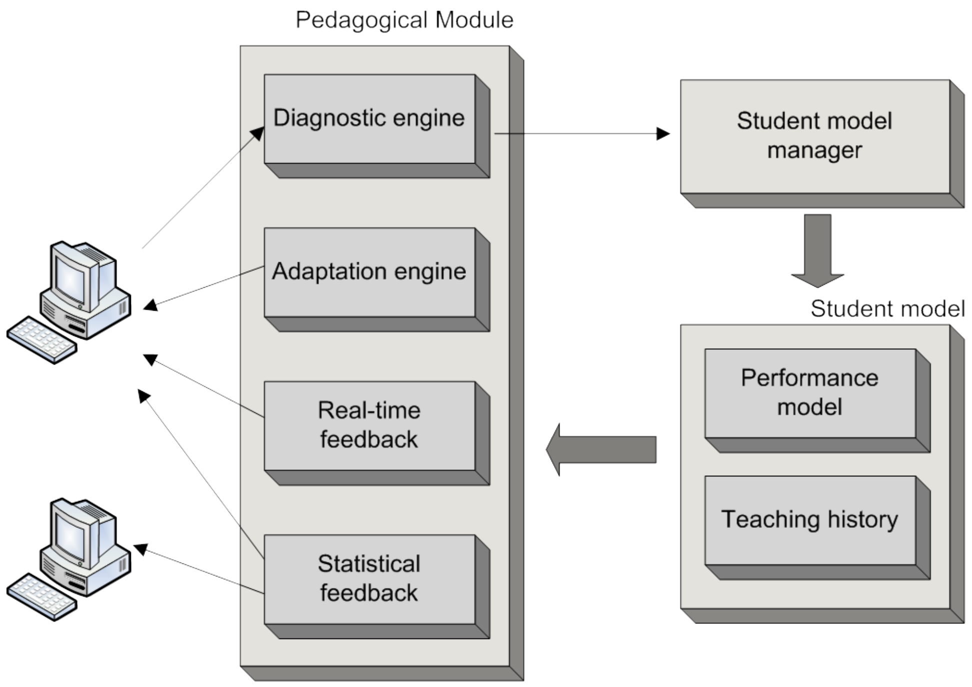


Figure 4. A closer look into the Pedagogical module and the Student model of DEPTHS

Information about the student’s knowledge level as well as other relevant information concerning the current learning session are stored in the student model and updated dynamically throughout the learning process. The *Student model manager* is responsible for handling this information.

The *Pedagogical module* uses some pieces of information from the *Student Model* to perform the adaptation of the teaching material and provide the student with adaptive navigation support (*Adaptation engine*), as well as to provide him with hints and recommendation for further work (*Real-time feedback*). Specifically the part of the *Student model* that is used for the adaptation purposes is named *Performance model* and it stores data related to the assessments of the student's overall knowledge, as well as data about the student's background knowledge, learning style, etc. The other relevant part of the *Student Model*, named *Teaching history* keeps track about the student’s progress so far. Even though the information it stores is not used for adaptation, it is very important for providing the student and the teacher with statistical analysis of the learning session in general, and some domain concepts, in particular. *Statistical feedback* engine in DEPTHS can compare performance of the particular student with performance of the group the student belongs to. It also provides the teacher with many specific analysing options such as, comparing the performance levels of different students, or different groups with respect to a specific domain concept or the whole course.

5 Student knowledge diagnosis

Student knowledge indicates the student’s level of understanding of the domain concepts. In DEPTHS, we used the overlay student modelling approach which actually means that we have associated each domain concept with the student’s knowledge status in relation to that concept. The computation of this status is based on fuzzy sets and certainty factor theories [Mendel, 95]. For each concept, a fuzzy value (ranging from 0 to 6) is calculated as the quantitative measure of system’s belief about the student’s level of understanding of the respective concept.

We use a collection of fuzzy membership functions and rules to reason about the student’s knowledge. The rules used in this reasoning have a form similar as the one in the following figure (Figure 5).

```
IF
      test difficulty is ‘easy’
      AND duration is ‘long’
      AND success is ‘good’
THEN
      knowledge is ‘enough’
```

Figure 5. Example of a rule used in reasoning process

In figure 5, ‘easy’ is one of the possible values of the linguistic input variable ‘test difficulty’ described by the membership function of the corresponding fuzzy set. The ‘test difficulty’ variable depends on the difficulty of each particular question of a test[2]. It is calculated as the average value of difficulty of all questions that the student has to resolve in the test. ‘Test difficulty’ has the range from zero (‘very easy’) to 100 (‘very difficult’), and is described by five fuzzy sets..

‘Long’ value, in the previous example, means that the student has spent more time solving the test than he should have spent. It is one of the possible values of the ‘duration’ variable. Specifically, three fuzzy sets are defined for the ‘duration’ variable: ‘short’ corresponds to fast resolving of a test, ‘middle’ for average speed of resolving a test, and ‘long’ for very slow test resolving. The value of the variable depends on the average time that the teacher has defined for solving a particular set of questions in a test.
‘Good’ is one of the possible values of the linguistic input variable ‘success’ and it depends on the correctness of the student answers on the test (i.e. the percentage of correct answers). It takes values in the range from 1 to 100, and is defined by five fuzzy sets.

‘Enough’ is one of the possible values that can be used to represent the system’s believes about the student’s degree of mastery of a concept. This value belongs to the ‘knowledge’ variable and takes values in the range from 0 to 6. It is defined by six fuzzy sets.

The logic for diagnosing the student’s knowledge for a concept is encoded in a set of fuzzy rules using the FuzzyJ Toolkit [Friedman-Hill, 03]. For our example rule, we would have one FuzzyRule holding three sets of FuzzyValues representing the antecedents (test difficulty, duration and success), and one set of FuzzyValues representing the conclusion (knowledge). The antecedent (the rule’s premise) describes to what degree the rule applies, while the conclusion (the rule’s consequent) assigns a fuzzy set to the given input combination. A rule is defined for every possible combination of antecedents that may occur. The inputs (i.e. antecedents) are combined logically using the AND operator. A firing strength for each output membership function is computed. These output values are combined (using a union of fuzzy values) to give a global fuzzy value in the process called global contribution. Finally, this global fuzzy value output must be converted to a crisp value, which is a representative value for the student knowledge. This process is called defuzzification [Mendel, 95]. The output value is passed to the student model as the system’s belief about the student knowledge of the concept being learned.

If the student fails to give all correct answers on the test, the system recommends him a learning path to follow in order to reach better results. If the student chooses to accept the advice, he is presented with some additional lessons to learn, and subsequently his knowledge is tested again. This time, the system selects only questions related to the student’s misunderstandings observed in the previous test. The new test is assessed in the same way as the previous one. We use the student’s results on all tests related to a specific concept when calculating the student’s overall knowledge of the concept. The formula we use for that purpose is as follows:

$$DM = \frac{\sum_{i=1}^{n} TK_i \cdot QN_i}{\sum_{i=1}^{n} QN_i}$$

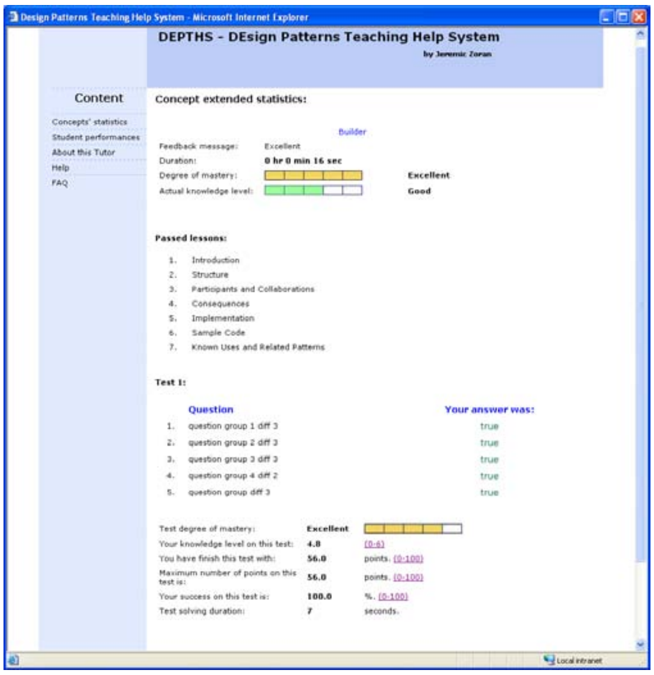
DM – the system’s belief about the student’s degree of mastery of a specific concept
i – ordinal number of the test for the current concept
TK – student knowledge shown at a particular test
QN – number of questions in a particular test

[2] The difficulty level of individual questions is defined by the author of teaching materials (e.g. a teacher).

Based on the concepts' assessments, the system creates the overall assessment about student's understanding of the domain. This value is calculated as the average value of all concepts degrees of mastery.

5.1 Reflective learning in DEPTHS

In order to provide better support for reflective learning, the system provides a student with a possibility to preview his performance data stored in the student model in the form of statistical analysis (Figure 6 and 6). Student can choose to preview the overall teaching history by selecting the page that contains main information about all passed concept. He can also take a closer look at each concept by selecting the page which provides details about a concept of interest. Such a page contains more details about each lesson learned, questions that were asked and the student's answers, time spent on solving tests, number of points on each test etc. (Figure 6) Based on these data, the student can see which concepts he did not pass well, and learn them again if he wishes.



DEPTHS - DEsign Patterns Teaching Help System

by Jeremic Zoran

Concept extended statistics:

Builder

Feedback message:	Excellent	
Duration:	**0 hr 0 min 16 sec**	
Degree of mastery:	🟨🟨🟨🟨🟨	**Excellent**
Actual knowledge level:	🟩🟩🟩⬜⬜	**Good**

Passed lessons:

1. Introduction
2. Structure
3. Participants and Collaborations
4. Consequences
5. Implementation
6. Sample Code
7. Known Uses and Related Patterns

Test 1:

	Question	Your answer was:
1.	question group 1 diff 3	true
2.	question group 2 diff 3	true
3.	question group 3 diff 3	true
4.	question group 4 diff 2	true
5.	question group diff 3	true

Test degree of mastery:	**Excellent**	🟨🟨🟨🟨⬜
Your knowledge level on this test:	**4.8**	(0-6)
You have finish this test with:	**56.0**	points. (0-100)
Maximum number of points on this test is:	**56.0**	points. (0-100)
Your success on this test is:	**100.0**	%. (0-100)
Test solving duration:	**7**	seconds.

Figure 6. Aiming to stimulate reflective learning, the system provides students with their performance records on a selected concept

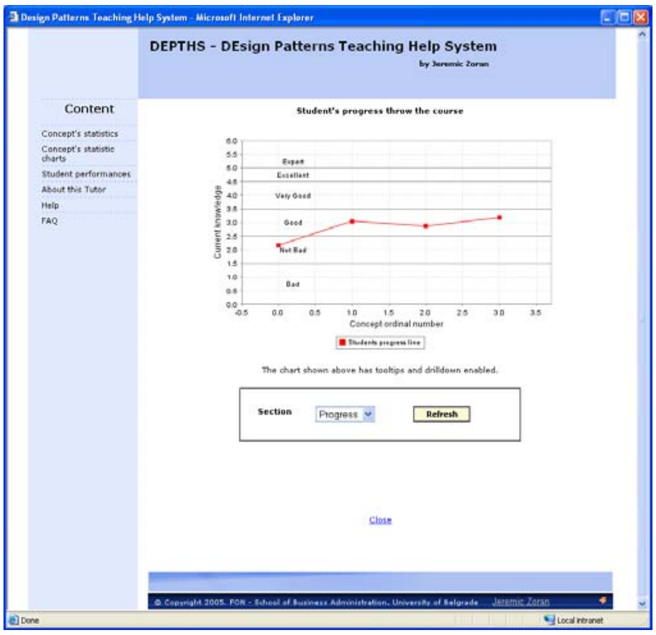

Design Patterns Teaching Help System - Microsoft Internet Explorer
DEPTHS - DEsign Patterns Teaching Help System
by Jeremic Zoran
Content
Concept's statistics
Concept's statistic charts
Student performances
About this Tutor
Help
FAQ
Student's progress throw the course
Expert
Excellent
Very Good
Good
Not Bad
Bad
Current knowledge
Concept ordinal number
Students progress line
The chart shown above has tooltips and drilldown enabled.
Section
Progress
Refresh
Close
© Copyright 2005. FON - School of Business Administration, University of Belgrade
Jeremic Zoran
Done
Local intranet

Figure 7. In order to foster reflective learning, DEPTHS provides a student with the information about his progress through the course

Statistical analysis provides a student with chart diagrams about his progress through the course material, as well as with the data about each concept separately (Figure 8). Student can choose to preview his own progress or to compare his progress with the group he belongs to.

6 Evaluation

Evaluation is a paramount success factor of any tutoring system. It provides relevant feedback for both system and course developers and is a quintessential part of quality assurance.
The evaluation of DEPTHS has been performed in 2005 in the context of course we teach at the Military Academy of Belgrade, Serbia. The focus of the evaluation was the learning effectiveness of the system as a whole, as well as the accuracy of the student knowledge diagnosis. In addition, we have evaluated students' attitudes towards learning with DEPTHS system.

The methods we used to test the system's effectiveness included:

- A survey aimed at capturing students' reactions to the training program, and
- An experiment set up with the objective of comparing pre-test and post-test results of an experimental group and two control groups.

The survey was conducted using an interview as a mean for collecting data about the students' satisfaction. The interview was designed so that reaction can be easily tabulated and manipulated by statistical techniques. Most of the questions were based on the Likert scale, but we also used free-form questions in order to encourage the participants to give additional comments not elicited by the provided set of questions.

Generally, DEPTHS received high marks from students. Some of the DEPTHS advantages, as reported by the students, include the organisation of the course material, the novelty of the online approach and convenient access to online research materials. The majority of the students felt that the DEPTHS system is very easy to use, and they have enjoyed very much learning with DEPTHS. They found that, one hour spent with DEPTHS is more useful than one hour in a traditional classroom. Furthermore, the feedback that the system provides to students is found as very useful. However, the students noted some negative aspects of the system as well. The major recognized drawback is the lack of collaborative learning support.

In order to evaluate the system’s effectiveness, we used a variant of ‘pre-test, post-test, control group – experimental group’ design. One experimental group and two control groups were involved in this experiment. All of the students were tested at the outset of the experiment to evaluate their knowledge in the subject matter. The pre-test was designed to test the student’s elementary knowledge in software engineering and design patterns. In addition, this test was aimed at identifying the students’ expectations from the course and its delivery model. Students in the experimental group were learning with the DEPTHS system, whereas students in the control groups were learning in the traditional way. After the course, all students had to take a test created by a teacher. This type of test (post-test) deduces overall judgements and criticism on the learning effectiveness. We performed a comparison of tests results in experimental and control groups in order to analyze the effect of the learning environment on students’ learning and overall performance. The findings indicate that DEPTHS had a beneficial effect on students’ motivation and led to a more rapid improvement in their performance over time.

For evaluation of the accuracy of the student knowledge diagnosis we compared the system’s assumptions about student’s knowledge with the information obtained through an external test designed by a group of domain experts. We were monitoring progress of each individual student by examining the student’s records (kept in his student models), and then comparing these values with external test results. In that way we were able to test the accuracy of the student model. We found, that most assumptions were in congruence with the test performance. However, we found that the external test shows slightly lower knowledge level. However, that difference was something we could have expected, because the results that a student shows immediately after learning a particular lesson (student model) are often better than results that he shows at the end of the whole course. The difference is greater for lessons that were learned at the beginning of the course and it was gradually decreasing towards the end of the course.

7 Conclusions

A flexible approach to student modelling, using a combination of stereotype and overlay techniques, is presented in this paper. We have also presented a fuzzy logics based method for assessing student knowledge that we applied in DEPTHS. One should note that the student model developed for the DEPTHS system is domain independent (i.e. it is not specifically aimed for the domain of design patterns). Such a model may be applied in any ITS (regardless of the subject domain) either in its present form, or with a few changes depending on the requirements of a specific ITS.

The flexible design of the DEPTHS system opens numerous possibilities for upgrading. One of the upgrades we are currently working on is the support for collaborative learning in DEPTHS, which will enable collaboration among students, as well as interaction between a student and a teacher. Another important upgrade is related to the student model itself and it is about making it easily interchangeable with other learning systems. To that end our intention is to make the model compliant with emerging standards for learner modelling, such as The PAPI Learner Standard (http://edutool.com/papi/), as well as to define it in OWL (Ontology Web Language, http://www.w3.org/2004/OWL/), the standard Semantic Web language.

References

[Ballim, 91] A. Ballim, Y. Wilks, Beliefs, stereotypes and dynamic agent modelling. User Modeling and User Adapted Interaction, vol. 1, No. 1, 33-66, 1991.

[Brusilovsky, 94] P. Brusilovsky, ISIS-Tutor: An Intelligent Learning Environment for CDS/ISIS Users. Proceedings of the Interdisciplinary Workshop on Complex Learning in Computer Environments (CLCE'94), Joensuu, Finland, 29-33, 1994.

[Bunt, 03] A. Bunt, C. Conati, Probabilistic Student Modelling to Improve Exploratory Behaviour, Journal of User Modeling and User-Adapted Interaction 13(3), 269-309, 2003.

[De Bra, 98] P. De Bra, L. Calvi, AHA: a Generic Adaptive Hypermedia System. Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98, Pittsburgh, USA . 20-24, 1998.

[Denaux, 05] R. Denaux, V. Dimitrova, L. Aroyo, Integrating Open User Modeling and Learning Content Management for the Semantic Web. In the Proceedings of the 10th International Conference on User Modeling, Edinburgh, UK, 9-18, 2005.

[Devedzić, 01] V. Devedzić, Knowledge Modelling – State of the Art. Integrated Computer-Aided Engineering, Vol.8, No.3, 257-281, 2001.

[Eklund, 97] J. Eklund, P. Brusilovsky, E. Schwarz, Adaptive Textbooks on the WWW. Proceedings of AUSWEB97 – The Third Australian Conference on the World Wide Web, Queensland, Australia, 186-192, 1997.

[Friedman-Hill, 03] E.J. Friedman-Hill, Jess in Action, Rule-Based Systems in Java. Sandia National laboratories, Manning Publications Co., Greenwich, CT, 2003.

[Friedman-Hill, 03B] E.J. Friedman-Hill, *Jess, The Expert System Shell for the Java Platform.* Sandia National Laboratories, Livermore, CA, 2003.

[Gamma, 95] E. Gamma, et.al., Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, USA, 1995.

[Huang, 91] X. Huang, G. McCalla, J.E. Greer, E. Neufeld, Revising deductive knowledge and stereotypical knowledge in a student model. User Modeling and User Adapted Interaction, vol. 1, 87-115, 1991.

[Kay, 00] J. Kay, Stereotypes, student models and scrutability. In Proceedings of the 5th International Conference on Intelligent Tutoring Systems, Lecture Notes in Computer Science, vol. 1839, Springer-Verlag, 19-30, 2000.

[Mendel, 95] J.M. Mendel, Fuzzy logic systems for engineering: A tutorial. Proceedings of IEEE, vol. 83, 345-377, 1995.

[Murphy, 97] M. Murphy, M. McTear, Learner modeling for intelligent CALL. Proceedings of the 6th International Conference on User Modeling, Springer, 301-312, 1997.

[Prentzas, 02] J. Prentzas, I. Hatzilygeroudis, J. Garofalakis, A Web-Based Intelligent Tutoring System Using Hybrid Rules as Its Representational Basis. In Proc. of 6th Intern. Conf. ITS, France and Spain, 119-128, 2002.

[Rich, 79] E. Rich, User modeling via stereotypes. Cognitive Science, vol. 3, 329-354, 1979.

[Silva, 98] D. Pilar da Silva, R. Van Durm, E. Duval, H. Olivie, Concepts and documents for adaptive educational hypermedia: a model and a prototype. Proceedings to the 2nd Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, USA, 20-24, 1998.