Elsevier Editorial System(tm) for Expert Systems With Applications
Manuscript Draft

Manuscript Number: ESWA-D-10-01092

Title: Student Modeling and Assessment in Intelligent Tutoring of Software Patterns

Article Type: Full Length Article

Keywords: Intelligent tutoring systems, student model, knowledge assessment, adaptive presentation, fuzzy logic

Corresponding Author: Dr Zoran D Jeremic, Ph.D.

Corresponding Author's Institution: Military academy

First Author: Zoran D Jeremic, Ph.D.

Order of Authors: Zoran D Jeremic, Ph.D.; Jelena Jovanovic; Dragan Gasevic

Student Modeling and Assessment in Intelligent Tutoring of Software Patterns

Z. Jeremić[a *], J. Jovanović[a], D. Gašević[b]

[a] *FON – School of Business Administration, University of Belgrade, POB 52, Jove Ilića 154, 11000 Belgrade, Serbia*
[b] *School of Computing and Information Systems, Athabasca University, Canada*

Received 15 May 2010;

Abstract

This paper presents the design, implementation, and evaluation of a 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 modeling, but choosing the right one depends on intended functionalities of an intelligent system that the student model is going to be used in. Those functionalities often determine the kinds of information that the student model should contain. The student model used in DEPTHS is a result of combining two widely known modeling approaches, namely, stereotype and overlay modeling. The model is domain independent and can be easily applied in other learning domains as well. To keep student model update during the learning process, DEPTHS makes use of a knowledge assessment method based on fuzzy rules (i.e., a combination of production rules and fuzzy logics). The evaluation of DEPTHS performed with the aim of assessing the system's overall effectiveness and the accuracy of its student model, indicated several advantages of the DEPTHS system over the traditional approach to learning design patterns, and encouraged us to move on further with this research.

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

1. Introduction

The major concern of today's software engineering education is to provide students with the knowledge and skill required for solving different kinds of software problems. The good news is that many of these problems have already been solved by other developers, and (re)using their experiences for solving our own problems could spare us a lot of time, and help us to develop well-designed solutions. These previous experiences are very often documented in a specific form and referred to as software patterns [1]. However, a bad news is that there are several different forms used for pattern representation, such as the Gang of Four form or the Alexandrian form [2]. Students need to be familiar with all these forms in order to be able to use patterns when working on software projects. Moreover, the increasing number of pattern collections spread all over the Web, makes it difficult to find appropriate one using regular search engines. Finally, "one-size-fits-all" approach used in online tutorials on software patterns is inappropriate for many students. Those tutorials are often too difficult for novice students and too simple and insufficient for more advanced students. More intelligent solutions, which provide adapted learning material, could solve these problems. Lessons tailored to the student's previous knowledge, using the (software pattern) form that the student is familiar with and offering examples in the programming language that the student knows well, seems to be a promising solution for managing the identified problems.

Intelligent Tutoring Systems (ITS) belong to an advanced generation of computer-based instructional systems that provide students with highly personalized learning experience, by adapting the content and its presentation to the student's needs and preferences. These features heavily rely on a Student Model, an obligatory component of any ITS, which, among other student relevant data, keeps data about the student's knowledge of the subject domain under study [3]. 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 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 learning behavior, 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 [4, 5].

Personalization of the learning process in an ITS is highly dependent on the quality of the information that represents the system's believes about a student's knowledge of the subject domain being studied [6]. This information is acquired through the assessment process, which is about inferring what the student knows from a set of observable facts. However, determining a student's knowledge is not a straightforward task, since it often depends on and is reflected through things that cannot be directly observed and measured. Even when substantial

* Zoran Jeremić, Tel: +381-11-3552-960; fax: +381-64-1485-328;
E-mail: jeremycod@yahoo.com

data about a student's behavior are available, knowledge assessment tends to be highly uncertain process as it is often questionable how to interpret the available data. Due to these open issues, the assessment of students' knowledge is still an open and challenging research task (see Section 2 for further details).

DEPTHS (Design Patterns Teaching Help System) is an ITS for learning software (design) patterns[*]. The system supports tutorial mode of learning the design patterns, as well as independent study of patterns for more advanced students. It performs an intelligent selection and representation of educational material adjusted to the characteristics of each particular students such as his background knowledge, performance in the domain of study, his learning style, the desired level of details, and preferred programming language. In the tutorial mode, DEPTHS gradually introduces the concept of design patterns and proceeds to teach a student the 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 modeling a student's knowledge as well as the rule-based technique applied for assessing 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 the description of the assessment 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. Background in Student Modeling and Assessment

In order to be able to adapt itself to each individual student, an ITS has to keep track of student's actions, and subsequently analyze them in order to deduce how the student's characteristics evolve over time. 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 characteristics in a point of time is called student model [8]. A student model contains all the information that the system knows about the student such as his learning style, background knowledge, job situation, colors 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 modeling.

A student model is generally initialized either with default values or by asking students to fill in a questionnaire. Thereafter, the student model is maintained by the system, although the student may be able to review and edit it later. The Assessment engine combines the student model with other models of the system to derive new "facts" about the student. Accordingly, the Assessment 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' [9]) 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 [10] in the system called GRUNDY. Since then, stereotypes have been used in many systems [11, 12, 13]. An appealing property of stereotypes is that they enable a system to start quickly with a customized interaction with the student [14]. 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 modeling is to first model a subject domain (e.g., by using the knowledge of a domain expert), and subsequently use a subset of the domain model as the student model. That is how well-known overlay student model is built [8]. The advantage of this approach is that the same representation can be used for domain and student modeling. If the student model is a real subset of the domain model (i.e., incorrect knowledge is not included in the model), we talk about a strict overlay model [8]. 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 [15], AHA (Adaptive Hypermedia Architecture) system [16], and ISIS-Tutor [17]. 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 [18] 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.

Recently, researchers have begun to adopt Semantic Web technologies in developing student models. Student models developed with a semantic web ontology language have the advantages of formal semantics, easy reuse, easy portability, availability of effective design tools, and automatic serialization into a format compatible with popular logical inference engines [19, 20, 21, 22]. A lot of research effort has already been put on development of ontology-based framework for sharing student profiles between different learning systems [23]. Furhtermore, a semantic learner model based on the FOAF ontology is proposed [24] in order to support automation of the process of grouping students while preserving the individual's personal needs and interests. However, even though these solutions rely on a more advanced knowledge representation technology, their approach to students'

[*] Design patterns are the most commonly used type of software patterns [7]. They typically show relationships or interactions between classes or objects in a system. DEPTHS is currently primarily focused on, but not limited to this type of software patterns.

knowledge (and/or competencies) modeling is often based on the well-known overlay model – students' knowledge (competencies) is expressed in terms of concepts of the domain ontology.

Though a lot of research has been dedicated to student modeling, only a few researchers have been dedicated to opening those models to students [25, 26, 27, 28]. An important reason for opening student model is to help students to better understand their learning and therefore enhance the learning process. It offers a source of information about their knowledge about the domain of study that is otherwise unavailable, encouraging them to reflect on their knowledge and on the learning process.

Besides the research in student modeling, considerable research work has been done over the past two or three decades on how to accurately diagnose a student's cognitive state. As stated by Ohlsson [29], diagnosis is "the process of inferring a student's cognitive state from his/her performance", and as such, is a central activity in any system which aims to build a dynamic student model. The term "diagnosis" has often been used synonymously with the term "assessment" to denote the process of building a representation of a student's knowledge. However, these terms have been distinguished from each other, describing two different aspects of the process of assessing a student's knowledge.

The problem of assessing a student's knowledge could be regarded as a process of estimating what the student actually knows about the subject being taught. It is usually performed through a set of quizzes generated by a tutor and solved by students. We use "student knowledge assessment" or just "assessment" term for this process. However, estimation of a student's knowledge about the domain matter is not as simple as rating how many correct answers the student had on a quiz. Many other factors influence the final judgment about his knowledge level, such as the difficulty level of a quiz, time spent to solve the quiz, student's motivation, etc. A part of the assessment process aimed at estimating a student's knowledge level of a specific domain based on the quiz results and diverse factors influencing the overall judgment, has to include advanced numerical techniques that have more rigorous mathematical foundation and could generate more predictable and accurate diagnoses. In this work, we used the term "student knowledge diagnosis" or just "diagnosis" to indicate this sub-process of the overall assessment process.

An assessment provides very detailed information about a student's competence at all points during instruction. Assessments are used to guide the selection of the next instructional actions. As the quality of the student model and the personalization of the instruction process depend on the quality of information gained from the assessment process, this process needs to be based upon more rigorous principles and its quality has to be constantly and carefully examined.

Many diagnostic techniques are described in the ITS literature [4]. However, most of them are based on two basic approaches for getting a set of attributes about the student's knowledge. One is to infer them, given a set of test items and data about student's performance on the items. The other one is to infer them based on the student's overall interactions with the system (i.e. followed links, used help…). Most of these techniques produce very detailed descriptions of the students' knowledge, while some of them (e.g., issue tracing) produce less detailed description. However, it has not been proved that these coarse-grained student models are less effective.

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 his cognitive characteristics, background knowledge and performance in the subject domain.

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. Each 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).

Figure 1 presents a snapshot of a typical content page in DEPTHS. The system utility menu is displayed at the top of the page (Figure 1C). Content menu on the left side of the page provides students with navigational links to the available learning topics in the course material (Figure 1A) as well as with information about the accomplished topics (Figure 1B). The rest of the page is reserved for the content of the currently selected learning topic and navigational utilities (Figure 1D and 1E).

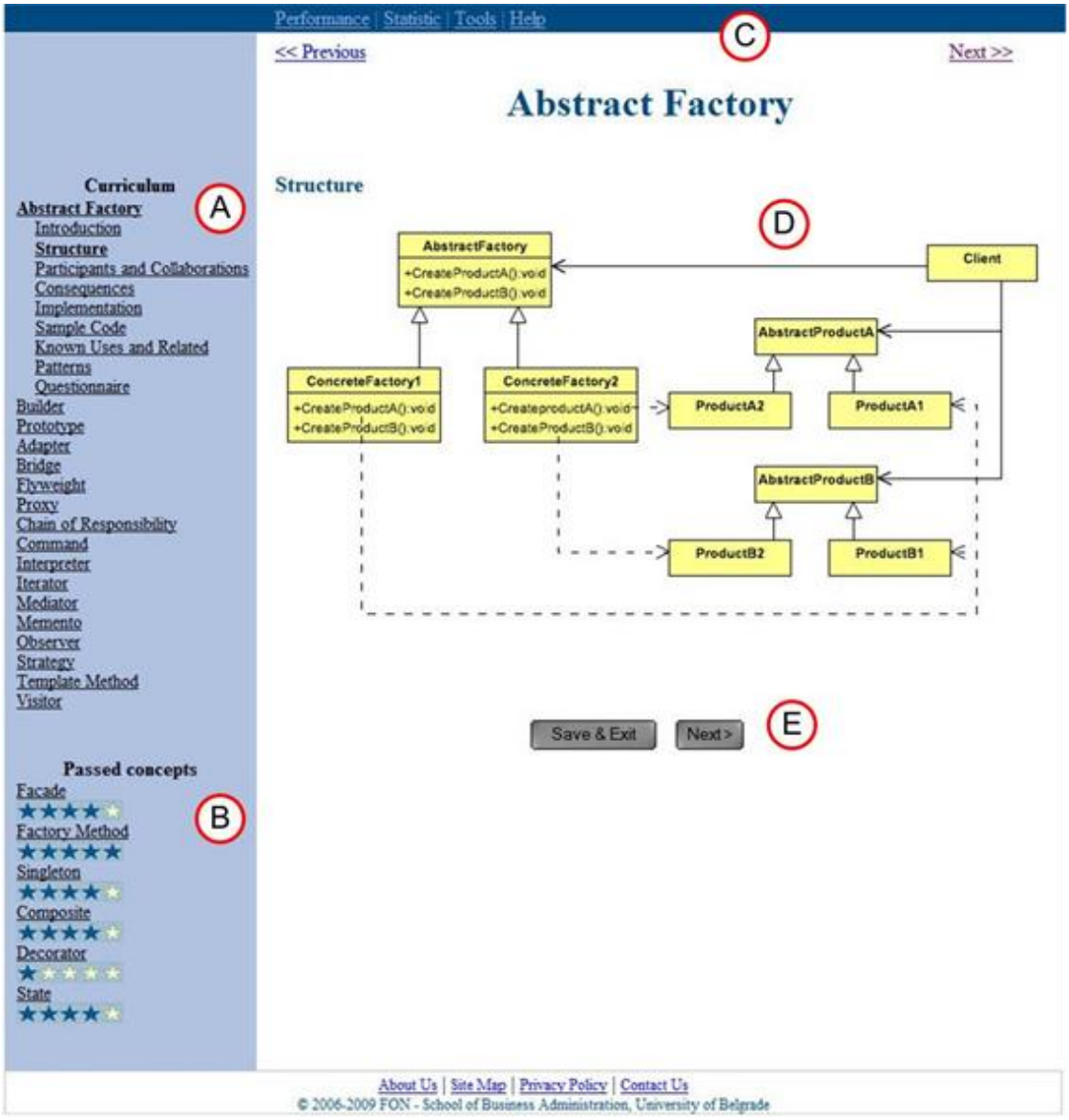

Fig. 1. Web based GUI of DEPTHS.

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 1E). The destination of the 'Next' button is dynamically determined by the system.
- Link removal – Advanced students can choose the concept to learn by selecting appropriate links from the Content menu (Figure 1A). However, the links that the system considers inappropriate (e.g., too advanced topics) are removed, i.e. they are made unavailable.

Regardless of the working mode, a student must achieve sufficient score on a test for the concept he is currently studying in order to move forward 'safely'. 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 assessment related data are stored in the student model. The system uses these data to prepare instructional plans, as well as to provide the student with recommendations for further work [30]. Furthermore, these data are used for adapting the presentation of the instructional material that is to adjust the presentation 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 in solving the problem and the test difficulty level, 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.

3.2. DEPTHS Architecture

Figure 2 depicts the system architecture of DEPTHS. Domain module contains knowledge about software design patterns and the related teaching material. It is designed as a semantic network of concepts. Each concept corresponds to a single software 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 from which the link originates. Each concept is related to one or more content units. Each unit defines a unique way of organizing and presenting learning content about the given concept (i.e., design pattern). Both the number of content units and the size of each unit are arbitrary. DEPTHS uses unit variants technique [31, 32, 33], which essentially means keeping an alternative content unit (i.e. an HTML page) for each recognized 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 the 'OR' relationship. The student model and the concept relationships defined in 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 (see Section 3.2);

(2) decides how to present the teaching material to the student (see Section 5);

(3) evaluates the student's performance (see Section 5), and

(4) provides both, formative and summative feedback to the student [34].

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. Moreover, it uses Expert module to make decisions related to curriculum sequencing, presentation planning, feedback provision, and evaluating and updating the student model. These decisions are made according to the information about the student stored in the Student model and information about learning content stored in the Domain module. Expert module uses Jess (Java Expert System Shell) as a rule-based inference engine based on the Rete inference algorithm, which is further elaborated in the next subsection [35].

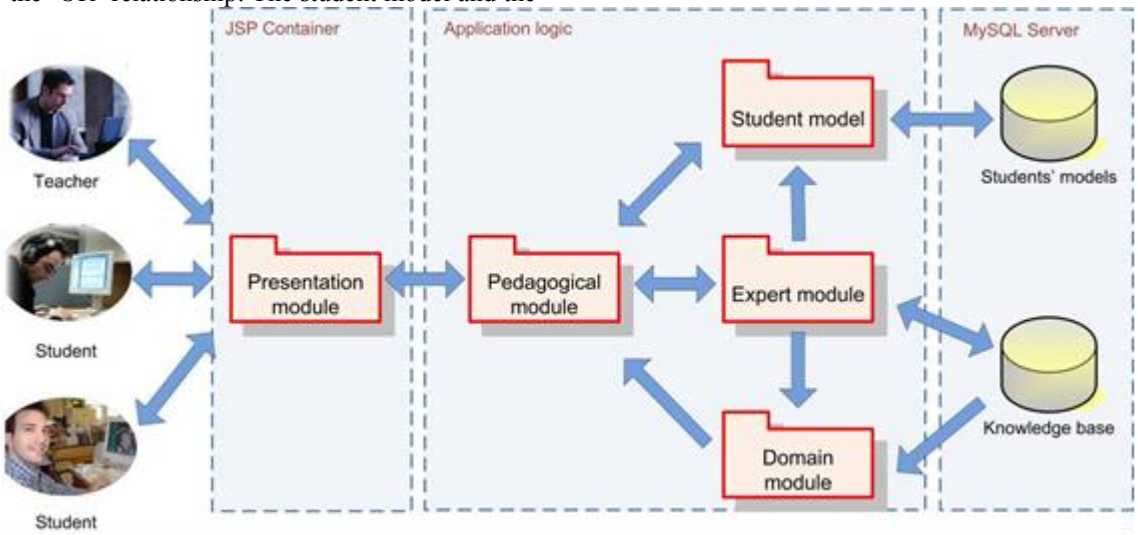

Fig. 2. DEPTHS architecture – the system components and their relationshions

3.2. Curriculum planning in DEPTHS

One of the primary functionalities of Pedagogical and Expert modules is curriculum planning, which is a complex process of selecting appropriate learning items (concepts, content units, fragments and test's questions) that best fits the student's characteristics. A curriculum in DEPTHS comprises a set of concepts that student should learn (concept plan), as well as a set of content units (content units plan) and a set of questions related to the concept that the student currently learns (test plan). A concept plan is created at the beginning of each learning session and after the student's performance has been significantly changed.

DEPTHS uses Jess's inference engine for selecting appropriate learning items (concepts, content units and questions) that should be included in the curriculum. Jess keeps in its working memory a set of facts about learning

items and the student's current state of knowledge, whereas its knowledge base contains a set of rules for selecting appropriate concepts, as well as queries for selecting appropriate content units and questions.

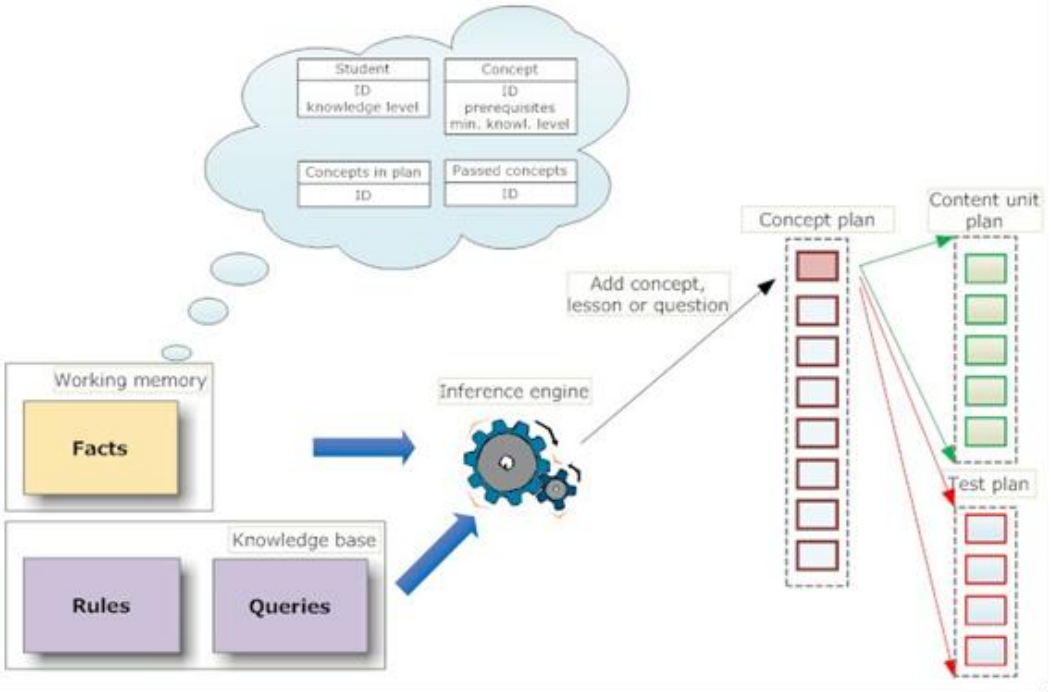

Fig. 3. Curriculum sequencing in DEPTHS

We define four frames (types of facts) that are used in the process of selecting concepts: (1) *Concept* – describes each concept from the Domain module, (2) *Passed_concepts* – keeps data about the concepts that student have already been learned, (3) *Concepts_ in_plan* – keeps data about the concepts that have already been added to the concept plan and (4) *Student* – keeps data about the student's current knowledge state (Figure 3). Each time a new fact is added into the Jess working memory, it is compared against the rule for selecting concepts (*check_concept* in Figure 4). According to this rule, the concept should satisfy the following conditions in order to be added to the concept plan:

- Student's current knowledge state should be the same as or lower (if possible) than the minimal knowledge level predefined for learning that concept,
- The concept should not be learned earlier,
- All prerequisite concepts should be already learned,
- The concept should not be already in the concept plan.

```
defrule check_concept
?fact1 <- ( concepts (ID ?cID) (prerequisite $?first ?one $?rest)
(minKnowledgeLevel ?mKnLev) (ordinalNumber ?ordNum))
(student (ID ?sID) (knowledgeLevel ?sKnLev))
(passed_concepts (ID ?one))
(concepts_added_in_plan (ID ?caipID))
(test (and (<= ?mKnLev ?sKnLev) (neq ?cID ?caipID) ))
(not (and (concepts (ID ?cID) (prerequisite $?first ?one $?rest)) (not
(passed_concepts (ID ?one)))))
=>
(retract ?fact1)
(call ?pok addConceptToPlan ?cID ?ordNum ))
```

Fig. 4. The rule for selecting concepts expressed in Jess language

Each time some of the concepts satisfy the described conditions, the rule is fired, and the concept is added to the end of concepts' plan. If the system decides that the student's performance has significantly changed, the existing concepts plan will be created from scratch.

When the student selects a concept to learn about, a detailed lessons and a test plan are created for that concept. For the selection of appropriate lessons, the system uses three frames, namely: *Student, Concept* and *Lesson* (keeps data about each lesson in the Domain module). As DEPTHS uses the lessons variant method (more than one lesson could be defined for different knowledge levels) [36], a Jess query is used to find those lessons that are related to the current concept and have difficulty level suitable to the student's current knowledge level. That way

DEPTHS selects those lessons that it believes the student will be able to understand and learn.

Similar approach is used for creating a test plan. At the end of studying each of concepts, DEPTHS assesses the student's knowledge using a test that it creates by combining questions from different question sets. Each question is described in Jess with frame containing information about the concept to which the question belongs, the difficulty level that the question best fits, the lesson that the question is related to and the group to which the question belongs. We defined a Jess query that selects a question, which is related to the current concept, which best fits the student's current knowledge level and which has not already been asked. In addition, each question comes from a different question group.

4. Student modeling

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) [37]. 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 usually remain unchanged throughout the learning sessions However, some of these data (e.g., email, organization, preferred programming language, etc.), student can directly change through available options menu. On the other hand, dynamic features are those that the system constantly updates during learning sessions based on the collected data. These data come directly from the student's interactions with the system, and include, 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. Most of these data are used for the history purposes, i.e. to provide some meaningful feedback information both to the student and teacher about what has been done, how successful the student was, and what needs to be improved. However, some of these data are used by the system itself, for the purposes of the adapted curriculum planning as described in the previous section.

4.1. Student model in DEPTHS

In DEPTHS, three basic categories of the students' characteristics are considered (Figure 5):

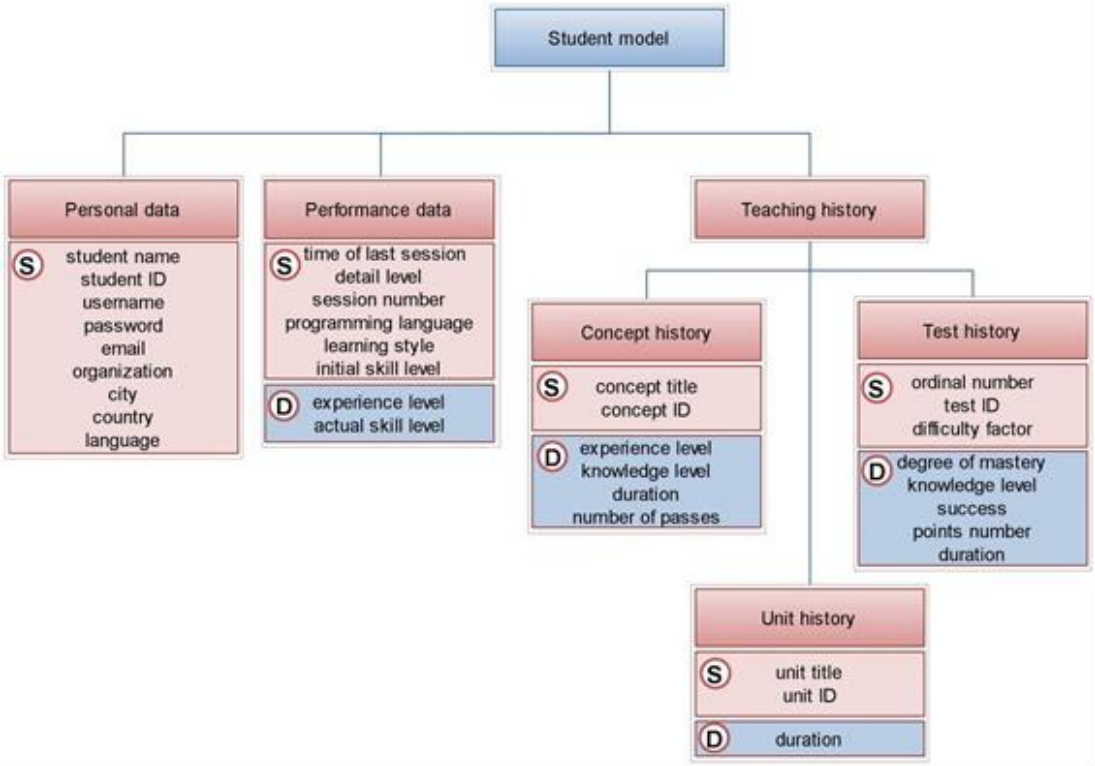

Fig. 5. Student model attributes grouped into categories (S-static data, D-dynamic data)

(1) Personal data – personal characteristics of the student (e.g., name, ID, and 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. This data is not essential for the adaptation of the teaching material.

(2) Performance data – cognitive and individual characteristics of the student. This part of the student model consists of a mixture of static and dynamic data. Static data, such as the desired detail level (when presenting content), the initial skill 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 and experience level. 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 assess precisely and accurately student's traits, it will fail to provide a good teaching material, too. Some of the data in this group affect the selection of content (i.e., teaching material) and other affect the presentation of content. The actual knowledge level, for example, indicates the student's level of mastery of a certain domain topic (i.e., a design pattern). This parameter is used by the system to provide the student with the lesson which is the most suitable for his current knowledge level. On the other hand, the programming language attribute is used for presenting 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 a learning session. This part of the student model is aimed at keeping track of everything that the student has done during the learning process. It also keeps track about the student's performance and other specific data related to the learning session, such as the time spent on solving tests and the student's success on a particular test. 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. In particular, to support reflective learning, the system uses these data to provide the student with feedback about his learning process: 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 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 [8]. When a new student registers with the system, he has to select a stereotype according to his self-judgment of his mastery level of the subject domain. The following categories (i.e. stereotypes) are available: beginner, intermediate and advanced. Subsequently, the system creates a student model for that student with attributes set to the 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 initial knowledge the system had 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. Student Knowledge Diagnosis

Student knowledge indicates the student's level of understanding of the domain concepts. In DEPTHS, we use the overlay student modeling 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, 1995). This choice was driven by the very nature of the assessment process in online learning environments and the fact that it is very difficult to precisely measure a student's knowledge in such learning settings. The most commonly used method for determining a student's achievements in a specific domain is to use tests. However, when judging about a student's achievements, besides test results, many other factors should be considered (e.g., test difficulty, time limitations, motivation, etc.). All this information is rather imprecise, error prone and its interpretation is vague and uncertain. Moreover, each teacher has their own criteria for evaluation of students' knowledge, and the method of knowledge diagnosing that is suitable for one teacher may not be suitable for other teachers. Fuzzy logic techniques are capable of approximating human-like diagnosis of a student's knowledge. Fuzzy models successfully handle reasoning with imprecise information, and enable representation of student knowledge in the same way human teachers do. This promising aspect of fuzzy logic has been recognized by the ITS/learning technology community, which applied fuzzy logic in various contexts [38, 39, 40].

For each concept, a fuzzy value (ranging from 0 to 5 and above) is calculated as the quantitative measure of the 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 process have a form similar to the one in Figure 6.

In Figure 6, '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. It is calculated as the average value of difficulty of all questions that the student has to resolve on the test. 'Test difficulty' has the range from zero ('very easy') to 100 ('very difficult'), and is described by five fuzzy sets (Figure 7-a).

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

Fig. 6. Example of a rule used in the reasoning process

The 'Long' value in the 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 (Figure 7-b) are defined for

the 'duration' variable: 'short' corresponds to fast resolution 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 reflects the correctness of the student's 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 (Figure 7-c).

'Enough' is one of the 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 'degree of mastery' variable and takes values in the range from 0 to 6. It is defined by six fuzzy sets (Figure 7-d).

The logic for diagnosing the student's knowledge for a concept is encoded as a set of fuzzy rules using the FuzzyJ Toolkit [41] since this toolkit allows for modeling fuzzy concepts and doing fuzzy reasoning in Java settings. For our example rule (Figure 6), 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 (degree of mastery). The antecedent (the rule's premise) describes the conditions that have to be satisfied for the rule to be activated, 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. In our case, we defined 75 rules obtained as the combination of each value of the test difficulty, duration and success. The inputs (i.e., antecedents) are combined logically using the AND operator (as shown on Figure 6). A firing threshold for each output membership function is computed. These output values are combined (using a union of fuzzy values) to give a global fuzzy value. Finally, this global fuzzy value output is converted to a crisp value (in the process called defuzzification [42]) in order to get a representative value for the student knowledge. This (crisp) output value is passed to the student model as the system's belief about the student knowledge of the concept being studied.

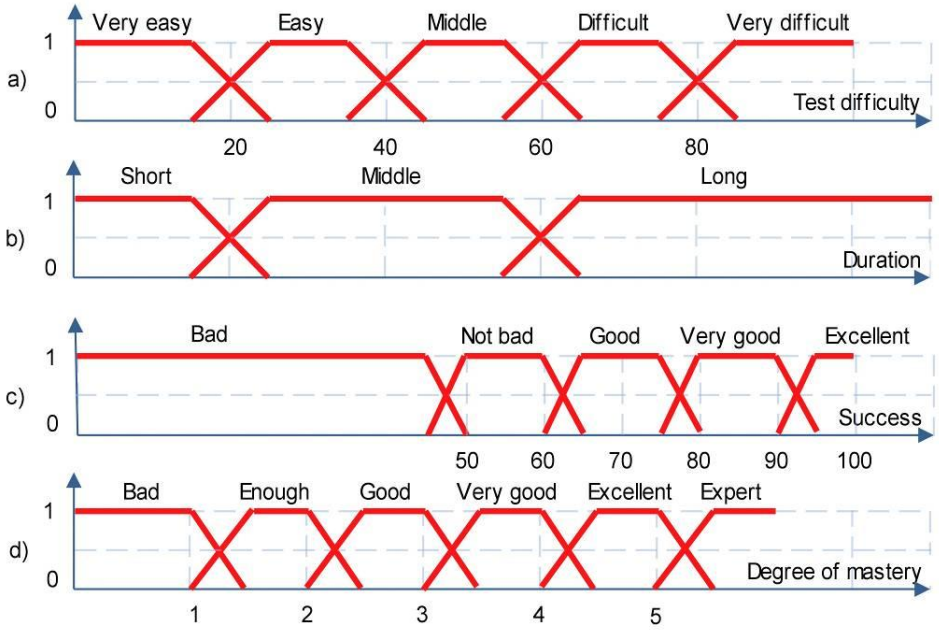

Fig. 7 Fuzzy variables: a) test difficulty, b) duration, c) success, d) degree of freedom

We use the student's results on test(s) related to a specific concept (the initial test and if it was unsuccessful, also the repeated test(s)) when calculating the student's overall knowledge of the concept. The formula we use for that purpose is as follows:

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

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

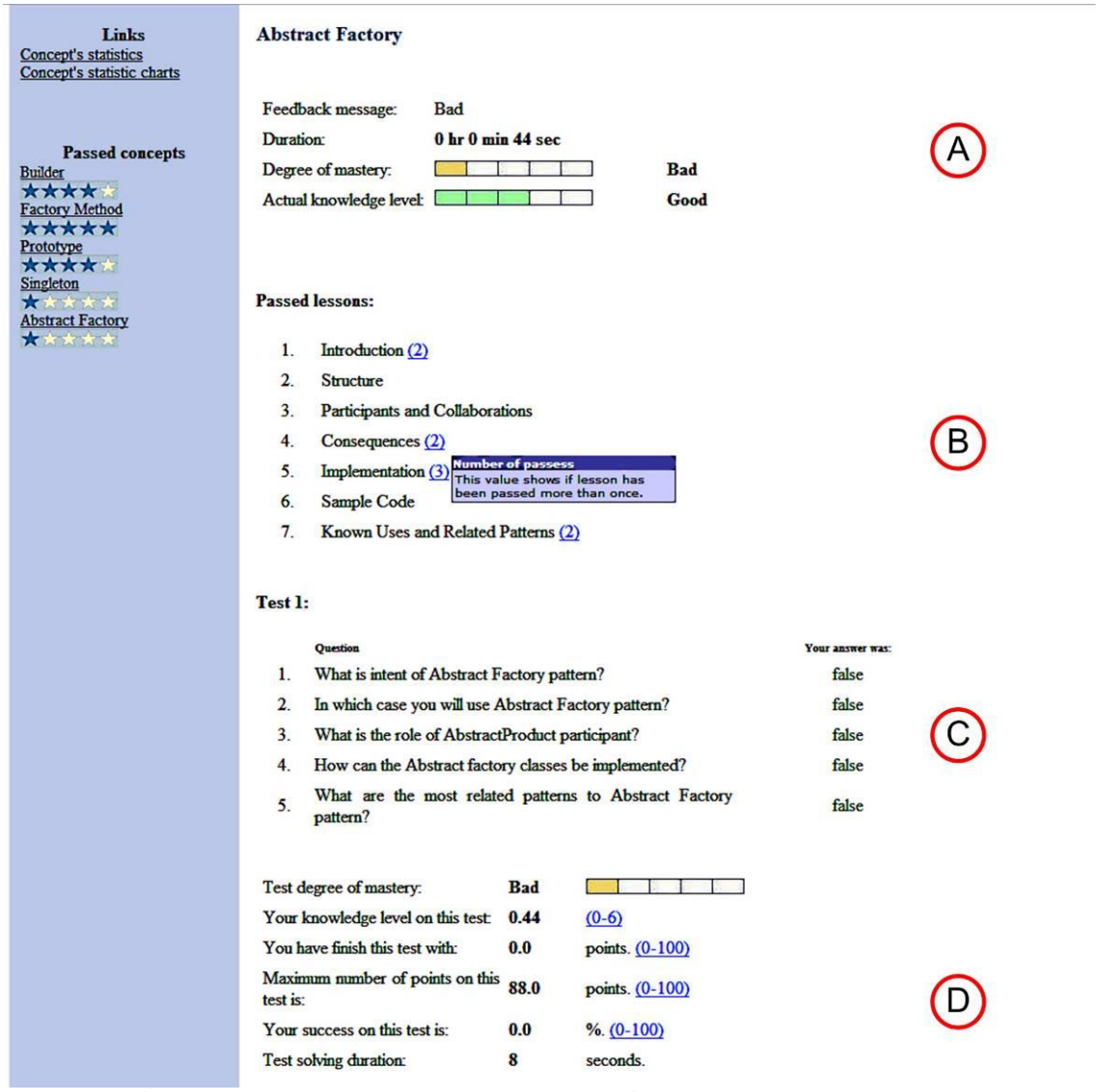

Fig. 8 Aiming to stimulate reflective learning, the system provides a student with his performance records on each domain concept (in this case, the Abstract Factory design pattern)

Based on its diagnoses of all the studied concepts, the system creates an overall diagnosis about the student's understanding of the domain. This value is calculated as the average value of all concepts degrees of mastery (formula 2).

$$Kn = \frac{\sum_{i=1}^{n} DM_i}{i} \quad (2)$$

Kn – the system's belief about the student's overall understanding of the domain
i – ordinal number of the passed concept
n – number of passed concepts
DM – the system's belief about the student's degree of mastery of a specific concept

4.2. Open Student Modeling in DEPTHS

In order to provide better support for reflective learning, the system provides a student with open access to his student model. For example, the student can choose to preview the overall teaching history by selecting the page that contains main information about all studied concepts. He can also take a closer look at each concept by selecting the page, which provides details about the concept of interest (Figure 8). Such a page contains general information about the student's knowledge of the concept in question (Figure 8A), details about each lesson learned (Figure 8B), questions that were posed and the student's answers (Figure 8C), time spent on solving tests, number of points on each test etc. (Figure 8D). Based on these data, the student can see which concepts he did not pass well, and learn them again if he wishes.

In addition, the student is provided with chart diagrams about his progress through the course material, as well as with the statistical data about each concept separately. Student can choose to preview his own progress or to compare his progress with the group that he belongs to.

5. Leveraging the Student Model for Personalized Learning in DEPTHS

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 have sufficient and accurate information about the student, such as his age, background knowledge, and interests. However, for the purpose of adaptation, the most commonly used data is his knowledge of the domain under study. One of the most common solutions to get information about what a student knows in a particular domain 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's previous knowledge level. The *Pedagogical module* (Figure 9), its *Diagnostic engine* to be more precise, is responsible for diagnosing the student knowledge level based on the test results. As we explained in detail in the previous section, the *Diagnostic engine* uses a set of pedagogical rules and domain knowledge in order to diagnose test results and infer the student's knowledge level based on these results. 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 system'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 (formula 1), as well as the student's overall knowledge level of the domain (formula 2). 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 knowledge level.

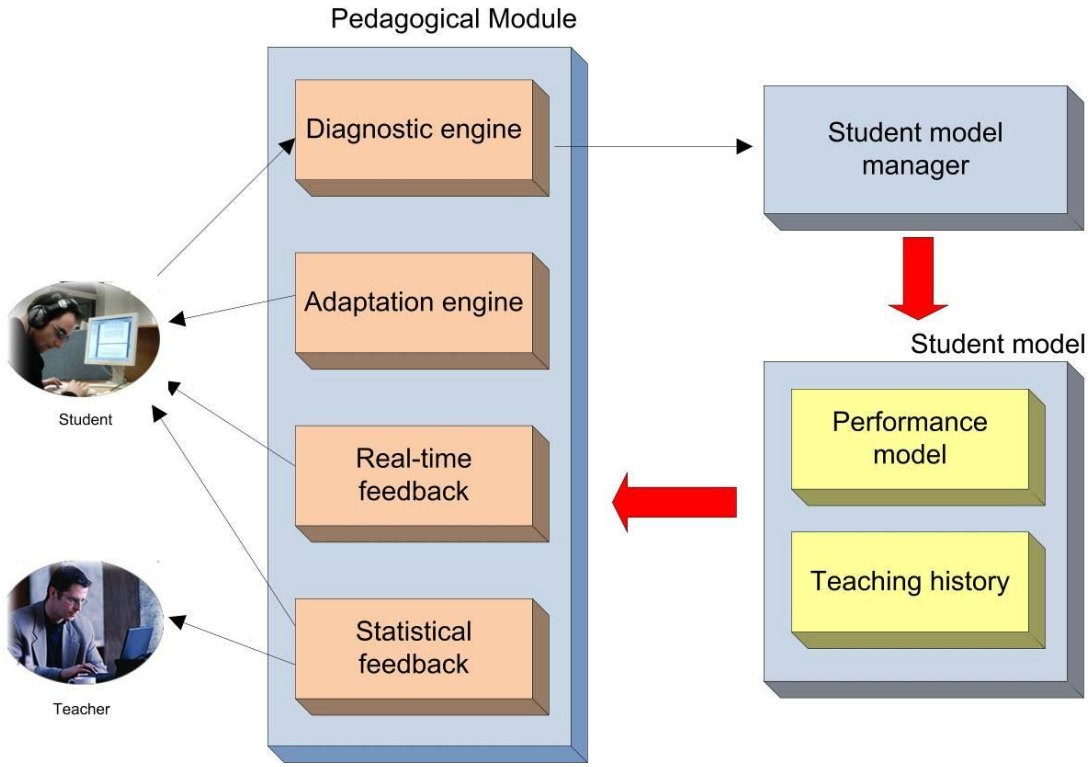

Fig. 9 Pedagogical module and the Student model of DEPTHS

The 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*), Figure 9-A.. This information is also used to provide the student with hints and recommendation for further work (*Real-time feedback*). More specifically the part of the *Student model* that is used for the adaptation purposes is named *Performance model* (Figure 9) and it stores student's performance data. Based on these data, the

Pedagogical module creates Real-time feedback related to the current learning situation. For example, after a test has terminated, the student is directed to the test results page (Figure 10) that gives him more details about results. Here, he is presented with the correct answers to the questions he did not do well (Figure 10D), as well as the test statistic that comprises the time spent on solving the test, the success rate and the final test score (based on the success rate and test difficulty) as shown in Figure 10C. Moreover, the student is presented with a personalized advices and information about his knowledge level on this test (calculated as described in Section 5 in both numerical and human-like interpretation) as shown in Figure 10B. Finally, the student is presented with personalized suggested readings (Figure 10E), that should give him more detailed explanation of the concepts that the system believes he did not understand well.

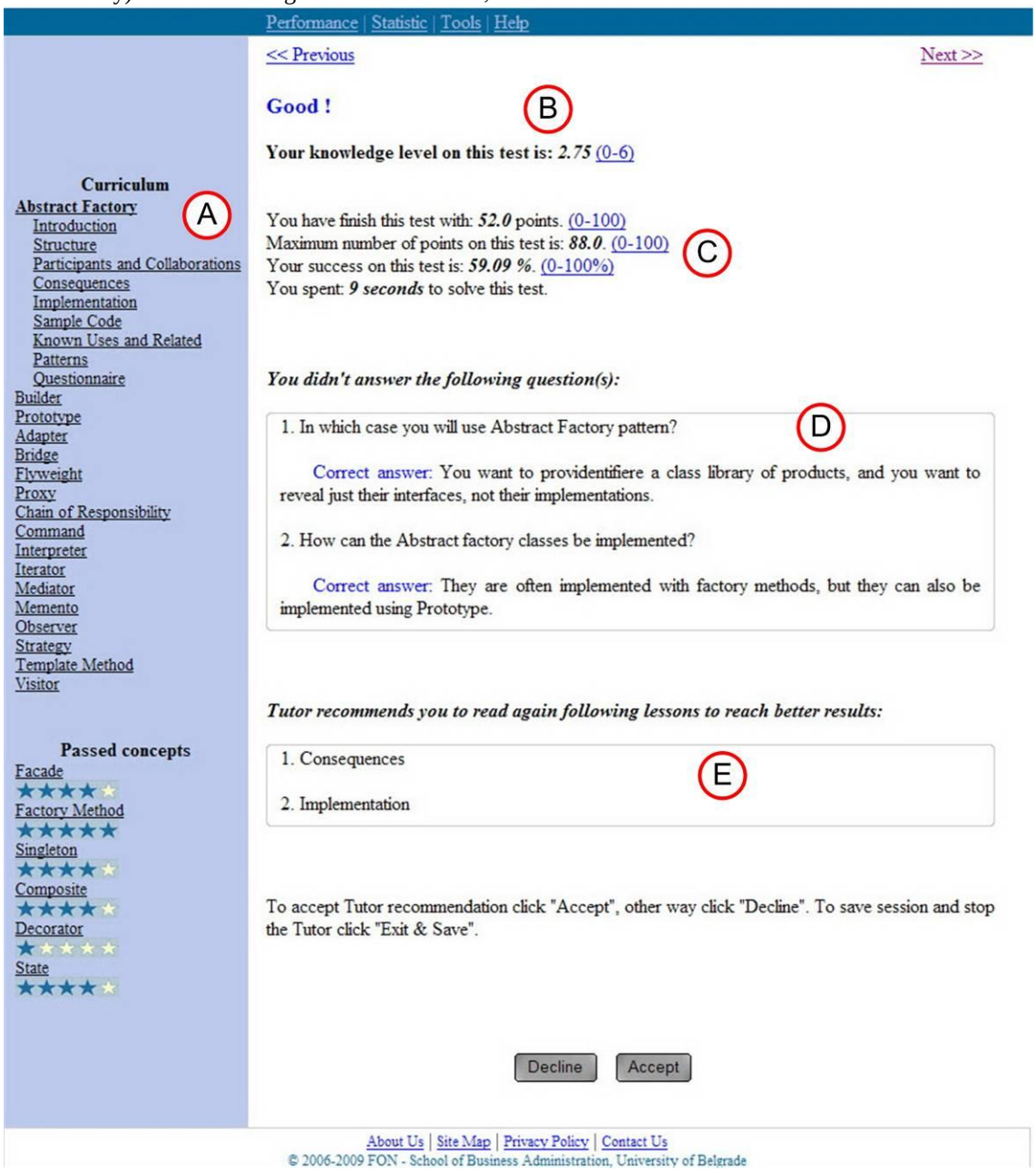

Fig. 10 Pedagogical module and the Student

Regardless of the fact that the Teaching history 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. The *Statistical feedback* engine in DEPTHS can compare performance of any particular student with performance of the group to which the student belongs. It also provides the teacher with many specific analyzing options such as, comparing the performance levels of different students, or different groups with respect to a specific domain concept or the whole course.

6. Evaluation

Evaluation is a paramount success factor of any tutoring system. It provides relevant feedback for system and course developers and is a substantive part of quality assurance.

The evaluation of DEPTHS was performed in the context of a course we teach at the Military Academy of Belgrade, Serbia [43]. 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 and the accuracy of the student knowledge diagnosis included:

- A survey aimed at capturing students' reactions to the training program; an interview was used as a mean for collecting data about the students' satisfaction.
- An experiment set up with the objective of comparing pre-test and post-test results of an experimental group and two control groups (14 computer science students of the third year of undergraduate studies in each group). The students already had some elementary knowledge in the domain of software design patterns from their previous education. The pre-test was designed to test the student's elementary knowledge in software engineering and design patterns. Students in the experimental group were learning with the DEPTHS system, whereas students in the control groups were learning in the traditional way. All of the students were tested at the outset of the experiment to evaluate their knowledge in the subject matter.
- A comparison of the system's assumptions about student's knowledge with the information obtained through an external test designed by a group of domain experts; this enabled us to evaluate the accuracy of the student's knowledge diagnosis.

Generally, DEPTHS received high marks from students. Students felt that the DEPTHS system was very useful, intuitive and easy to use. They found particularly useful the student directed feedback that the system provides. The system's major drawback as recognized by the students is the lack of collaborative learning support.

A comparison of tests results in the experimental and control groups indicated that DEPTHS had a beneficial effect on students' motivation and led to a more rapid improvement in their performance over time. Finally, by examining the knowledge record of each individual student (kept in his student model), and then comparing these values with external test results, we found that most system's assumptions about students' knowledge were in congruence with the external test performance. More precisely, we found that the external test showed a 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 when the process of forgetting the acquired knowledge has already started. The difference is greater for lessons that students learned at the beginning of the course and it was gradually decreasing towards the end of the course.

Accordingly, we concluded that the comparison between the external test and the student model could not be a realistic indicator of the student model accuracy. This is due to the significant difference in points in time when the students' knowledge is assessed (end of the semester in case of the external test, and immediately after the learning a concept in case of our system, i.e., its diagnostic engine). Therefore, we decided to analyze the system's log data in order to get some more information about the accuracy of its student model.

By analyzing the system's log data, we found that some questions, annotated by their creators (i.e., teachers) as very easy, were not performed well by students. Contrary, some other questions described as very difficult were surprisingly well done. Having analyzed this issue further, we concluded that it originates from the fact that the Diagnostic module uses data about a question's difficulty level and time necessary to solve it provided by a human teacher or question developer. However, a teacher can make a mistake when estimating the difficulty level and the required time for a specific question. Such a mistake leads to an inappropriate adaptation of tests. For example, if a teacher describes a difficult question as very easy, the system will use it to test students with low level of knowledge. Based on this conclusion we have improved our Diagnostic module. It now performs analysis of the system's logs data, compares this data with question's difficulty level and time necessary to answer the question, and changes it if a significant variance exists. We strongly believe that this model could significantly improve the previous solution. However, we have not evaluated this model yet.

Detailed explanation of the methodology applied in this evaluation study and analyses of the results obtained, are out of scope of this paper and can be found in [43].

6. Related Work

The student and domain knowledge models are central tools for adaptation in intelligent learning environments (such as ITSs), but they are difficult to acquire and cannot easily be updated or customized during use. Aiming to address these problems, researchers have proposed many different approaches. However, there is no broadly accepted approach yet. A less computation intensive approach is to take the normative stereotyped classes of students, but this approach does compromise expressivity. Other approaches, like Bayesian inference networks are criticized for overwhelming computational complexity [44].

The student model in ALI system [45] shares many common aspects with the DEPTHS's student model. ALI uses the overlay model to store information about the important domain concepts that a student has learned. The system assesses a student's knowledge based on a small

quiz that the student completes at the end of each domain concept. However, the attributes of the ALI's student model are represented as simple flags: whether or not the knowledge element has been accessed, whether it has been explained, and whether the question related to an element was answered correctly or incorrectly. Comparing to ALI, DEPTHS gives precise information about the student's knowledge level of a specific domain concept. Moreover, it does not take into account only results gained on a quiz, but also other aspects like the difficulty of the quiz and time spent to solve the quiz.

Computer-based tutoring system called Stat Lady [46] teaches introductory descriptive statistics. In contrast to DEPTHS that is intended for Web-based learning, Stat Lady was originally developed strictly as a tool for student modeling research in Armstrong Laboratory TRAIN facility. It employs an intelligent student modeling component, called SMART (Student Modeling Approach for Responsive Tutoring), The SMART student modeling paradigm is designed to enhance the efficacy of automated instructional systems across a broad range of domains [47]. It tracks a student's performance and can be relied upon to make decisions regarding which curriculum objectives need to be taught, when the student has mastered a particular topic, and so on. SMART represents a student's level of understanding of each knowledge item with a fuzzy set, which only has one predicate, "mastered". This predicate is identified through a discrete representation such as remedial, intermediate and mastery with two possible values for each of them: "low" and "high". SMART increases or decreases this value based on the feedback about the student's success in solving a problem, where feedback level 0 means that the student has given a correct answer at the first trial. However, the way that SMART updates a student's knowledge has some drawbacks; for example, knowledge cannot be updated until the student provides a correct solution, which is not the case with DEPTHS's student model.

PatternGuru [48] is an educational tool that provides learning of software patterns in a collaborative manner and presents them as an integral part of software development. The basic idea behind a PatternGuru is that learning software patterns should be in the context by involving students directly into concrete software development problems. The tool is developed by extending ArgoUML, an well-known open source project, so that it can be used for both software engineering and education. The system provides a set of services for collaboration between the teacher and his students. Comparing to DEPTHS, PatternGuru has an advantage of using a problem-based approach to learning software patterns. However, it does not support any kind of adaptation of learning material, as it does not implement a student model. We found this tool as a good and natural extension of the DEPTHS system.

User adapted system for teaching Object-Oriented Design Patterns (OODP) supports students in learning OODP by leveraging on existing technologies (software generation facilities, modeling languages, specific and general standard metamodels [49]. The tool is embedded in the well-known software development platform Eclipse. While drawing their OO class diagrams, students can ask the system for advice on how to use OODP. The system suggests learners with the structural class diagrams which most closely match the current learner's software design style for a selected portion of OO class diagram. Suggestions are retrieved from a predefined library of recurring designs drawn from standard software design patterns. Comparing to DEPTHS this system targets the same domain (OODP teaching). However, exact comparison of the effectiveness of the two systems is not possible, because it would be needed to use related evaluation procedures.

An approach to representing student's knowledge similar to the one used in DEPTHS is presented in [50]. The student's knowledge is represented by concepts, topics, and subject. A subject consists of several related topics, whereas a topic consists of sevaral related concepts, and a concept is the basic unit of knowledge. The student model in this work models the relationship between the student's answers and his level of understaning of each concept, as well as the aggregation relationships between different knowledge units suchs as subjects, topics and concepts. Comparing to DEPTHS, a different approach to diagnosing the student knowledge level is used here. A Bayesian network is used to measure student's ability at different levels of granularity, allowing substantial simplifications when specifying the parameters (conditional probabilities) needed to construct the Bayesian Network that describes the student model, and supports the Adaptive Diagnosis algorithm. However, this approach is not evaluated with a real students. Instead, simulated students are used to test the accuracy of each system's component, which makes it difficult to make a comparisson in its effectiveness with DEPTHS.

8. Conclusions

A flexible approach to student modeling in an ITS, using a combination of stereotype and overlay techniques, is presented in this paper. We have also presented our approach to diagnosing students' knowledge that relies on fuzzy logics and rules. Both approaches are tested in DEPTHS, our Intelligent Tutoring System for the domain of software design patterns. 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 slight changes to meet the requirements of a specific ITS.

The flexible design of the DEPTHS system opens numerous possibilities for upgrading. One of the upgrades that we are currently working on is the support for collaborative, project-based learning in order to enable collaboration among students in DEPTHS. The new, extended version of DEPTHS integrates several existing learning systems and tools such as a Learning Management

System, software modeling tool, diverse collaboration tools and relevant online repositories of software DPs [51]. This integration of knowledge from different educational tools facilitated by a common ontology framework [52] provides a solid base for securing personalized and context-aware learning. It is expected to improve students' learning effectiveness and efficiency by providing them with context-aware educational services actively supporting the collaborative learning process. We believe that this approach will significantly improve DEPTHS as it assures active students' participation in the learning process.

References

[1] C. Alexander, S. Ishikawa, M. Silverstein, A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York, 1997.

[2] L. Rising, The Patterns Handbook: Tech., Strategies, and Applications, Cambridge U. Press, NY, 1998.

[3] E.L. Ragnemalm, Collaborative Dialogue qith a Learning Companion as a Source of Information on Student Reasoning. In the Proceedings of ITS'96, Montreal, 1996.

[4] K. Van Lehn, Student modelling. In: M. C. Polson and J. J. Richardson (eds.), Foundations of Intelligent Tutoring Systems. Hillsdale, NJ: Lawrence Erlbaum Associates Publishers, 1988, p. 55-76.

[5] P. Holt, S. Dubs, M. Jones, J. Greer, The state of student modelling. In Greer, J. E. and McCalla, G. I., editors, Student Modelling: The Key to Individualized Knowledge-Based Instruction, NATO_ASI Series F, p. 3–35. Springer-Verlag, 1994.

[6] P. Brusilovsky, Methods and techniques of adaptive hypermedia. User Modeling and User Adapted Interaction 6, 2-3, p.87-129, 1996.

[7] E. Gamma, R. Helm, R. Johnson, J.M. Vlissides, Design Paterns – Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, USA, 1995.

[8] R. Sison, M. Shimura, Student Modeling and Machine Learning. International Journal of Artificial Intelligence in Education., 9(1-2), 1998, p.128-158.

[9] 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, 9-18, Edinburgh, UK, 2005.

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

[11] V. Dimitrova, J. Self, The interactive maintenance of open learner models. In: Lajoie, S., Vivet, M. (eds.): Artificial Intelligence in Education, 1999, p.405-412.

[12] V. Tsiriga, M. Virvou, Initializing the student model using stereotypes and machine learning. In: A. El Kamel, K. Mellouli and P. Borne (eds.), Proceedings of the 2002 IEEE International Conference on System, Man and Cybernetics, 2002.

[13] K. Tourtoglou, M. Virvou, User Stereotypes Concerning Cognitive, Personality and Performance Issues in a Collaborative Learning Environment for UML. New Directions in Intelligent Interactive Multimedia, 2008, p.385-394.

[14] 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, 2000, p.19-30.

[15] D.P. 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, p.20-24, Pittsburgh, USA, 1998.

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

[17] 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, 1994, p.29-33.

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

[19] M. Winter, C. Brooks, J. Greer, Towards Best Practices for Semantic Web Student Modelling,Proceedings of the 12th International Conference on Artificial Intelligence in Education AIED-2005, 2005.

[20] J. Jovanović, D. Gašević, C. Brooks, V. Devedžic, M. Hatala, T. Eap, G. Richards, LOCO-Analyst: Semantic Web to Analyze the Use of Learning Content. International Journal of Continuing Engineering Education and Life-Long Learning, Vol. 18, No. 1, 2008, p. 54-76.

[21] J. Jovanović, D. Gašević, V. Devedžić, TANGRAM for Personalized Learning Using Semantic Web Technologies. Journal of Emerging Technologies in Web Intelligence, Vol. 1, No. 1, 2009, p. 6-21.

[22] P. Dolog, B. Simon, T. Klobucar, W. Nejdl, Personalizing access to learning networks. ACM Transactions on Internet Technologies, 8(2), 2008.

[23] P. Dolog, M. Schaefer, A framework for browsing, manipulating and maintaining interoperable learner profiles. Proceedings of the 10th International Conference on User Modeling, UM2005, Edinburgh, UK. Lecture Notes in Computer Science, Vol. 3538, p. 397–401, Berlin/Heidelberg: Springer, 2005.

[24] A. Ounnas, H.C. Davis, D.E. Millard, Semantic modeling for group formation. Paper presented at the workshop on Personalization in E-Learning Environments at Individual and Group Level (PING) at the 11th International Conference on User Modeling UM2007, Corfu, Greece, 2007.

[25] S. Bull, Supporting learning with open learner models. Proceedings of fourth Hellenic Conference on Information and Communication Technologies in Education, Athens, Greece, 2004, p.47-61.

[26] V. Dimitrova, P. Brna, J.A. Self, The Design and Implementation of a Graphical Communication Medium for Interactive Open Learner Modelling. In Proceedings of the 6th International Conference on Intelligent Tutoring Systems (June 02). S. A. Cerrin, G. Gouarderes, and F. Paraquacu, Eds. LNCS vol. 2363. Springer-Verlag, p.432-441, London, 2002.

[27] J. Kay, Learner Know Thyself: Student Models to Give Learner Control and Responsibility, in Z. Halim, T. Ottoman & Z. Razak (eds), Proceedings of International Conference on Computers in Education (AACE), 1997, p.17-24.

[28] V. Dimitrova, G.I. McCalla, S. Bull, Open Learner Models: Future Research Directions, Special Issue of the IJAIED. International Journal of Artificial Intelligence in Education 17(2-3), 2007.

[29] S. Ohlsson, Some principles of intelligent tutoring, Instructional Science, 14, 1986, p.293-326.

[30] 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, 2002, p.119-128, France and Spain.

[31] I. Beaumont, I. User modeling in the interactive anatomy tutoring system ANATOMTUTOR. User Models and User Adapted Interaction 4 (1), 1994, p.21-45.

[32] J. Kay, R.J. Kummerfeld, An Individualised Course for the C Programming Language, Retrieved March 20, 2009, from http://www.cs.su.oz.au/~bob/kay-kummerfeld.html.

[33] D.W. Kim, WING-MIT: Das auf einer multimedialen und intelligented Benutzerschnittstelle basierende tutorielle Hilfesystem für das Werkstoffinformationssystem WING-M2. Paper presented at the Workshop Adaptivität und Benutzermodellierung in interactiven Systemen, München, Germany, 1995.

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

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

[36] A. Kobsa, J. Koenemann, W. Pohl, Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. The Knowledge Engineering Review, 16, 2, 2001, p.111-155.

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

[38] O. Nykanen, Inducing Fuzzy Models for Student Classification, J. Educational Technology and Sociaty 9(2), 2006, p.223-234.

[39] Z. Ševarac, Neuro Fuzzy Reasoner for Student Modeling. Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, ICALT2006, 2006, p.740-744.
[40] A.P. Ayala, Teaching-Learning by Means of a Fuzzy-Causal User Model. Proceedings of the 8th Mexican International Conference on Artificial Intelligence, 2009, p. 521-532.
[41] E.J. Friedman-Hill, Jess in Action, Rule-Based Systems in Java. Sandia National laboratories, Manning Publications Co., Greenwich, CT, 2003.
[42] J.M. Mendel, Fuzzy logic systems for engineering: A tutorial. Proceedings of IEEE, vol. 83, 1995, p.345-377.
[43] Z. Jeremić, J. Jovanović, D. Gašević, Evaluating an Intelligent Tutoring System for Design Patterns: the DEPTHS Experience, Educational Technology & Society, 12(2), p.111-130, 2009.
[44] E.L. Ragnemalm, Student Modelling based on Collaborative Dialogue with a Learning Companion. Dissertation No 563, Linkoping University, S-581 83, Linkoping, Sweden, 1999.
[45] A. D'Souza, J. Rickel, B. Herreros, W. Johnson, An Automated Lab Instructor for Simulated Science Experiments, In Proceedings of AIED 2001, 10th International Conference on Artificial Intelligence in Education, p. 65-75, San Antonio, 2001.
[46] V.J. Shute, K.A. Gluck, Stat Lady Descriptive Statistic Tutor: Data Organization and Plotting Module. Brooks AFB, TX: Armstrong Laboratory, 1994.
[47] V.J. Shute, SMART evaluation: Cognitive diagnosis, mastery learning and remediation. In J. Greer (Ed.), Proceedings of AI-ED 95, 1995, pp. 123-130, Charlottesville, VA: AACE.
[48] M. Boškovic, D. Gaševic, V. Devedžic, PatternGuru: An Educational System for Software Patterns, Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05), 2005, p.650-654.
[49] M. Marinilli, Cognitive Modeling of Personalized Software Design Styles - A case Study in e-learning, in E-Learning Networked Environments and Architectures A Knowledge Processing Perspective, series: Advanced Information and Knowledge Processing, Springer Book Series, 2009.
[50] E. Millán, J.L. Pérez-de-la-Cruz, A Bayesian diagnostic algorithm for student modeling and its evaluation. User Modeling and User-Adapted Interaction: 12, 2002, p.281-230.
[51] Z. Jeremić, J. Jovanović, D. Gašević, Semantic Web Technologies for the Integration of Learning Tools and Context-aware Educational Services. In Proceedings of the 8th International Semantic Web Conference (ISWC 2009), Semantic Web in Use Track, 860-875, Washington, DC, USA, 2009.
[52] J. Jovanović, C. Knight, D. Gasevic, G. Richards, Ontologies for Effective Use of Context in e-Learning Settings, Educational Technology & Society, 10(3), 2007, p. 47-59.