Computing and Informatics, Vol. 32, 2013, 629–659

# APPLYING RECOMMENDER SYSTEMS AND ADAPTIVE HYPERMEDIA FOR E-LEARNING PERSONALIZATION

Boban Vesin, Aleksandra Klašnja-Milićević

*Higher School of Professional Business Studies*
*Vladimira Perića-Valtera 4, 21 000 Novi Sad, Serbia*
*e-mail:* `vesinboban@gmail.com, aklasnja@yahoo.com`

Mirjana Ivanović, Zoran Budimac

*Department of Mathematics and Informatics*
*Faculty of Science, University of Novi Sad*
*Trg D. Obradovića 4, 21 000 Novi Sad, Serbia*
*e-mail:* {`mira, zjb`}`@dmi.uns.ac.rs`

Communicated by Pavol Návrat

**Abstract.** Learners learn differently because they are different – and they grow more distinctive as they mature. Personalized learning occurs when e-learning systems make deliberate efforts to design educational experiences that fit the needs, goals, talents, and interests of their learners. Researchers had recently begun to investigate various techniques to help teachers improve e-learning systems. In this paper we present our design and implementation of an adaptive and intelligent web-based programming tutoring system – *Protus*, which applies recommendation and adaptive hypermedia techniques. This system aims at automatically guiding the learner's activities and recommend relevant links and actions to him/her during the learning process. Experiments on real data sets show the suitability of using both recommendation and hypermedia techniques in order to suggest online learning activities to learners based on their preferences, knowledge and the opinions of the users with similar characteristics.

**Keywords:** e-learning, recommendation, personalization, adaptive hypermedia, ontology, semantic web, tutoring system

**Mathematics Subject Classification 2010:** 68T05, 68N15, 68N19

## 1 INTRODUCTION

Design and implementation of web-based educational systems have grown exponentially in the last years, spurred by the fact that neither learners nor teachers are bound to a specific location and that this form of computer-based education is virtually independent of any specific hardware platforms [33]. These systems accumulate a vast amount of information which is very valuable in analyzing learners' behavior. However, due to the vast quantities of data these systems can generate daily, it is very difficult to manage manually and to provide an individual approach for each learner. A very promising area to attain this objective is the use of recommender systems. Educational tool/system should recommend learners materials that are easily understandable according to their level of knowledge and are interesting enough to keep the learners' attention.

The task of delivering personalized content is often framed in terms of a recommendation task in which the system recommends items to an active user [29]. Recommender systems use the opinions of the community of users to help individuals more effectively identify content of interest from a potentially overwhelming set of choices [31]. Such systems have become powerful tools in many domains from electronic commerce to digital libraries and knowledge management [34]. Some recommender systems have also been applied to e-learning systems for recommending lessons (learning objects or concepts) that learners should study next [26] or for providing recommendation about lessons offered that contribute to the learner's progress towards particular goals [12]. Recommender system in the e-learning domain have specific requirements not present in other domains, most importantly the need to take into account pedagogical aspects of the learner and the need to recommend sequences of items in a pedagogically effective order. We suggest the construction of effective automatic recommendation system for web-based learning environments that takes into account profiles of on-line learners and their access history and uses simple data mining techniques.

Another approach for course personalization is the use of *adaptive hypermedia* methods and techniques that are used in *adaptive educational hypermedia systems* [8]. Adaptive Hypermedia [3], and especially its application in the educational field, adaptive educational hypermedia, is a field that has been developing for about 20 years, centered around personalization of content to a learner in a Web environment. Adaptive educational hypermedia systems can adaptively sort, annotate, or partly hide the links on web pages to make it easier to choose or to recommend to the learners where they should go from a certain point based on his/her goals, preferences and level of knowledge. Learners can also be informed about importance and relevance of certain links.

In this paper we present our design and implementation of an adaptive and intelligent web-based PRogramming TUtoring System – *Protus* that applies recommendation and adaptive hypermedia techniques. The implemented system aims at automatically guiding the learner's activities and recommend relevant links and actions to him/her during the learning process.

The rest of this paper is organized as follows. Section 2 presents related work from the area of research. Personalized options with usage of recommender systems and adaptive hypermedia are considered in Section 3. Section 4 explains the architecture and design of Protus. Learner interface and usage of Protus are covered in Section 5. Results and an outlook on our future work in the area of personalization of e-learning content conclude this paper in last two sections.

## 2 RELATED WORK

Computer technology has been used to develop a vast array of educational software, from early computer-based training systems to web-based adaptive hypermedia, multimedia courseware, and educational games. These systems have given learners access to a great variety of pedagogical approaches that supplement classroom learning and provide resources outside the classroom. This variety has been helpful in reaching learners who don't do well with traditional lecture and textbook instruction. During our research, we focused attention on a specific kind of tutoring systems only. These can be roughly classified into two categories:

1. programming tutoring systems and
2. tutoring systems that use different recommendation techniques in order to suggest the most appropriate online learning activities to learners, based on their preferences, knowledge and the browsing history of other learners with similar characteristics.

### 2.1 Programming Tutoring Systems

Most of the tutoring systems for learning programming languages found on the Web are more or less only well-reformatted versions of lecture notes or textbooks [2]. As a consequence these systems don't have implemented interactivity and adaptivity. The functions that such systems can perform vary. Some of them are used for learner assessment, like JavaBugs [37] and JITS [38, 39], or some of them are adaptive web-based tutorials [33, 36]. One step further in implementation of adaptation was made by systems like JOSH-online [2], iWeaver [44] and CIMEL ITS [17, 43].

JavaBugs examines a complete Java program and identifies the most similar correct program to the learner's solution among a collection of correct solutions. After that it builds trees of misconceptions using similarity measures and background knowledge [37]. They focused on the construction of a bug library for novice Java programmer errors, which is a collection of commonly occurring errors and misconceptions.

The WWW-based introductory LISP course ELM-ART (ELM Adaptive Remote Tutor) is based on ELM-PE [4], an on-site intelligent learning environment that supports example-based programming, intelligent analysis of problem solutions, and advanced testing and debugging facilities. For annotating the links, the authors use the traffic light metaphor. A red ball indicates pages which contain information for which the user lacks some knowledge, a green ball indicates suggested links, etc.

Java Intelligent Tutoring System – JITS is a tutoring system designed for learning Java programming [38]. JITS implements JECA (Java Error Correction Algorithm), an algorithm for a compiler that enables error correction intelligently changing code, and identifies errors more clearly than other compilers. This practical compiler intelligently learns and corrects errors in learners' program [39].

iWeaver is an interactive web-based adaptive learning environment, developed as a multidisciplinary research project at RMIT University Melbourne, Australia [44]. iWeaver was designed to provide an environment for the learner by implementing adaptive hypermedia techniques to teach the Java programming language. It implements several established adaptation techniques, including link sorting, link hiding and conditional page content. The current version of iWeaver does not support adaptive navigation, which is one of the best researched areas of adaptive environments.

JOSH is an interpreter for the Java programming language [2] originally designed to make easier teaching Java to beginners. Recently the interpreter was restructured into a server based interpreter applet and integrated into an online tutorial on Java programming called JOSH-online.

CIMEL ITS is an intelligent tutoring system that provides one-on-one tutoring to help beginners in learning object-oriented analysis and design. It uses elements of UML before implementing any code [17]. A three-layered Learner model is included which supports adaptive tutoring by deducing the problem-specific knowledge state from learner solutions, the historical knowledge state of the learner and cognitive reasons about why the learner makes an error [43]. This Learner model provides an accurate profile of a learner so that the intelligent tutoring system can support adaptive tutoring.

Most of the existing e-learning platforms for teaching programming have not yet taken the advantage of adaptivity [11, 13, 20, 38], possibly because the expected profit has not justified the high effort of implementing and authoring adaptive courses. Moreover, most of the adaptive tutoring systems do not support e-learning standards. Our system recommends a media experience that is most likely to be chosen in the current learning context by the current learner. This recommendation mechanism is attempting to accommodate to a possible variation in a learner's learning style profile. Also, up to now most if not all systems do not take into consideration the important aspect of learning styles preferences or how and when to adjust the presented topic based on the preferred presentation method of the learner.

## 2.2 Tutoring Systems with Implemented Recommendation

A personalized recommender system that uses web mining techniques for recommending a learner which (next) links to visit within an adaptable educational hypermedia system was described in [33]. They presented a specific mining tool and a recommender engine that they have integrated in the AHA! system, in order to help the teacher to carry out the whole web mining process. They made several experiments with real data in order to show the suitability of using clustering and sequential pattern mining algorithms together for discovering personalized recommendation links [3].

Another system described in [36] allows all learners to collaborate their expertise in order to predict the most suitable learning materials to each learner. This smart e-Learning system applies the collaborative filtering approach [36] that has an ability to predict the most suitable documents to the learner. All learners have the chance to introduce new material by uploading the documents to the server or pointing out the web link from the Internet and rate the currently available materials.

My Online Teacher 2.0 (MOT 2.0) successfully combines Web 2.0 features (such as tags, rating system, feedback, etc.) in order to support both learners in personalized systems, as well as authors [14]. Authors focus on a study of how to more effectively use and combine the recommendation of peers and content adaptation to enhance the learning outcome in e-learning systems.

In the last few years, some research studies have been conducted on developing an approach that identifies learning styles automatically from learners' behavior in an online course [15, 18].

All these systems claim to be innovative and stress the importance of content, but unfortunately, none of these tutoring systems are being used by a large and worldwide community outside the research area. The adaptive response of existing environments is often restricted to pictures and text instead of multimedia presentations, with some exceptions like the iWeaver [44]. Systems like Logic-ITA, ProGuide and Jeliot 3 gave us good ideas and perspective which functionalities could be included in new web-based tutoring system [27, 30]. Compared to current tutoring systems which are only executed on stand alone machine (JavaBugs, JITS, CIMEL ITS, Jeliot 3) or have just basic interactivity and adaptivity implemented (JOSH-online, JavaBugs, Logic-ITA), Protus system integrates content and link adaptation in order to accomplish completely functional web-based tutoring system with personalization options. Protus e-learning system offers a constant, on-the-fly adaptation of the course units and their presentation to the current needs and preferences of the individual learner. This guarantees a significant, individual success of a learner.

None of the above-mentioned systems implements full use of the recommender techniques (like collaborative filtering, association rule mining and clustering), just the basic data mining techniques. Second, besides learning content ranking, Protus also supports learning path generation and personalization based on the learning styles identification.

Our work differs from previous mentioned papers in several aspects. First, we combine several adaptation techniques, both recommendations of material and adaptive hypermedia, in order to personalize lessons presentation to learners. Second, besides learning content ranking, Protus also supports learning material clustering and learning path generation. Third, despite of the great variety of tutoring systems in the literature we chose to focus our attention on programming tutoring system that defines scalable and adaptable architecture. Protus provides the posibility to import knowledge from various domains in our system so that the proces of learning can be performed in whatever domain of knowledge. This choice enabled us to develop a system for knowledge presentation and acqusition that tries to be independent from the specific domain.

## 3 ADAPTIVE LEARNING AND PERSONALIZATION OF CONTENT

Different techniques need to be implemented to adapt content delivery to individual learners according to their learning characteristics, preferences, styles, and goals [21]. Protus provides two general categories of personalization in system based on adaptive hypermedia and recommender systems:

**Content adaptation** – presenting the content in different ways, according to the domain module and information from the learner model. All learners and contents are grouped into classes of similar objects in order to recommend optimum resources and pathways. The principle of clustering is maximizing the similarity inside an object group and minimizing the similarity between the object groups. Such clusters needed to be defined in Protus in order to provide learner with the most suitable learning material and to form the most suitable pathway. The system maintains different versions of pages it presents to the learners or in same cases different versions of page fragments within the page, and selects the version to show to the learner according to the information in learner model. System also hides advanced content from a novice learner or shows suitable additional content to more advanced learner.

**Link adaptation** – the system modifies the appearance and/or availability of every link that appears on a course web page, in order to show the learner whether the link leads to interesting new information, to new information the learner is not ready for, or to a page that provides no new knowledge. System makes some links inaccessible to the learner if the system estimates from the learner model that such links take him/her to the irrelevant information. System assumes that less successful learners will be interested in additional materials, which are given in form of block diagrams or exact syntax rules. Therefore, those learners may click the link for additional material on the interface.

Several approaches can be used to personalize the material presented to the learner. As mentioned earlier, our approach is based on applying adaptive hypermedia and recommender systems in combination with learner modeling to enable a personalization process in Protus. In this paper, we describe system architecture that utilizes these techniques to form a scalable, reliable, and robust structure.

### 3.1 Adaptive Hypermedia

Adaptive hypermedia methods and techniques make it possible to inform learners that certain link leads to material they are not ready for, to suggest visiting pages the learner should consult, or automatically provide additional explanations at the pages the learner visits, in order to scaffold his/her progress [8]. There are two main reasons for using adaptive hypermedia: to avoid problems that occur when reading material out of intended order and to better serve the individual differences between users [7].

In an online course with navigational freedom the learner cannot know whether a reference to some material is a forward or a backward reference. Protus tracks every learner's path through the course text and thus system is provided with information whether for the current learner at this time a reference is a forward or backward reference.

This allows authors to create multiple versions of the references and system can choose and present the appropriate one. This kind of content adaptation is useful in order to provide learner with additional (further elaboration of concept that is already known) or comparative (interesting comparison with other, already known concept) explanation.

The system provides additional explanations using conditional pages – buttons that lead to different segments or pages are turned on and off and presented to the learner as needed, based on whether his/her learner model meets some conditions or not. Conditions are results of collaboration filtering and association rule mining, which will be presented later in this chapter. In some cases, the system simply maintains different versions of pages it presents to the learners and selects the version to show according to the learner model. In that vein, all learning objects are given certain role that provides information about its use. For example, if concept has *fact* or *definition* role it is used to increase basic knowledge and if its role is *example*, than it is used to increase learner's practical skills.

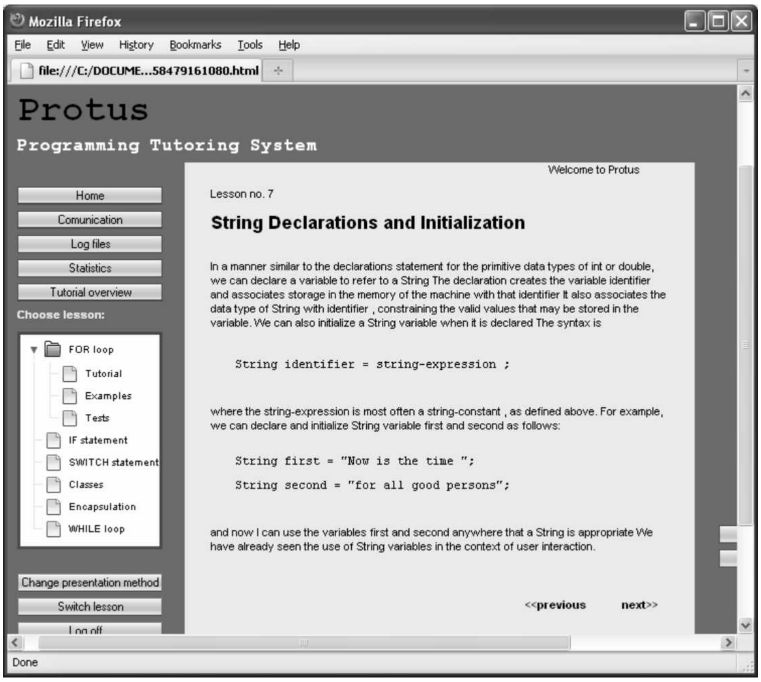
Protus not only adopts content to the knowledge level of the learner. Preferred learning styles for every particular learner are also considered. Every learner has his/her own learning style that indicates a preference for some media type(s) over others. Protus distinguishes three user's learning styles (among other identified in [10]) and therefore use three different presentation methods:

**Textual.** Learners that prefer to perceive materials as text are provided with lessons, which are in form of text pages with rich formatting and highlighted source code.

**Visual.** Learners that prefer to perceive materials in form of pictures are provided with illustrations, figures, diagrams, flowcharts, etc.

**Interactive.** Learners that prefer to interact physically with learning material are provided with interactive flash animation.

Figure 1 shows a presentation of the topic of *String Declarations* and *Initialization* to a learner with a preference for textual material. Figure 2 shows the presentation of the material to a learner with a visual preference. Based on the visual preference, the topic about the *FOR loop* is presented as a block diagram.



Lesson no. 7

String Declarations and Initialization

In a manner similar to the declarations statement for the primitive data types of int or double, we can declare a variable to refer to a String The declaration creates the variable identifier and associates storage in the memory of the machine with that identifier It also associates the data type of String with identifier , constraining the valid values that may be stored in the variable. We can also initialize a String variable when it is declared The syntax is

```
String identifier = string-expression ;
```

where the string-expression is most often a string-constant , as defined above. For example, we can declare and initialize String variable first and second as follows:

```
String first = "Now is the time ";
String second = "for all good persons";
```

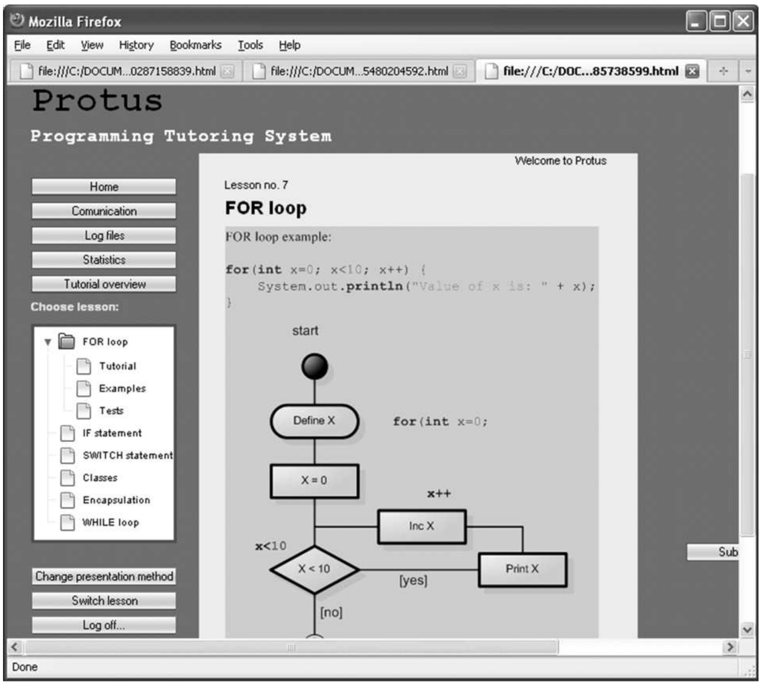
and now I can use the variables first and second anywhere that a String is appropriate We have already seen the use of String variables in the context of user interaction.

Fig. 1. Verbal presentation

Preferred style for every learner is determined with questionnaire at the beginning of the course and with optional filled-in questionnaire at the end of every completed lesson. It is considered that surveys and questionnaires are intrusive and distracting in a learning environment [45]. Therefore, they are optional through course, but still necessary for improving process of recommendation. Protus also tracks improvement that specific learners make while using specific presentation method and update learner model accordingly. In every moment learner can manually switch between different presentation methods.

## 3.2 Recommender Systems

Recommender systems are often implemented as software agents that anticipate the needs of web users and provide them with recommendations to personalize their navigation [1]. They are becoming very popular in e-commerce applications to recommend the online purchase of some products [23]. These agents can be very useful in an e-learning environment to recommend actions, resources or simply links to follow. In this section, we present our proposed personalization approach taking into account both the course access history of learners as well as the content of the learning material. Our approach is based on applying data clustering, collaborative



Lesson no. 7

**FOR loop**

FOR loop example:

```
for(int x=0; x<10; x++) {
    System.out.println("Value of x is: " + x);
}
```

Fig. 2. Visual presentation

filtering and association rule mining techniques [24].

**Data clustering.** Learners are grouped into smaller, identifiable and manageable clusters, based on their common attributes (e.g. class, age), and on the preferable categories of content delivery. Parts of the instruction are then tailored to the groups, and are applied in the same or similar way to all members of a segmented group. The categorization is accomplished by different surveys (that the learner is prompted to fill during the registration on the system and optionally after every lesson) and by monitoring the learner's actions, progress and overall performance. Questions and tasks within tests are also categorized. Tests in our system contain three types of questions [42]:

**Multiple-choice of syntax check.** This type of test is used to ask the learner to trace the correct sample code.

**Multiple-choice of execution results.** This type of test is used to ask the learner to choose correct result after execution of offered code portion.

**Code completion.** The problem is presented in form of skeleton program with the specific area for entering appropriate code snippet according to program specification.

After clustering is performed, learners are categorized into clusters based on their learning interest. However, recommendations cannot be made at this point, because even for learners with similar learning interests, their ability to solve the task can vary due to the dissimilarity of their knowledge level. Therefore, during this process, recommendations will be made not on the whole pool of learners as most recommender systems do [19] but on the clustered areas as illustrated in Figure 3 below. Accordingly, the number of recommended lessons will also shrink.

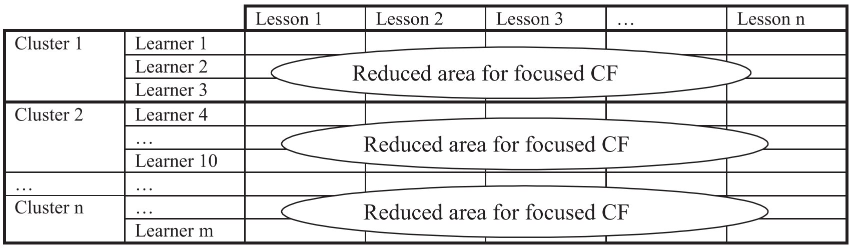


Fig. 3. An illustration for focused *Collaborative filtering*

**Collaborative filtering – CF.** When learner needs suggestion about which location to visit or which test or example will provide the most benefits, the learner profile is compared to the collective to find similar profiles [24]. A selection from these similar profiles is used to produce a recommendation. For this matter, we needed good and trusted ratings entered by the learners. The learners are prompted to fill short survey after every lesson. This survey is optional because entering ratings could be considered intrusive. If a learner refuses to fill the survey, the system uses history logs by other similar learners as input for his/her profile. Therefore, learners' inaction and failure to cooperate have no significant influence to recommendation as system is independent of learners' response. Consequently, grading is not the primary input for constructing recommendations. The system tracks navigation patterns of learners and uses these patterns for creating a recommendation list according to the ratings of frequent sequences, which will be addressed later in this section.

**Association rule mining.** When setting up the structure of the course, authors of courses have a certain navigation pattern in mind and assume all learners would follow a consistent path, materialized by some hyperlinks. Learners, on the other hand, could follow different paths based on their preferences and generate a variety of learning activities. Often these activities are not in the optimal order, and probably it differs from the order of learning activities intended by the author. All those varieties of different sequences of learning activities are noted down, sorted by success of learners that performed them and recommended to next learners [25]. Therefore, the automatic recommendation in Protus is based on the author's intended sequence of navigation in the course material, or on navigation patterns of other successful learners. This technique is used for recommending shortcuts or jumps to some resources to help learners better navigate the course materials [24]. Protus recognizes different patterns of learners' interaction with the system and compares benefits it caused to learner. After that, Protus completes personalized recommendation of the learning content according to the ratings of these patterns.

Before significant database of lessons ratings (both entered by the learners and based on successful patterns) is formed, Protus uses default lessons sequence and chosen presentation methods (in pre course survey) for initial recommendation. Therefore, Protus does not have critical minimal group of learners for successful recommendation.

In order to support adaptive learning and personalization of a content delivery, the learner's knowledge and progress must be measured, and learner model must be constantly updated. According to that, the course can be possibly redirected.

## 3.3 Learner Model

Building of the learner model and tracking related cognitive processes are important aspects in providing personalization. The learner model is a representation of information about an individual learner that is essential for an adaptive system to provide the adaptation effect. The system uses the information from learner model to predict the learner's behavior, and thereby adapt to his/her individual needs. Data from learner model in Protus is classified along three layers that are suggested in [24]:

**Objective information,** which includes data supplied directly by the learner, such as: personal data, previous knowledge, preferences, etc. The learner edits this data during his/her registration on the system.

**Learner's performance,** which includes data about level of knowledge of the subject domain, his/her misconceptions, progress and the overall performance for a particular learner.

**Learning history,** which includes information about lessons and tests learner has already studied, his/her interaction with system, the assessments he/she underwent, etc.

The log file that collects information about learner's interaction with system contains: IP address of the request, the user name of the learner who interacts with system, the date and time of the request, the result of the request (error, success, failure, etc.), numeric data presented in request, etc. A log entry is automatically updated each time a request or action reaches the web server.

In order to accomplish successful categorization of learners we tracked characteristics of the learner and collected a variety of useful information:

- information about the learner, including cognitive, affective and social characteristics,
- information about the learner's perspectives on the content itself, including the learner's feedback on the content, the learner's knowledge of the content (as determined, for example, by a test administered during the learner's interactions with the system),
- information about the technical context of use, including characteristics of the learner's software and hardware environment,
- information about how the learner interacts with content, including observed metrics such as dwell time, number of learner keystrokes, patterns of access.

Gathered information will be classified along three layers in learner model that is presented in Figure 4.

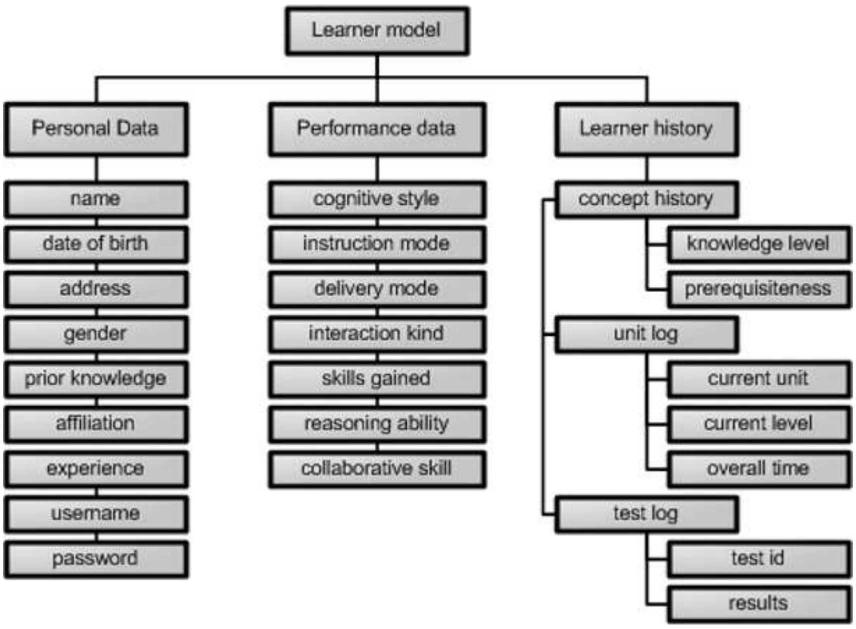


Fig. 4. Layers in learner model

## 4 SYSTEM ARCHITECTURE AND DESIGN

The general architecture of Protus is presented in this section as a means for implementing programming course with personalization options. Java programming course as the first fully implemented course in Protus from a learners' perspective will be presented in Section 5.

Protus is a tutoring system designed to help learners in learning basics of programming languages [41]. In spite of the fact that this system is designed and implemented as a general tutoring system for different programming languages, the first completely implemented and tested version of the system was for introductory Java programming course. Java is chosen because it is a clear example of an object-oriented language and is therefore suitable for teaching of object-oriented concepts [42]. The environment is designed for learning programming basics for learners with no object-oriented programming experience. It is an interactive system that allows learners to use teaching material prepared within appropriate course and also includes part for testing acquired knowledge.

Protus is designed as an extension of existing web-based Java tutoring system – Mag [42]. This previous version only used basic adaptation techniques and did not contain any recommendation of material involved in personalization process. In order to allow the specification of ontologies we used open standards, like XML, RDF and OWL. They allow standardization and formalization of content and enable the reuse and interoperability. These standards have not been implemented in Mag. Therefore, we propose completely redefined architecture for Protus that provides modular semantic web-based adaptive hypermedia architecture as a service-oriented system [22]. Protus architecture is based on experiences gained from similar web-based learning systems [5, 28] and an architecture for ontology-supported adaptive web-based education systems suggested in [6] and [8]. Figure 5 depicts the general architecture of redesigned and extended version of our previous system.

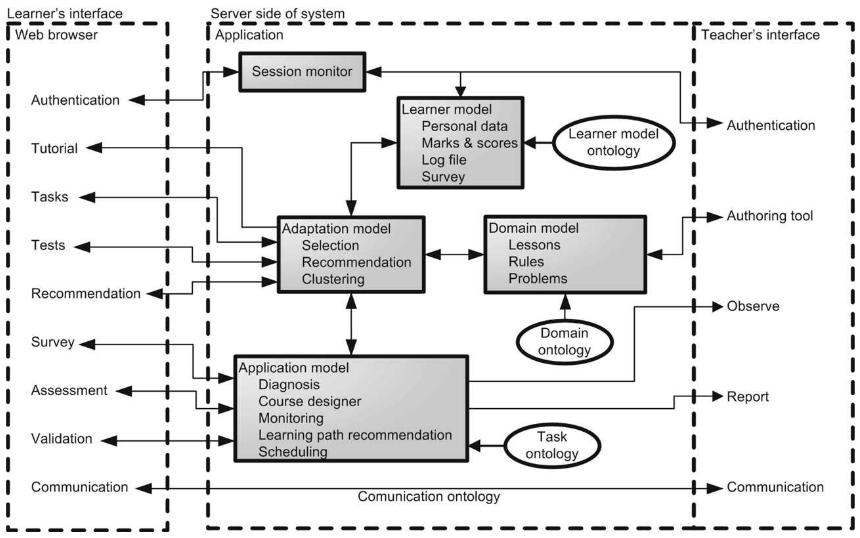


Fig. 5. Protus system architecture

Beside being beneficial for providing learners with personalized learning experience, the implemented architecture and the reasoning that is performed over it are also useful for generating feedback for other key participants in the learning process – content authors and teachers. Likewise, the framework can be used to provide feedback to teachers about the learners' activities, their performance, achieved knowledge level and so on. In both cases, the feedback can help in improving the learning process. To support this statement, several goals had been fulfilled in the Protus system [22]:

- separation of the two different interfaces – for learners and teachers,
- a strict separation of different modules: domain, application, adaptation and learner, in order to ensure a good modularization of the system components,
- permanent administration of learning progression, preferences and personal data of learners within sharable and dynamic learner model,
- enabling communication and collaboration among learners and between learners and teachers,
- assessment of knowledge and increasing competency level of learners,
- functionalities for creation of new learning content and migration of content from external sources,
- semantically rich descriptions of the components' functionality, in order to allow effective interoperability among system components, and
- providing effective coordination and communication between the system components.

As a result of the Protus design, essentially a centralized architecture of the system has been reached [22]. The core of the system includes the *adaptation*, *application* and *domain modules* as well as *learner model*, all of them stored on a central server.

*Domain module* presents storage for all essential learning material, tutorials and tests. It describes how the information content is structured. The whole course is divided into units, all consisting of several lessons (Figure 6) [25]. Every lesson (out of eighteen) contains three basic parts: theory session (tutorials), examples and tests. To every lesson unlimited number of examples and tests can be attached. Teachers can add new learning material using appropriate authoring tool.

*The adaptation module* is responsible for building and updating learner's model characteristics and also for personalization of content to be presented to the learner. It processes changes of learner's characteristics based on learner's activities and provides an adaptation of visible aspects of the system for specific learner. Its main tasks also include storage and management of learning material, presenting that material to learners, generating of reports and test results, etc.

Each *learner model* is a collection of both static and dynamic data about the learner. Static data include personal data, specific course objectives, etc. Dynamic data include scores, time spent on a specific lesson, marks, etc. In addition to the above-mentioned data, the learner model also contains a representation of the

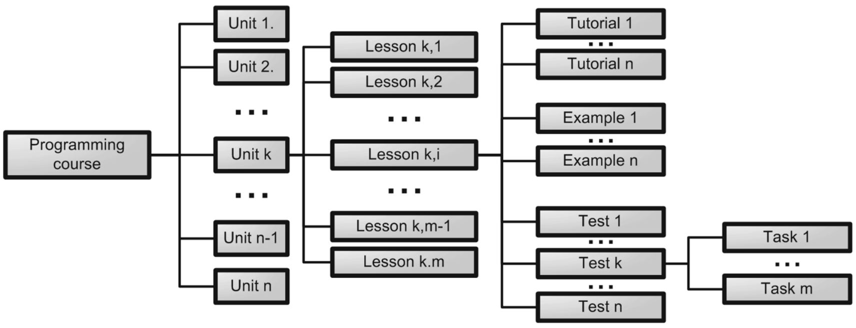


Fig. 6. Course material hierarchy

learner's performance and learning history. The system uses that information in order to predict the learner's behavior, and thereby adapt to his/her individual needs.

Within the *session monitor* component, the system gradually re-builds the learner model during the session, in order to keep track of the learner's actions and his/her progress, to detect and correct his/her errors and possibly redirect the session accordingly. At the end of the session, all learners' preferences are recorded in learner model. The learner may change this information at any time by editing his/her preferred learning style. Therefore, if a learner does not agree with the system's assumptions about his/her preferences, he/she can inspect his/her learner model and make changes in it during learning session [25]. Learner model is then used along with other information and knowledge to initialize the next session for the same learner.

The *application module* performs the adaptation. To be exact, the adaptation module follows the instructional directions specified by the application module. These two components are separated in order to make easier process of adding new content clusters and adaptation functionalities.

Protus provides means for learners to improve their learning experience by online communication and collaboration activities. It helps learners identify their problems and eventually find solutions by activities such as chat and forum.

Application of ontology engineering is a key aspect for the success of our web-based educational systems. Educational ontologies for different purposes can be included in architecture design, such as presenting a domain (*domain ontologies*), building learner model (*learner model ontologies*) or presenting of activities in the system (*task ontologies*) [8]. A repository of ontologies was built to achieve easier knowledge sharing and reuse, more effective learner modeling and easier extension of a system. Ontologies are structured following the SCORM (Sharable Content Object Reference Model) e-learning standard [35]. This ontological representation (OWL/RDF) enables not only to represent meta-data but also reasoning in order to provide the best solution for each individual learner.

The original architecture of Mag, and later Protus, did not bring any kind of homogenous representation of components. Each one was represented by different formats, using a variety of tools. The purpose of our current research activities, is to represent each component of the system in form of the ontology. Each component will be responsible for specific tasks. According to that, the level of abstraction of this architecture will be higher. This approach will make it easier to understand the role of each component and, consequently, to promote interoperability among the components of the architecture.

The minor adjustment can be performed on Protus in order to adapt it for performing different courses other than programming. That adjustment would include adding new course material and modifying testing abilities of the system.

### 4.1 Domain Module

One of the main goals of the learning process is to understand and to acquire a body of knowledge for a given domain [3]. Domain module presents storage for all essential learning material, tutorials and tests. It describes how the content intended for learning has to be structured. The domain module is structured as a taxonomy of concepts, with attributes and relations connecting them with other concepts, which naturally leads to the idea of using ontologies to represent this knowledge. Domain ontology within Protus includes:

- *Course taxonomy* (nodes that presents resource types in the programming domain). The most general resource type is *DomainResource*. DomainResource has three subtypes: *CourseMaterial* (theoretical explanations), *AdditionalMaterial* (practical explanations) and *ExaminationMaterial*. All these subtypes are further decomposed.
- *Domain knowledge* (nodes that presents actual learning objects). The root concept includes numerous sub-concepts like *Syntax*, *LoopStatements*, *ExecutionControl*, *Classes*, etc. Implementation of additional programming courses only implies adding new elements into existing ontology.

## 5 USER INTERFACES

Two main roles exist in the system, intended for two types of system's users:

**learners** – they are taking the Java programming course and will be using the system in order to gain certain knowledge and

**teachers and content authors** – the lesson and learner database administrator; they track the learning process of learners and help them with their assignments.

Therefore, separated user interfaces are provided for learner and teacher [42].

Teacher's interface helps in process of managing data about a learner and course material. Learner's interface is a series of web pages that provide two options: taking lessons and testing learner's knowledge. All data about learner and his/her progress in the course, as well as data about tutorials, tests and examples are stored in the system's server.

## 5.1 Learner's Interface

This section describes the user interface and explains the guidelines that were taken into account for its design. The new learner registers in the system by using an appropriate form in order to create a personal profile. After that, learners ought to answer short optional survey where they need to choose their own preferred learning style that indicates a preference for some previously mentioned presentation methods over others. These results are stored in a learner model, which will be used for the initial adaptation in Protus [22].

When a learner is logged in, a session is initiated based on learner's specific data and sequence of lessons is recommended to him/her. Lessons are grouped into units. Initial order of lessons in implemented Java programming course is presented in Figure 7. A learner has the possibility to change the order in which he/she will attend lessons. After selecting a lesson from the collection of lessons available in Protus (Figure 8), system chooses presentation method of the lesson based on the learner's preferred style. For the rest of the lesson, learners are free to switch between presentation methods by using the media experience bar.

For every lesson the same sequence of activities has to be followed. At the beginning of a lesson, participants are pre-tested with multiple choice questions and fill in appropriate entering questionnaire. In the next step, participants are shown a short introductory text on the lesson's topic (Figure 9), additionally explained with appropriate examples (Figure 10).

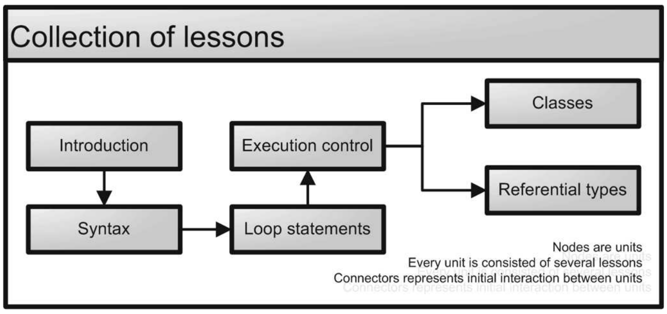


Fig. 7. Collection of lessons and their interaction

At the end of each lesson a post-test is conducted. The test contains several multiple-choice questions and code completion tasks. This time Protus provides feedback on their answers and gives the correct solutions after the test. The post-test section is followed by a lesson summary and a lesson feedback form. Through

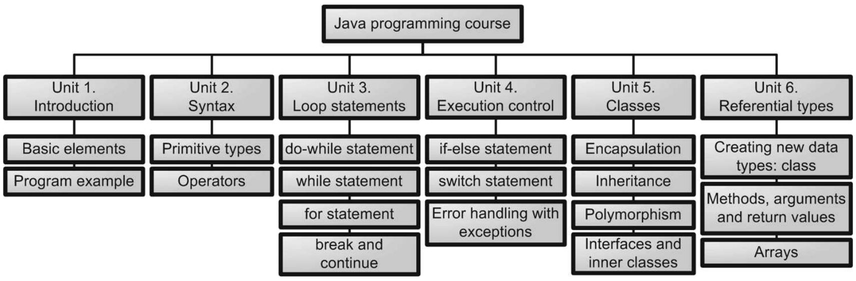
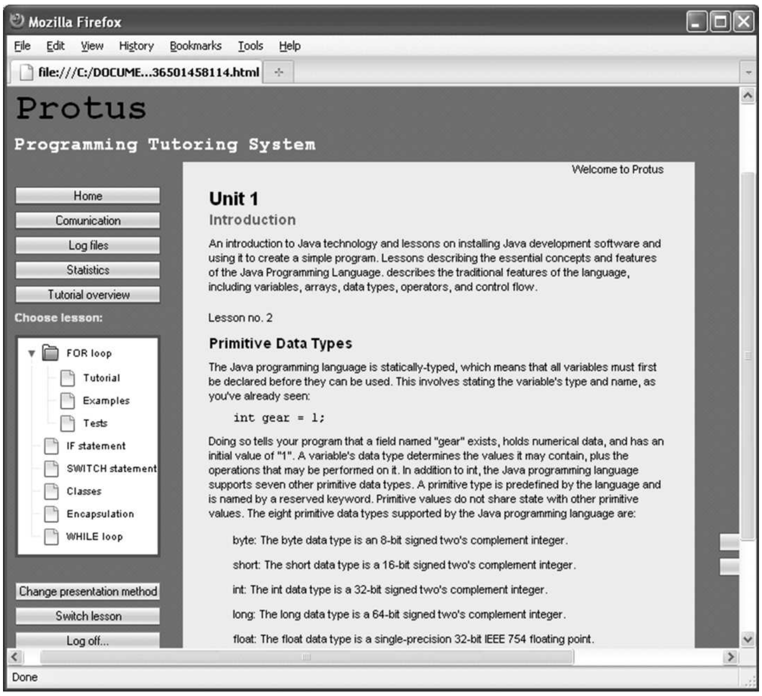


Fig. 8. Lessons hierarchy in Java programming course

this form, participants rate the presentation method(s) they used in that lesson and answer questions about their perceived enjoyment, progress, and motivation. Given grades are in the range from 1 (lowest satisfaction) to 5 (highest satisfaction). Participants could also leave additional comments in free-text fields.

When the learner completes the sequence of learning materials, the system evaluates the learner's knowledge degree for each lesson. The test contains several multiple-choice questions and code completion tasks. Protus then provides feed-



Unit 1

Introduction

An introduction to Java technology and lessons on installing Java development software and using it to create a simple program. Lessons describing the essential concepts and features of the Java Programming Language. describes the traditional features of the language, including variables, arrays, data types, operators, and control flow.

Lesson no. 2

Primitive Data Types

The Java programming language is statically-typed, which means that all variables must first be declared before they can be used. This involves stating the variable's type and name, as you've already seen:

```
    int gear = 1;
```

Doing so tells your program that a field named "gear" exists, holds numerical data, and has an initial value of "1". A variable's data type determines the values it may contain, plus the operations that may be performed on it. In addition to int, the Java programming language supports seven other primitive data types. A primitive type is predefined by the language and is named by a reserved keyword. Primitive values do not share state with other primitive values. The eight primitive data types supported by the Java programming language are:

byte: The byte data type is an 8-bit signed two's complement integer.

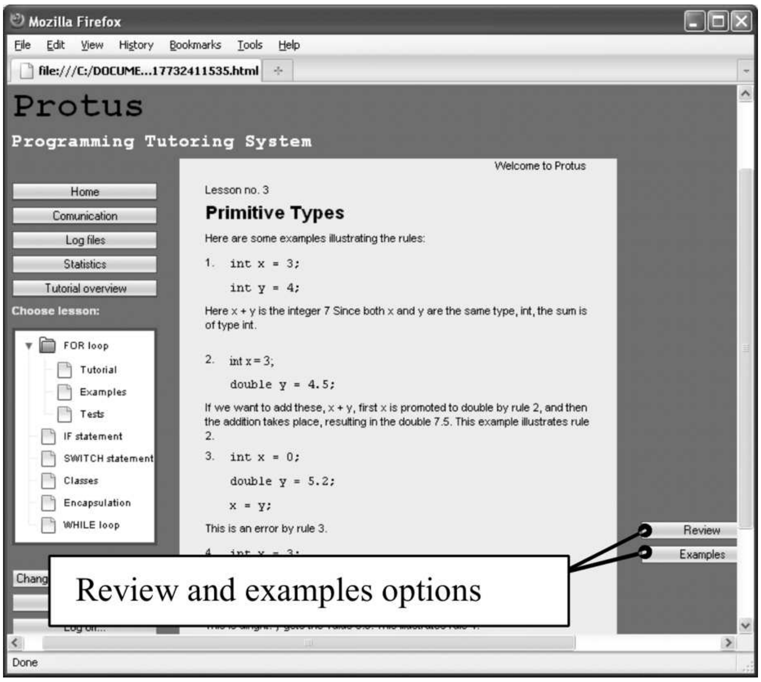
short: The short data type is a 16-bit signed two's complement integer.

int: The int data type is a 32-bit signed two's complement integer.

long: The long data type is a 64-bit signed two's complement integer.

float: The float data type is a single-precision 32-bit IEEE 754 floating point.

Fig. 9. Lessons tutorial



Lesson no. 3

**Primitive Types**

Here are some examples illustrating the rules:

1. `int x = 3;`

   `int y = 4;`

Here x + y is the integer 7 Since both x and y are the same type, int, the sum is of type int.

2. `int x = 3;`

   `double y = 4.5;`

If we want to add these, x + y, first x is promoted to double by rule 2, and then the addition takes place, resulting in the double 7.5. This example illustrates rule 2.

3. `int x = 0;`

   `double y = 5.2;`

   `x = y;`

This is an error by rule 3.

4. `int x = 3;`

Fig. 10. Lessons example

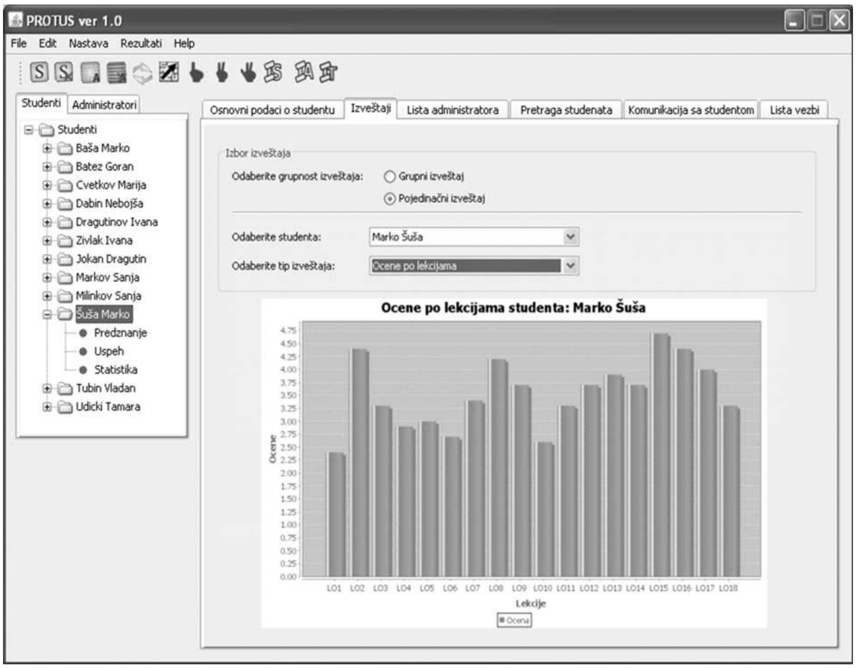
back to the learner on his/her answers and gives the correct solutions after the test.

When the learners have visited all lessons within one unit they have to fill-in a final test to evaluate their accepted knowledge about the unit.

### 5.2 Teacher's Interface

Besides being beneficial for providing learners with personalized learning experience, the Protus system is also useful for generating feedback for other participants in the learning process-content authors and teachers. Content authors are typically subject matter experts who create learning content, that is subsequently used by teachers who wrap that content into a learning design. Protus can be used to provide feedback to teachers about the learners' activities, their performance, achieved collaboration level and the similar activities. In both cases, the feedback helps in improving the learning process.

Protus aims at helping teachers rethink the quality of the learning content and learning design of the course they teach. To this end, the system provides teachers with feedback about the relevant aspects of the learning process taking place in the online learning environment they use. The provided feedback is based on the



Odaberite grupnost izveštaja: Grupni izveštaj [✓ Pojedinačni izveštaj]

Odaberite studenta: Marko Šuša

Odaberite tip izveštaja: Ocene po lekcijama

Fig. 11. Teacher's interface

analyses of the context data collected in the learning environment. In particular, Protus informs teachers about:

- the activities the learners performed during the learning process,
- the usage of learning content they had prepared and deployed in the tutoring system,
- the peculiarities of the interaction among learners.

Figure 11 depict the graphic interface that represents the real interaction of the teachers with Protus and where the assessment and tracking data and statistics are generally shown. This form facilitates data retrieval and provides appropriate results for the teacher. The teacher can combine parameters and filters in order to obtain reports that will be presented in form of charts and tables. The chart type varies according to the selected filters. For example, the teacher could know what specific material was more used by learners, what kind of learning style they preferred or what grades they earned for every particular lesson. These reports can show results for group of learners or for every learner separately.

## 6 EXPERIMENTAL RESULTS

In order to test our designed system, as it was presented in Sections 3–4, we have carried out some experiments on an educational dataset. The experiment was running for almost four months, from October 2009 until January 2010. We selected 70 students of Higher School of Professional Business Studies, Novi Sad University. These students firstly underwent an individual pre-test to assess their prior knowledge of Java programming language. Then, they were divided into two groups: Experimental group and Control group, each group consisting of 35 students. Students in Control group learned in a normal way and did not receive any recommendation or guidance through the course. The course was delivered to them based on Mag e-learning platform [42] (without recommendation and personalization options), while the students in Experimental group were required to use the Protus system [25]. Learners from both groups did not take any parallel traditional course. They were required not to use any additional material or help except that already provided in the system, because we presumed that it would be sufficient in terms of controlling variables and, if that is the case, experiment results will show uninfluenced improvement of the learners.

Both experiments consisted of three independent sessions. In the Theory session the teaching unit on the essential Java concepts and features had been studied by both groups, but only the students from the Experimental group were guided by recommendation generated by recommendation component of the system. In the second (Exercising) session students worked on simple programming exercises, and in the third (Final exam), they had to solve a complex programming task consisting of representative tasks which require knowledge of essential concepts presented in delivered lessons (Figure 12). All exercises and exams are also integrated in the system.

In order to assess whether the means of two groups are statistically different from each other, the t-test was utilized. Both groups of learners completed the Norm-referenced test which allows us to compare learners' intellectual abilities [16]. Results of this test were combined with grades that learners earned at a basic computer literacy course in the first semester of their studies. The aim of the computer literacy course was to teach data structures and algorithms by presenting exercises of algorithm simulations to the learners. Programming coursework in any programming language was not assessed [25]. The most important outcome was therefore the introduction of general problem solving concepts, rather than focusing on teaching the syntax of a specific programming language.

The predetermined alpha level adopted for hypothesis testing was 0.05, as significance levels of less than 0.05 are considered statically significant, degree of freedom (df) for the test was 68. Table 1 reports the obtained $t$-test results. Since the calculated value of t (0.81) is not greater than table value of $t$ (2.00), we can conclude that the differences between the experimental and the control group are negligible and there is no need for additional equalization of groups.

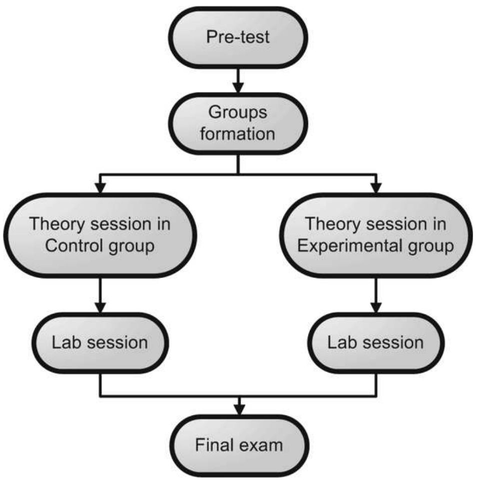

Pre-test
Groups formation
Theory session in Control group
Theory session in Experimental group
Lab session
Lab session
Final exam

Fig. 12. Graphic depictions of the Experimental procedures

| Type of test | Group | N | Df | Mean | $t$ (calculated value) | $t$ (table value) |
|---|---|---|---|---|---|---|
| Intelectual abilities | Experimental | 35 | 68 | 80.25 | 0.81 | 2.00 |
| | Control | 35 | | 74.69 | | |
| Level of significance $\alpha = 0.05$ | | | | | | |

Table 1. The analysis of the test score difference between the two groups

During the experimental period of four months, grading was made after every lesson. We monitored the completion of lessons and the time needed by the students for these completions. The experiment results showed that the students in the Experimental group should be able to complete a course in less time than students in the Control group. We further assumed that significant number of students in the Experimental group should reach a goal of eighteen delivered lessons in less time. After the experiment, all the students were given a post-test. To try and control confounding variables related to the students' history, no other Java lessons were taught during the period of the experiment and the students were not given any homework on the topics. Finally, comparison of the results obtained by each student, in the pre-test and in the post-test, allowed us to evaluate not only the gained knowledge but also how much the personalized course had contributed to improve their knowledge.

A pedagogy driven recommender system for e-learning that takes into account learner characteristics and specific learning demands should also be evaluated by learning evaluation criteria. As suggested in [9] we used educational research measures. Educational research measures are needed to evaluate whether learners actually do benefit with usage of a recommender system. Therefore we suggest the following measures for the analysis of the recommender systems suitability in e-learning.

From an educational point of view, learners only benefit from learning technology when it makes learning more effective, efficient, or more attractive. In educational research, efficiency, effectiveness and satisfaction are the common measures. Efficiency indicates the time that learners needed to reach their learning goal. Effectiveness is a sign of the total amount of completed, visited, or studied lessons during a learning phase. It is related to the efficiency variable through counting the actually study time. Satisfaction reflects the individual satisfaction of the learners with the given recommendations. Satisfaction is close to the motivation of a learner and therefore a rather important measure for learning.

Figure 13 shows grades comparison between Control and Experimental groups as a measure of efficiency. Furthermore, Figure 14 demonstrates that the Experimental group needed less time to complete the lessons successfully and shows that the Experimental group continuously completed more lessons successfully than the Control group.

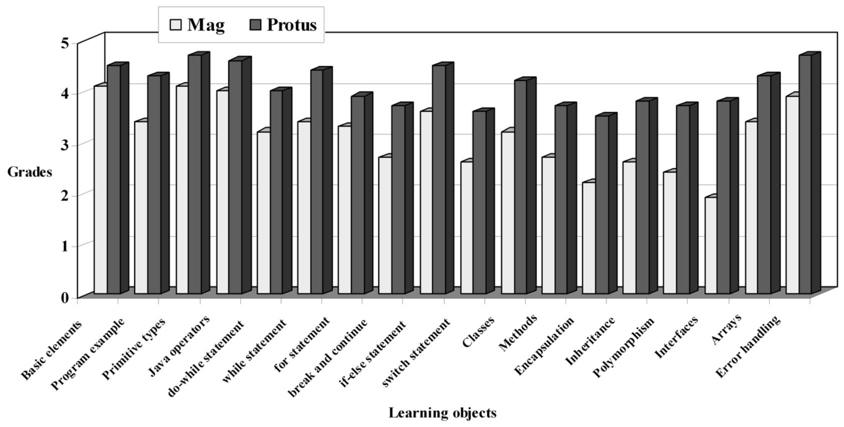


Fig. 13. Grades comparison between Control and Experimental groups

## 6.1 Evaluation Based on Learning Curve

The *learning curve* is a popular method to evaluate the students' learning effects. It is widely used in to estimate effectiveness of teaching methods and training models. In this section, we discuss the average learning curve comparison between Control group and Experimental group students. As mentioned earlier, both courses (taken by Mag and Protus) contain eighteen lessons, which have been grouped into six units. After completion of each unit, students were given appropriate tests in order to assess their gained knowledge. Only for those students who pass given tests,

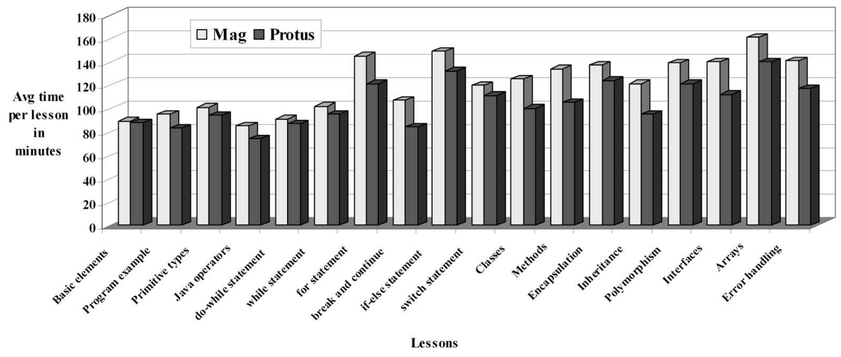


Fig. 14. Efficiency comparison between Control and Experimental groups

a final exam was provided. The main motivation for this activity was to certify if the usage of the Protus system really helped students enhance learning effect.

The detailed achievement comparison for all units between Control and Experimental groups can be seen in Figure 15. For students in the Experimental group we recognize an ascending trend with higher acceleration than for students the in Control group.

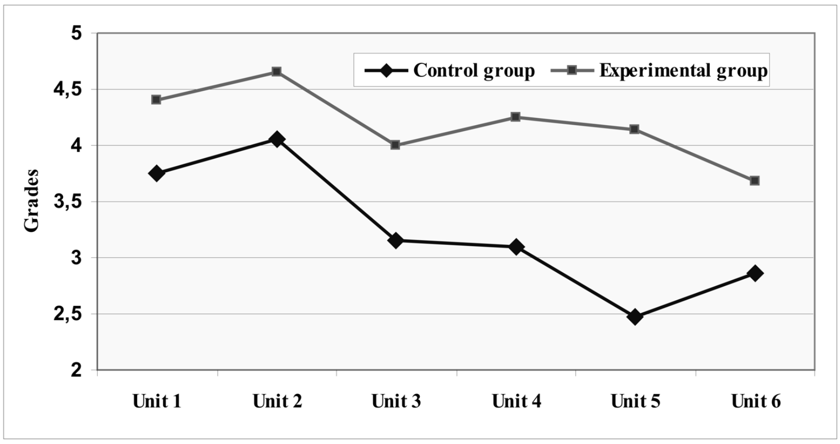


Fig. 15. Achievement comparison between Control and Experimental groups

## 6.2 Evaluation of Satisfaction

To obtain more subjective evaluation of our system, at the end of the course we prepared a non-mandatory questionnaire to be filled-in by students (from the Experimental group). The questionnaire helped us collect students' opinions about the main features of the system. Out of 35 students, 30 students filled the questionnaire. Results of processed answers are shown in Figure 16.

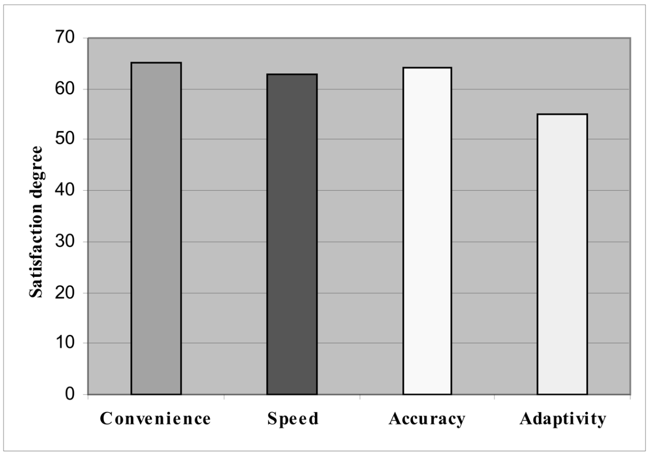


Fig. 16. Students' opinions about the main features of the system

The figure shows that more than 60 % of students considered the system convenient, meaning that it has helped learning. Two principal features of system's speed and accuracy were evaluated. 63 % and 64 % of the students are satisfied with the speed and accuracy, respectively. Most learners found that the material was processed faster while working with the system, compared to other courses they attended in the traditional way. Also, the accuracy of the selection of appropriate examples is high. Moreover, more than 50 % of students considered the system as adaptive.

Our experimental results show that a combination of the learners' learning styles recognition and mining the frequent sequences in the web logs, which can be used in the collaborative filtering approach, has the potential to improve the quality of an intelligent tutoring system, as well as to keep the recommendation up-to-date.

The measure of influence of the implemented adaptation techniques is performed in global, not for each recommendation or adaptive hypermedia technique separately. Therefore, based on the previously presented results we conclude that the system has achieved a remarkable impact on learners' self-learning. Taking advantage of the system, the learners have gained more knowledge in less time, which was confirmed by efficiency checks throughout the course.

## 7 CONCLUSIONS

E-learning can use different recommendation techniques in order to suggest the most appropriate online learning activities to learners, based on their preferences, knowledge and the browsing history of other learners with similar characteristics. The ultimate goal is improving the teaching and learning process by providing the learners with personalized course.

In this paper we presented an approach in developing adaptive web-based programming tutoring system for learning essential concepts of Java programming language. The main goal of the system was to offer features for adaptation and personalization of both the content and the navigation to learner's needs and goals, in order to demonstrate how recommendation and adaptive hypermedia techniques can enhance the adaptation and interoperability of an e-learning system. The proposed system aims at automatically guiding the learner's activities and recommend relevant links and actions to him/her during the learning process.

To determine the efficiency of this system, we performed three kinds of evaluations at the end of the semester during which the system was tested. They include pre-test and post-test learning grading, learning curves, and subjectively perceived satisfaction of this system. The empirical results show that the system can support learners to enhance their learning effort towards an ascending learning curve and better grades. Also, most students who used the system were satisfied with its efficient and helpful services. Experiments also show the suitability of using both recommendation and hypermedia techniques in order to suggest online learning activities to learners based on their preferences, knowledge and also on the opinions of a community of learners with similar characteristics.

Provided with recommendations, learners can learn more conveniently than before with a system that meets their needs and interests. It has been shown that including learner's learning style into recommendation strategy is useful for better interpretation of the learner cluster, which can be used to identify frequent sequences of navigational pattern in each cluster. These sequences are important for generating recommendations based on the collaborative filtering approach.

The developed system is highly modular, which allows better flexibility and future replacement and addition of various components as long as they comply with the current interface. Although ontologies have a set of basic implicit reasoning mechanisms derived from the description logic which they are typically based on (such as classification, instance checking, etc.), they need rules to make further inferences and to express relations that cannot be represented by ontological reasoning. Thus, ontologies require a rule system to derive/use further information that cannot be captured by them, and rule systems require ontologies in order to have a shared definition of the concepts and relations mentioned in the rules. For the future work we plan to present details of ontologies employed in our system along with rules that will allow adding expressiveness to the representation formalism, and reasoning on the instances. Also, we aim to carry out a more detailed study involving more students, and perform experiments to evaluate used recommendation techniques.

**Acknowledgment**

This paper is part of the research project "Infrastructure for Technology Enhanced Learning in Serbia" supported by the Ministry of Education and Science of the Republic of Serbia [Project No. 47003].

## REFERENCES

[1] Bădică, C.—Budimac, Z.—Burkhard, H.—Ivanović, M.: Software Agents: Languages, Tools, Platforms. Computer Science and Information Systems, Vol. 8, 2011, No. 2, pp. 255–296.

[2] Bieg, C.—Diehl, S.: Education and Technical Design of a Web-Based Interactive Tutorial on Programming in Java. Science of Computer Programming, 2004, pp. 25–36.

[3] Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Modelling, Vol. 6, No. 2–3. 1996, pp. 87–129.

[4] Brusilovsky, P.—Schwarz, E.—Weber, G.: ELM-ART: An Intelligent Tutoring System on the World Wide Web. Third International Conference on Intelligent Tutoring Systems, Montreal, Canada, 1996, pp. 261–269.

[5] Chen, C. M.: Ontology-Based Concept Map for Planning a Personalized Learning Path. British Journal of Educational Technology, Blackwell Publishing, 2008, pp. 1–31.

[6] de Bra, P.—Aroyo, L.—Chepegin, V.: The Next Big Thing: adaptiveWeb-Based Systems. Journal od Digital Information 5, 2004, Article No. 247.

[7] de Bra, P.: Web-Based Educational Hypermedia. In: C. Romero and S. Ventura (Eds.): Data mining in e-learning WIT Press, Southampton, Boston, UK 2006, pp. 3–19.

[8] Devedžić, V.: 2006 Semantic Web and Education 1st ed. Springer Science, New York.

[9] Drachsler, H.—Hummel, H.—Koper, R.: Identifying the Goal, User Model and Conditions of Recommender Systems for Formal and Informal Learning. Journal of Digital Information 10, 2009, pp. 4–24.

[10] Dunn, R.—Dunn, K.: Teaching Students Through Their Individual Learning Styles: A Practical Approach. Reston Publishing, Reston 1978.

[11] Emurian, H.: A Web-Based Tutor for Java: Evidence of Meaningful Learning. International Journal of Distance Education Technologies, Vol. 4, 2006, No. 2, pp. 10–30.

[12] Farzan, R.—Brusilovsky, P.: Social Navigation Support in a Course Recommendation System. Proceedings of 4th International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Dublin, 2006, pp. 91–100.

[13] Fertalj, K.—Hoiíc-Bošić, N.—Jerković, H.: The Integration of Learning Object Repositories and Learning Management Systems. Computer Science and Information Systems, Vol. 7, 2010, No. 3, pp. 387–407.

[14] Ghali, F.—Cristea, A.: MOT 2.0: A Case Study on the Usefuleness of Social Modeling for Personalized E-Learning Systems. Frontiers in Artificial Intelligence and Applications, Vol. 200, 2009, pp. 333–340.

[15] García, P.—Amandi, A.—Schiaffino, S.—Campo, M.: Evaluating Bayesian Networks' Precision for Detecting Students' Learning Styles. Computers and Education, Vol. 49, 2007, pp. 794–808.

[16] Glaser, R.: Instructional Technology and the Measurement of Learning Outcomes. American Psychologist, Vol. 18, 1963, pp. 510–522.

[17] Glenn, B.—Shahida, P.—Wei, F.—Moritz, S.: A Web-Based ITS for OO Design. 12th International Conference on Artificial Intelligence in Education, Amsterdam 2004, pp. 59–64.

[18] Graf, S.—Liu, T.—Kinshuk, C.: Analysis of Learners' Navigational Behavior and Their Learning Styles in an Online Course. Journal of Computer Assisted Learning, Vol. 26, 2010, No. 2, pp. 116–131.

[19] Herlocker, J. L.—Konstan, J. A.—Borchers, A.—Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. Proceedings of SIGIR '99 – The 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkley, USA 1999, pp. 230–237.

[20] Holland, J.—Mitrovic, A.—Martin, B.: J-Latte: A Constraint-Based Tutor for Java. In 17th International Conference on Computers in Education, Hong Kong, 2009, pp. 142–146.

[21] Ivanović, M.—Pribela, I.—Vesin, B.—Budimac, Z.: Multifunctional Environment for E-Learning Purposes. Novi Sad Journal of Mathematics, Vol. 38, 2008, No. 2, pp. 153–170.

[22] Klašnja-Milićević, A.—Vesin, B.—Ivanović, M.—Budimac, Z.: Integration of Recommendations Into Java Tutoring System. The International Conference on Information Technology, Amman, Jordan, CD proceedings, 2009, paper No. 524.

[23] Klašnja-Milićević, A.—Nanopoulos, A.—Ivanović, M.: Social Tagging in Recommender Systems: a Survey of the State-of-the-Art and Possible Extensions. Published online in Artificial Intelligence Review, Springer 2010, pp. 187–209.

[24] Klašnja-Milićević, A.—Vesin, B.—Ivanović, M.—Budimac, Z.: Integration of Recommendations and Adaptive Hypermedia Into Java Tutoring System. Computer Science and Information Systems, Vol. 8, 2011, No. 1, pp. 211–224.

[25] Klašnja-Milićević, A.—Vesin, B.—Ivanović, M.—Budimac, Z.: E-Learning Personalization Based on Hybrid Recommendation Strategy and Learning Style Identification. Computers and Education 56, 2011, pp. 885–899.

[26] Ksristofic, A.: Recommender System for Adaptive Hypermedia Applications. Proceeding of Informatics and Information Technology Student Research Conference, Bratislava 2005, pp. 229–234.

[27] Merceron, A.—Yacef, K. A.: Web-Based Tutoring Tool with Mining Facilities to Improve Learning and Teaching. Artificial Intelligence in Education, IOS Press 2003, pp. 201–208.

[28] Merino, P. J. M.—Kloos, C. D.: An Architecture for Combining Semantic Web Tehniques With Intelligent Tutoring Systems. ITS 2008, Springer-Verlag Berlin Heidelberg 2008, pp. 540–550.

[29] Mobasher, B.: Data Mining for Personalization. The Adaptive Web: Methods and Strategies of Web Personalization, Springer-Verlag, Berlin Heidelberg 2007, pp. 1–46.

[30] Myller, N.: Automatic Prediction Question Generation During Program Visualization. Electron. Notes Theor. Comput. Sci. 178. 2007, pp. 43–49.

[31] Resnick, P.—Iacovou, N.—Suchak, M.—Bergstrom, M.—Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Proceedings of ACM Conference on Computer Supported Cooperative Work, Chapel Hil., 1994, pp. 175–186.

[32] Romero, C.—Ventura, S.—Delgado, J. A.—de Bra, P.: Personalized Links Recommendation Based on Data Mining in Adaptive Educational Hypermedia Systems. Creating New Learning Experiences on a Global Scale, 2007, pp. 292–306.

[33] Romero, C.—Ventura, S.: Preface. In: C. Romero and S. Ventura (Eds.), Data mining in E-learning, WIT Press, Southampton, Boston, UK 2006, pp. 3–19.

[34] Schafer, J. B.: The Application of Data-Mining to Recommender Systems. Encyclopedia of data warehousing and mining, Hershey, PA. Idea Group, 2005, pp. 44–48.

[35] SCORM: Sharable Content Object Reference Model e-learning standard 2010. Retrieved 10/02/2010 from `http://www.adlnet.gov/Technologies/scorm/default.aspx`.

[36] Soonthornphisaj, N.—Rojsattarat, E.—Yim-Ngam, S.: Smart E-Learning Using Recommender System. ICIC Springer-Verlag, Berlin Heidelberg 2006, pp. 518–523.

[37] Suarez, M.—Sison, R.: Automatic Construction of a Bug Library for Object-Oriented Novice Java Programmer Errors. ITS 2008, Springer-Verlag Berlin Heideberg 2008, pp. 184–193.

[38] Sykes, E. R.—Franek, F.: An Intelligent Tutoring System Prototype for Learning to Program Java. The third IEEE International Conference on Advanced Learning Technologies (ICALT '03), Athens, Greece 2003, pp. 485–492.

[39] Sykes, E.: Developmental Process Model for the Java Intelligent Tutoring System. Journal of Interactive Learning Research Chesapeake, Vol. 18, 2007, No. 3, pp. 399–410.

[40] Vesin, B.—Ivanović M.—Budimac, Z.: Learning Management System for Programming in Java. Annales Universitatis Scientiarum de Rolando Eotvos Nominatae, Sectio Computatorica 31, 2009, pp. 75–92.

[41] Vesin, B.—Ivanović, M.—Budimac, Z.—Pribela, I.: MILE – Multifunctional Integrated Learning Environment. IADIS Multi Conference on Computer Science and Information Systems (MCCSIS 2008), Amsterdam, Netherlands 2008, pp. 104–108.

[42] Vesin, B.—Ivanović, M.: Modern Educational Tools. Proceedings of PRIM2004 – 16th Conference on Applied Mathematics, Budva, Montenegro 2004.

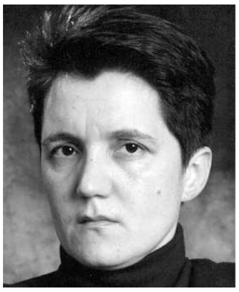
[43] Wei, F.—Moritz, S. H.—Parvez, S. M.—Blank, G. D.: A Student Model for Object-Oriented Design and Programming. Journal of Computing Sciences in Colleges, 2005, pp. 260–273.

[44] Wolf, C.: iWeaver: Towards 'Learning Style'-Based E-Learning in Computer Science Education. Australasian Computing Education Conference, Adelaide, Australia 20, 2003, pp. 273–279.

[45] Zaiane, O. R.: Recommender Systems for E-Learning: Toward Non-Intrusive Web Mining. In: C. Romero and S. Ventura (Eds.): Data Mining in E-Learning. WIT press, Southampton, Boston, UK, 2006, pp. 79–96.

**Boban Vesin** has graduated from the Faculty of Science, University of Novi Sad, in 2002. He received his master degree at the same faculty in 2007 and is working towards his Ph. D. thesis. Currently he is a lecturer at Higher School of Professional Business Studies, University of Novi Sad, Serbia. His major research interests include e-Learning and personalization in intelligent tutoring systems. He has published a number of scientific papers in the field.

**Aleksandra Klašnja-Milićević** is a lecturer of computer science at Higher School of Professional Business Studies at University of Novi Sad, Serbia. She graduated in electrical and computer engineering from Faculty of Technical Sciences at the University of Novi Sad in 2002. She joined the graduate program in computer sciences at Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad in 2003, where she received her M. Sc. degree (2007). Her research interests include recommender systems, information retrieval, user modeling and personalization, and electronic commerce. She has published more than 20 referreed papers on her work in national and international conferences and journals.



**Mirjana Ivanović** holds the position of Full Professor at Faculty of Sciences, University of Novi Sad, Serbia since 2002. She is the Head of Chair of Computer Science and a member of University Council for Informatics. She is author or co-author of 13 textbooks and of more than 230 research papers on multi-agent systems, e-learning and web-based learning, software engineering education, intelligent techniques (CBR, data and web mining), most of which are published in international journals and conferences. She is/was a member of Program Committees of more than 80 international conferences and is the Editor-in-Chief of Computer Science and Information Systems Journal.

**Zoran Budimac** holds the position of Full Professor at Faculty of Sciences, University of Novi Sad, Serbia since 2004. Currently, he is the Head of the Computing Laboratory. His fields of research interests include educational technologies, agents and WFMS, case-based reasoning, and programming languages. He was principal investigator of more than 20 projects. He is the author of 13 textbooks and more than 220 research papers, most of which are published in international journals and international conferences. He is/was a Program Committee member of more than 60 international conferences and is an Editorial Board Member of Computer Science and Information Systems Journal.