DISTRIBUTED SYSTEM FOR THE ACQUISITION OF SKILLS IN JAVA PROGRAMMING

Bandáková Jana, Čierny Marián, Samuelis Ladislav

ABSTRACT
This contribution describes briefly a distributed application, which main aim is to support testing procedures of students' during their Java programming skills acquisition and to introduce them interactively into the component based programming through a case-study. Application enables teachers to automize repetitive tasks of building groups of random tests, assignments and their evaluation. It also supports archiving of the results for their later evaluation and tracking of students' progress.

KEYWORDS
E-learning, three-layer architecture, Java tests, component based programming, tracking of students' progress.

MOTIVATION AND THE PURPOSE OF THE SYSTEM

Motivation for the development of the application originates from the observation that the most effective way of learning a new programming paradigm for novice programmers is by doing experiments with the already existing code. This observation is not new in principle and it was utilized for various purposes in various contexts in the past (e.g. Pšenáková I., 2000). The Java enabled distributed technology enables to build robust environment for the provision of learning-by-experimenting. From the e-learning systems engineering point of view, it is a learning management system (LMS, e.g. Learning Technology Standards Observatory, 2005) with limited capabilities. In other words the aim of this application is to support project-based learning environment for novice students. But a question still remains. What type of code (case-study) will we present to the learning community? We have decided that the case-study will be devoted to the development of a typical 3 layer distributed application. The case-study simulates the work of the automatic teller machine. From the architectural point of view, the case-study has the same architecture as the platform (LMS), which offers the case-study.

From the pedagogical point of view the purpose of the system is twofold.

1. The application, on one hand, automizes some boring repetitive pedagogical tasks, i.e preparation of tests, their random grouping and evaluation.
2. On the other hand, it provides a case-study and of chunks of codes for their later incremental change and in this way to better understanding of the code behavior.

These pedagogical tasks are not only automized as far as possible, but the system also provides a base for the later assessment of students' progress. We also state that a consequent evaluation of the progress of students' during their learning provides teachers with highly valuable information, which is necessary for raising the quality of the future learning materials. For this reason we have embedded archiving facilities for thorough tracking of the successfulness or failure of students' learning progress.

In the next paragraph we briefly describe the architecture and several selected functions of the system, which are available through the systems' graphical user interface.

ARCHITECTURE OF THE APPLICATION

The application implements the popular three-layer architecture (Horstmann C., 2002), as depicted on Figure 1. This architecture partitions the system into three logical layers:

- the user interface layer
- the business rules layer
- the database access layer

The three-layer architecture is used when an effective distributed client/server design is needed that provides (when compared to the two-layer) increased performance, flexibility, maintainability, re-usability, and scalability, while hiding the complexity of distributed processing from the user. The second layer (middle layer server) is between the user interface (client) and the data management (server) components. This middle layer provides process management where business logic and rules are executed and can accommodate hundreds of users by providing functions such as queuing, application execution, and database staging. There are a variety of ways of implementing this middle layer, such as transaction processing monitors, message servers, or application servers. In addition the middle layer adds scheduling and prioritization for work in progress. The three-layer client/server architecture improves the performance for groups with a large number of users. In the two-layer architectures database access functionality and business logic were often contained in the client component, any changes to the business logic, database access, or even the database itself, often required the deployment of a new client component to all users of the application. Multi-layer client/server architecture enhances the two-layer client/server architectures in two ways:

- It makes the application less fragile by further insulating the client from changes in the rest of the application.
- It allows more flexibility in the deployment of an application.

Multi-layer client/server reduces application fragility by providing more insulation and separation between layers. The user interface layer communicates only with the business rules layer, never directly with the database access layer. The business rules layer, in turn, communicates with the user interface layer on one side and the database access layer on the other. Thus, changes in the database access layer will not affect the user interface layer because they are insulated from each other. This architecture enables changes to be made in the application with no affects to the client component.

At the present time all three layers are implemented on Intel-based PCs and Linux type operating system. The application server hosts the Tomcat software, which is the open source implementation for servlets and the JSP technology, (http://phoenix.mis.cycu.edu.tw/Class/2001Su_J2EE). Tomcat also interprets Java commands and enables the communication with database through libraries.

The client is usually a browser such as Internet Explorer, Netscape, Opera etc. Browsers interact with the server through protocols. There are many protocols available on the Internet, which are used for this process. In our application the HTTP protocol (HyperText Transfer Protocol) is utilized. The architecture of the application is on the next Figure 1:

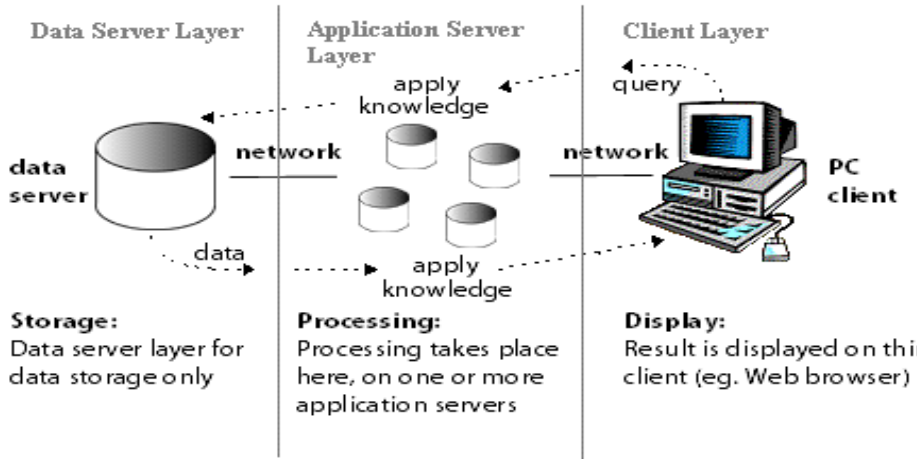

Figure 1. The three-layer architecture of the application

The database is processed by the MYSQL RDBMS, which may be located remotely. All pieces of information, which are necessary for the application, such as students' and teachers' usernames and passwords, personal information and students' test results, are stored in the database. The application server is used to process the queries to the database server. For example, if student or teacher enters into the system their usernames and passwords are sent to the application server, which forwards the query to the database server. The database server sends back the appropriate information. The database contains: test questions and answers, test results, students' and teachers' personal information, other pieces of auxiliary information.

JSP source code and the compiled code, which manages the communication between server and database and between server and client, are stored in specific directories. The compilation runs on the server with authorized access. If the compilation is successful the information message will be displayed. Servlets are products of the JSP code compilation.

The recent version of the application was developed on local PCs with the installed Borland JBuilder 2005 Enterprise software. Development of the application requires installation of the Tomcat software (Tomcat 4.0) for processing the JSP (Java Server Pages) technology (http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial), (Hall M., 2000) and for the cooperation with the MYSQL database (http://dev.mysql.com/doc/,2004).

GROUP OF FUNCTIONS PROVIDED FOR STUDENTS

Students enter the system through their "username" and "password". If the entry to the system is unsuccessful, an error message appears and the client can try to enter the system again. Successful entry into the system consists of two steps. Firstly, the system checks the entered "username" and "password" against their values in the database. Secondly, the system checks, whether the authorized client is already logged in. This approach allows the following:

- Only that student who is registered may enter the system.
- It is possible to define a definite time period for opening a session in the system.

The aim of the time interval definition is to control the access of students to tests during the practical labs and in this way to avoid cheating.

*Function "Registration"***.** This function is used for the registration of new students. Registration consists of two steps. Students register into the system and the basic information is stored into the database. Teachers later admit students to enroll into a specific course e.g. Java course. This facility enables teachers to control the access to specific courses. Teacher has exclusive rights to update the list of students who are eligible to take a specific course. Students are notified automatically about the

registration via e-mail. In case of successful entry into the system, the following functions are available for students:
*Function "Personal information"***.** Student's personal information is stored in the database and he/she may modify them later.
*Function "Test results"***.** This function provides students with viewing the test results, percentage of successfulness, teacher's classification and history of particular tests with correct and incorrect answers. When student finishes the test, system grades the performance and delivers appropriate notes in accordance to given criteria. Teacher is able to modify the automatically generated grade.
*Function "Type of the test"***.** This function provides student with the selection of the test type, which he/she intends to accomplish. During the training phase student can choose more questions and at the end of the test is able to view the successfulness in percentage and the correct answers. Teacher controls the number and the type of the questions.
*Function "Change password"***.** Student who entered the system can change his/her login and password and later logout the system

GROUP OF FUNCTIONS FOR TEACHERS

Teachers enter the system through "username" and "password" like students. If the entry to the system is unsuccessful, an error message appears and the client can try to enter the system again. If the entry to the system is successful then the following functions are available:

Function "Students". This function displays the list of enrolled students.
Function "New account". Teacher is able to add new students to the system.
Function "Security". This function is used to specify student's "username" and "password" to ensure that a specific student makes the test in a definite time-period. Teacher may create these usernames and passwords manually or automatically by software generator. Teacher can also set the time interval, during which the "username" and "password" are set for the first entry. When the time interval elapsed, which was set for passing the test, the system resets the initial settings for the student. To sum up, teacher has the following rights:

- To view and update students' username and password.
- To view and update the time interval, during which the student is allowed to enter the system. If a student does not make the test in the defined time interval then it is necessary to generate new username and password for a new time-interval. This feature contributes to the enhancement of the system security.

The purpose of these teacher's rights is to prevent the following attempts for cheating:

- Student gets the username and password directly before he/she passes the test so it is difficult to send these data to other persons.
- Student's username and password is generated for a definite time-interval, during which student is allowed to pass the test.

Function "Search". Teacher can search students by name, by surname or by group.
Function "Presence". Teacher is able to control the presence of students in labs in number of weeks. The number of the weeks may vary for the Java courses.
Function "Questions". Questions are divided into three groups: exam questions, test questions and credit questions. Teacher can add new questions or modify the existing questions in the system.
Function "Results". Teacher is able to track the study progress of his/her students in sorted tables. System evaluates the results and offers noted grades, which are subject to the teacher's approval.

IMPLEMENTATION OF THE CASE-STUDY AND THE PROJECT-BASED APPROACH TO LEARNING

As it was already mentioned in the introduction, the case-study is a part of the dedicated e-learning management system. The main purpose of the case-study is to help students in gaining practical skills in

mastering Java programming language by developing a complete application, which simulates the automatic teller machine example. The core architecture of the application follows the example introduced in (Horstmann C., 2002).

The specification of the task is as follows. The application has three actors: Client of the bank, Employee of the bank and Administrator of the bank.

1. The Client is provided with the following abilities: Clients have their own customer number and a personal identification number (pin), which are necessary for the entry into the bank system. Every client has two accounts:
    a. Saving account and
    b. Checking account (puts some limits on the balance)

   The client can add or withdraw money from an account or to transfer between accounts.

2. The Employee of the bank enters the system through "username" and "password". The bank employee has the next functions available in addition:
    a. Registration of the new client;
    b. Clearing the account of the existing client (it removes the client from the bank)

3. The administrator of the bank also enters the system through "username" and "password" like an employee of the bank. The administrator has the next functions available in addition:
    a. Registration the new employees of the bank;
    b. Removal of bank employees from the database;
    c. Changing the personal information of the bank employee.

From technical point of view this case-study was prepared from a running java application developed in Borland JBuilder 2005 Enterprise environment, it uses the JSP (Java ServerPages) technology and cooperates with MYSQL database server. The main purpose of using these technologies was the necessity to provide dynamic HTML pages and to monitor and archive the students' learning progress.

This case-study's code and explanation is reachable from the above mentioned dedicated learning management system. It offers students an anatomized online view to the selected developmental phases of the case-study for study purposes. It also suggests additional learning materials and exercises in order to support the project-based learning.

In the following text we briefly describe the available functionalities for students.

Button "**Visited chapter**". This button triggers a table with two columns "Name of chapter" a "Date of the last visit". The number of rows in the table is equal to the number of chapters of the case-study. If the chapter was not visited then the appropriate field in the row is marked as "Not visited". If the chapter was already visited then the background color of the row of the table changes to grey color and in the appropriate field of the row is updated with the actual datum of the visit.

Button "**Dictionar**y". – enables access and detailed explanation of items, which are referenced in the case-study.

Button "**Links**" – points to web links that are related to topics dealt in the chapter. Web links are divided into two parts: Java related and MYSQL related links. In spite of the fact that the focus of the case-study is on the Java skills acquisition, it is necessary to know the basics of databases because of the case-study's distributed nature. In case those students prefer other databases, they are free to study the appropriate database.

Button "**Quiz**" - provides very simple questions and refers to the most important topics dealt in the chapter. Not every chapter is finished with quiz.

Button "**Content**" – triggers a menu to access any chapter of the case-study.

The students' interface provides also a scroll menu for quick orientation and picking up chapters of the case-study.

An important feature of the case-study is the guided incremental change of the already available code. These options are available through assignments or projects. Students are forced to change the code and experiment with the results.

CONCLUSION

To sum up, the main goal of the course is to uncover the developmental process of a non-trivial distributed application through the provision of the case-study. Students have at disposal the Java code in every step of the development. At the end of each module there are suggestions for tasks (or projects), which are constructed as requirements for minor incremental changes in the existing case-study code. This approach assures that a student (or group of students) is not "lost" in the main thread of the application development. Teacher consults the specification of the projects and later conducts discussions around the observed problems. This project-oriented learning aim is to make learning more efficient for students. We plan to evaluate the obtained data statistically later.

REFERENCES

Pšenáková, I.(2002): Internet v dištančnom vzdelávaní. In: Sborník příspevku z mezinárodní konference Modernizace vysokoškolské výuky technických předmetů. 1. vyd. Hradec Králové: Gaudeamus, 2000, s. 171-173. ISBN 80-7041-723-4.

Learning Technology Standards Observatory: http://www.cen-ltso.net/Users/main.aspx, (2005-04-21)

Hall M., (2001). Core Servlets and JavaServer Pages (JSP). Neocortex spol. s.r.o., 2001

Horstmann C. (2002). Big Java. RR Donnelly & Sons Company, 2002

http://dev.mysql.com/doc/ (2005-02-07)

http://phoenix.mis.cycu.edu.tw/Class/2001Su_J2EE/J2EE01, (Client_Server_Architecture).pdf (2005-02-08)

http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial (2005-02-07)

Bandáková Jana (bandakovaj@centrum.sk) and Čierny Marián (cierny@intrak.sk) are graduate students at the Faculty of Electrical Engineering and Informatics, Technical University of Košice.

Samuelis Ladislav Ing., Ph.D.
Assistant professor
Dept of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 0402 Košice, Slovakia
Ladislav.Samuelis@tuke.sk