A 3-Tier Distributed Architecture Approach to Design of an Education and Testing Application in the Academic Intranet Environment

Miloš A. Kovačević

Abstract- this paper focuses on a possible application architecture intended for the exam preparation and testing in an academic intranet. The environment has specific qualities pertaining to testing conditions, hardly ever taken into account in the WBE (Web Based Education) applications intended for the corporation intranets or the Internet in general. By using the EUKLID application an attempt was made to meet the following requirements: ensure maximum system resistance to failure during testing in the software architecture domain, disable as much as possible student collaboration during exams, enable transparency of the existing faculty database for the applicable (business) logic, and finally ensure application portability in relation to the hardware platform and the operating system. The use of the Java distributed object model, with adequate assumptions mentioned in the paper, proved satisfactory.

Key words: intranet, WBE, reliable, testing, portability, 3-Tier architecture, JDBC, RMI, Applets.

1 INTRODUCTION

The Internet is probably the most important phenomenon of the late 20th century. It is almost impossible to find a single domain of human activity not at least partly covered by the phenomenon. Generally accepted network communication protocols, especially HTTP, enabled free flow of multimedia information as well as easier accessibility from relatively simple applications - browsers such as Internet Explorer or Netscape Navigator. Since information is the "blood" that runs through the "veins" of the Internet, and since it is the basic element of the knowledge, all prerequisites are present for taking a new approach to the process of education. Nowadays, even persistent advocates of the classic education show more interest in WBE and WBT techniques. The fact that only in the USA around 1.6 million students participated in the Internet courses from 1997 to 1998, as well as that the figure is constantly increasing [1] proves the fact that WBE is an excellent addition to the classic education.

Parallel to the Internet growth the number of companies with own intranet network is increasing. Their networks are separated from the "rest" of the Internet by means of special network hardware - firewalls. These devices enable information filtering to/from the Internet allowing usually only HTTP and SMTP protocols (WWW and E-mail). Companies are very interested in development and implementation of WBE systems in their own intranet environment because of the excellent results in personnel training and lower expenses in relation to the classic methods. High schools and universities are also among institutions that can promote considerably their educational function by using WBE applications. Some functions of the educational software must take into account certain specific qualities of the academic environment. This firstly refers to testing process specially arranged so that testing results can be reused in other school administration applications as well. The testing process is expected to be resistant to a possible hardware failure during testing, and to prevent possible fraud during exams. By presenting experiences in development of the EUKLID (system for training and testing designed at the Faculty of Civil Engineering in Belgrade) this paper will try to suggest the possible WBE system architecture in an academic intranet.

II PROBLEM STATEMENT

In the application domain, the problem is: to develop an application that will enable students to undergo certain exams by means of computers, realizing at the same time the following goals:

A. Test results can be used in an application utilized by the department of student administration
B. Undergoing an exam must be resistant to possible failures of computer resources.
C. Testing process must to maximum extent impede unallowed collaboration during exam
D. Students can prepare for an exam in the system that will be used during examination

The following problem arose in the implementation domain: how to choose among numerous software technologies those that can solve the problem so that the resulting application is also:

I. Portable
II. Open to most RDBMS (Relational Database Management System) systems on the market
Goal A naturally imposes itself as a consequence of previous existence of the faculty student database in ORACLE 8 system.

Goal B clearly points out to specific characteristic of a testing process in an academic environment. Namely, due to restricted computer resources and fixed exam terms it cannot be allowed to start an exam and then, after a network infrastructure or a server application failure, break it off and fix new exam terms.

Goal C also points out to specific requirements of the academic environment. It is quite clear that if the testing result is of legal and not commercial importance (student pays a course on the Internet) then attempts of unallowed collaboration during examination should be minimized.

Goal D has, apart from the educational function (student tests his knowledge and learns by correct answer inspection) the psychological importance. Since in a training process student uses the same user interface, he thus "gets used to" the system, which in the process of official exam can ease the tension.

Goals determined in the domain of the adequate software architecture choice (goal I and II) would enable the implementation of the solution in the other environments of interest. This refers primarily to the official educational institutions.

III EXISTING SOLUTIONS

A great number of applications enabling WBE exist on the market today. One of the most famous are WebCT and SocratEase.

Development of the WebCT application started in 1995 at the University of British Columbia. By the end of June 1999 more than two million people in 820 institutions and 42 countries used WebCT [2]. WebCT was primarily designed for the Internet/intranet courses. It has an extremely rich set of features for multimedia presentation creation, collaboration in the learning process (mail, chat, blackboard), various tests for knowledge examination, instructor insight into certain student activities and so on. The application is designed as a classic client-server 2-Tier architecture. Namely, the server component with the integrated internal database on courses and students is installed on the web server (IIS, Apache…). The client part is in fact a web browser like Internet Explorer or Netscape Navigator. User interface is made mostly of pure HTML forms, while the smaller part is made of Java applets for functions such as chat.

WebCT cannot completely meet the needs of the university testing. It is quite evident that goal A in the application domain and goal II in the implementation domain are not met. There is indeed a possibility of data transfer from the internal database to some of more famous RDBMS systems, but the process depends on the destination and is very complex [3].

Goal B is not completely met because the questions are delivered as a sequence of HTML forms, in accordance with answering order. If for example, one hour after the beginning of an exam, a server error or a network failure occurs (link between the client and the server goes down), the exam will be interrupted. This is the consequence of the two facts: firstly, the client side cannot memorize the progress of the test (HTML is stateless protocol) and secondly, all test questions for a student are not arriving simultaneously in a package. Therefore, at least one student will have to repeat the exam.

Goal I is not achieved either because WebCT can be installed only on Windows NT and UNIX platforms.

SocratEase, Eutectics company product was designed as classic client-server 2-tier application, written in Java. Server component consists of integrated Java web server (Servlet Engine), integrated internal database, and indispensable JRE 1.3 (Java Runtime Environment). The client interface was mainly implemented through HTML forms and Java Script, although Java applet with rich possibilities was used when creating the course/test content. SocratEase is an extremely adequate for most environments since it enables prompt installation and does not require previously installed web server. The usage of Java language enables maximum portability (goal I in the implementation domain). Clients at which applets will be started for course creation demand JRE 1.3, which can be freely downloaded from the Internet [4].

Eutectics affirms that SocratEase is the best software for WBE when price-performance ratio is taken into account. It is consequently used by big companies such as Compaq, Lucent, General Electric or institutions like USAF, USDA and others [5].

Just like WebCT application, SocratEase cannot fulfil goal A and goal II initially; usage of databases such as ORACLE or MS SQL is optionally supported. Goal B is not fulfilled due to the same reasons.

Both applications fulfil goal D, while goal C is partially achieved.

Clearly, the existing solutions cannot meet entirely all the application needs for student preparation and testing in an academic environment. It is evident that the reason for this is the fact that the existing applications were primarily intended for DL (distance learning) on the Internet and for WBE in the corporation intranets, where testing conditions are less rigid in comparison with the academic environment.

IV PROPOSED SOLUTION - EUKLID

EUKLID is an application intended for exam preparation and testing, developed at the faculty of Civil Engineering, University of Belgrade. The application enables a student to undergo exams and to prepare for an arbitrary number of exams. Three main applicative functions of the package are: administrative, testing and
preparatory. Before we start with the EUKLID architecture analysis and analysis of impact of solutions in the architecture domain on realization of the determined goals in section II, we shall first consider these applicative functions.

The administrative function enables instructor to create questions for his course. Questions can be grouped according to type and difficulty level. They can be testing or training questions (not used in exams). The types of question that can be used are:

- Mark the correct statements
- Choose the correct answer
- Enter the correct number

Instructor defines the following test characteristics:

- Duration
- Whether student can see the correct answers after a test or not
- Type of questions and how many questions from each chosen group will be used in an exam
- How many questions from each group should be answered in order to pass the exam
- Number of points for different grades

The testing function enables a student to undergo an exam after the following verification process:

1. Candidate fills out form on the screen
2. Instructor verifies particulars and identity of a candidate by means of the matriculation book picture
3. Instructor enters a password for given exam and then the test begins for the candidate

Candidate can "hand in" his test before the determined time limit. After the end of the exam candidate can see the test results as well as the test solutions, if allowed.

The preparatory function implies that students can solve a test in an environment identical to the one during exam (user interface, timer). Thus goal D from the implementation domain is achieved. The only difference is in the choice of questions provided by the system – all questions have a training attribute. Moreover, during exercises students can turn off the timer and see the correct answers to all questions. During exercises students can communicate in a chat mode, thus realizing collaboration aspect. Also, students can send a mail to a specialized instructor during preparation if they have some questions or suggestions.

It is worth mentioning that the administrative function enables manipulation with questions, courses, students and tests registered in the underlying database. For example, it is possible to see which student had more than N points in mathematics at the last exam, or to see the results in physics which achieved student X Y in the current exam.

Let us turn our attention now to the EUKLID architecture. During the design process a problem came up; it refers to the choice of the software technologies existing today in the Internet/intranet programming domain. Software RISC principle was applied. In accordance with it, it is necessary to find a minimum group of elements; by further combination of the elements the target complex domain can be covered [6]. In this case elements are in fact technologies such as HTML, XML, DCOM, ActiveX, CORBA, ODBC, JDBC and so on.

All the EUKLID components are written in Java. The components of the applied 3-tier distributed architecture are displayed in figure IV – 1.

---

**Figure IV-1**

- **Client browsers (IE, NN)**
- **HTML documents**
- **Java Applets**
- **HTML with applets**
- **Applet Storage**
- **RMI calls**
- **RMI Registry**
- **Web Server**
- **Main application and DB interface**
- **PC Clients**
- **HTTP request**

---

1. Candidate fills out form on the screen
2. Instructor verifies particulars and identity of a candidate by means of the matriculation book picture
3. Instructor enters a password for given exam and then the test begins for the candidate
Now, the EUKILD architecture will be described starting from the Main application and DB Interface (MADB) component (tier No. 2). The component uses JRE 1.3 enabling thus maximum portability i.e. realizing goal I at the application server side. Here we can find a part of the so-called EUKILD’s business logic referring to test generation, based on random question sequence from a database, chat server for student collaboration during process of preparation and so on. The second part of the business logic is situated at the client side in applets, which is not common practice with 3-Tier architecture. As it will be seen there is a good reason for it. All operations related to the database manipulation are also found in the component i.e. in the Java class Database which implements the JDBC (Java Database Connectivity) interface to the underlying database. The so-called JDBC-3-tier scenario [7] is displayed in figure IV-2. By isolating all operations that manipulate with the underlying database in one class, easy EUKILD modification was enabled, if the database structure is changed. Also, choice of the JDBC-ODBC bridge for the Java application driver, enables compatibility with the most existing RDBMSs, supplied by the adequate ODBC driver. MADB also comprises DBmaker class. This class is responsible for creation of relational tables and corresponding relations when the system starts. Based on the mentioned facts it is clear that goals A and II are completely achieved.

![Figure IV-2](image)

The client part of the application (tier No.1) consists of set of HTML pages, which include numerous applets, downloaded from the Applet Storage on the web server. Applets in fact represent the most part of the EUKILD’s user interface. As it is already explained, a part of the business logic is situated in applets. In the applet used for candidate testing, validation of candidate’s answers is performed. When a candidate in his browser downloads a HTML page together with the test applet, the process of its initialization includes downloading of complete test, as well as correct answers. If during the exam a server failure occurs and the exam time limit expires, or candidate hands in his test, the test will be checked in despite of failures (applets are executed in the client JVM (Java Virtual Machine)). Test results will also be available and a message displayed on the screen to call the duty instructor. The same procedure applies when a network failure occurs i.e. the link between the client and the server goes down. If the error is mended during the exam, candidates will not even notice it. Namely, at the end of exam, the test with candidate answers and results are transferred to the application server. The adequate logic implemented in the MADB component classes (tier No. 2) will put the results in a database on the database server (tier No.3). In this way the goal B from the implementation domain is achieved to a large extent. As evident from the mentioned facts, the flow of the complex data structures between the applets on the client and the MADB on the application server happen at the very beginning and end of testing. The same applies to the collaboration applets (chat), where the communication is completely asynchronous. Consequently, it is necessary to ensure software communication infrastructure that will support complex data interchange among Java objects executed in different JVMs. RMI (Remote Method Invocation), the best solution for distributed Java-to-Java application [7] was chosen. It firmly connects tier No.1 and tier No.2. EuclidClientManager representing the so-called remote Java interface [8] is implemented inside MADB. By means of the methods it contains, it provides indispensable services to the client side. The sequence of RMI calls in communication between the test applet and the EUKILD server is shown in figure IV-3. The direction of the communication is always from the client to the server. All RMI calls in the EUKILD application are implemented in that way. RMI also gives a possibility of bi-directional communication (peer to peer) [9].

![Figure IV-3](image)
Another component depicted in figure IV–1 is RMI Registry. When starting the MADB, an object of the class which implements the EuclidClientManager interface is registered under adequate name in the registry. Clients who will call the EuclidClientManager methods first obtain a reference to the remote interface by sending query to the RMI Registry component. Figure IV-3 shows that TestForm applet inside its init() method uses therefore RMI API call Naming.lookup(). After obtaining remote object reference, RMI calls are directly forwarded to the EuclidClientManager.

Finally, the goal C from the implementation domain remains to be considered. In order to thwart unallowed collaboration in exams each student receives test questions randomly chosen from the database. Overlapping is possible, but if the overall number of questions (in the database) from the course is big enough in comparison to the number of questions per test, then the probability of overlapping is reduced. In order to further lessen the possibility of unallowed collaboration the following solution was used: a question with the same ordinal number in the database after the i+1

selection does not appear the same as after the i

selection. It was achieved in the following way: let us suppose the question X is of type – mark the correct statements. When a candidate receives the question X, it consists of a question formulation and five offered statements. Database representation of the question X consists of three semantically equivalent formulations, five correct and five incorrect statements. When choosing the question X, MADB logic will randomly choose a formulation and five statements. Assuming that the random numbers generator acts in an ideal uniform manner, the probability that X in the i+1

selection looks completely the same as in the i

selection is:

\[ p = \frac{1}{3^5 \binom{10}{5}} = 0.0013 \]

This type of question forming is especially important in the academic environment, because students may undergo an exam several times (in more exam terms). As evident from the already mentioned, goal C is also considerably achieved.

V CONSTRAINTS AND ASSUMPTIONS OF THE SUGGESTED ARCHITECTURE

Let us consider now the constraints of the suggested architecture imposed at the client side exclusively (tier No. 1). The fact that certain commercial browsers such as the Internet Explorer contain incomplete JVM, not supporting the RMI API, represents the most important restriction of the architecture. Consequently, applets remain without JVM, which is the basis of the distributed Java model. Swing API which considerably improves the user interface in comparison to old AWT API, is not supported either. In order to overcome this very serious restriction the following solution was used: the applets using RMI execute in the standard Sun’s JVM. It means that clients need JRE 1.3. Sun therefore offered the so-called plug-in technology used by the EUKLID as well. It is based on the following procedure: first a HTML document is downloaded to the client; if it contains the specifically marked applet (with the OBJECT tag) than it checks weather JRE 1.3 is installed on the client. If it is not installed then download of JRE 1.3 is initiated from the web server specified in the OBJECT tag. In this case that is the EUKLID web server (tier No. 2). After downloading, installation to a local disk starts, and after installation, the applet is loaded and executed by the complete Sun’s JVM. The procedure is done only once per client; after that each subsequent downloading of a HTML page with an applet, omits browser’s JVM during the applet execution. The classic applet tag is used when an applet is expected to work inside the browser’s JVM. The initial download from an intranet web server on a local area network and the very JRE installation is a swift process since the size of compressed JRE 1.3 file is approximately 5 MB. This process takes about 2 minutes on the network at the Faculty of Civil Engineering in Belgrade (10 Mbit/s client segments). Most of the time in the above process is used for installation of JRE. Each subsequent applet execution inside JVM on the local disk is transparent. In accordance with the above mentioned facts it is supposed that EUKLID will be executed in an intranet environment.

If this architecture is applied on the Internet then the initial download of JRE through modem lines can be time consuming which has a negative psychological impact on possible users.

The following shortcoming refers to the reduced RMI performance on the Internet, when RMI requests are forwarded through the HTTP protocol to port 80. It refers to a situation when firewalls disable traffic through the implied TCP port 1099 for the RMI communication [9]. It is presumed again that the application will be used in an intranet environment where there are no port restrictions.

Finally, it is presumed that the clients have 64 MB of RAM memory for smooth functioning of plug-in and JRE 1.3. 32 MB is a minimum, which is quite acceptable today.

Let us consider now some presumptions not pertaining to the system architecture.

A good generator of uniformly distributed random numbers is necessary for a quality choice and question creating from a database. EUKLID uses generator of the uniformly distributed random numbers situated in the main Java API packages. The initial seed for the generator is a system time in ms starting from the reference date January 1st, 1970.

The last presumption refers to the duty instructor presence during exams, because of the already mentioned candidate verification process in section IV. Namely, the academic environment and numerous
companies are not supposed to dispose of the peripheral devices capable of unambiguous verification of each candidate (e.g. fingerprints scanner).

VI SYSTEM PERFORMANCE ANALYSIS

System performance analysis aims at measuring system response in two cases: the first case relates to the RMI call getTQustions() from N clients almost simultaneously with candidate’s test creating from the database (Figure IV-3). The second case applies to the RMI call addTTestForm(), which is expected from M clients at the same time when client’s answers and results are sent to the database. The second case is less critical because it is expected that M<N since some students will finish their exam before predetermined time limit. Based on the author's triennial practice at the faculty it is expected that M be by 20% (maximum) less than N. Performance testing will be made in September 2000. The experiment will be tested on a test group with M=N=30. For M=N=3 response was prompt!

We shall consider now solutions from the architecture domain supposed to accelerate the system response. Firstly, each RMI request coming from the different JVM is processed on the application server in a separate thread. Significant acceleration can be obtained by using the multiprocessor server platform.

Each time when an operation with the database is initiated an instance of class Database, which supports given operation is created. Related object constructor creates a connection with the database, lasting until the operation is finished. In order to form a SQL request the PreparedStatement API is used whenever possible, since it significantly accelerates work with the database [9]. In that case, already optimized and precompiled SQL query is used, to which parameters for each new request are sent. Data response to competitive access with more open connections is not going to be considered here, but clearly it depends on numerous factors (database server, data type, transaction type).

VII TOOLS USED FOR DESIGNING AND IMPLEMENTATION

When designing the relational database model and the object- oriented model of the EUKLID application, Rational Rose was used. Class diagrams and use case diagrams supported iterative process of finding out the optimal EUKLID structure. By means of RR the software documentation was generated.

The Java code was written in Borland JBuilder 3.

HTML pages were designed partly in FrontPage 2000, and partly by means of a text editor. Pictures in HTML pages were created in Adobe PhotoShop 5.5.

VIII CONCLUSION

The paper focuses on a possible architecture of one distributed web application intended for exam preparation and testing. By using a pure Java platform it is possible to create a substantially portable system, in respect to the operating system and to the database at the back of the application. By applying 3-Tier distributed model with partial transfer of the applicative (business) logic to the clients (tier No.1) the higher degree of the system resistance to a possible failure in concrete use was achieved. The application portability in relation to the database level (tier No.3) was enabled by the database interface implementation in only one class of the application level (tier No.2). Finally, it was undoubtedly found that the combination of the applets, JDBC and RMI technologies is extremely powerful if used for the development of the distributed intranet applications. However, once again it turned out that the maximum portability must inevitably result in certain performance degradation. The proposed solution was primarily intended for an intranet environment.

The issue of overcoming the problem, which appears in the Internet environment exclusively on the client side, remains open. Clearly, the subject under discussions is the problem of JVM incompatibility in commercial browsers. After all, when HTML and Java Script technologies are applied, the same problem appears again.

An ideal universal solution still remains just ideal. However, one should always aim at it.

LITERATURE:

  SocratEase/customers.htm