Tenth Int. Conf. on IFSs, Sofia, 28-29 Oct. 2006
NIFS Vol. 12 (2006), No. 3, 30-34

ONLINE CONSULTATIONS SYSTEM WITH INTUITIONISTIC FUZZY ESTIMATIONS

Daniela Orozova[1], Teodor Tzhechev[2]

[1] Bourgas Free University, Bourgas 8001, e-mail: orozova@bfu.bg
[2] Bourgas Free University, Bourgas 8001, e-mail: tzhechev@gmail.com

Systems that improve students' and teachers' access to information are among the most visible hallmarks of a modern university. They hasten and simplify communication and in doing this they improve the quality of education. An example are online consultation systems – software, which allows students and teachers to achieve communication in fewer steps. The online consultations system is a standardized, centralized structure, which is accessible to all student and teachers. Such a system would for example allow a teacher to send a message to all of his/her students simultaneously, or to send a message to only the participants in a particular course, or to create a table of frequently asked questions in order to avoid answering the same questions over and over.

At this moment we're building such a system in the Bourgas Free University[1]. Currently the following modules of the system are complete – registration in the system, access management and management and organization of the database containing information on the registered users.

The online consultations system, which within the framework of the project we have named the "virtual university" is composed of several modules, which will provide different functionality and information. Some of these functionalities will have wide-ranging effects on the entire system and that imposes the need of security measures – more specifically access control. An effective and widely used method is separating users in accordance with their status and defining different categories for different levels of access. Apart from obvious micro-categories (for instance, the user who has received a message is the one who has access to the contents of the message), we define macro-categories. One of them is named "student" and it encompasses all users who are enrolled as students in the university. Users who belong to that category have the right to communicate with teachers, but don't have the right change information in the FAQ, for instance.

Creating such a system is possible with a number of technological tools – the simplest and most accessible way is with a web server, which can be accessed with a web browser. There exists a diversity in the technologies that can viably be used as web servers – for the needs of the current project was chosen a Java based web server – Tomcat [2] with a Firebird database back end. The reasons for this selection are numerous, but some of the most significant are that these technologies are free to use, meticulously documented, modern and powerful. The Java environment is particularly suited to the creation of large, stable projects, because of its pronounced scalability. Java as a language emphasizes the usage of

[1] A project by Science Research Activity at Bourgas Free University "Online consultations system" with a team of: D. Orozova, B. Sendov, T. Zhechev, G. Petkov.

standardized coding and modularity. Firebird is a free to use branch of the widely used and professional InterBase.

For the creation of the application we're using the Java programming language [3] and more specifically the Java JSP (Java Server Pages) [4] and servlet libraries.

Because of the necessity for frequent access to the database, we're creating an auxiliary Java class, which aims at speeding up and simplifying access to the data. The class, called DBAccess, executes most queries to the database through its static methods for entering and extracting data. DBAccess accesses the database through a JDBC-ODBC bridge driver by Sun and executes previously defined transactions, while simultaneously securing against SQL injections. The methods return the data or information on whether the transaction was successful.

Access management is done in three basic stages: authentication, authorization and access control [5].

When a user accesses the web server, the first page they are taken to is the login page. That is 'Login.jsp' – a dynamically generated page, which greets users and according to the situation prints extra information, (if, for instance, the attempt to log in has failed for some reason). Login.jsp presents to users an HTML form in which they enter their username and password. Upon activating the log in button their information is sent via the POST method to the web server, which is then accepted by a servlet. The servlet then attempts to authenticate the user, accepting as parameters the username and password. If such are present in the request, it makes a call to a method in DBAccess, which checks for matches in the database and returns either a 'null' value (if there was no match) or a 'User' object – an instance of a specialized class which contains the user's information in capsulated form. From that object is extracted the 'role' (the level of access) of the user and the latter is forwarded to a dynamic page reflecting their status. Before forwarding the request, the Login servlet creates a new session object for the client and places in it the User object.

The session object is generated by the server, it follows the movement of the client through the pages of the web server. If we were to place some sort of information in the session object, this information would be available to all pages and servlets the client accesses afterwards. This allows for the creation of an access management system, with the session object used as a foundation. For a user to be granted access to a page with limited access, the user's session must contain a User object, which contains information about the role of the user. The only way a User object can be placed in the user's session is if it is done by the Login servlet, as such a user who hasn't been authenticated cannot be granted one. Furthermore, the user has no access to the session object – it is created and maintained only on the server.

If the Login servlet authenticates a user, the user is granted a User object in their session and is then forwarded to the page according to their role. If the Login servlet fails to authenticate the user's information, the user is sent back to Login.jsp along with a parameter explaining the cause of the failure. This is achieved as such:

```
HttpSession session = request.getSession(true);
session.invalidate();
request.setAttribute("error", "Bad password or username.");
RequestDispatcher dispatcher =
        request.getRequestDispatcher("../login.jsp");
dispatcher.forward(request, response);
```

If a user attempts to access a page or servlet, for which the user has no permission, the user must be stopped or turned back. While it is possible for each page and each servlet to check for the user's User object and role individually, but this would require too much extra code and would present the possibility of creating discrepancies. A better solution would be the filter. The filter is a Java interface, which allows the creation of classes, which Apache Tomcat can then use to filter clients' requests. The filter can change a request, forward it, stop it or let it through. If we use a filter, we can check a user's authenticity before we grant the user's request. Filters are applied directly to addresses on the server, by being noted in the Web.xml file, which is used to configure Tomcat. Filters can be applied to individual or multiple possible addresses through the use of wildcards:

```
<filter-mapping>
  <filter-name>Path Mapped Filter</filter-name>
       <url-pattern>/servlet/*</url-pattern>
</filter-mapping>
```

Filters can be layered one over the other, meaning that there can be more than one filter applied to a particular address.

The usage of filters makes managing access much easier, because they are independent of servlets and pages and are largely invisible to them, which in turn allows them to be freely modified to apply different security models.

The filter we're using incorporates a check on whether the user's session contains a User object, whether the data is properly entered in the object and whether the role of the user is sufficient for access to the requested resource.

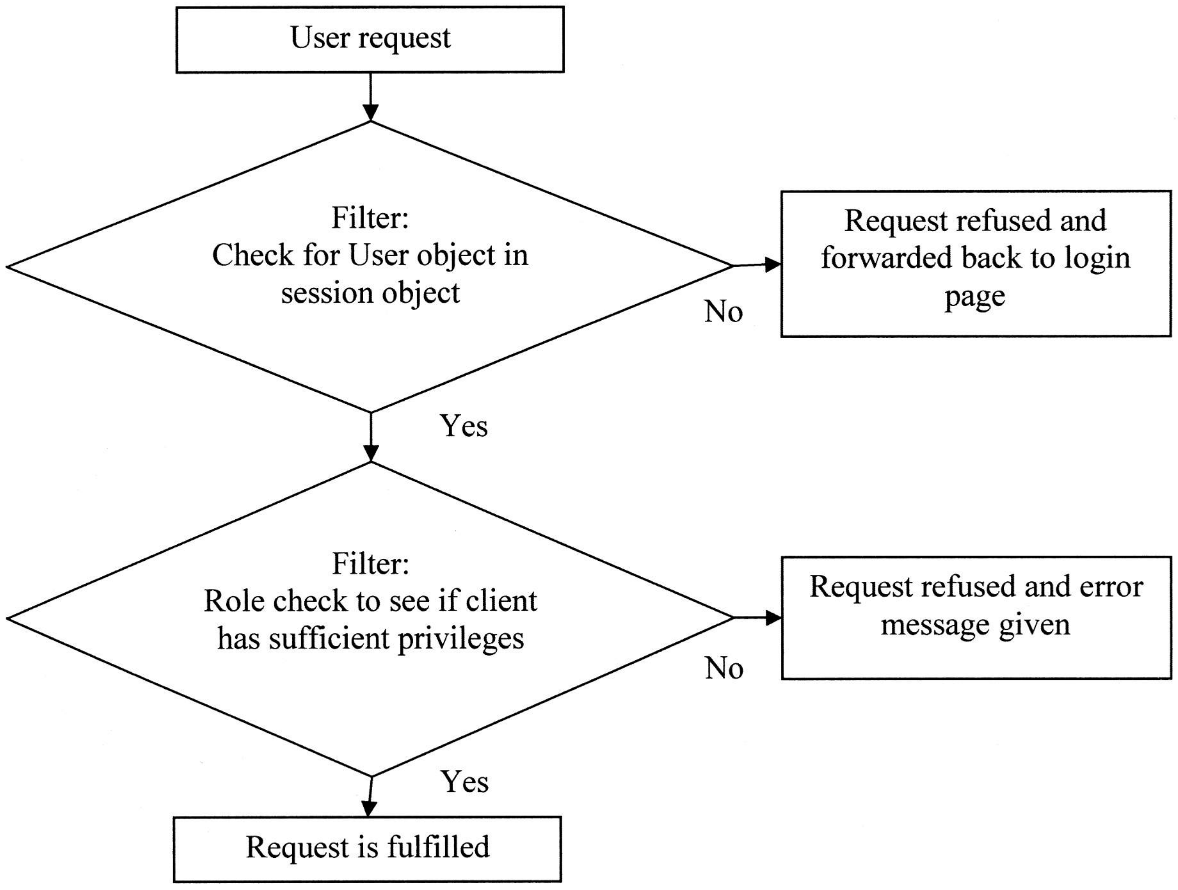
The database contains information on all users: ID, name, password, role, permission to send messages, temporary or indefinite suspension of the account as well as extra information, such as address, phone number etc. The database, after the project is complete, will also contain a number of other tables, which will deal with storing and sending messages, maintaining forums and an archive etc.

Management of the system will be done through the administration panel – a jsp, which grants the administrator the ability to perform operations on the database through HTTP requests with parameters defining the actions to be taken. The jsp would check the parameters and execute the operations while generating HTML code with a description of the operation that was executed as well as information pertaining to its success.

Using sessions to track users through the system is a defensive approach – the session is an object in the java environment, which exists only to the server. The client has no access to it and no way to modify it or the information contained inside. This is in contrast to cookies, for example, which are stored on the client's machine, whereas the session is completely inaccessible. Furthermore, the session will be set to become invalid after a certain time of inactivity, so even in a situation in which a user has failed to log out of the system before letting someone else access their computer, the second user would only have access if the session was still unexpired.

Filters will also prevent attempts to go around the login process or attempts to perform requests without passing authentication. Even if a user tries to feed parameters into a request to the administration panel, the filter responsible for that page would intercept the request before it reaches the jsp. Filters can cover a large number of pages and coexist without interfering with each other. Using identification placed in the session and monitoring permissions through filters allows the system to become resistant to malicious HTTP requests.

This is the functional structure of the system:


User request
Filter: Check for User object in session object
No
Request refused and forwarded back to login page
Yes
Filter: Role check to see if client has sufficient privileges
No
Request refused and error message given
Yes
Request is fulfilled

The Java environment is often criticized as being inefficient – this is a reasonable comment, but only when applied to desktop application, because the virtual machine loads a large number of extra classes and consumes a large amount of resources. This isn't a problem with server applications, however, because the extra classes are only ever loaded once and are used by every java application that is currently running.

The work of the system is estimated by the intuitionistic fuzzy characteristic [1] $<\mu^{\alpha}, \nu^{\alpha}>$, where μ^{α} и ν^{α} denote respectively the levels of preparedness and unpreparedness, and $\mu^{\alpha}, \nu^{\alpha} \in [0,1]$ and $\mu^{\alpha} + \nu^{\alpha} \leq 1$, and for each β-token – intuitionistic fuzzy estimations $<\mu^{\beta}, \nu^{\beta}>$ determining the degree of effectiveness (μ^{β}) and ineffectiveness (ν^{β}) of the fixed algorithm with the respective inequality again holding: $\mu^{\beta} + \nu^{\beta} \leq 1$ for $\mu^{\beta}, \mu^{\beta} \in [0,1]$.

μ is the count of fulfilled requests divided by the total number of requests.

η is the count of unfulfilled requests divided by the total number of requests.

In [6, 7] intuitionistic fuzzy estimations are introduced for the work of the connections with wireless local area networks.

Unfulfilled requests can be caused by a failed authentication, insufficient privileges (or other restrictions), found viruses, a server crash, a request to a wrong address and so on. These are the cases in which the system receives a request from the client and doesn't fulfill it, due to technological reasons or some other conditions not being met.

Then $\pi = 1 - \mu - \eta$ reflects the number of requests, which get no response, divided by the total number. This coefficient expresses the probability of the system not responding at all in a situation in which a user has sent a request.

The system has potential for further development, such as adding web chat, message compression (which can be achieved through filters), automatic modules for event notification (the server sending the administrator an SMS in the event of an error), e-mail or RSS notification when a message is received and so on. In [8, 9, 10, 11, 12] the authors have described different processes related to information exchange between students and teachers. The present online consultation system describes one of the different processes flowing in a modern university.

Literature:

[1] Atanassov K., Intuitionistic Fuzzy Sets, springer, Heidelberd, 1999.

[2] Marty Hall, Servlets and JavaServer Pages, A Sun Microsystems Press/ Prentice Hall PTR Book, 2002: www.coreservlets.com/

[3] Documentation and training materials presented by the authors of Java programming language: http://java.sun.com/docs/index.html/

[4] Marty Hall, „More Servlets & JavaServer Pages"2004: www.coreservlets.com/

[5] Microsoft, „Internet Information Server Access Control", MSDN – Online documentation for Microsoft products.

[6] K. Atanassov , S. Sotirov, V. Kodogiannis, Intuitionistic fuzzy estimations of the Wi-Fi connections, First Int. Workshop on IFSs, GNs, KE, London, 6-7 Sept. 2006, 75-80

[7] S. Sotirov, V. Kodogiannis, R. Elijah Blessing, Intuitionistic fuzzy estimations for connections with Low Rate Wireless personal area networks, First Int. Workshop on IFSs, GNs, KE, London, 6-7 Sept. 2006, 81-87

[8] Langova-Orozova, D., E. Sotirova, Using Java Technologies to Develop an Intranet Forum, Jangjeon Mathematical Society, 2004, Vol. 7, No 1, 2004, 15-29

[9] Langova-Orozova, D., E. Sotirova, Generalized Net Model of a Multiuser Chat-Server Realization by Java Programming Tools, Jangjeon Mathematical Society, 2004, Vol. 7, No 1, 2004, 31-42

[10] Shannon, A., E. Kerre, E. Szmidt, E. Sotirova, I. Petrounias, J. Kacprzyk, K. Atanassov, M.Krawczak, P. Melo-Pinto, S. Mellani, T. Kim, Intuitionistic Fuzzy Estimation and Generalized Net Model of E-learning within a University Local Network, Advanced Studies in Contemporary Mathematics, 2004, Vol. 9, No 1, 41-46

[11] Shannon, A., D. Langova-Orozova, E. Sotirova, K. Atanassov, P. Melo-Pinto, T. Kim, Generalized Net Model for Adaptive Electronic Assessment, Using Intuitionistic Fuzzy Estimations, – In :- Computational Intelligence, Theory and Applications, Computer Science I, Springer, 2005, 291-297

[12] Melo-Pinto, P., T. Kim, K. Atanassov, E. Sotirova, A. Shannon and M. Krawczak, Generalized net model of e-learning evaluation with intuitionistic fuzzy estimations, Issues in the Representation and Processing of Uncertain and Imprecise Information, Warszawa, 2005, 241-249

[13] Shannon, A., D. Langova-Orozova, E. Sotirova, K. Atanassov, P. Melo-Pinto, T. Kim, Generalized Net Model of a Training System, Advanced Studies in Contemporary Mathematics Vol 10, No 2, 2005, 175-179