Computer Security Lab

Term Project

Phase 3 report

Department of Computer Science & Engineering

Changwoo James Jung, cse93227

Alexey Naumov, cse93165

Apr 2013

Contents

1. Design

To meet the requirements we selected simplified ICONIX process as design methodology for the project.

ICONIX process is UML Use Case driven but more lightweight than Rational Unified Process. ICONIX provides sufficient requirement and design documentation, but without analysis paralysis. The ICONIX Process uses only four UML based diagrams in a four step process that turns use case text into working code. This design process was an ideal decision for a small team of two and complex project like this.

After analysis of requirements we created

- GUI prototypes,

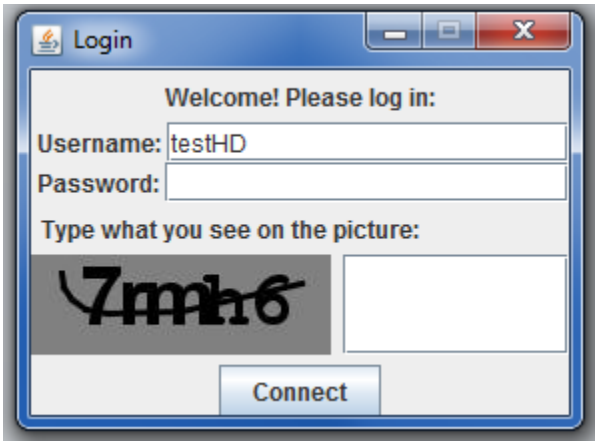

Figure 1: Help desk user client login

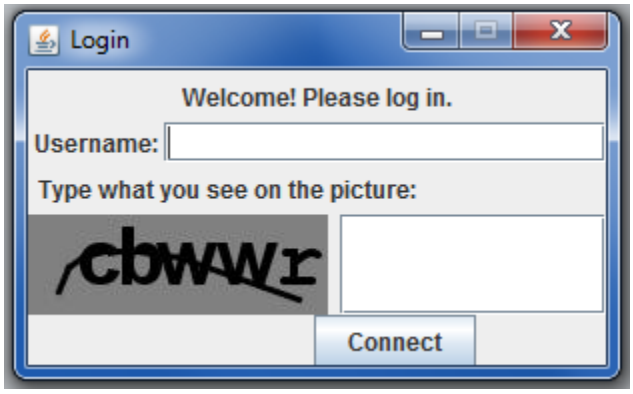

Figure 2: Anonymous user client login

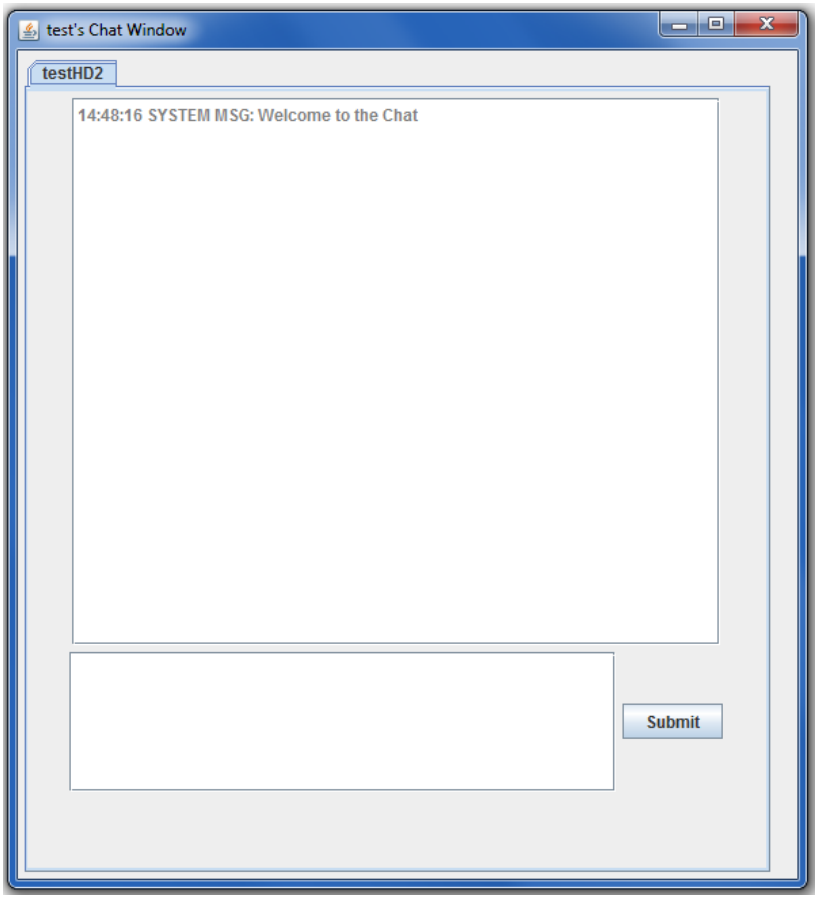

Figure 3: Anonymous user chat window

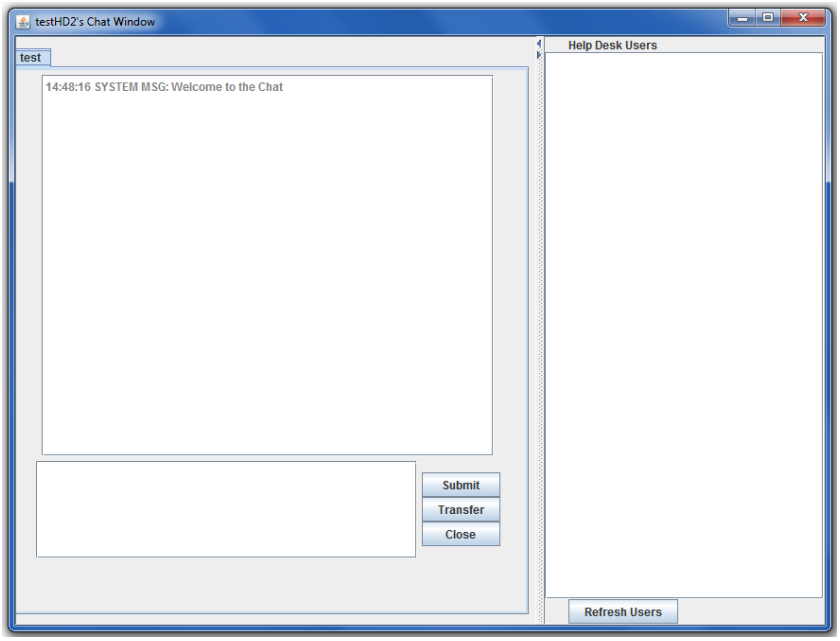

Figure 4: Help desk user chat window

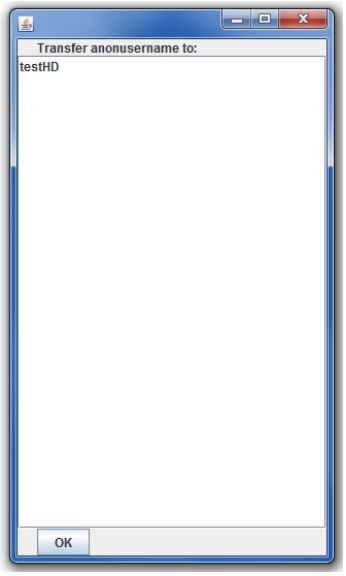

Figure 5: User transfer window

- Domain model as prototype of static structure of the system (Fig. 6)

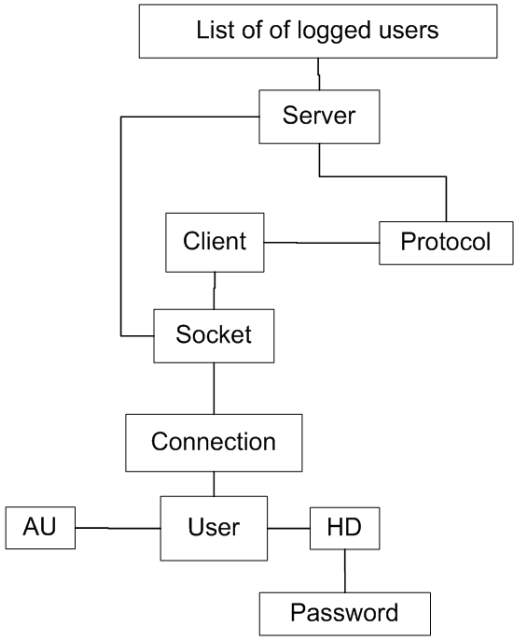

Figure 6: Model prototype

- Use Case diagrams to define the behaviour of the system. (Fig 7)

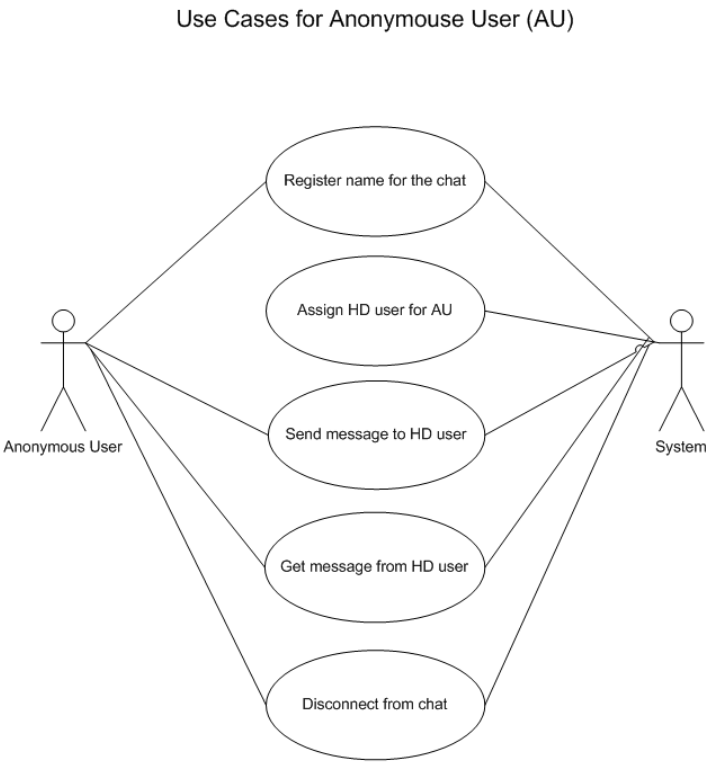

Figure 7: Anonymous user use case

Figure 8: Help desk user use case

- Robustness diagram was used to assign behaviour to the classes of the system(Fig. 9)

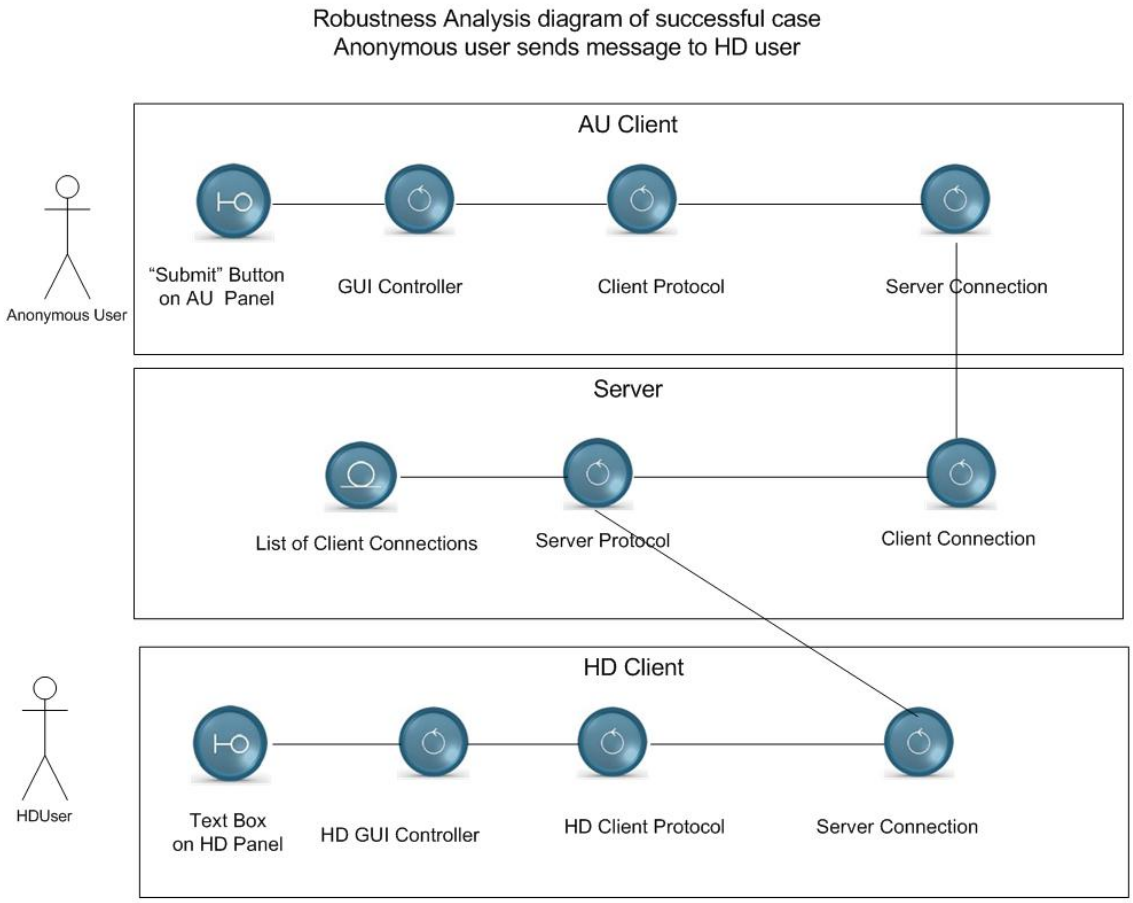

Figure 9: Robustness analysis diagram

Also Model-View- Controller (MVC) is used as the main design pattern. This decision is based on the requirement to create Client –Server application, since MVC separates the representation of information from the user's interaction with it.

In our project, Server components represent is Controller and Model, Client side responsible for View. But to make client's View working we need a set of client-side controllers and client-side model.

Design of the client side was the most complex part of the project and at this moment ICONIX Robustness analysis diagram (Figure 9) helped us to design client part of the application.

Server Connection Controller is responsible for communications with the Server. It sends messages to the GUI Controller that reacts in corresponding changes of the view.

Client architecture

The anonymous user client and the help desk user client have similar qualities – both users have a graphical user interface which they interact with, a connection to the server, a protocol they should follow, all of which combined is a way to easily send messages.

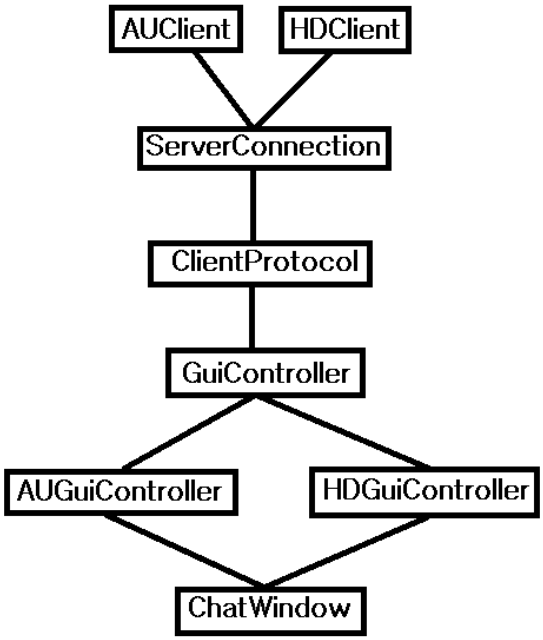

Figure 10: Client side architecture

This model was chosen as they share those qualities. GuiController is an interface that both the AUGuiController and HDGuiController implements, both of which include a chat window provided by ChatWindow class. HDGuiController contains more features such as being able to transfer users, having multiple tabs, being able to chat with an online help desk user of their choice, and so on. For the GUI to be able to send requests to the server, the GUI must use the ClientProtocol . The ClientProtocol class provides various methods for easily sending requests to the server, such as transferring a user, sending a message, requesting a list of all online help desk users, etc. For some methods in the HDGuiController class, they should only be used for help desk users. Some methods are not implementations of GuiController and must be used by casting it to a HDGuiController instance. Other methods may be implementations of GuiController but are also only to be used for HDGuiController. In the current implementation, the program creates logs and also receives warning messages from the server of possible security bugs or invalid commands.

Server architecture

The Server class listens on the port for any incoming connections, and makes an instance of ClientConnection. Once a client connection is established, the ServerProtocol class has many methods for parsing commands from the client and responding with the appropriate response. Such methods include authenticating a user, transferring a user, sending a message, and so on.

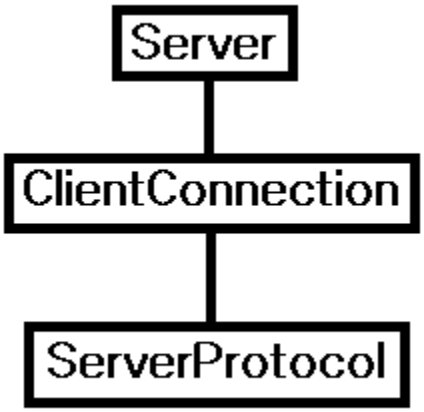

Figure 11: Server side architecture

2. Testing

HD= help desk user, AU = anonymous user.

Users

Test Description	Results	Verification status	Corresponding requirement
1) Start the server 2) Connect to the server as HD 3) Connect to the server as AU	HD and AU should see different screens and have different functionality	PASSED	REQ_1_1

User Authentication

Test Description	Results	Verification status	Corresponding requirement
1) Start the server 2) Start HD application 3) Start AU application 4) Provide invalid username/password/captcha	Both users fail to login with invalid username/password/captcha	PASSED	REQ_2_1
1) Start the server 2) Start HD application 3) Start AU application 4) Provide valid username/password/captcha	Both users can successfully log into the system	PASSED	REQ_2_1

Automatic selection of help-desk user

Test Description	Results	Verification status	Corresponding requirement
1) Start the server 2) Connect to the server as HD1 3) Connect to the Server as HD2 4) Connect to the server as AU	Only one help desk user has chat with connected AU. It can be HD1 or HD2	PASSED	REQ_3_1
1) Start the server 2) Start AU application 3) Connect to the server	User sees message “There are no agents available. Please try again later”	PASSED	REQ_3_1

Anonymous/Help-desk users

Test Description	Results	Verification status	Corresponding requirement
1) Start the server 2) Connect to the server as HD 3) Connect to the server as AU 4) Send message to HD from AU	HD user can see message that was send from AU; AU can see message that was send to him from HD	PASSED	REQ_4_1

5) Send message to AU from HD			

1) Start the server 2) Connect to the server as HD 3) Connect to the server as HD1 4) As HD1 double click on the name of HD in the list of all help desk users 5) In a new tab send message to HD	A new tab should appears for HD and the message send from HD1 should be displayed there	PASSED	REQ_4_2
1) Start the server 2) Connect to the server as HD 3) Connect to the server as AU 4) Connect to the server as HD1 5) As HD on the AU tab click Transfer button. In a new window select HD1 and click OK.	AU tab for HD is cloused. HD1 has a new tab for communication with AU. HD1 and AU can send and receive messages.	PASSED	REQ_4_3
1) Start the server 2) Connect to the server as HD 3) Connect to the server as HD1 4) Send message from HD1 to HD 5) Send message from HD to HD1	HD1 should be able to see his messages and the messages that were received from HD. Same for HD	PASSED	REQ_4_4

Network Communication/Server

Test Description	**Results**	**Verification status**	**Corresponding requirement**
1) Run StressTest from appendix 2) Connect to the server as HD 3) Connect to the server as AU	AU and HD still can communicate	PASSED	REQ_5_1
1) Start the server 2) Start packet sniffer 3) Connect to the server as HD 4) Connect to the server as AU 5) Send msg from AU to HD	Network sniffer will display only TCP packets for this communication	PASSED	REQ_5_1
Start server on the occupied port	Server doesn't starts. Exits with message "Could not listen on port: <portnum>"	PASSED	REQ_5_2
1) Start server. 2) Start several clients 3) Check ports and associated processes	Server process is listening only on one port.	PASSED	REQ_5_2

Client

Test Description	Results	Verification status	Corresponding requirement
1) Start the server 2) Start HD program and provide host name instead of IP	HD user can connect to the server and work as usual	PASSED	REQ_6_1
1) Start the server 2) Log in as HD user 3) Log in as another HD user 4) HD user should be able to see logged in users	HD user can see all logged in HD users, and if not, the 'Refresh List' refreshes the list to see all logged in HD users	PASSED	REQ_7_2

3. How to build the application

In order to make code reusable and easy to build the application we decided to use maven as our build tool. In order to make it flexible easy to configurable profile system was used.

To build the project maven should be installed on your system. If maven is installed and configured go to the root folder of the project and run on of the following:

- “mvn –P Server clean install” – in order to build server,
- “mvn –P AU clean install” – in order to build anonymous user client,
- “mvn –P HD clean install” – in order to build helpdesk user application.

After execution of this command you can find an executable jar under target folder.

To run the server: java –jar server.jar <port>

To run the HD client: java –jar HD.jar <hostname> <port>

To run the AU client: java –jar AU.jar <hostname> <port>

For this phase help desk user names are limited to the following list “testHD,testHD[1-10]” and accept any password.