

# Agent Based Software Engineering

(SENG 609.22) Instructed by: Dr. B. Far

## 1. Intention

The main motivation and purpose of designing this simple project is to experience the multiple agent system development methodology. The methodology followed by this project is Prometheus. The major steps are taken.

## 2. Introduction

Nowadays, more and more people need internet service when they are on their business trip or even at home. More than only providing the internet service physically, the service providers also need to control the registered user, gives them authorization to update their personal profile and query their personal bill, etc. Portal Server is a system between users and the background communication.

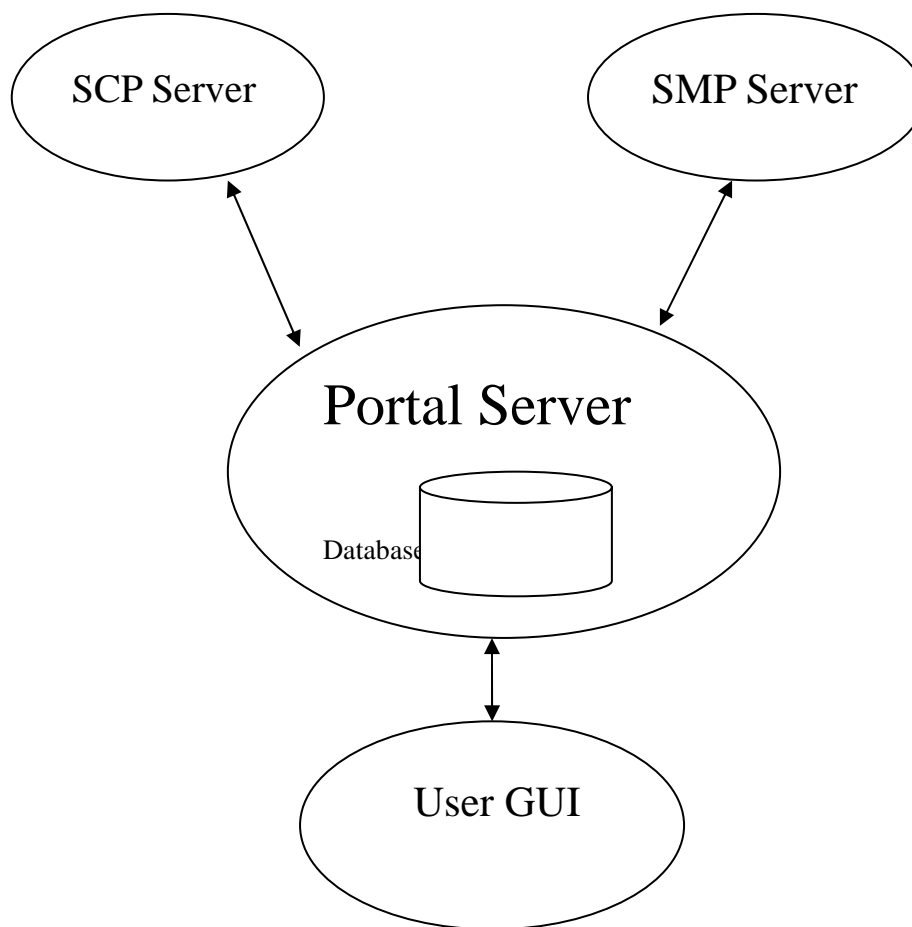
In this document, I discuss a multi-agent system for portal server. The document structure is as follows:

- For the system analysis, the system specifications are presented. Apart from the business case and system description, I make great efforts on the requirement specification in which the system use case is discussed in detail.
- For the general design, the system architecture is discussed, and I apply the “methodologies fro agent-based analysis and design” for role identification and agent description.
- For the detail design, OOAD and UML technology are used to specify the classes as well as sequences, which are lower level abstractions that can be implemented directly.

### 3. Business Case

#### 3.1 Business Case Overview

Portal Server is a web-based system that involves multi-functions such as sending / receiving messages to System Management Part Server (SMPerver), sending / receiving messages to System Control Part Server (SCPServer), showing the information content to the customers, logging every operation from the user. Etc. Portal Server is just wok as a server to manage all the users information, control their login / logout operations and. The system overview is shown in Figure 3-1



Portal server needs to connect to another two systems to work together. They are SCP and SMP. SCP is a system under the level of the Portal Server and serves for Portal Server. It implements the following operations about user

1. Users web authorization
2. User account activation and update user profile online
3. User close service
4. Query activated operations status
5. User exit portal server

SMP is a system under the level of the Portal Server as well; it provides more functions for user to operate online by them. They are included:

1. User pre-paid card charging
2. Modify user's login name
3. Modify user's password
4. Register a login name
5. User query charging record
6. User query the detail bill on user card

And so on

The users who use Portal Server are registered user, but they are divided in to two different types of users under different service type, because unregistered user can not use any function from portal server. One of the service type is prepaid, those kind of user can query how many dollars left on their account, if less then 10\$, they will be asked to recharge money in their account; another type of user is monthly payment user, they will have a detail bill every month, but prepaid money in their account is not necessary.

## 3.2 System Use Case

### 3.2.1 SCP Use Case

Table 3.1 Web Authorization case

Use Case	Users web authorization
Short description	Validate user information with login ID
Pre-conditions	None
Post-conditions	User information is authorized
Main flow	<ol style="list-style-type: none"> <li>1. Make sure system has been initialized</li> <li>2. Make sure SCP session buffer is still available (There is a maximum number of SCP session number limit in the SCP communicate protocol)</li> <li>3. Obtain user information based on user's IP address</li> <li>4. Make sure User has not been pre-authorized in this session</li> <li>5. Start to count user's online time</li> <li>6. Collect all the data needed to make a message and send to SCP Server</li> <li>7. Receive message from SCP, collect all the received data and feed back to the user.</li> </ol>
Alternative flow	If any of the 2 or 4 above is failed or SCP return failed information, then case failed
Use Case type	SCP Use Case

Table 3.2 User account activation and update user profile case

Use Case	User account activation and update user profile case
Short description	User account activation and update user profile case
Pre-conditions	Web Authorization case
Post-conditions	User account has been activated
Main flow	<ol style="list-style-type: none"> <li>1. Make sure user has been authorized</li> <li>2. Make sure user status is not activated, otherwise end the</li> </ol>

	<p>case</p> <ol style="list-style-type: none"> <li>3. Collect all the data needed to make a message and send to SCP Server</li> <li>4. Receive message from SCP, collect all the received data and feed back to the user</li> <li>5. If successfully activated user's new service type, then store to local database</li> </ol>
Alternative flow	If any of the 1 or 2 above are failed, or returned message failed, then case failed
Use Case type	SCP Use Case

Table 3.3 User close service case

Use Case	User close service
Short description	User service close
Pre-conditions	Web Authorization case
Post-conditions	User account has been deactivated
Main flow	<ol style="list-style-type: none"> <li>1. Make sure user has been authorized</li> <li>2. Make sure user status is activated, otherwise end the case</li> <li>3. Collect all the data needed to make a message and send to SCP Server</li> <li>4. Receive message from SCP, collect all the received data and feed back to the user</li> <li>5. If successfully deactivated, stop to record the time of user online</li> </ol>
Alternative flow	If any of the 1 or 2 above are failed, or returned message failed, then case failed
Use Case type	SCP Use Case

Table 3.4 Query activated operations status

Use Case	Query activated operations status
Short description	User query the status of if he/she is activated or not
Pre-conditions	Web Authorization case
Post-conditions	Return a result of if user account is at that time activated or not
Main flow	<ol style="list-style-type: none"> <li>1. Make sure user has been authorized</li> <li>2. Obtain the user status from the SCP session</li> </ol>
Alternative flow	N/A
Use Case type	SCP Use Case

Table 3.5 User Exit Portal Server

Use Case	User Exit Portal Server
Short description	Opposite operation of web authorization

Pre-conditions	Web Authorization case
Post-conditions	User exit Portal Server
Main flow	1. Make sure user has been authorized 2. Delete user from SCP session buffer
Alternative flow	N/A
Use Case type	SCP Use Case

### 3.2.2 SMP Use Case

Table 3.6 User Register New Login Name

Use Case	User Register New Login Name
Short description	After login with user ID, user can pick a specific login name for him/her self
Pre-conditions	Web Authorization
Post-conditions	User has a new login name
Main flow	1. Obtain all the user information from SMP session 2. Collect them and make a message, send to SMP Server 3. Receive message from SMP, collect all the received data and feed back to the user
Alternative flow	N/A
Use Case type	SMP Use Case

Table 3.7 User Change Own Password

Use Case	User Change Own Password
Short description	User can change own password
Pre-conditions	Web Authorization
Post-conditions	User password has been changed
Main flow	1. Obtain all the user information from SMP session 2. Collect them and make a message, send to SMP Server 3. Receive message from SMP, collect all the received data and feed back to the user
Alternative flow	N/A
Use Case type	SMP Use Case

Table 3.8 User Change Login Name

Use Case	User Change New Login Name
Short description	After user register the new login name, user can change his/her login name
Pre-conditions	Web Authorization & Register a new login name
Post-conditions	User login name has been changed
Main flow	4. Obtain all the user information from SMP session 5. Collect them and make a message, send to SMP Server

	6. Receive message from SMP, collect all the received data and feed back to the user
Alternative flow	N/A
Use Case type	SMP Use Case

Table 3.9 User Query Charging record

Use Case	User Query Charging record
Short description	After recharging money in his account, user can query the charging record and new balance
Pre-conditions	Web Authorization & User recharged money in his account
Post-conditions	The record of new balance showed up on the screen
Main flow	<ol style="list-style-type: none"> <li>1. Obtain all the user information from SMP session</li> <li>2. Collect them and make a message, send to SMP Server</li> <li>3. Receive message from SMP, collect all the received data and feed back to the user</li> </ol>
Alternative flow	N/A
Use Case type	SMP Use Case

Table 3.10 User Query Detail Bill

Use Case	User Query Detail Bill
Short description	User can check his detail bill for each month
Pre-conditions	Web Authorization
Post-conditions	The record all the bill for the current month showed up on the screen
Main flow	<ol style="list-style-type: none"> <li>1. Obtain all the user information from SMP session</li> <li>2. Collect them and make a message, send to SMP Server</li> <li>3. Receive message from SMP, collect all the received data and feed back to the user</li> </ol>
Alternative flow	N/A
Use Case type	SMP Use Case

Table 3.11 User Pre-paid cards charging

Use Case	User Pre-paid card Charging
Short description	User can recharge money in his prepaid card
Pre-conditions	Web Authorization
Post-conditions	The new balance will come to user's card
Main flow	<ol style="list-style-type: none"> <li>1. Obtain all the user information from SMP session</li> <li>2. Collect them and make a message, send to SMP Server</li> <li>3. Receive message from SMP, collect all the received data and feed back to the user</li> </ol>
Alternative flow	N/A

## 4 Agent-based Analysis and Design

In this section, the functionalities will be grouped and assigned to the identified agents. For the uses description in the above section, we can find that some use cases use the same data. We can group the use cases that use the same data together and assign these functionalities to an agent. Then the interaction between agents is reduced and that is the core design thought in Prometheus methodology.

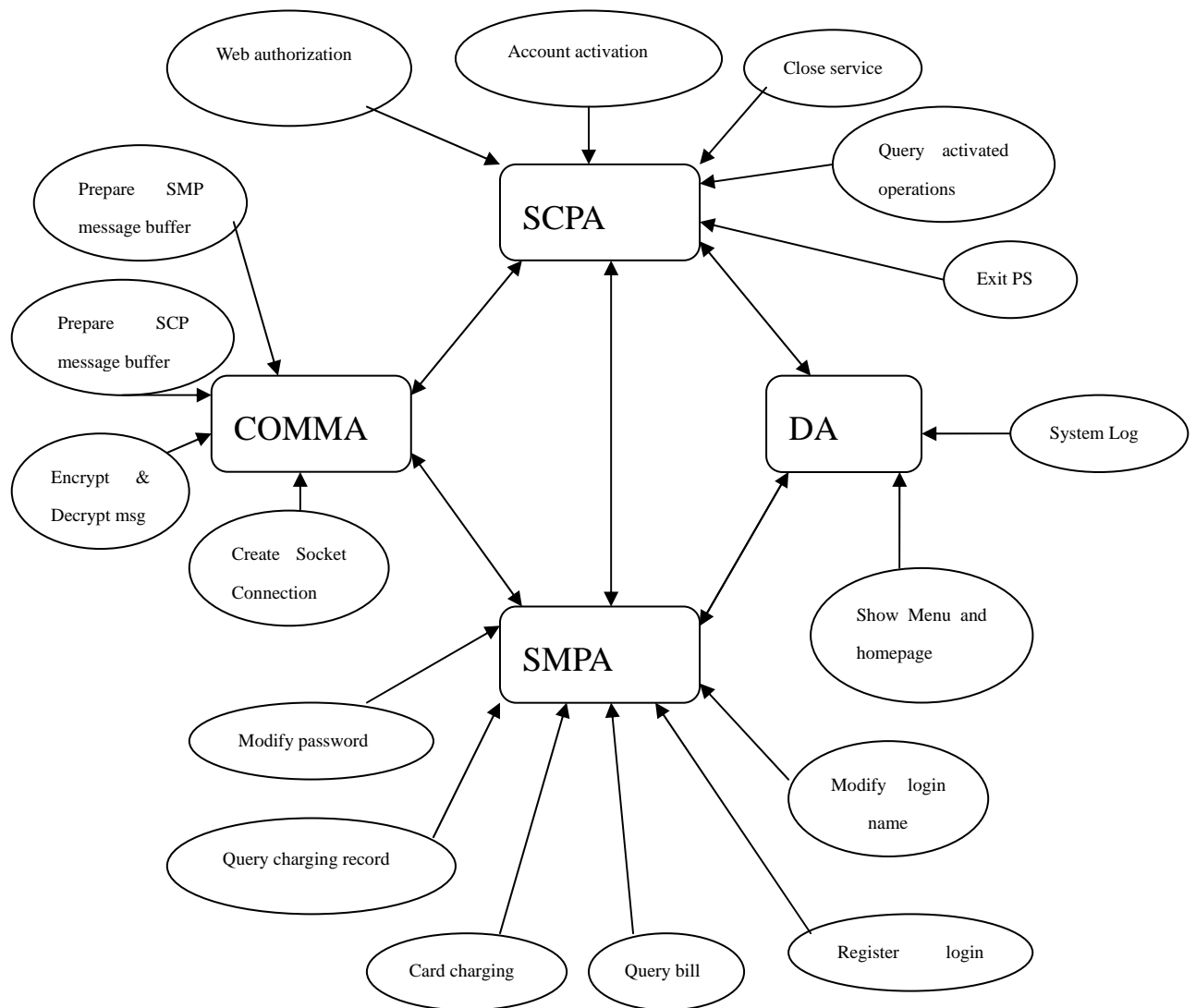
All the use cases for SCP can be handled with the SCP Agent (SCPA). SCPA can finish all the functions that users intend to do through SCP Server. Apply for the user data logic, users web authorization, user account activation and update user profile online, user close service, query activated operations status and user exit portal server are all grouped and assigned to the SCPA

All the use cases for SMP can be handled with the SMP Agent (SMPA). SMPA can finish all the functions that users intend to do through SMP Server. User pre-paid card charging, modify user's login name, modify user's password, register a login name, user query charging record, user query the detail bill on user card are all grouped and assigned to the SMPA.

All the communication work, like sending and receiving the messages, encrypt the message before sending and decrypt after receiving, prepare message buffer and session buffer for user, etc all belong to Communication Agent (COMMA). This agent is a base level agent, it has nothing to interact with users, only process the message and prepare the socket tube for the message.

All other operations including display the content of Portal Server, provide advertising information in the corner, record system log are all belong to Display Agent (DA).

The Figure 4-1 gives an architectural picture of what the system look like and interactions among major components of system.



## 4. Detailed design

### 4.1 Agent overview diagram

The agent overview diagram is a good representation of the internal structure of agent. The letter envelope icon represents the requests to the agent. The round corner rectangle icon represents the capability of agents. Figure 4-1, 4-2, 4-3, 4-4 are the agent overview diagrams

Figure 4-1 The SCP agent has the several capabilities to deal with different requests as following

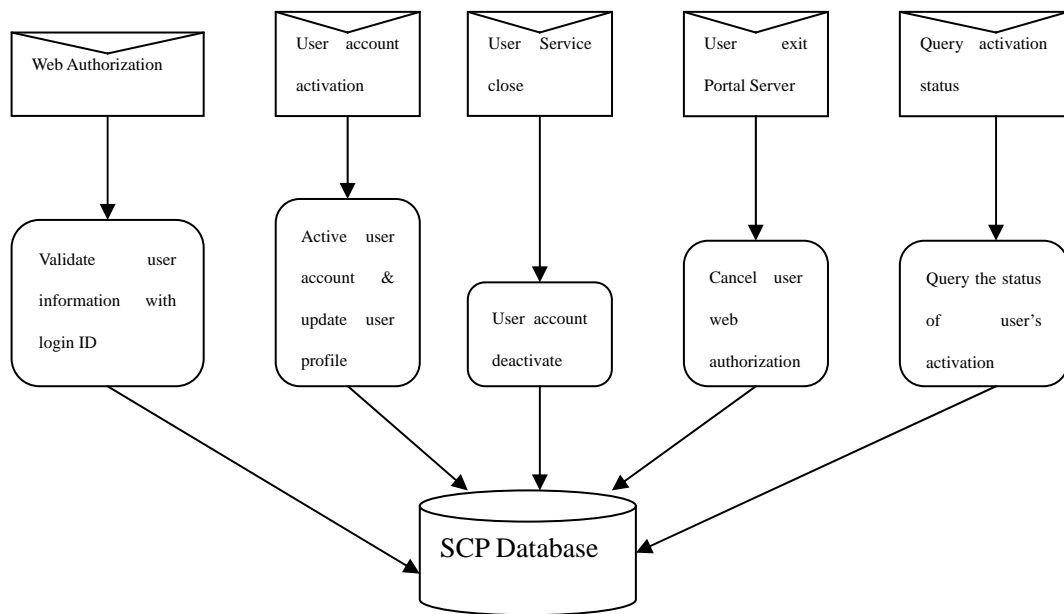


Figure 4-2 SMP Agent has the several capabilities to deal with different requests as following

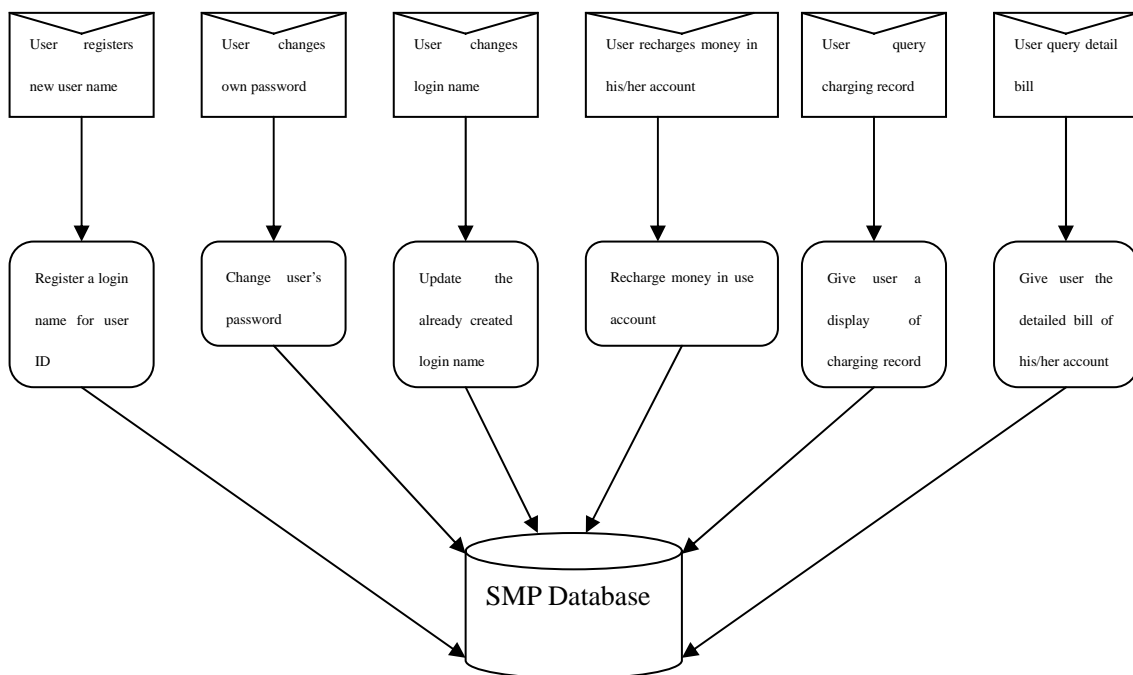


Figure 4-3 Communication Agent has the several capabilities to deal with

different requests as following

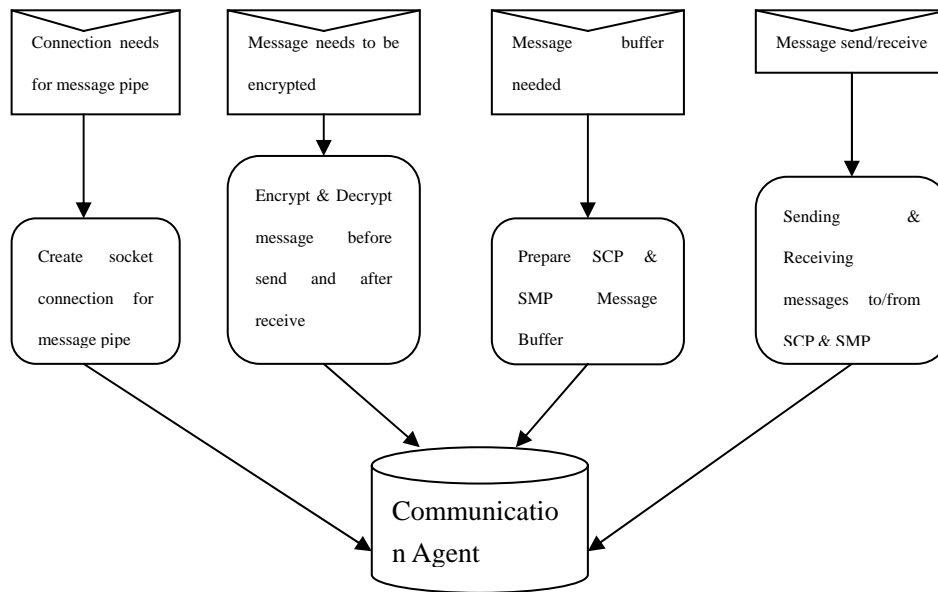
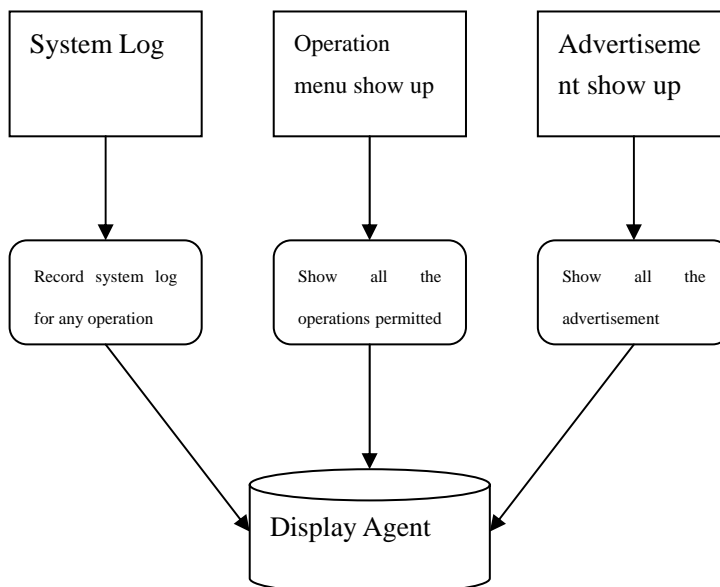
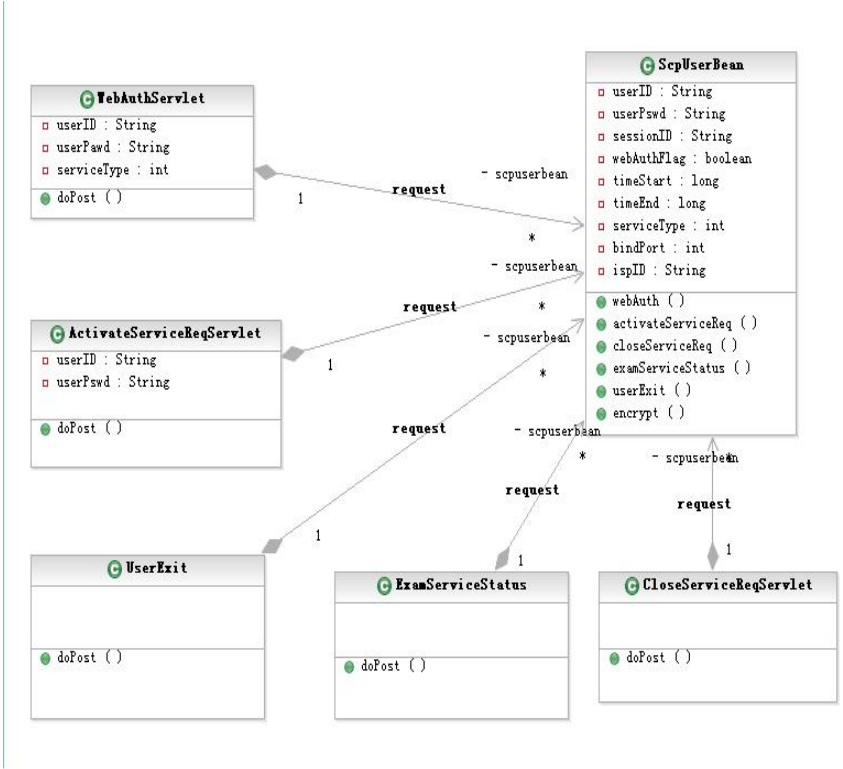


Figure 4-4 Display Agent has the several capabilities to deal with different requests as following

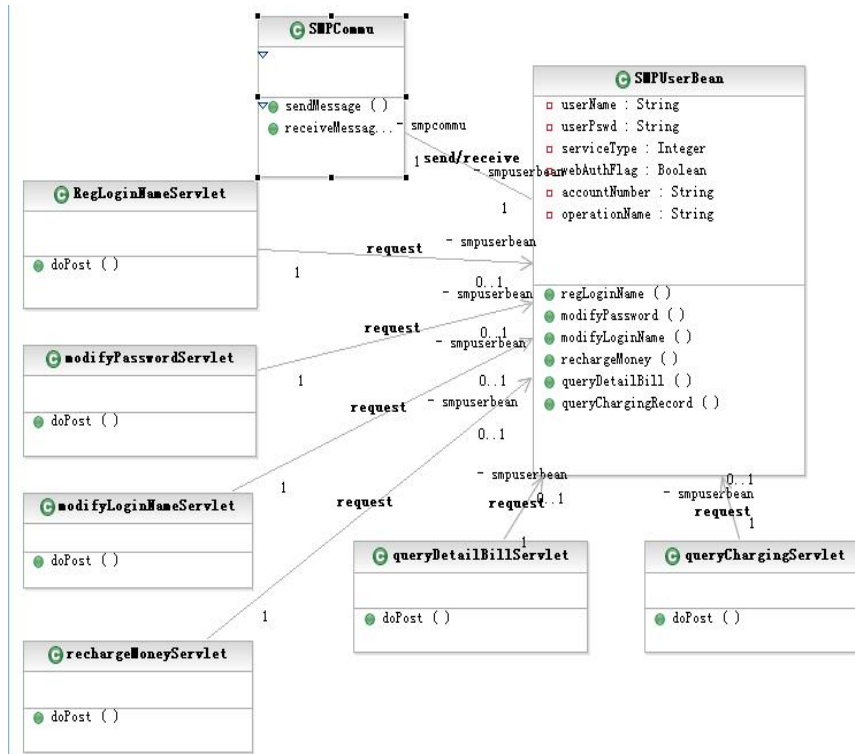


## 4.2 Class Diagram

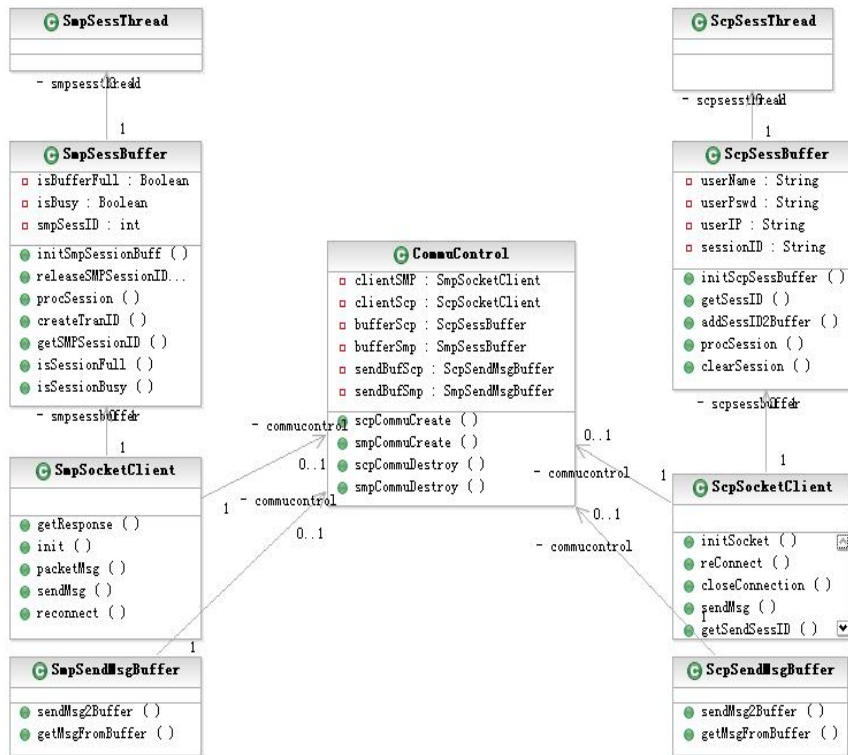
1. Class diagram of SCP Agent (Figure 4-5)



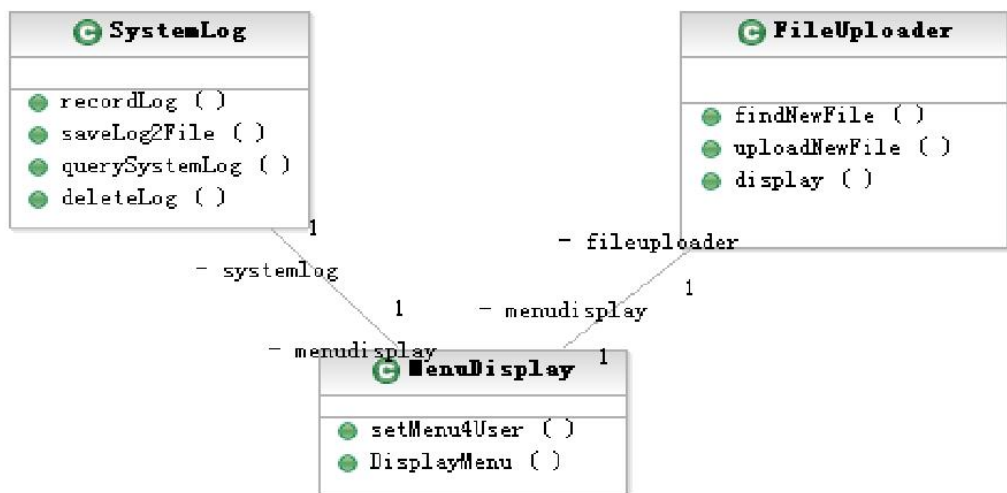
2. Class diagram of SMP Agent (Figure 4-6)



3. Class diagram of Communication Agent (Figure 4-7)



4. Class diagram for Display Agent (Figure 4-8)



### 4.3 Interaction diagram

The sequence diagram is good representation of interaction protocol between agents and environment. So only the scenarios with interactions between agents are represented in the interaction diagrams. Here, I will show only one of the interaction diagrams from SCP use case and SMP use case.

Figure 4-9 SCP web authorization case

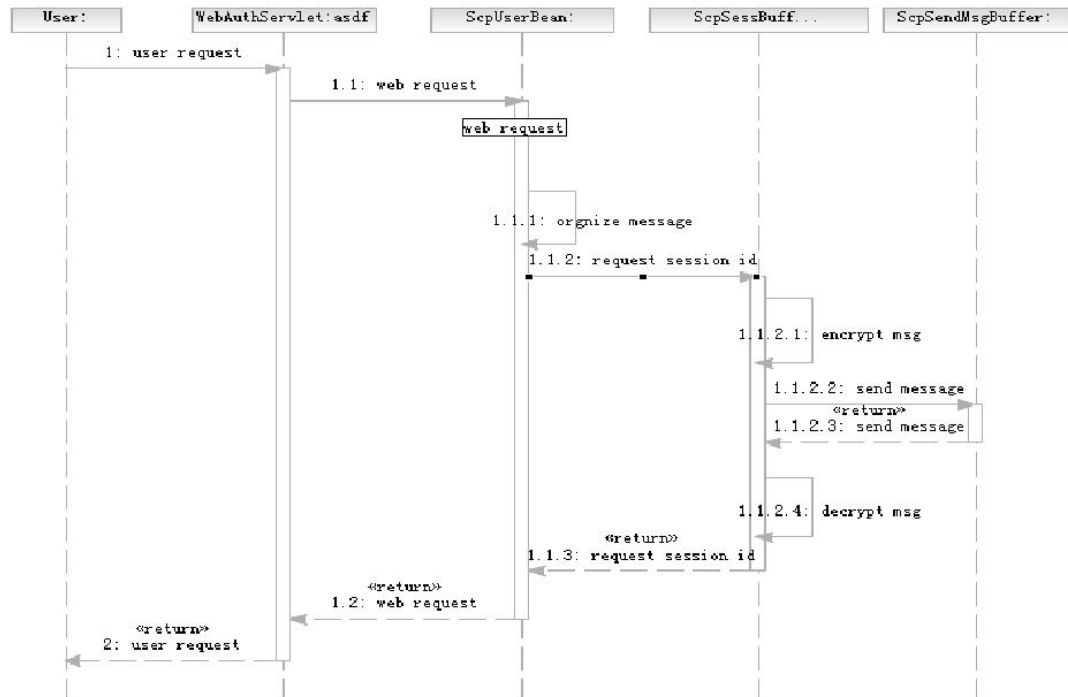
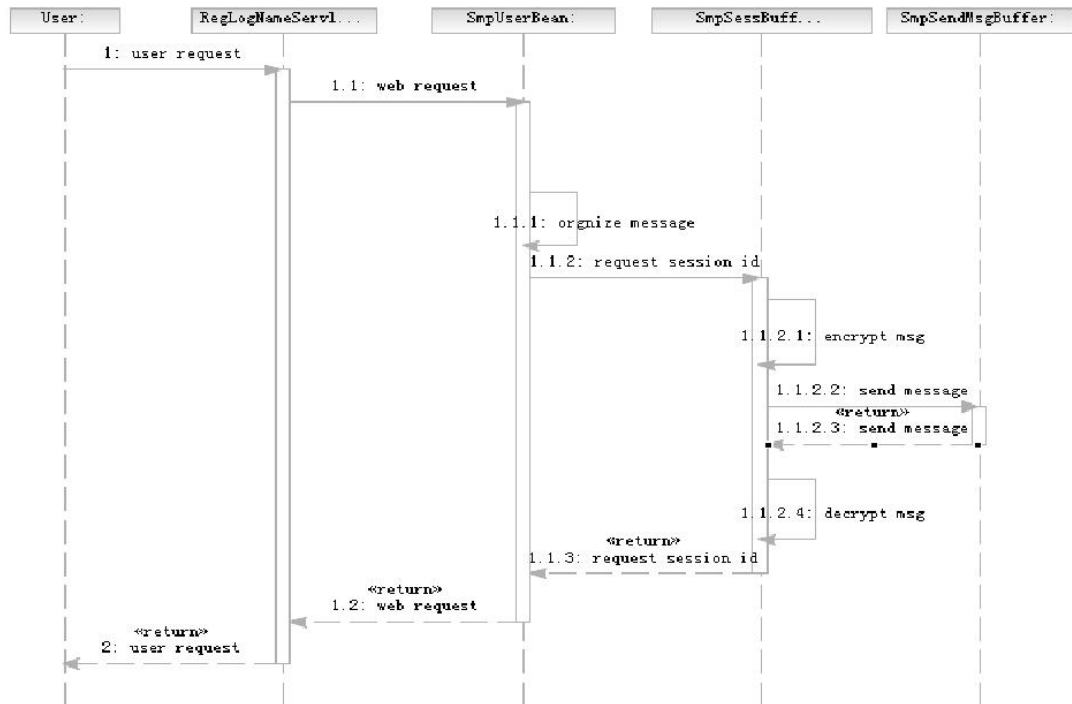


Figure 4-10 SMP Register new login name case



## 5. Conclusion

The design and development of this project basically follows the Prometheus that is an agent system design methodology. I find that Prometheus is simple, clear and very easy to use. The visual modeling used by this method is very efficient and straightforward. But this methodology uses a waterfall process, instead of an iterative process. In my design of this project, the iterative process is used. The use case descriptors and agent descriptors were omitted since the purpose of this project is to focus on evaluating the Prometheus process instead if the project itself. The project still has a lot of pitfalls that I do not have enough time to make it perfect, but it represents my thought of agent-based software system. In the following study and working, I will try to implement agent-based software in the real industry project, and make it better.

## 8. References

- [1] UML Distilled, Applying The Standard Object Modeling Language. Martin Fowler
- [2] L. Padgham and M. Winikoff: Prometheus: A Methodology for Developing Intelligent Agents, Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), 2002.