

*Department of Electrical and Computer Engineering*

# **Agent Based Software Engineering**

## **(SENG 697)**

**Agent Based Online Bookstore System**

**Student Name: Zilan (Nancy) Yang**

**Student ID: 302267**

**Submission Date: Oct 29, 2006**

## **Table of Content**

---

<b>1. Introduction</b>	<b>2</b>
<b>2. System Overview</b>	<b>2</b>
<b>2.1 System Description</b>	<b>2</b>
<b>2.2 System Architecture</b>	<b>2</b>
<b>2.3 System Requirements</b>	<b>3</b>
<b>2.4 Assumptions</b>	<b>3</b>
<b>2.5 Wish List</b>	<b>3</b>
<b>3. System Design Documents</b>	<b>4</b>
<b>3.1 System Requirement Model</b>	<b>4</b>
<b>3.1.1 Domain Description Phase</b>	<b>4</b>
<b>3.1.2 Agent Identification Phase</b>	<b>5</b>
<b>3.1.3 Role Identification Phase</b>	<b>7</b>
<b>3.1.4 Task Specification Phase</b>	<b>10</b>
<b>3.2 Agent Society Model</b>	<b>12</b>
<b>3.2.1 Ontology Description Phase</b>	<b>13</b>
<b>3.2.2 Role Description Phase</b>	<b>14</b>
<b>3.2.3 Protocol Description Phase</b>	<b>15</b>
<b>3.3 Agent Implementation Model</b>	<b>15</b>
<b>3.3.1 Multi-Agent Structure Definition Phase</b>	<b>15</b>
<b>3.3.2 Multi-Agent Behavior Description Phase</b>	<b>16</b>
<b>3.3.3 Single-Agent Structure Definition Phase</b>	<b>22</b>
<b>3.3.4 Single-Agent Behavior Definition Phase</b>	<b>24</b>
<b>3.4 Code Model</b>	<b>24</b>
<b>3.5 Deployment Model</b>	<b>25</b>
<b>3.5.1 Deployment Configuration Phase</b>	<b>25</b>
<b>4. Conclusion</b>	<b>25</b>
<b>5. References</b>	<b>26</b>

## 1. Introduction

As the internet almost covers every place in the world, online bookstore becomes a smart and efficient way to sell books for bookstore owners. Customers also get benefits to find and purchase the most suitable book they want from home. But nowadays online bookstores are not smart enough, and we want to add more intelligence into it which can optimize the book search quality. Multi-agent technology may be a good option of solutions to handle this issue. So we propose this agent-based online bookstore system.

In this project, we will try to adopt PASSI methodology in the analysis and design of this system. PASSI covers the whole software development life cycle and supports Rational Rose.

This paper is organized as followed. In Section 2 we will go through this system generally. Then details design will be illustrated following PASSI methodology in Section 3. Finally the conclusion is drawn in Section 4.

## 2. System Overview

### 2.1 System Description

This system is supposed to help customers find the most suitable book they want from different bookstores, and complete their purchase process with the abilities to handle all the financial and delivery issues online. In this system, we adopt CBR technology to deal with the book searching, and preference rules can be calculated and inferred based on the customer's profile and purchase history to optimize the search quality.

### 2.2 System Architecture

The system is a 3-tier web-centric multi-agent system. The customer makes the book searching and purchasing through web browser. The agent-based online bookstore web-centric system will search books from bookstore database (in order to simplify the system) and complete the payment process through bank web services. All the customer's profiles and purchase history will be saved in the online-bookstore database. The architecture is as followed.

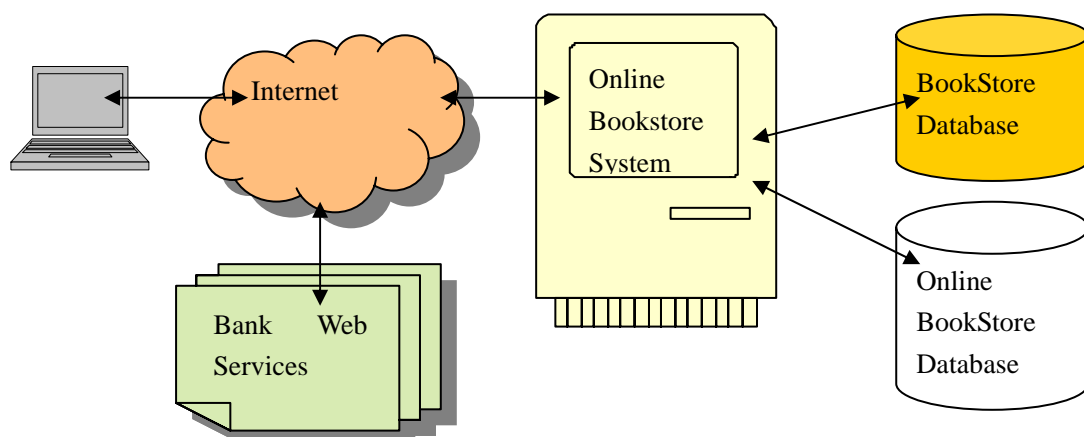


Figure 1. Agent-based Online-bookstore System Architecture

### 2.3 System Requirements

- Customers can register an account, edit his profile, and view his purchase history.
- Before the customer purchases, he must login. The customer can search for books without login.
- Allow customers to enter keywords for book title or author names (mandatory), as well as other book attributes such as language, edition which are optional.
- After choosing books and delivery method, the system offers online banking service for customers.
- After the customer login, the system will calculate and grab the preference rules based on his profile and purchase history and save them in the cache.
- Search is based on CBR technology. Similarity will be calculated for each book attribute and preference rules will overwrite default settings.

### 2.4 Assumption

- All the books info, delivery and bank info are saved in Bookstore database.
- Online-banking services are offered through different bank web service, and security is not a problem.
- Bank Web Services are agent-based web services which also conform to the system domain ontology.
- Only support credit card payment with pre-defined banks so far.

### 2.5 Wish List (Not implemented)

The wish list here lists all the functionalities which will be implemented during this iteration but will be add in the future iterations.

- After the customer login, system can recommend new books according to his preference.
- Add more purchase management functions such as delete purchase order based on some strategy, allow customers to track order process.
- Offer more banking service such as debit card, loan, etc.
- Allow system to find most suitable delivery services offered through Delivery suppliers' web services.
- The customer can search books from different online bookstores through their web services. One more agent "BookstoreBroker" will be added to handle the communication between system and bookstores web services.

### 3 System Design Documents

In this part, we are going to follow the PASSI methodology to do the analysis and design for this system. PASSI is a requirement-to-code methodology.

#### 3.1 System Requirement Model

System requirement model aims to analyze system requirement and identify agents based on use-case diagram.

##### 3.1.1 Domain Description Phase

In this phase, I decompose system functionalities and draw a use-case diagram using the classical object-oriented method. Here we have 8 use cases. Worth mentioning, I separate the Web interface from the system functionalities and make it one use case dealing with interaction between system and customers input in order to make it available as one agent in the next “Agent Identification Phase”.

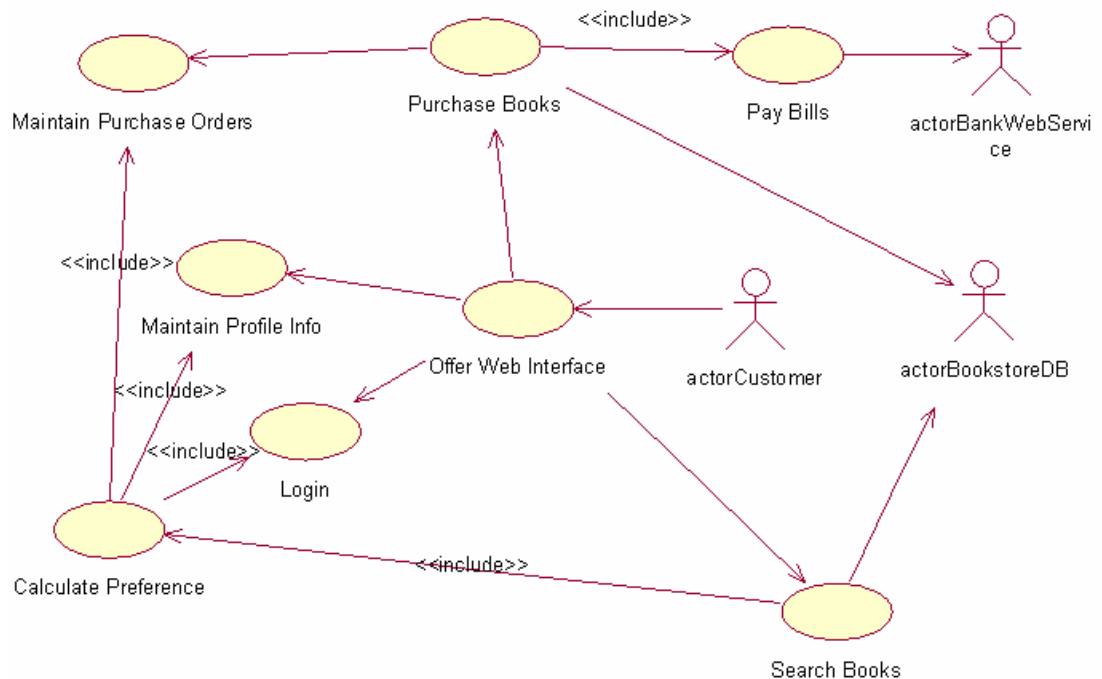


Figure 2. Domain Description Diagram

User case 1: Offer Web Interface

Actors: actorCustomer

Purpose: Offer Web Interface to customers

Overview: The system offer interfaces to customers to login, enter search conditions, choose books they want to purchase, enter financial information and confirm the purchases, etc.

#### User case 2: Login

Actors: none

Purpose: Verify customer's account info

Overview: The system verifies customer's username and password against online bookstore database.

#### User case 3: Maintain Profile Info

Actors: none

Purpose: Maintain customers' profiles

Overview: Customers can register a new account and edit his account info such as name, gender, address, etc.

#### User case 4: Maintain Purchase Orders

Actors: none

Purpose: Maintain customers' purchase history

Overview: Customers can view his purchase orders, and may delete or change his purchase in later release. Right now the system will add purchase record to the online bookstore database after successful purchase.

#### User case 5: Calculate Preference

Actors: none

Purpose: Calculate and induce customer's preference rules

Overview: Based on the customer's profile and purchase history, the system applied the data mining techniques to grab some preference rules such as if the customer purchases books in computer subject area, he prefers the book language to be Chinese, or the customer prefers large fonts edition, etc.

#### User case 6: Search Books

Actors: actorBookstoreDB

Purpose: Search the books according to search query customers offer

Overview: Based on the search conditions customers offer as well as customer's preference rules, find the most suitable books from the Bookstore Database.

#### User case 7: Purchase Books

Actors: actorBookstoreDB

Purpose: achieve the whole purchase process from online

Overview: starting from chosen books, customers choose delivery method from the delivery options information grabbed from Bookstore Database and offer financial information, finally sending request to update purchase history

#### User case 8: Pay bills

Actors: actorBankWebService

Purpose: talk outside with Bank web services

Overview: Based on the financial information offered from customers, choose the right bank web service to do the validation and return the result.

### 3.1.2 Agent Identification Phase

Since use cases 2,3,4,5 “Login”, “Maintain Profile”, “Maintain Purchase Orders” and “Calculate Preference” are all dealing with customer accounts, I group them into one agent called “AccountManager”. Other use cases form different agents by themselves.

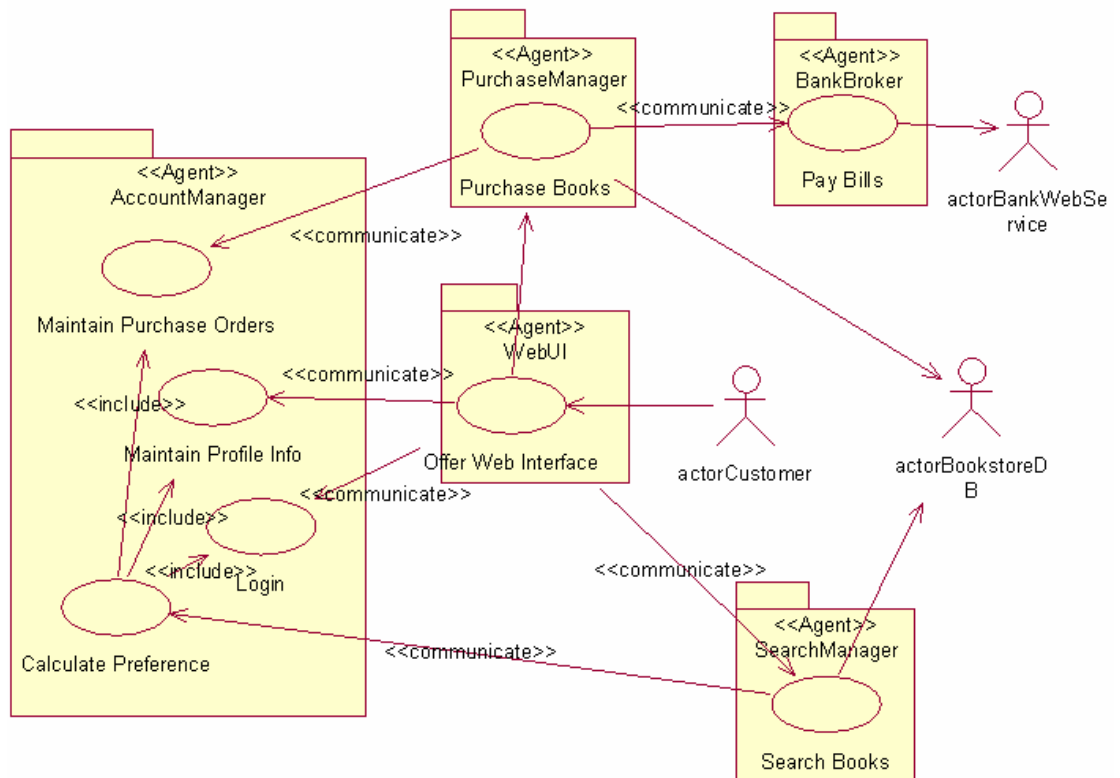


Figure 3. Agent Identification Diagram

- **WebUI Agent**

Act as Web interface between system and customers. Handle all the requests from the customers, validate the inputs and show results from system such as search result, profile edit, etc.

- **AccountManager Agent**

Deal with all the account issues including authentication, profile management, purchase history management. In addition, after login this agent will calculate and infer the customer preference rules based on the customer’s profile and purchase history. This is the only agent who talks with the online bookstore database.

- SearchManager Agent

Deal with all the search issues including grabbing preference rules from AccountManager, search the bookstore database based on the search query and preference rules, send search result to WebUI agent.

- PurchaseManager Agent

Deal with all the purchase issues including inquiring delivery method, grabbing payment information from customer and sending to BankBroker agent, sending request to AccountManager to update the customer's Purchase history.

- BankBroker

Act as a broker between PurchaseManager and the outside Bank Web services.

### 3.1.3 Role Identification Phase

In this phase, sequence diagrams are drawn to describe all the possible scenarios based on the communication paths between agents to help find all the possible roles for each agent. I gave 4 major scenarios – login & registration, search and purchase which cover all the communication paths. Here the object is represented as <role\_name>:<Agent\_name>.

Through this phase, we got the following roles for the agents.

- WebUI Agent roles

**AccountUIHandler** – take all the customer's inputs regarding login, registration and transfer to AccountManager Agent

**SearchUIHandler** – take all the customer's inputs regarding search query and transfer to SearchManager Agent

**PurchaseUIHandler** – take all the customer's inputs regarding chosen books, delivery method and personal financial information and transfer to PurchaseManager Agent

- AccountManager Agent roles

**PurchaseRecorder** – accept purchase update request from PurchaseManager Agent and write the record into online-bookstore database

**AccountProcessor** – accept login and registration information from WebUI Agent and do the authentication or generate/modify account information

**PrefInfoProvider** – offer preference rules to SearchManager Agent for optimal searching

- SearchManager Agent role

**SearchProcessor** – deal with all the search process

- PurchaseManager Agent role

**PurchaseProcessor** – deal with all the purchase process

- BankBroker Agent role

**Broker** – take customer’s financial information from PurchaseManager Agent and validate against corresponding bank web service

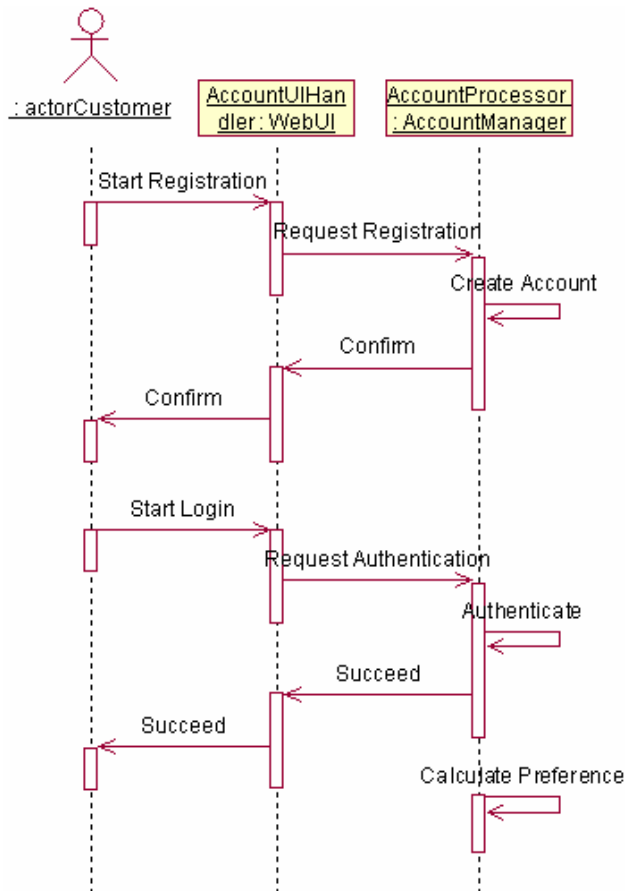


Figure 4. Role Identification Diagram – Login & Register

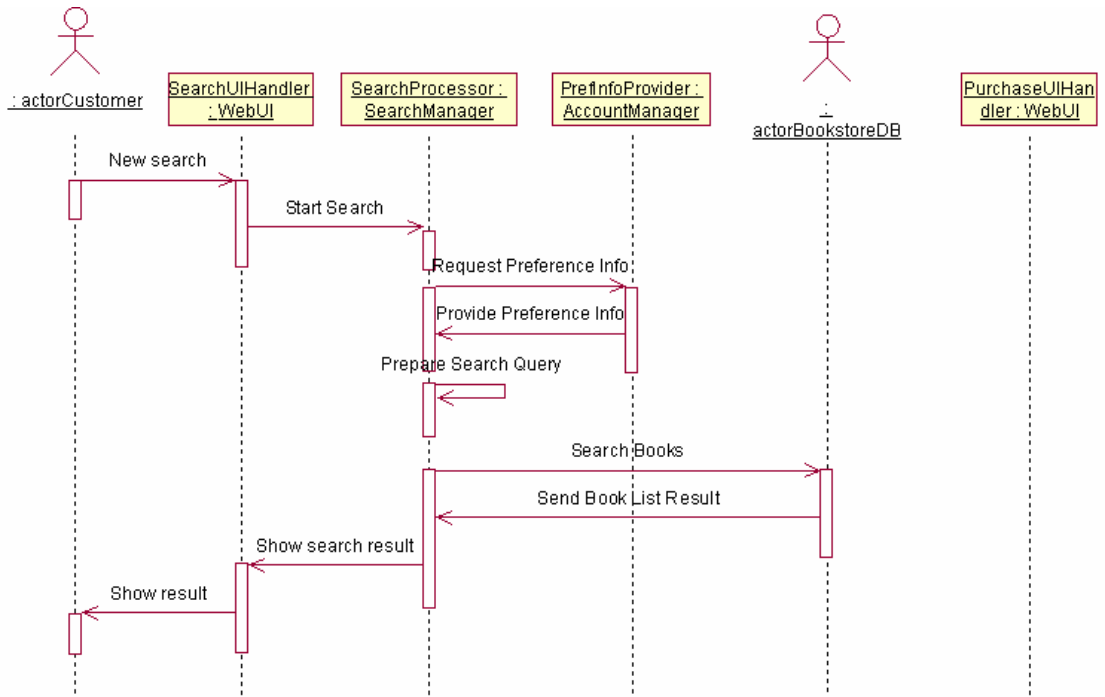


Figure 5. Role Identification Diagram – Search

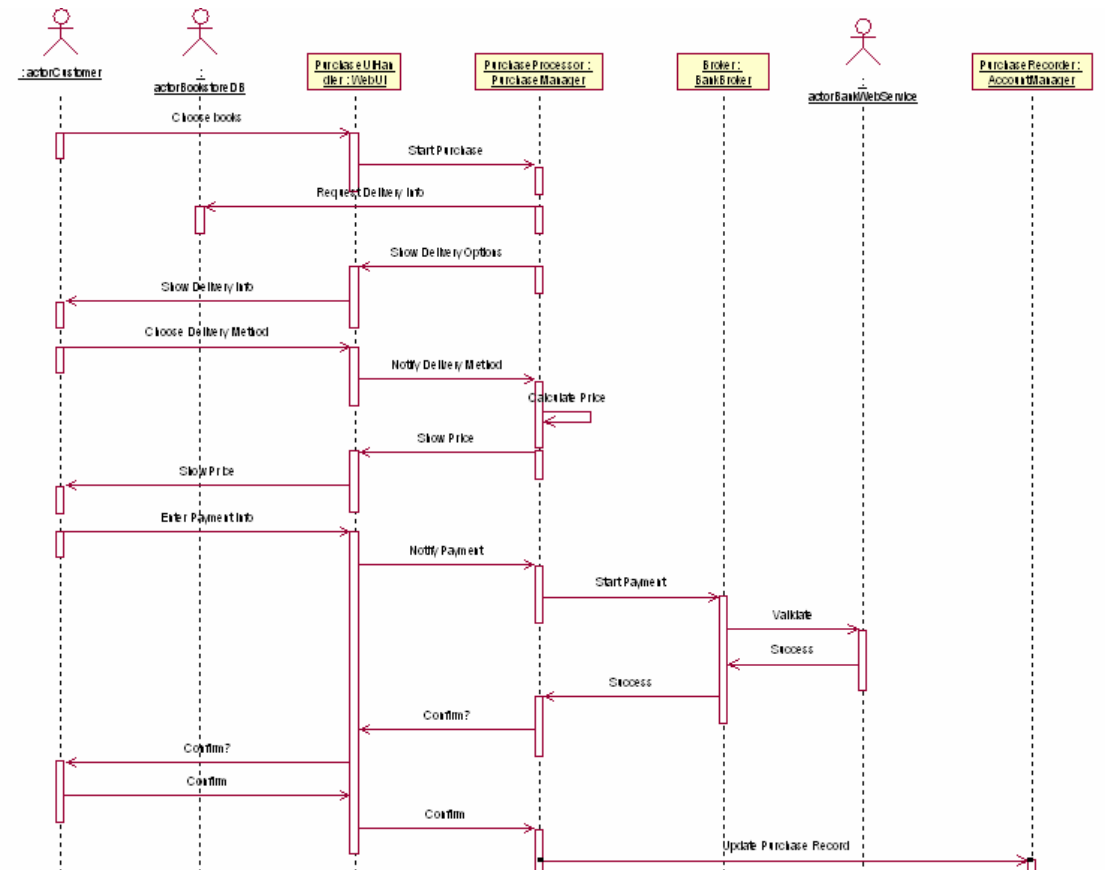


Figure 6. Role Identification Diagram – Purchase

### 3.1.4 Task Specification Phase

In this phase, we focus on each agent's behavior and try to find all the possible tasks in each agent ignoring agent roles information using activity diagrams. The diagram is a little different from object-oriented design. All the agent's tasks are in the right swim lane, while relative interacting agents tasks are in the left swim lane under "Other Agents". In this way we can explore all the interaction between agents and internal actions in the agent.

Also our system is based on FIPA-OS, so we give "Listener" task here for accepting all the incoming messages.

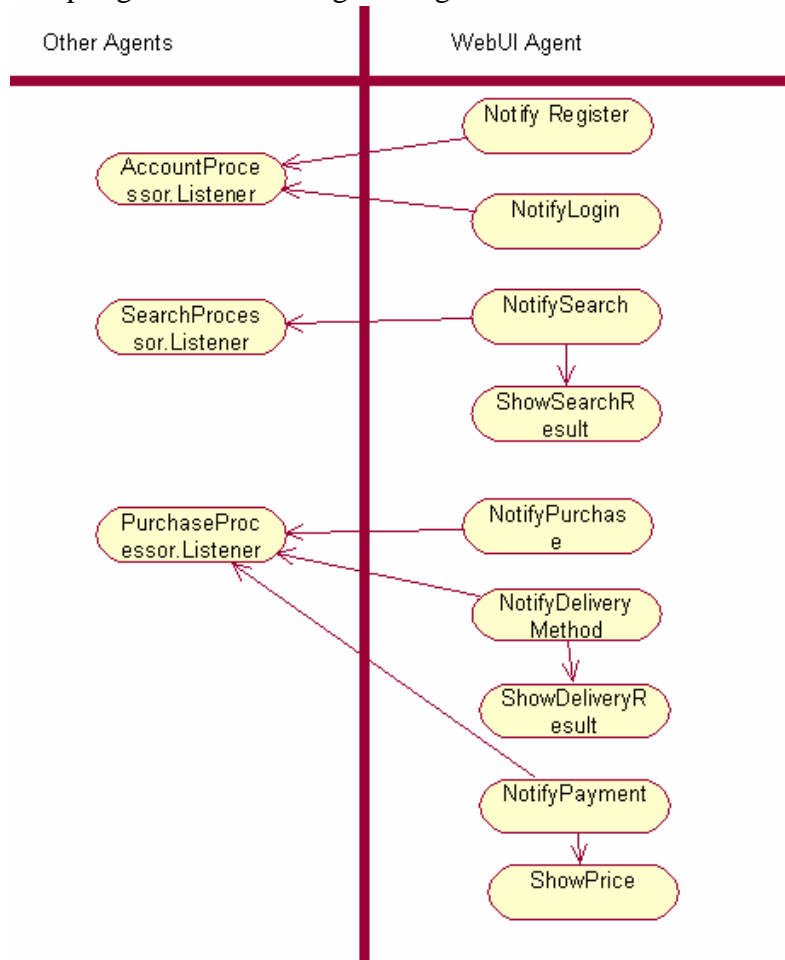


Figure 7. Task Identification Diagram – WebUI Agent

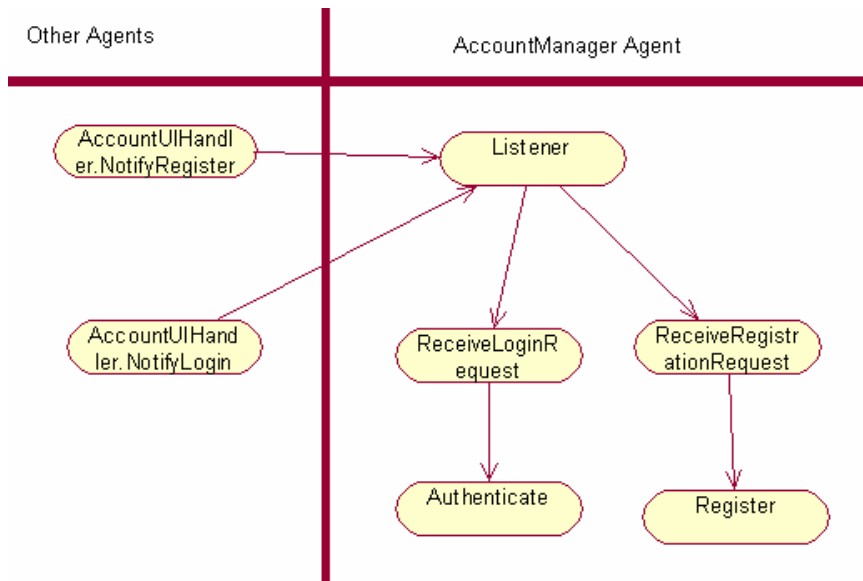


Figure 8. Task Identification Diagram – AccountManager Agent

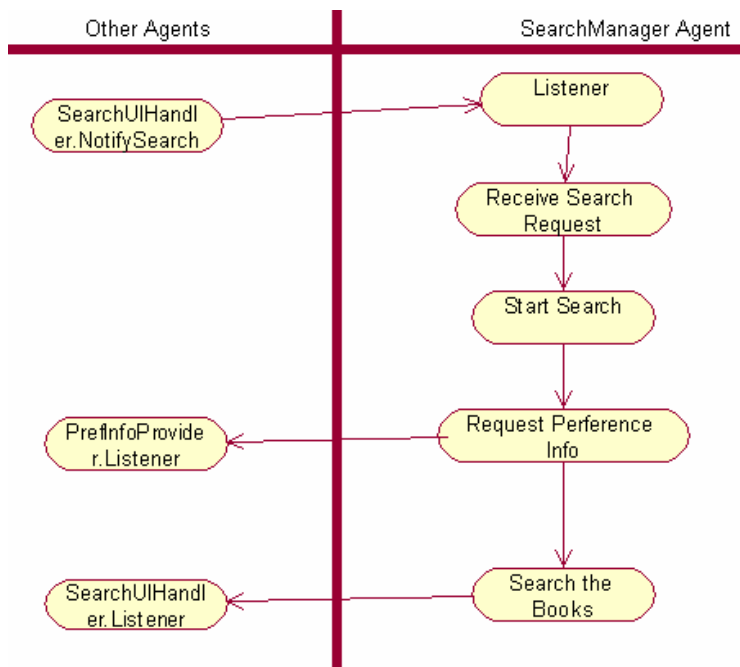


Figure 9. Task Identification Diagram – SearchManager Agent

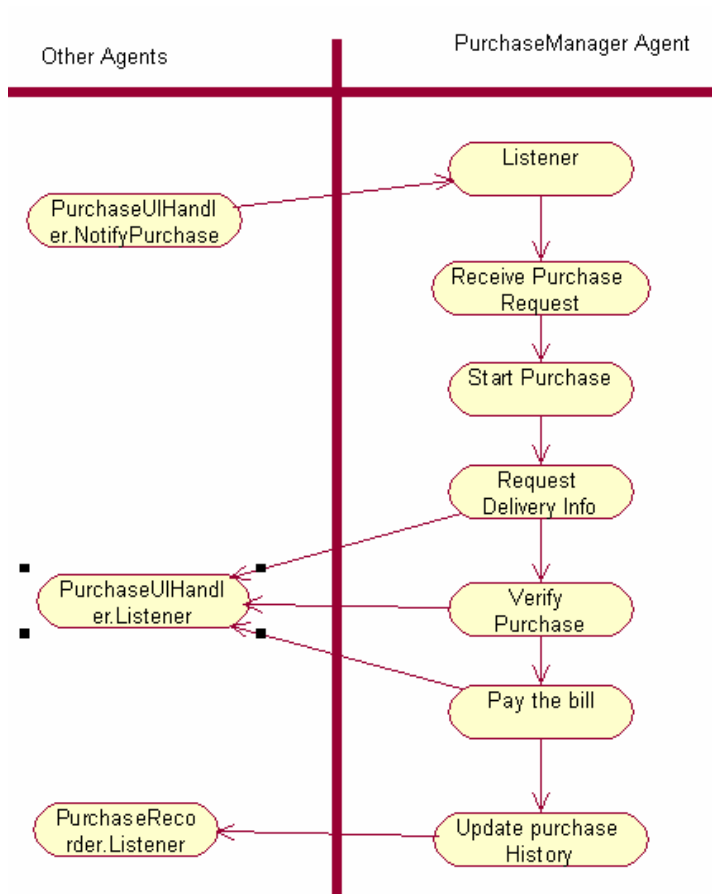


Figure 10. Task Identification Diagram – Purchase Agent

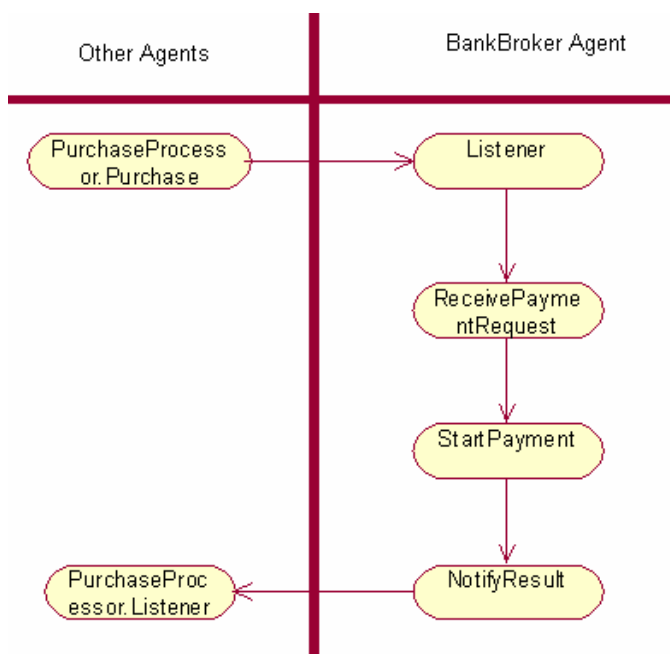


Figure 11. Task Identification Diagram – BankBroker Agent

### 3.2 Agent Society Model

This model deals with agency issues in a society point of view.

### 3.2.1 Ontology Description Phase

This phase contains two diagrams. The first one is domain ontology description which gives the domain knowledge for the whole system as we can see as below.

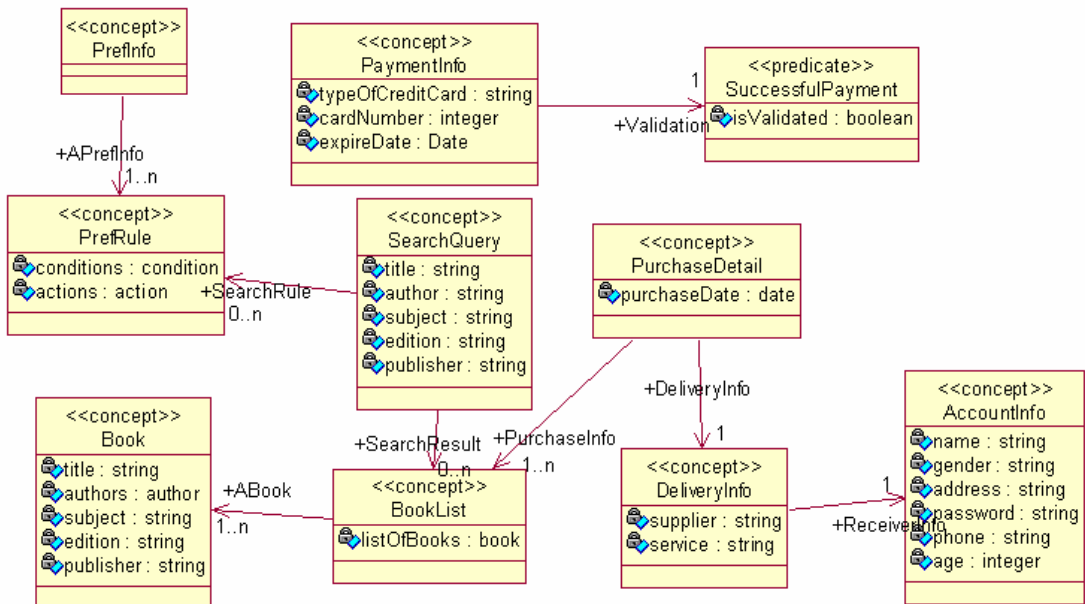


Figure 12. Domain Ontology Diagram

The second one is communication ontology description which describes the local knowledge for each agent and communications between agents in terms of ontology, language and protocol. Here Agents are represented as classes with local knowledge in its attribute fields. (Note: access control signs have no meanings here) The blue boxes represent the communications between agents which are association classes in implementation to relate the instances of the two participating agents.

In PASSI, the add-in tool can translate ontology diagrams into RDF formats used in communication. The protocol is standard FIPA interaction protocol using ACL.

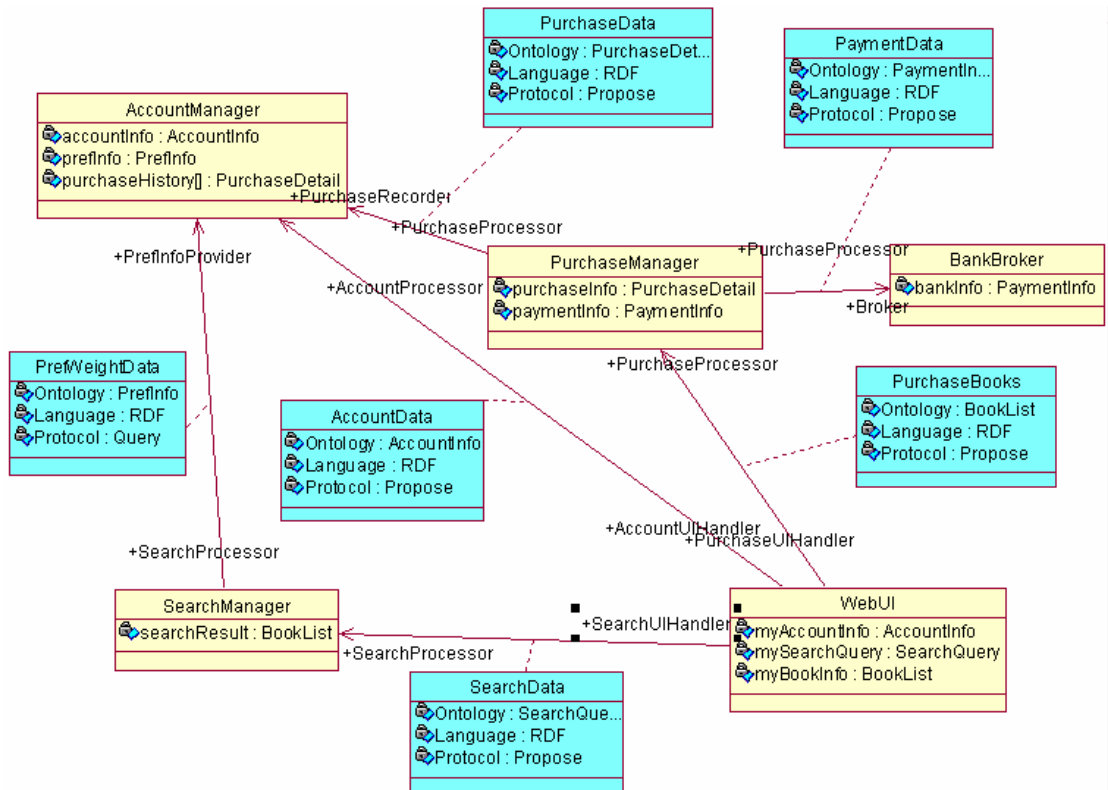


Figure 13. Communication Ontology Diagram

### 3.2.2 Role Description Phase

At this step, we emphasize on the role changes within each agent and communications and dependencies between agents. In the following diagram, roles are represented as classes in the agent package. Tasks are assigned to each role in order to help refine the task list. Arrow lines indicate the dependencies between agent roles and also define the communication data or services.

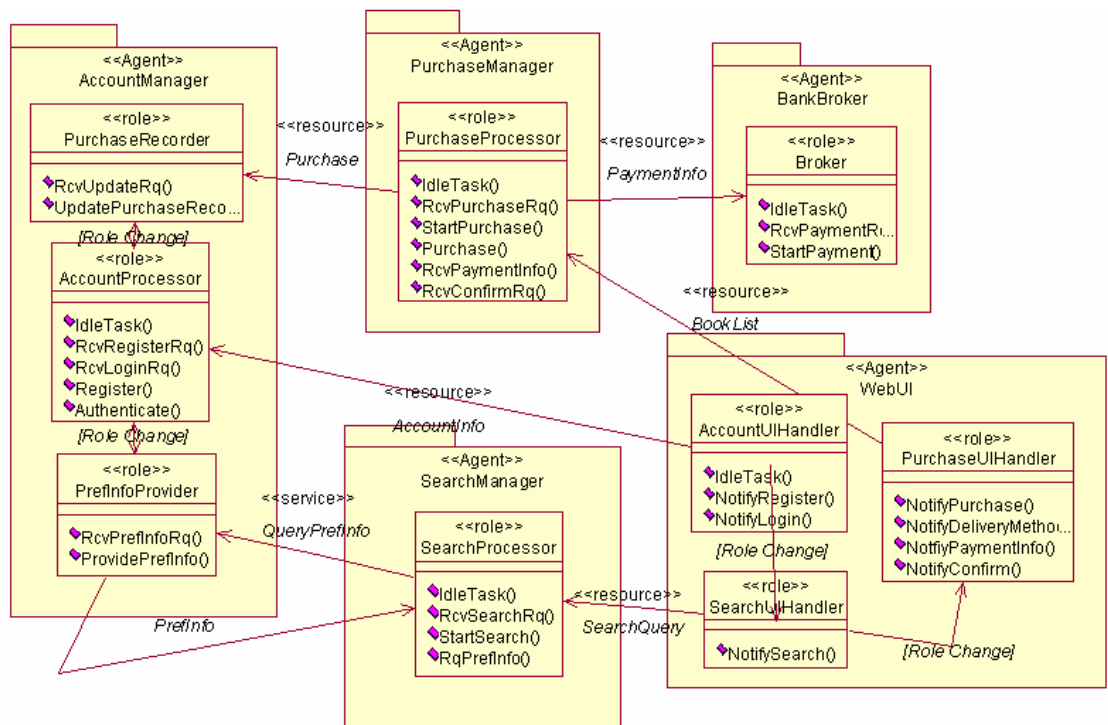


Figure 14. Role Description Diagram

### 3.2.3 Protocol Description Phase

Since our system is implemented on FIPA-OS platform which adopt standard FIPA interaction protocols, so we don't need to draw protocol description diagram here which defines the user-defined interaction protocol using AAML.

## 3.3 Agent Implementation Model

At this step we will deal with agent implementation from multi-agent view and single-agent view. In Multi-agent view we look at the agent society level, and each agent is an element of this society. In single-agent view, fully detailed structure for each agent is explored to produce code.

### 3.3.1 Multi-Agent Structure Definition Phase

In this phase, a class diagram is automatically generated by the PASSI add-in to represent the structure of system with a compact notation from communication ontology diagram and role description diagram. Here each agent is represented as the main agent class with knowledge in its attribute fields and all the tasks in its method fields. Environment and actors also play a part in it.

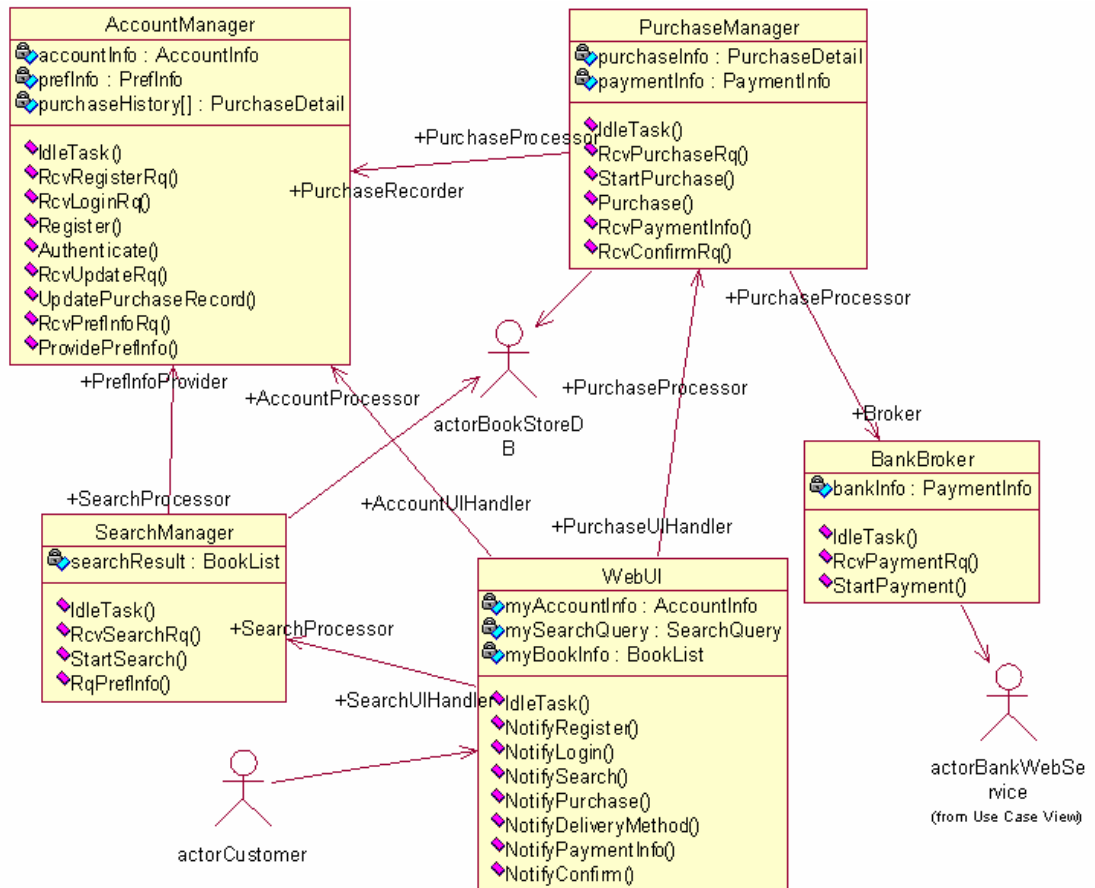


Figure 15. Multi-agent Structure Definition Diagram

### 3.3.2 Multi-Agent Behavior Description Phase

In this phase, activity diagrams are used to represent the behavior of the system at a level of detail that focus on the single method of the each agent task. Here we offer 4 diagrams illustrating the behaviors to handle login, register, search and purchase respectively. The label for each swim lane is represented as `<Agent_name>.<Task_name>`, and the content of each activity box is represented as `<Task_name>.<Method_name>`. Between different agents, the state transition line indicates “message” communication. Between different tasks within the same agent, the state transition line indicates “newTask” normally.

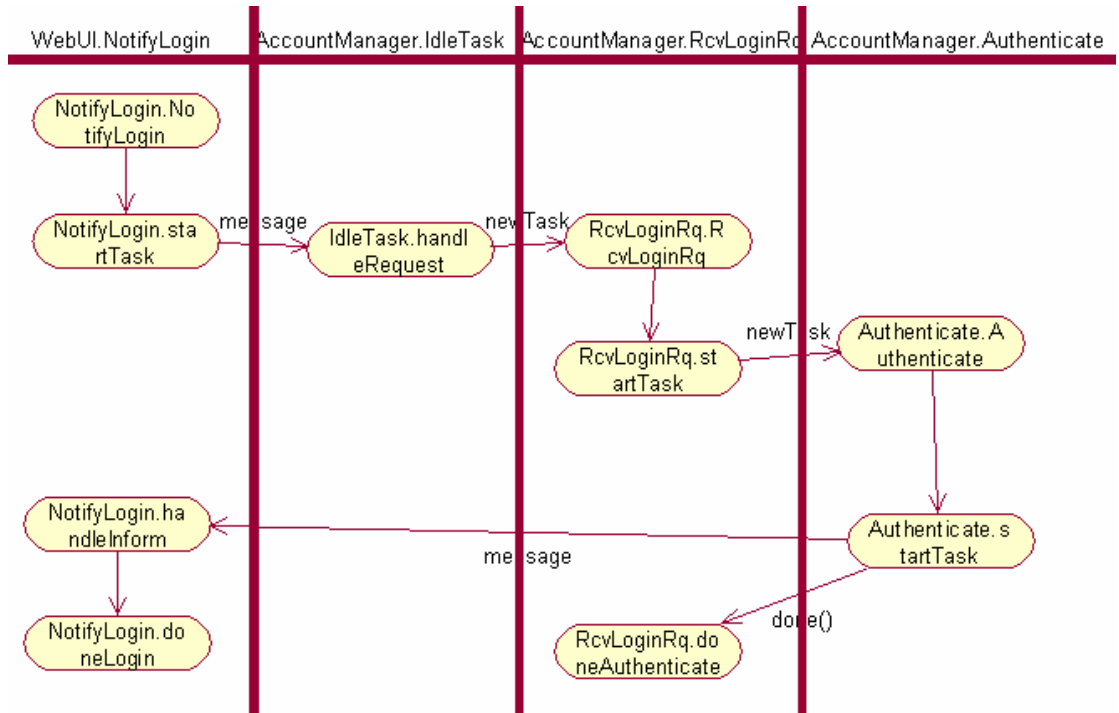


Figure 16. Multi-agent Behavior Description – Login

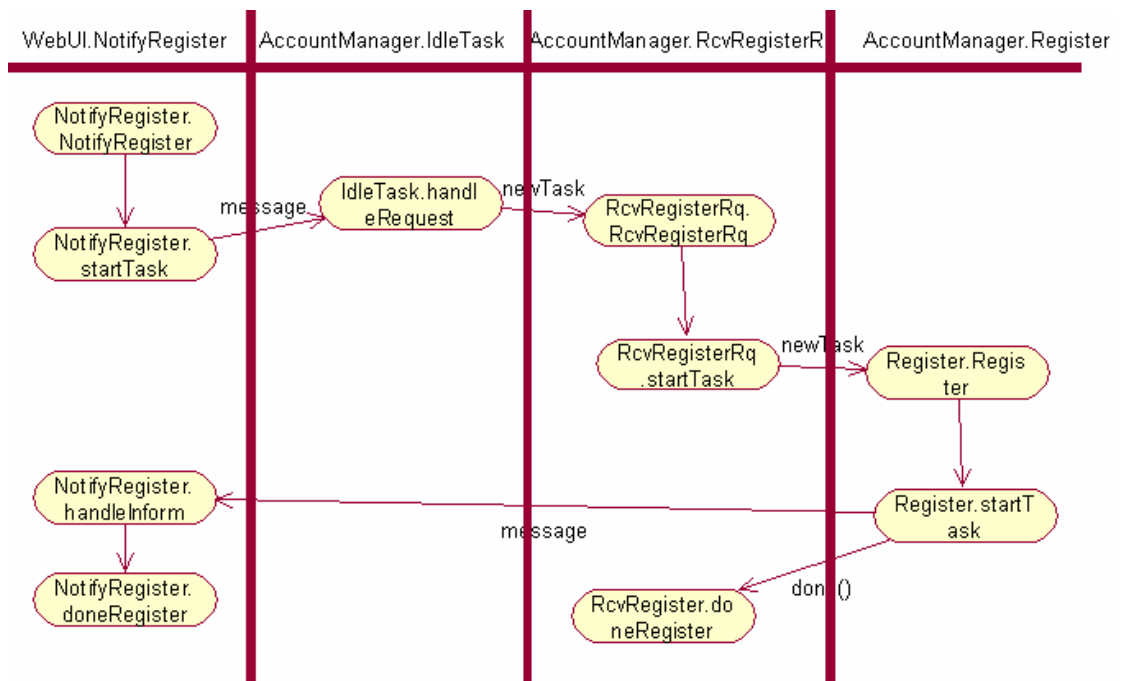
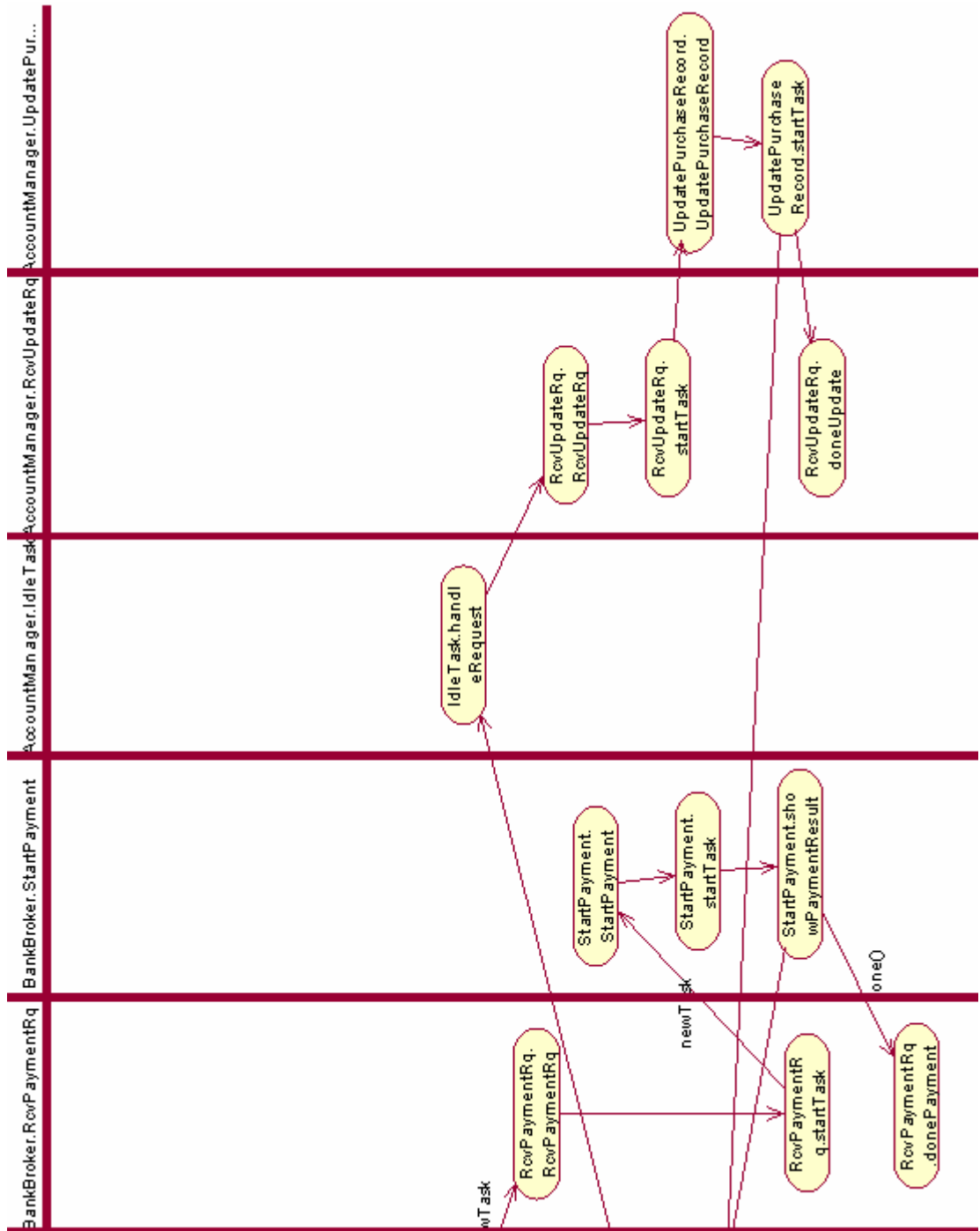


Figure 17. Multi-agent Behavior Description – Register



Figure 18. Multi-agent Behavior Description – Search







### 3.3.3 Single-Agent Structure Definition Phase

From the multi-agent behavior description diagrams, we can easily find almost all the methods for each agent task. In this phase, real implementation classes are considered. Here agent is represented as a class derived from base agent type class “FIPA\_Agent”, and tasks for this agent are classes derived from “Task” class. 5 Agent Structure Definition diagrams are given as followed.

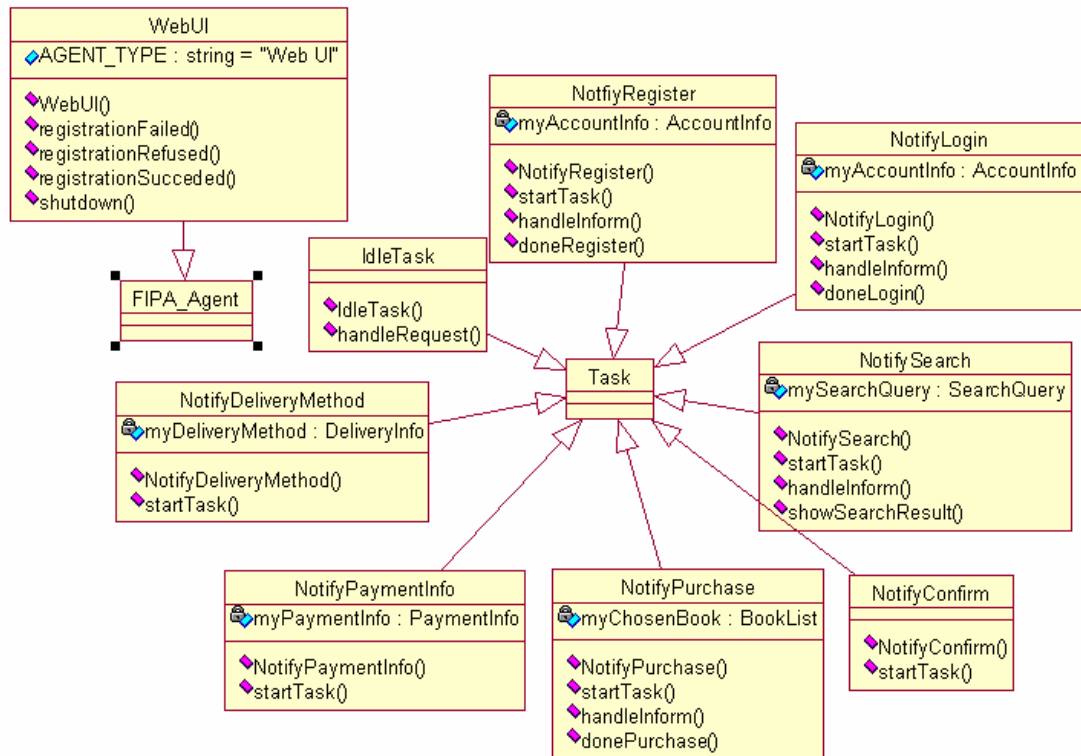


Figure 20. Single-agent Structure Definition Diagram – WebUI Agent

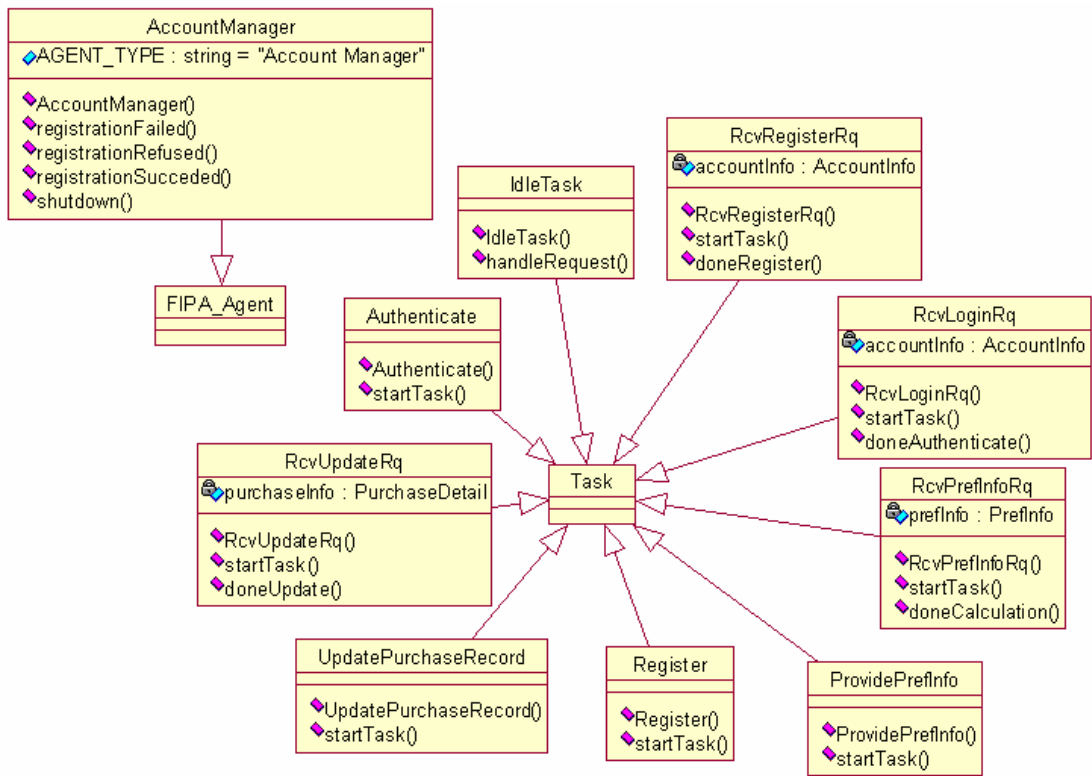


Figure 21. Single-agent Structure Definition Diagram – AccountManager Agent

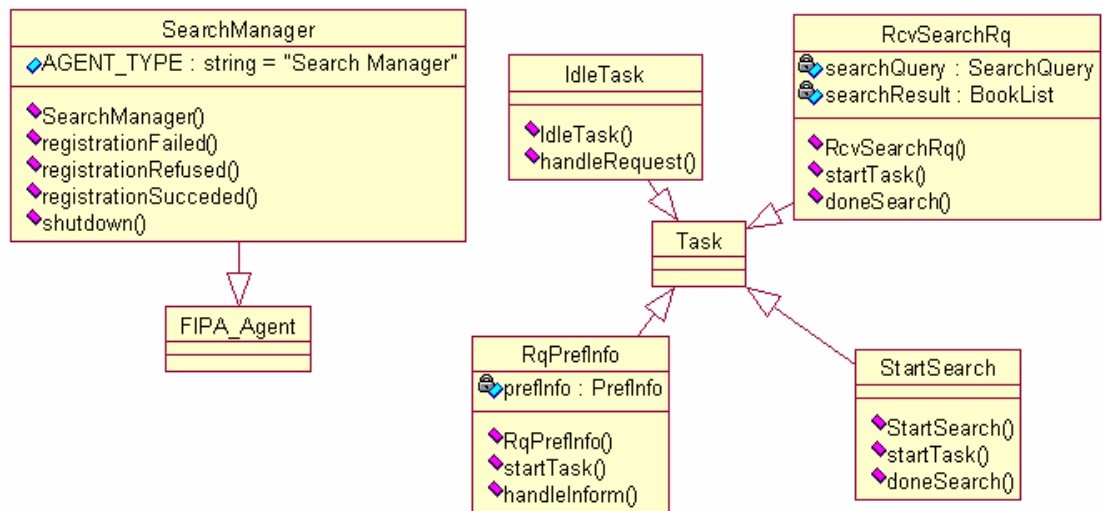


Figure 22. Single-agent Structure Definition Diagram – SearchManager Agent

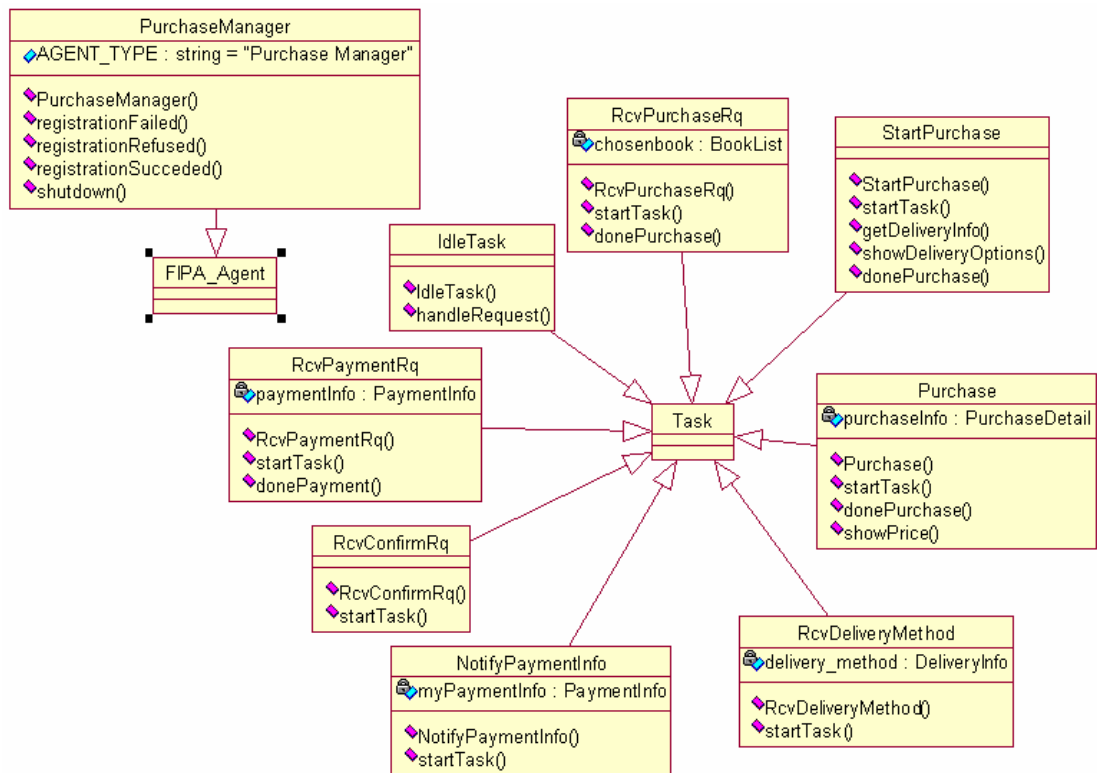


Figure 23. Single-agent Structure Definition Diagram – Purchase Agent

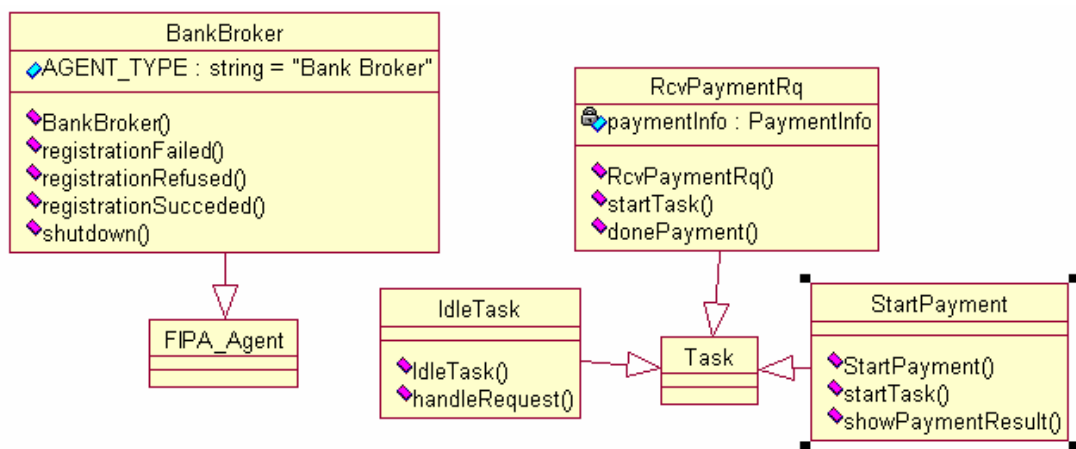


Figure 24. Single-agent Structure Definition Diagram – BankBroker Agent

### 3.3.4 Single-Agent Behavior Description Phase

In this phase, flowcharts, state diagrams or semi-formal text descriptions can be used to explain the implementation of the methods of the classes. Since some methods are common methods from the FIPA-OS API and others are self-explainable from previous diagrams, we don't illustrate them any more. FIPA-OS API documents can be referenced if needed.

## 3.4 Code Model

Our system is built on FIPA-OS platform, PASSI tool will automatically generate skeleton of the codes according to the predefined patterns of agents and tasks with the design diagrams. And programmers just need to complete the implementation of methods. Since I don't have this tool, I will leave this part open.

### 3.5 Deployment Model

#### 3.5.1 Deployment Configuration Phase

All the agents will be assigned to one Web server machine. The server will talk with Bank web services for validation of customer's financial information, get book and delivery information from Bookstore Database on another computer. In the next release, we will use different bookstore web services instead of one bookstore database for the book information. In addition, one online-bookstore database is needed for our system to save all the customers' profiles and purchase histories.

Since our system is built on FIPA-OS platform, the communication between agents is based on ACC (Agent Communication Channel). We assume the bank web services are agent-based web services which also conform to the system domain ontology. So the communication between BankBroker Agent and Bank Web Service is using SOAP as transport protocol, and ontology is PaymentInfo.

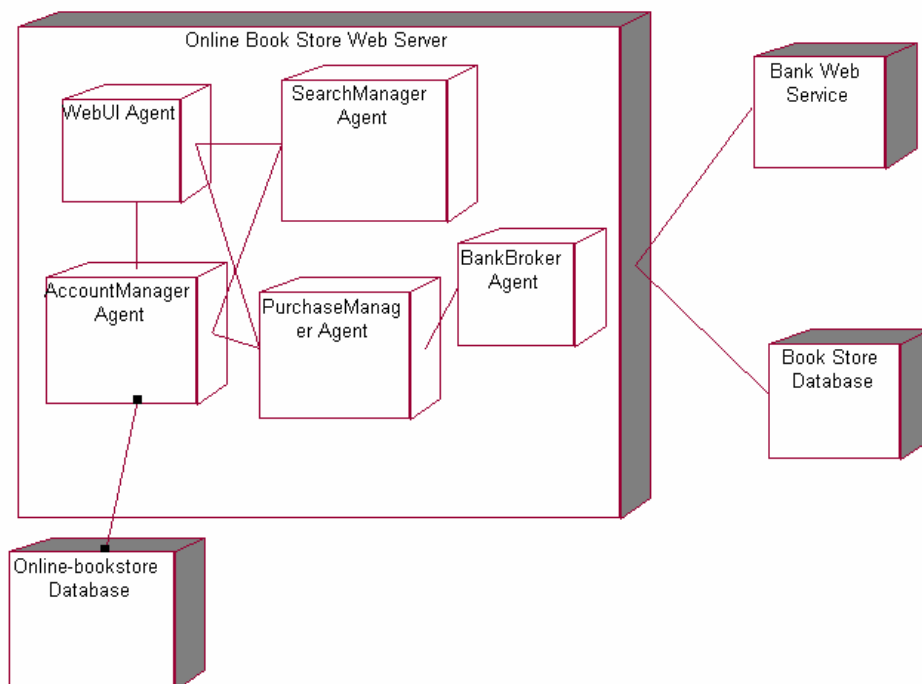


Figure 25. Deployment Configuration Diagram

## 4 Conclusion

The purposes of this project are first to apply multi-agent technology into this online

bookstore application, second to adopt PASSI methodology into the analysis and design process. Whether multi-agent technology is a good choice to apply to this application is not a consideration here. Actually considering the nature of this application which doesn't need much parallelism and is a relatively close system, the traditional web technology like J2EE may be a better solution. In terms of PASSI, it's very easy to use, and very helpful to guide the designers through the whole development process from requirement analysis to system deployment.

## 5 References

1. Dr. Behrouz H. Far, Agent-based Software Engineering, Course Slides, University of Calgary, 2006-10-25
2. Massimo Cossentino, Agent-Oriented Methodologies, Chapter IV, ICAR-CNR, Italy
3. Massimo Cossentino, Colin Potts, A CASE tool supported methodology for the design of multi-agent systems, Italy
4. Stefan Poslad, Phil Buckle, Rob Hadingham, The FIPA-OS agent platform: Open Source for Open Standards, UK, <http://fipa-os.sourceforge.net/docs/papers/FIPAOS.pdf>