

Course Project: MASE System

Chengyao Deng

cdeng@cpsc.ucalgary.ca

October 9, 2005

Agent Based Software Engineering

(SENG 609.22)

Instructor: Dr. Far

1. Abstract:

This project is build based on the MASE (MILOS-ASE)^[1] system which is a Tool Support for Agile Software Teams and has been implemented by the EBE group of University of Calgary. Though the source idea of the project is based on agile method of Software Engineering, I try to describe MASE in the multi agents' way, using the Prometheus methodology^[2] for building multi agents systems, which I learned this term in SENG 609.22, instructed by Dr. Behrouz Homayoun Far. In fact, the real system of MASE is too large for me to introduce clearly because of the limited time and energy. Therefore I only choose the key features to build my agents, which, however, is enough for demonstrate the multi agents' way and the Prometheus methodology.

2. Business Case:

The MASE (MILOS-ASE) stands for “Minimally Invasive Long-term Organizational Support for Agile Software Engineering”.^[1] This project investigates methods, techniques, and tools that support the coordination of agile software teams.^[3]

MASE system is based on WikiPage collaboration tool. Using WikiPage, the authorized user of the system can change the style and content of every Wiki page in any way they like. Of course, there is a namespaces and team-based access control permissions for Wiki pages.

Team members of an agile Software Engineering group are called the users. We divided the users into three parts: one who has not logged in is unauthorized user, He does not have the privilege of changing the page and other creating activities; a user who has logged in is authorized user, he has the changing and creating privileges but cannot create an iteration or a team; these things can only be done by the last type of user, the team manager.

An authorized user can also add a new person to the project team. If one is added to the team, he changes to an authorized user with the privilege of changing the pages, creating activities, editing project details and adding other users in this team.

An authorized user can do agile project planning with the MASE Whiteboard. On the Whiteboard one can create iteration, create story card and check estimate accuracy by assigning estimated time to every story made by him.

Besides what an authorized user can do to the project of his team, a project manager can also create another team with its project and create iteration for each project.

On the white board, team manager can create iterations for a project. Iteration is the short time plan within a project. In each iteration, team members of this team can

create story card, which is his work in this iteration. A project manager can also create story cards and assign them to other team members and he can also finalize iterations.

3. System Overall Specification

In this section, the overall system specification will be demonstrated.

3.1 Functional Requirements:

From the above business case of the MASE system, the following functional requirements can be retrieved in order to build the use case diagram, which illustrate the functionalities of the system.

F1. When any body, authorized users or unauthorized users, accesses to the website they can read the information in the form of WikiPage.

F2. Unauthorized users, who are the team members, can login to the system, using their email addresses as their usernames and their own password. If the information they hand out is right, they change into authorized users.

F3. When unauthorized users become authorized users through logging in, they can organize and edit the WikiPage in the style they like by pressing the button “Edit” and save the changes by pressing the button “Save”. The changed pages can be browsed by other people.

F4. Every user has his personal information. When he logged in to the system, he can edit and change his information such as email address, name and password. The personal ID is assigned to the user automatically.

F5. Unlike other systems, MASE do not allow people to register freely. Instead, only authorized user can add another person into his team to become a team member. He needs to edit the personal information for the new team member.

F6. Team member can create story card for their work in iterations. By creating a story card team members can keep track of features that are to be completed by the team. For creating a story card, they need to edit the name, description and work hours for the story card. Then the story card is presented on the whiteboard in iteration showing the name of the team member.

F7. Each project has its own special requirements. And the requirements in Agile method are usually changed by the customer. So team members need to edit the details of projects from time to time.

F8. MASE provides an experience base of team members' time estimates. Based on

the team's estimation accuracy (velocity), MASE will suggest to you the most appropriate size (person-hours) for the next iteration of your project.

F9. Iterations and story cards are all residing in Whiteboard, which is an online "information radiator". The idea is to put all story cards in a place where everyone can view and manipulate them. The Whiteboard works like a collapsible tree. However all of the items in the tree can be edited by just clicking on them. After making any changes, team members can simply click on the "Save" button to save them. And the whiteboard allows team members to track their estimates for each feature, as well as how much time they have spent on that feature.

F10. Iteration is a period of time in which the team will attempt to complete a number of story cards. Only the team manager can create iteration by specifying the name and completion data. When doing iterative development, it may be helpful for a team to organize story cards into iterations.

F11. Only team manager can create a team for completing an existing project.

F12. When one iteration is completed, only the team manager can finalize the iteration.

3.2 Use Cases:

Figure 1 is the use case of MASE, in which the functionalities, relationships among them and the interactions of the system with the environment are briefly presented. The use case descriptor is not showed since the purpose of this project is based on the approach of agent based Software Engineering and the Prometheus methodology.

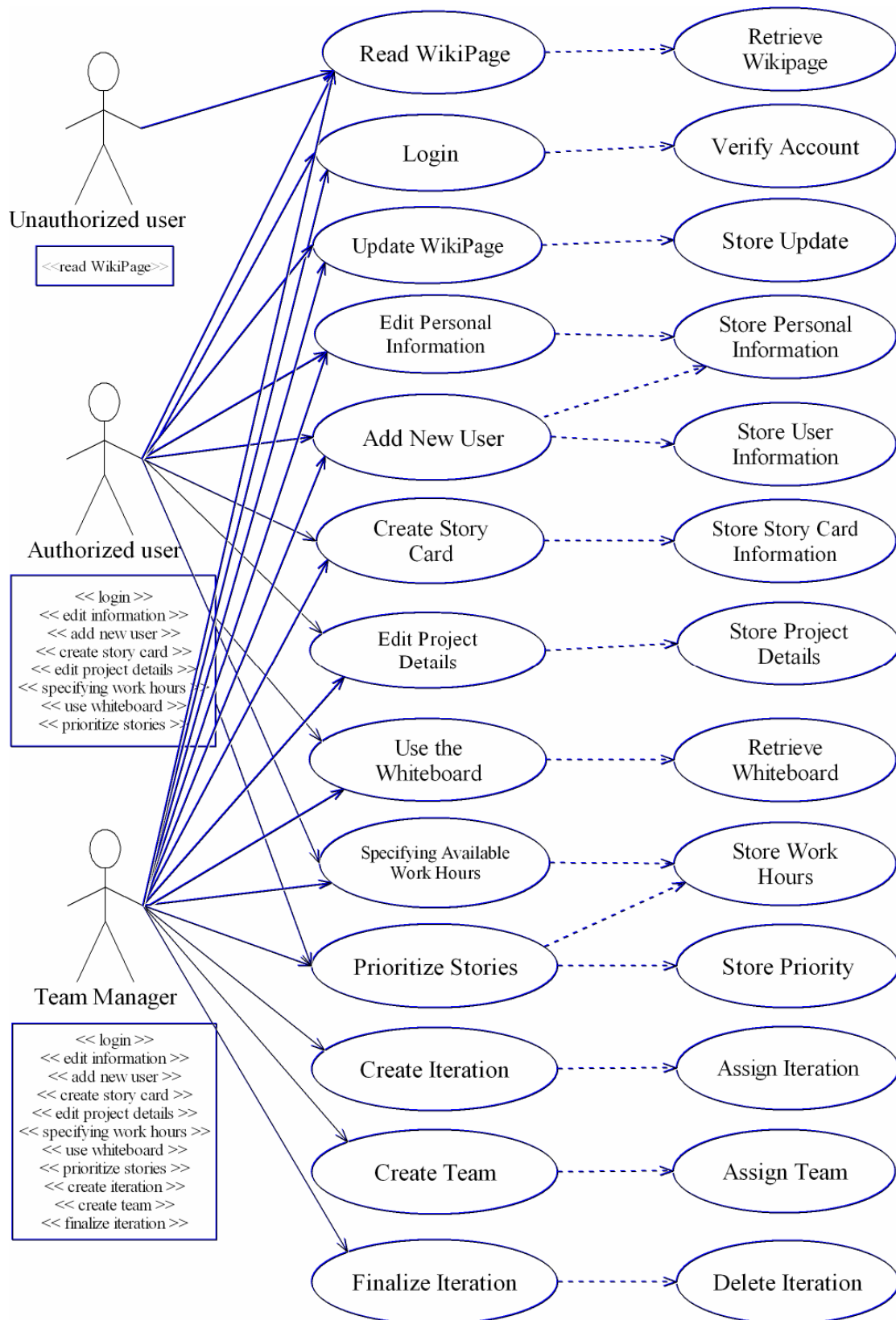


Figure1: Use Cases Diagram for MASE

4. System Architecture Design:

In this section, the agents to be used will be decided, by grouping the functionalities of the system. The rules for this process are as follows: If functionalities use the same data or they have significant interactions, group them to the same agent. And if they are clearly unrelated or they exist on different hardware platforms, separate them into different agents.

4.1 System Overview Diagram

The user account information is used by use cases Login, Edit Personal Information and Add Team Member. they all use the same data source as the Account Database. So these functionalities are grouped into the same agent named Account Management Agent, **AMA**.

MASE system is based on WikiPage which has specific feature. And the Update and Retrieve functionalities interacts with the WikiPage Database. So we need a WikiPage Management Agent, **WMA**, to deal with the updating and retrieving Wiki pages for other Agents.

Project Planning Agent, **PPA**, deals with the following functionalities: Create Story Card, Edit Project Details, Use the Whiteboard, Specifying Available Work Hours, Prioritize Stories. Since these functionalities have close relations with each other, we group them together.

Team Management Agent, **TMA**, consists of Create Team, Create Iteration and Finalize Iteration functionalities, which has strong relationships. In addition, they can only be executed by Team Manager.

Figure 2 is the System Overview Diagram for MASE, which describes how the system wholly functions with all the agents. The notations used bellow is defined in [2].

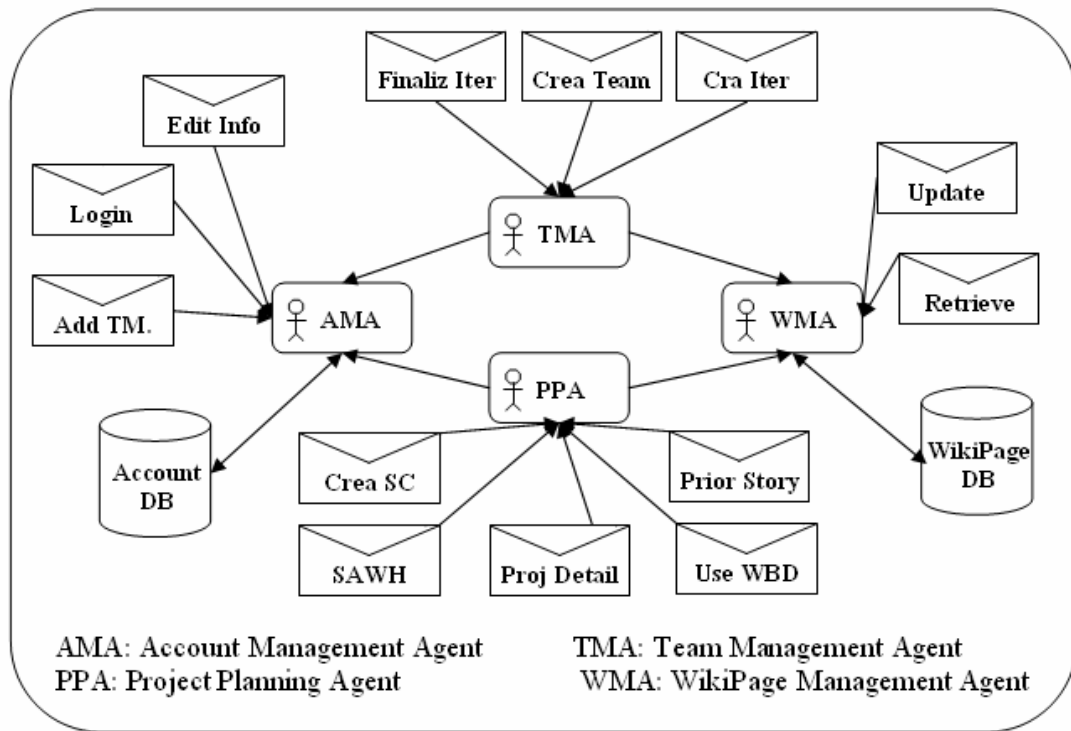


Figure2: MASE System Overview Diagram

4.2 Interaction Diagrams

We use interaction diagrams, which are borrowed directly from the Object Oriented Design, to display full interaction between agents. Therefore only the scenarios with the interactions among different agents are presented in the interaction diagrams.

Figure 3 shows the protocols among the TMA, PPA and WMA, while creating Iteration.

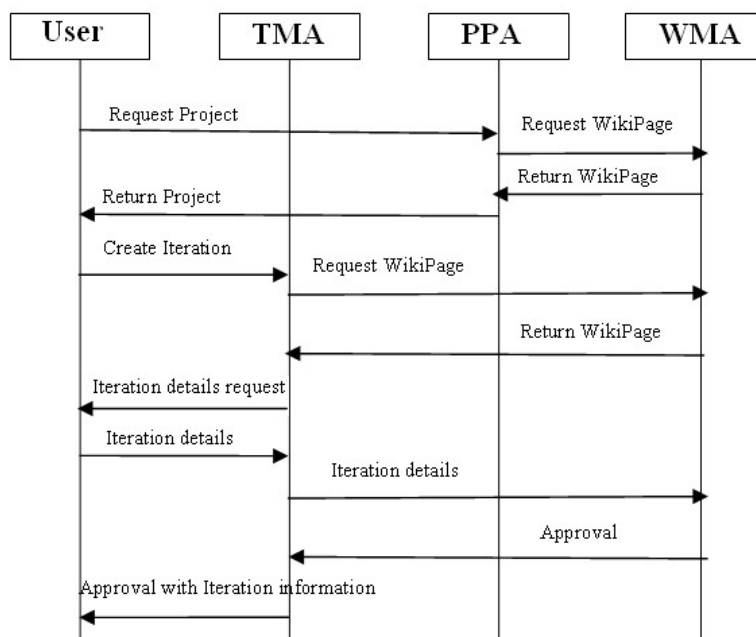


Figure 3: Create Iteration Interaction Diagram

Figure 4 describes the protocols among the TMA, PPA and WMA, while creating Story Card.

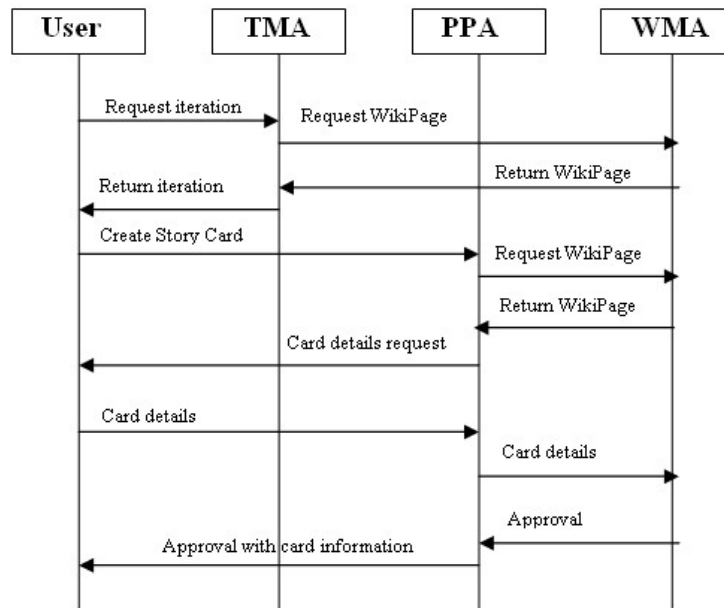


Figure 4: Create Story Card Interaction Diagram

4.3 Class Diagram

Since interaction diagrams give only a partial picture of the system's behavior, we need the interaction protocols, which define the sequences of the interaction, to have a precisely defined system.

Class Diagram^[4] is suitable for clearly displaying the interfaces between the agents and the databases. Figure 5 shows the Class Diagram of MASE.

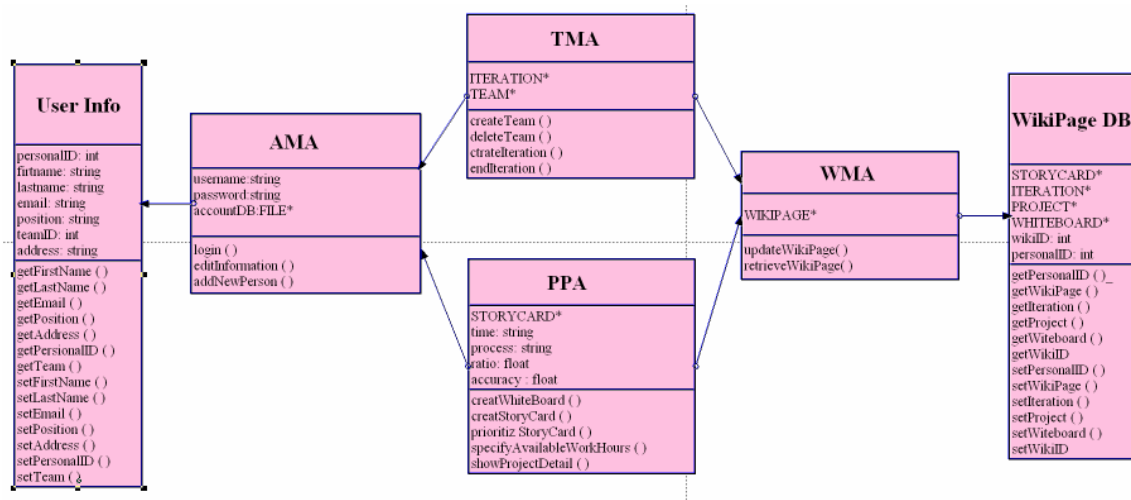


Figure 5: MASE Class Diagram

5. Detailed Design:

In this section, we focus on designing the detailed internal structure for each agent.

5.1 Agent Overview Diagram

From the agent overview diagram we can see the top level view of the agent internals, which is very similar in style to the system overview diagram. The letter envelope icon represents the functionalities of the agent. The round corner rectangle icon represents the capabilities of agents.

Figure 6 shows the Agent Overview Diagram for AMA agent, which has three capabilities to deal with different request. These capabilities includes: Create Person Information, Verify Account and Store Personal Information.

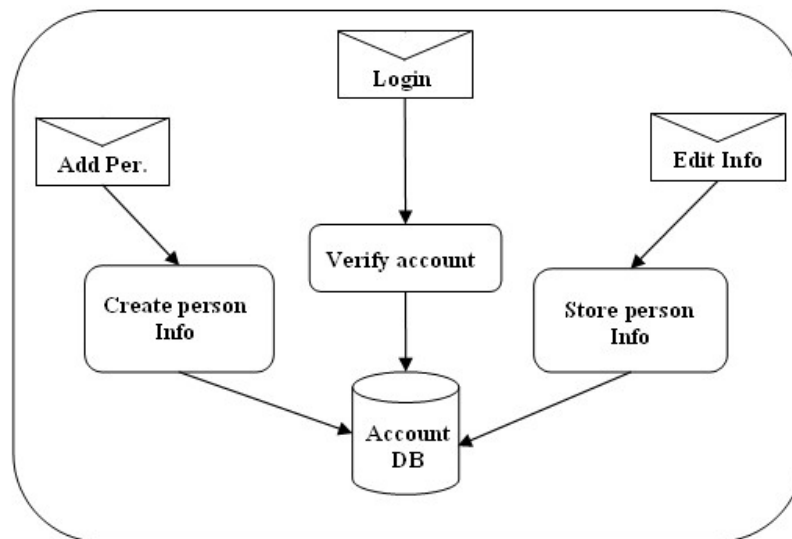


Figure 6: Agent Overview Diagram for AMA

Figure 7 shows the Agent Overview Diagram for WMA agent, which has two capabilities to deal with different request. These capabilities includes: Update WikiPage and Return WikiPage.

Figure 8 shows the Agent Overview Diagram for TMA agent, which has four capabilities to deal with different request. These capabilities includes: Delete Iteration, Return WikiPage, Get New Iteration and Shown on Whiteboard.

Figure 9 shows the Agent Overview Diagram for PPA agent, which has five capabilities to deal with different request. These capabilities includes: Get New Story Cards, Assign Work Hour, Update Project Details, Assign Priorities and Show Whiteboard.

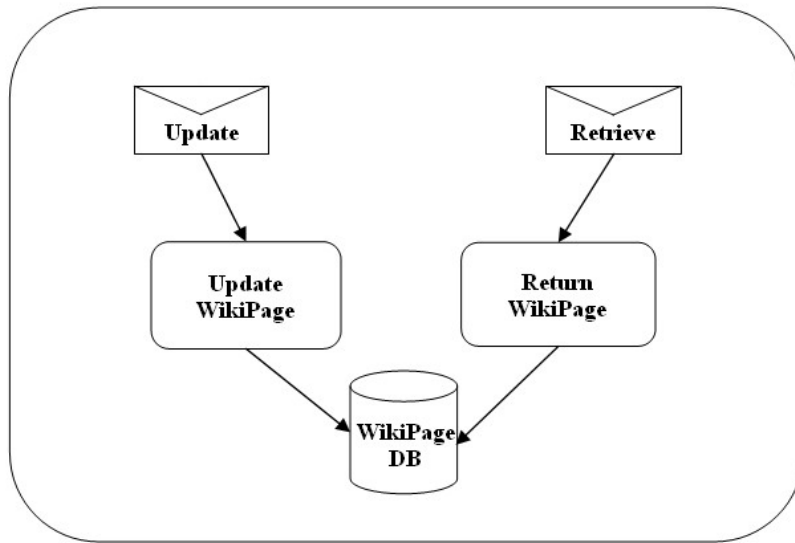


Figure 7: Agent Overview Diagram for WMA

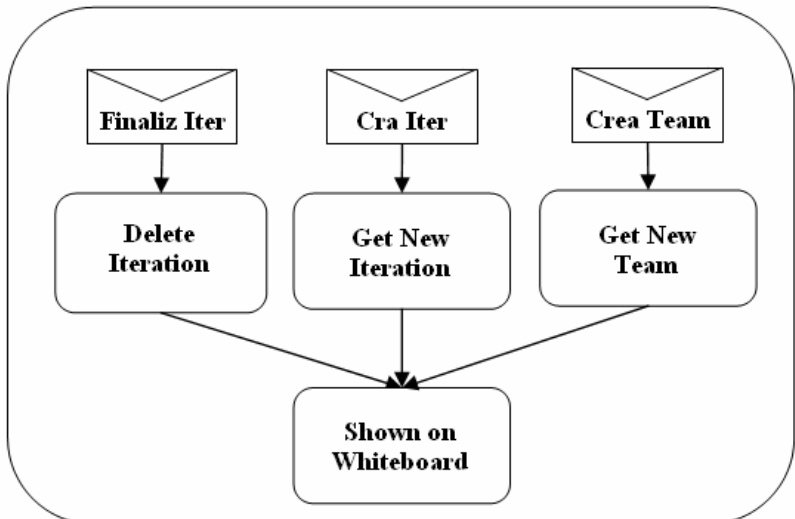


Figure 8: Agent Overview Diagram for TMA

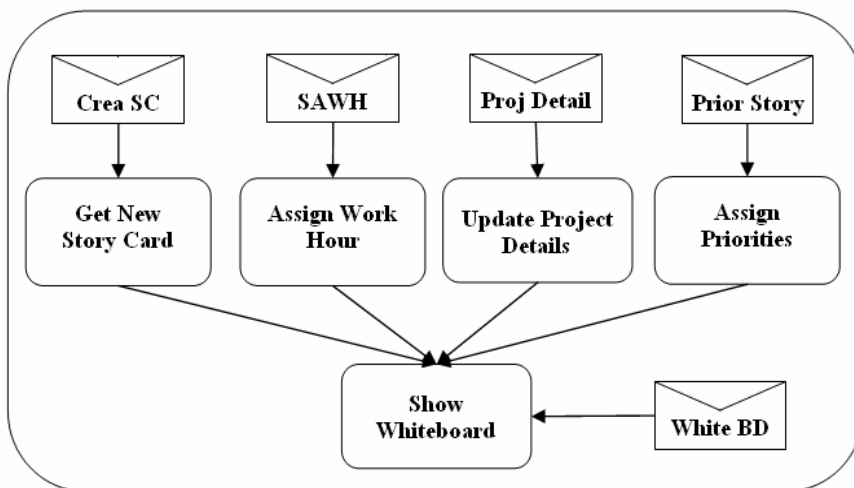


Figure 9: Agent Overview Diagram for PPA

5.2 Data Specification

Because I used databases in the project, I need to draw the Entity Relationship Diagram to demonstrate the relationships of tables in the project. Figure 10 shows the E-R diagram for MASE System.

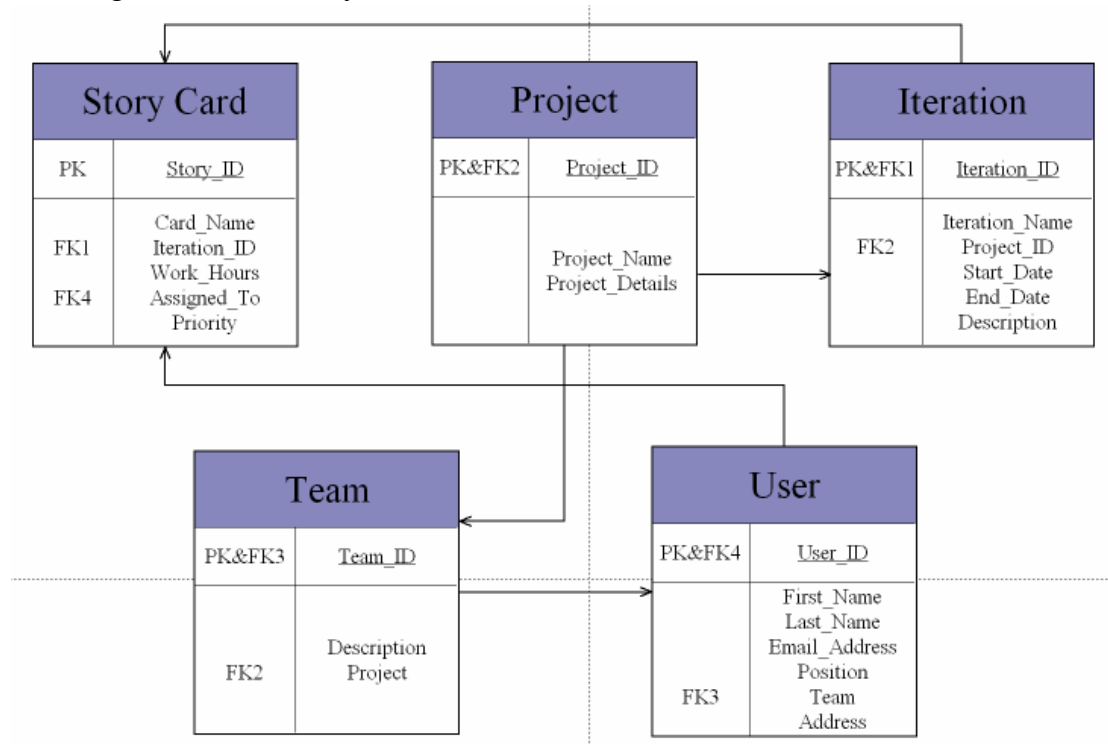


Figure 10: E-R Diagram for MASE

5.3 Inter-Agents Messages

In this section, the inter-agents messages will be generated. They are used for showing the input and output parameters for each function which are introduced below with XML format.

Figure 11 shows the input and output parameters for the functionality of Create Story Card.

Parameter	Description	Parameter	Description
<storyCard>	Story Card Related Information	<storyCard>	Display Story Card in the Iteration.
<name>String </name>			
<iteration>String</iteration>			
<workHour>Int</workHou>			
<assignTo>Int</assignTo>			
</storyCard>		</storyCard>	

Figure 11: Input (left) and output (right) of Create Story Card

Figure 12 shows the input and output parameters for the functionality of Create Iteration.

Parameter	Description	Parameter	Description
<pre> <iteration> <name>String </name> <startDate> Date </startDate > <endDate>Date</endDate> <description>String</description> </iteration> </pre>	Iteration Related Information	<pre> <iteration> <tmID>int</tmID> <projName>string</projName> <name>String </name> <startDate> Date </startDate > <endDate>Date</endDate> <description>String</description> </iteration> </pre>	Display Iteration on Whiteboard.

Figure 12: Input (left) and output (right) of Create Iteration

Figure 13 shows the input and output parameters for the functionality of Create Team.

Parameter	Description	Parameter	Description
<pre> <team> <tmID>Int</ tmID> <description>String</description> </team> </pre>	Team Related Information	<pre> <team> <tmID>Int</ tmID> <description>String</description> <tm>String</tm> . n . </team> </pre>	Display Team List with the New Created Team.

Figure 13: Input (left) and output (right) of Create Team

Figure 14 shows the input and output parameters for the functionality of Prioritize Story Card.

Parameter	Description	Parameter	Description
<pre> <prioritizeStoryCard> <storyName>String </storyName> <priority>Int</priority> </ prioritizeStoryCard > </pre>	Priority of Story Card.	<pre> <storyCards> <storyCard> <name>String </name> <priority>Int</priority> <workHour>Int</workHou> <assignTo>String</assignTo> </storyCard> . n . </prioritizeStoryCard > </pre>	Display all Story Cards with priorities on the Whiteboard.

Figure 14: Input (left) and output (right) of Prioritize Story Card

Figure 15 shows the input and output parameters for the functionality of Edit Project Details.

Parameter	Description	Parameter	Description
<projectDetails>	Project details	<projectDetails>	Display Project
<name>String </name>	Related	<name>String </name>	Details.
<details>String</details>	Information	<details>String</details>	
</projectDetails>		</projectDetails>	

Figure 15: Input (left) and output (right) of Edit Project Details

Figure 16 shows the input and output parameters for the functionality of Specifying Available Work Hours.

Parameter	Description	Parameter	Description
<workHours>	Work Hours	<workHours>	Display Work
<stName>String </stName>	Related	<stName>String </stName>	Hours
<itName>String</itName>	Information	<itName>String</itName>	According to
<workHour>Int</workHour>		<workHour>Int</workHour>	Iterations or Story
</workHours>		</workHours>	Cards.

Figure 16: Input (left) and output (right) of Specifying Available Work Hours

6. Date Dictionary:

MASE: Minimally Invasive Long-term Organizational Support for Agile Software Engineering

AMA: Account Management Agent

WMA: WikiPage Management Agent

PPA: Project Planning Agent

TMA: Team Management Agent

Info: Information

Finaliz Iter: Finalize Iteration

TM.: Team Member

DB: Database

Crea: Create

SC: Story Card

SAWH: Specifying Available Work Hours

Prior: Prioritize

WBD: Whiteboard

Iter: Iteration

Proj: Project

7. Conclusion:

The project is build basically on the approach of Prometheus. I have found this method of building multi-agent system very simple and straightforward, thus easy to implement. However I omitted the process of building descriptors for both user case and each agent. Because I think, by using the use case, the interaction diagrams and the class diagram, the system can be clearly specified, and the purpose of the project, which is trying to build a project in a multi-agent perspective, is achieved through the concise approach.

8. References:

- [1] Frank Maurer: *Supporting Distributed Extreme Programming*, Proceedings Agile Universe/ XP Universe 2002, Springer, 2002
<http://ebe.cpsc.ucalgary.ca/ebe/attach?page=Root.PublicationList%2FMaurer2002.pdf>
- [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),
<http://www.cs.rmit.edu.au/agents/Papers/aose02.pdf>
- [3] Frank Maurer, MASE website,
<http://ebe.cpsc.ucalgary.ca/ebe/Wiki.jsp?page=Root.MASE>
- [4] UML basics: The class diagram:
<http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html>