



**SENG 609.22 Agent s Based Software Engineering**

# On-line Examination Assistant System (OEAS)

--Based on Agent-Oriented Methodology

Instructor: Dr. B. H. Far

1. Introduction	2
2. System Overall Specification	2
2.1. SYSTEM GENERAL DESCRIPTION .....	2
2.2. REQUIREMENTS.....	2
2.3. ASSUMPTIONS .....	4
2.4. WISH LISTS (NOT IMPLEMENTATION) .....	4
3. System Design Document	4
3.1. AGENT-BASED SYSTEM ANALYSIS.....	4
3.2. ROLE SCHEMA .....	5
3.3. PROTOCOL DEFINITION .....	9
3.3.1. DEFINITION OF PROTOCOL ASSOCIATED WITH THE ROLE MARKER .....	9
3.3.2. DEFINITION OF PROTOCOL ASSOCIATED WITH THE ROLE EXAMINEE.....	10
3.3.3. DEFINITION OF PROTOCOL ASSOCIATED WITH THE ROLE COMMITTEE .....	11
3.3.4. DEFINITION OF PROTOCOL ASSOCIATED WITH THE ROLE TIMER .....	11
3.3.5. DEFINITION OF PROTOCOL ASSOCIATED WITH THE ROLE REGISTRAR.....	12
3.4. SYSTEM ARCHITECTURE .....	12
3.4.1. AGENT DESCRIPTION .....	13
3.4.2. AGENT INTERNAL ARCHITECTURE.....	14
3.4.3. TECHNOLOGY OVERVIEW .....	17
3.4.3.1. COMMUNICATION PROTOCOL: SOAP .....	17
4. Detailed Design Document	19
4.1. USE CASE.....	19
4.1.1. REGISTRAR AGENT .....	19
4.1.2. COMMITTEE AGENT .....	20
4.1.3. MARKER AGENT .....	20
4.1.4. TIMER AGENT .....	21
4.2. DETAILED DESIGN .....	22
4.2.1. OVERALL COMMUNICATION SEQUENCE CHART .....	22
4.2.1.1 DETAILED DESIGN OF MARKER AGENT .....	24
4.2.1.2 REGISTRAR AGENT .....	26
4.2.1.3 COMMITTEE AGENT .....	30
4.2.1.5 EXAMINEE AGENT .....	34
5. Data Specification	35
5.1. DATA AND/OR KNOWLEDGE SHARING SPECIFICATION .....	35
5.2. INFORMATION TABLE.....	36
5.2.1. EXAMINEEINFO TABLE.....	36
5.2.2. QUESTIONBANK TABLE.....	36
5.2.3. EXAMRESULT TABLE .....	37
5.2.4. GRADE TABLE.....	37
6. Inter-Agent Messages	37
7. Conclusion	42
8. References	43

## **1. Introduction**

On-line Examination Assistant System (OEAS) is a multi-agent system that allows registering for exam, authentication, making answer sheets and reporting.

This document is the design documentation for “On-line Exam Assistant System”. It specifies the overall system specification, proposes the design issue (from high level architecture to details of each components and implementation details). The design and implementation are based on the concepts of Agent-based Software Engineering. Section 2 specifies the system, section 3 proposes the system design document, and section 4 is about detailed agent design document, section 5 is data specification, in section 6 we introduce inter-agent messages, finally in section 7, we draw a conclusion.

## **2. System Overall Specification**

### **2.1. System General Description**

The “Online Examination Assistant System” is a powerful agent-based system to provide exams to measure personal ability to perform specific real-world job functions and provide a valid and reliable measure of technical proficiency and expertise. In this “Online Examination Assistant System”, innovative question types present a more realistic visual representation of the scope the examinee are being tested on so that examinee’s exam experience more consistently reflects the types of tasks the examinee might perform on the real world. Self-adapted is one important feature in this system. The system can adjust the types of questions and the difficulty levels of questions according to different situations. The system also can track the history of the examinee to provide the most appropriate questions sets for him. This system is designed to provide the access for any computer on the Internet. So the examinee can take the examination very conveniently.

### **2.2. Requirements**

The goal of “Online Examination Assistant System” is to provide the real-time customized examination system. In general, it should be able to provide the questions based on the preference of examinee, evaluate the question answers and make the report for the examination. The system requires the following features described as the following:

- **Examinee Registration:** When examinee wants to use this system, she/he has to register into the system by providing her/his personal information, including full name, position, expertise, and mailing address and email address. Then the system will record the information into the central database and generate user ID and password to email them to the email

address the examinee provided. Then examinee will be able to take the examination and check her/his examination history.

- **Authentication:** Before the examinee access the examination history or take the on-line examination, she/he has to provide the user ID and password to the system. After system authorizes the requests by validating the Examinee ID and password, the examinee can take the online examination or review his/her previous examination results. Otherwise, the system will give the unauthorized information to the examinee that provides the invalidated identification information.
  
- **Examination preference collection:** The examinee shall be able to choose the scope of examination. And the system could give the initial questions sets based on the choice of the examinee, the expertise of the examinee and the history of examination of the examinee.
  
- **Questions generation:** After the system provided the initial questions sets, the system shall be able to adjust the number of questions, type of questions and the difficulty level of the questions according to the answers the examinee has already made during the examination. For example, for a certain type questions, if the examinee got all the answers rights for the easy level questions, then the system could ask more advanced questions in this area to evaluate the examinee's expertise. If the examinee got most of the answers wrong in the same area, the system will terminate continuing to ask more advanced questions to save the time.
  
- **Examination administration:** The system shall be able to end the examination within the certain time. Or the examination shall be ended by the time the examinee submits the all the answers to the system. The administration sub-system shall be responsible for delivering all the answers to the marking sub-system when the examination ended.
  
- **Real time report:** The system shall give the results of the examination right after the end of the examination. The result includes the final grade, the percentage of right answers in each sub area. The system also saves the results to the central database as the future reference information for the examinee. The examinee can review the history results any time when he accesses the system.
  
- **Question Bank Maintains:** The system can adjust the questions properties in the question bank according to the examination results and the usage of the questions. For instance, if the ration of right answer to wrong answers keeps high, the difficulty of this question shall be decreased. And the longer the question has been used, the less probability the questions will be used.

## 2.3. Assumptions

- There are already predefined facilitator used in the system to provide register and discovery service on the Internet. This facilitator use UDDI as the standard to describing services, discovering businesses, and integrating business services.
- Facilitator has known how to locate registrar agent and timer agent. It is a universal knowledge to other agents that need the service from these two agents.
- How to select the optimized service providers is determined by agent, which requests the services.

## 2.4. Wish Lists (Not implementation)

- The facilitator will be customized to fit into this system rather than using existing facilitator on the Internet. Therefore, the facilitator could choose the best service provider to give to request agent rather than returned the service providers list to the request agent.
- The timer agent will be integrated into the examinee agent to reduce the heavy network traffic.
- The report function will be separated from the marker agent. Another dedicated agent “Report Agent” will be added to the system to be responsible for the report function.

## 3. System Design Document

### 3.1. Agent-based System Analysis

When the examinees want to take the on line examination, the examinee role is the key role. The examinee role sends the request to other roles. The scenarios are as follows:

If the examinees are first time to use this system, then they need register in the system, provide their information, then they can begin to select the examination questions. The second scenario is if the examinees had used this system before, they just need login. After getting authentication, they can select the examination requirements. So the first contact role is registrar role. The function of this role is to provide user ID and password to the examinees who first use the system, or to valid user information, like user ID, password.

After the examinees login, they can choose to take the examination or just get last time examination results. If they want to take the examination, then the examinees contact with the committee role, and inform the committee role which kind

examination question they want to take. At this moment, the timer begins to timing. The question will be sent to the examinee one by one. Only after submitting one answer, the examinees can get next question. At this stage, the examinees need notice the marker role. As soon as the marker gets the answer from the examinees, the marker marks it, and sends the result to the committee role. According to the result, the committee selects the next question's level. If the answer is right, then the committee send more difficult question. If the answer is wrong, then the committee send easier question to the examinees. During this period, the timer keeps indicating how much time left for this examination to the examinees.

The second scenario, examinees only want to get last time examination results, the examinees just need contact with the marker role.

From the timer role, we can know if the examinee finished the examination on time. When time becomes zero if the examinee haven't submitted the last question answer, which means the examinees haven't finished the examination on time.

As soon as the examinee finished the examination, they can get the results report, such as difficulty levels, category, percentage of correct answering, and how much time to finish this examination from the committee role.

So far, we have concluded that the following five roles are needed in our system:

- Examinee role: to provide information, take the examination
- Registrar role: to process all user information
- Committee role: to generate question
- Marker role: to mark an exam, and generate a report for the exam
- Timer role: to timing

Basically, the above five roles can make the system operate properly.

### 3.2.Role Schema

Role schema: marker			
Description:			
Receive answer from the examinee and mark the answer, then save the results and report the results to the examinee.			
Protocols and Activities:			
MarkingRequest, Marking, ProduceResults, SaveResults, NotifyCommittee, ReportResults			
Permissions:			
Reads	<b>supplied</b>	ExamineeId	// examinees' ID
	<b>supplied</b>	QuestionDetails	//question information
	<b>supplied</b>	Answer	// examinees' answer
		StandardAnswer	// right answer

Generates	ExaminationID Results	//Examination Identification // the examination results
Responsibilities:		
Liveness:		
	Marker=	((MarkingSingle) w. ProduceResults. SaveResults.ReportResults)   ReportResults) w
	MarkingSingle =	MarkingRequest.Marking.NotifyCommittee
Safety:		
	ResultsIsAvailable	(examineeid)

**Figure 3.2.1 Role Maker**

The Maker role is responsible for

- Get the marking request from Examinee.
- Process the answers for the examination.
- Notify the committee answer status of each question.
- Generate the report according to the answers based on the schema.
- Save the results to the central database for future reference.
- Retrieve the results from the central database.
- Response the examinee with the reports.

Role schema: Committee		
Description:		
	Generate questions to meet examinee's requirements.	
Protocols and Activities:		
	QuestionRequest, AnswerAnalysis, GenerateQuestion, ReturnQuestion	
Permissions:		
Reads	<b>supplied</b> ExamineeId	// examinees' ID
	<b>supplied</b> QuestionRequirement	//details of the examinee's requirement: i.e. question area.
	AnswerStatus	// Boolean for the status of answer for single question.
Generates	QuestionBank	// question bank
	ExaminationID	//Examination Identification
	ExaminationQuestion	// a paper consists of questions that meet the examinee's requirements and time limit of the examination
Responsibilities:		
Liveness:		
	Committee=	(QuestionRequest. GenerateQuestion   (AnswerAnalysis. GenerateQuestion).ReturnQuestion) w
Safety:		
	QuestionsIsAvailable	(QuestionRequirement)

**Figure 3.2.2 Role Committee**

The Committee role is responsible for

- Generate the question according to the examinee's requirements.
- Analyze the submitted answer.
- Generate the consequent questions according to the analyzed results.
- Retrieve the questions from the central database.
- Response the examinee with the questions.

Role schema: examinee	
Description:	Description: an individual requiring an on-line examination.
Protocols and Activities:	Registration, Authentication, GiveExaminationRequirement, StartExamination, TakeExamination, SubmitAnswer, EndExamination, GetResult
Permissions:	
Reads	<b>supplied</b> ExaminationQuestions //questions that are generated by committee ExaminationID //Examination Identification ExaminationResults // Examination results TimeLimitaionStatus // Boolean for exceeding time limitation
Generates	Answer // The examinee's answer
Responsibilities:	
Liveness:	Examinee= (Registration) +. Authentication. (GiveExaminationRequirement. StartExamination. TakeExamination. SubmitAnswer. EndExamination.GetResult)  GetResult
Safety:	TimeLimitationStatus = false

**Figure 3.2.3 Role Examinee**

The Examinee role is responsible for

- Provide registration information.
- Give examination requirement.
- Generate answers of the examination questions.
- Submit the answer to marker.
- End the examination.

Role schema: timer	
Description: Timing the examination duration.	
Protocols and Activities: StartTimerRequest, timing, timeupnotification, stoptiming	
Permissions:	
Reads	supplied TimeLimit // The time limit of the exam SubmitStatus // Boolean for the submit status
Change	LeftTime // The time left for the examination
Generates	TimeLimitaionStatus // Boolean for exceeding time limitation
Responsibilities:	
Liveness:	Timer =
	(starttimerrequest.timing.timeupnotification starttimerrequest.timing.stoptiming)+
Safety:	
	LeftTime>=0 SubmitStatus=false

**Figure 3.2.4 Role Timer**

The Timer is responsible for

- Change the left time of the examination.
- Generate and notify the examinee the TimeLimitationStatus.

Role schema: Registrar	
Description: Provide registration service and authentication.	
Protocols and Activities: AwaitRegistration, ProcessRegistration, AwaitAuthentication, ProcessAuthentication	
Permissions:	
Reads	Supplied ExamineeDetails // information of the examinee
Generates	ExamineeId // The examinee's identification Password // password of the examinee's identification AuthenticationResult // Boolean for authenticationRequest
Responsibilities:	
Liveness:	Registrar= ((AwaitRegistration. ProcessRegistration) (AwaitAuthentication. ProcessAuthentication))ω
Safety:	True

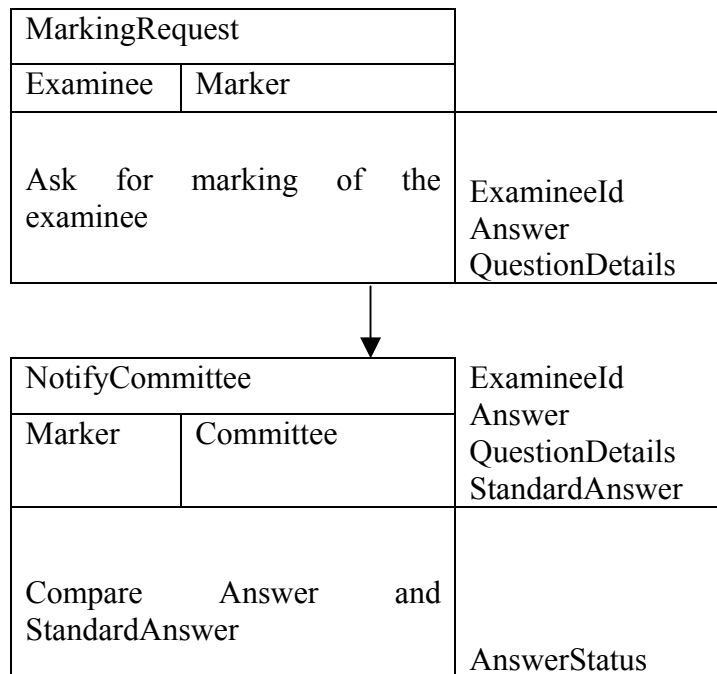
**Figure 3.2.5 Role Registrar**

The Registrar role is responsible for

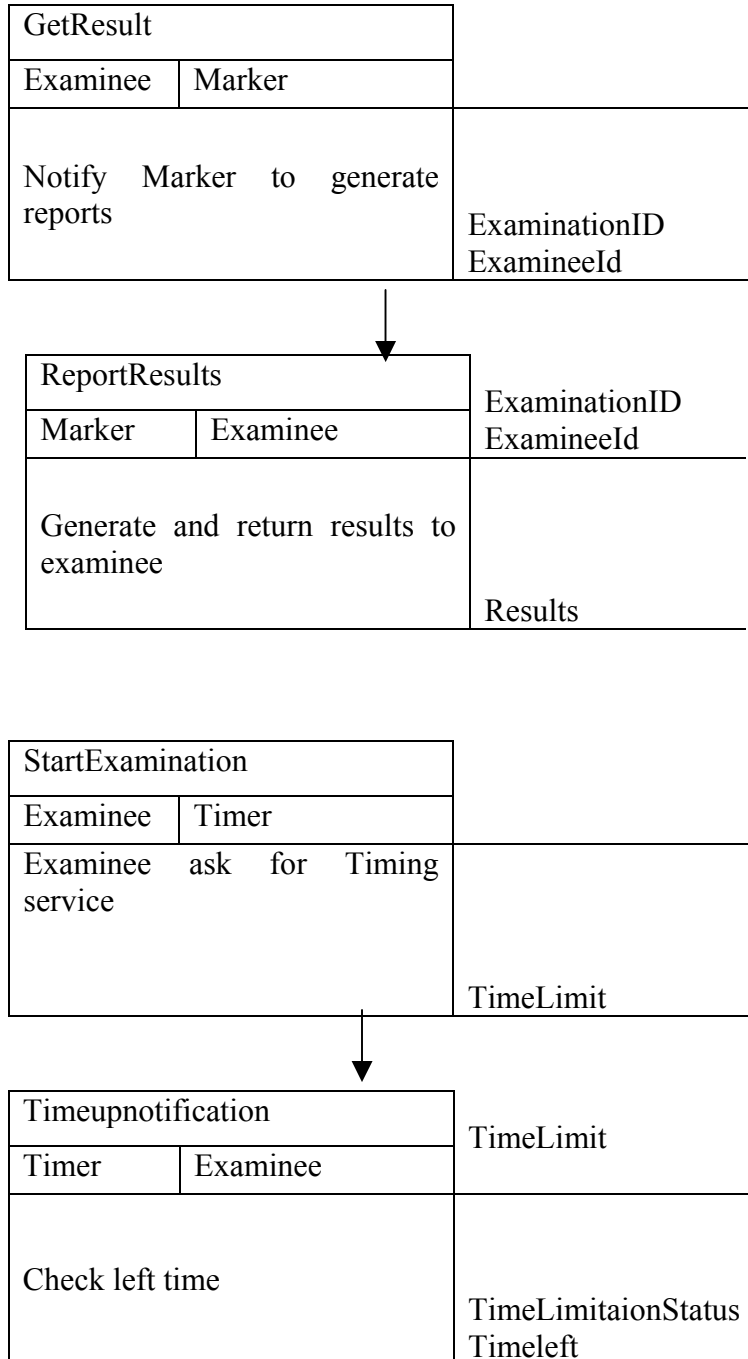
- Process the information provided by the examinee.
- Generate examinee id and password for the examinee.
- Validate the examinee's identification.

### 3.3. Protocol Definition

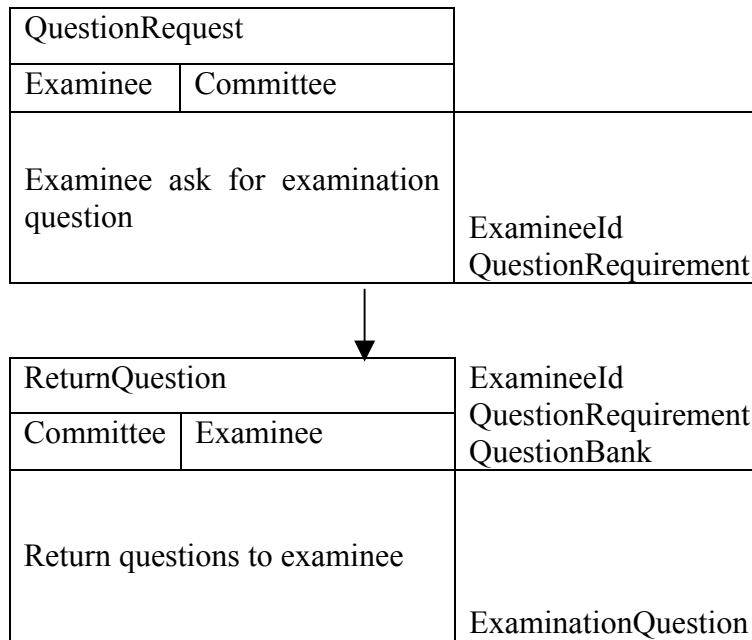
#### 3.3.1. Definition of Protocol Associated with the Role Marker



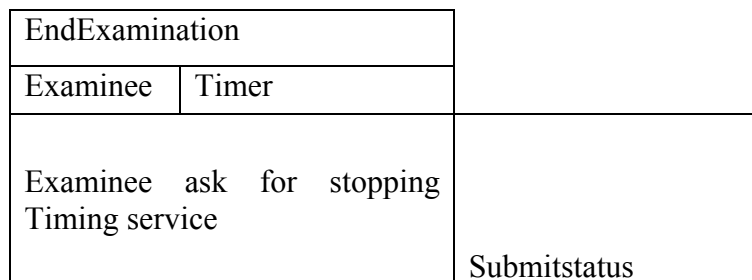
### 3.3.2. Definition of Protocol Associated with the Role Examinee



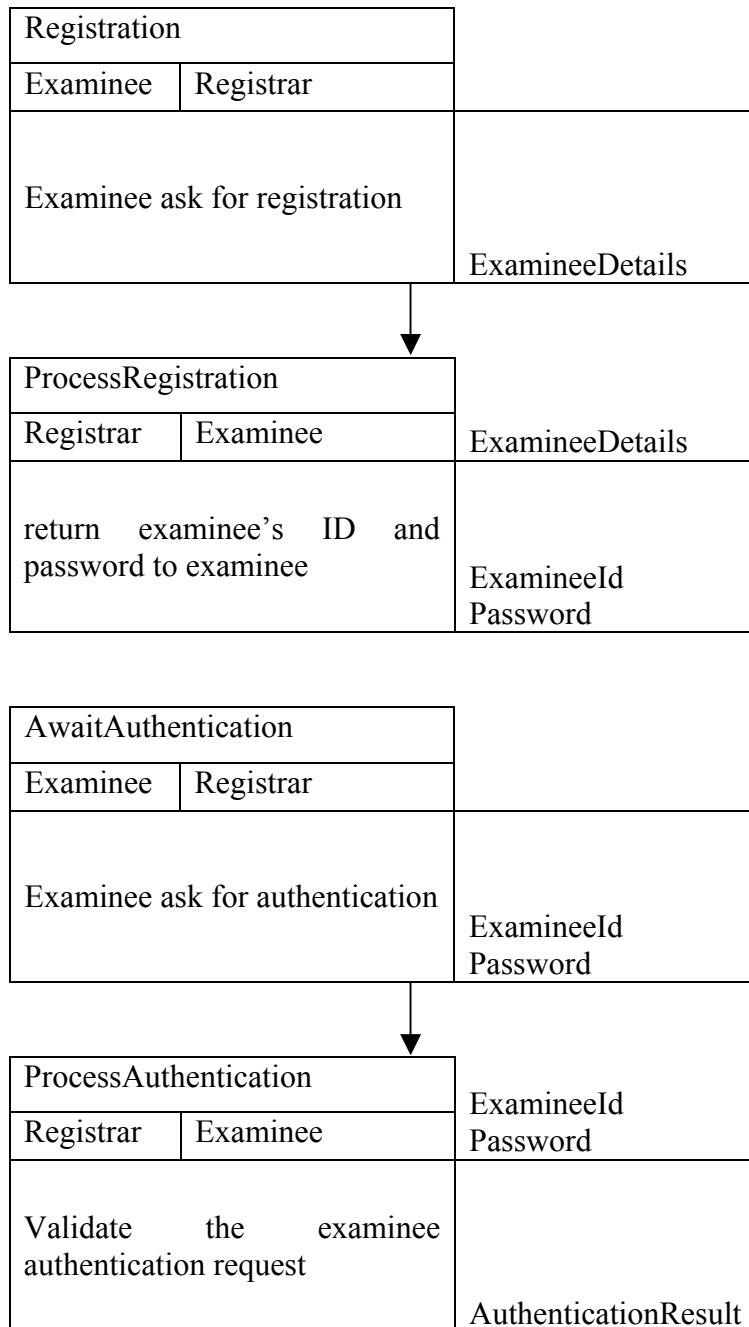
### 3.3.3. Definition of Protocol Associated with the Role Committee



### 3.3.4. Definition of Protocol Associated with the Role Timer



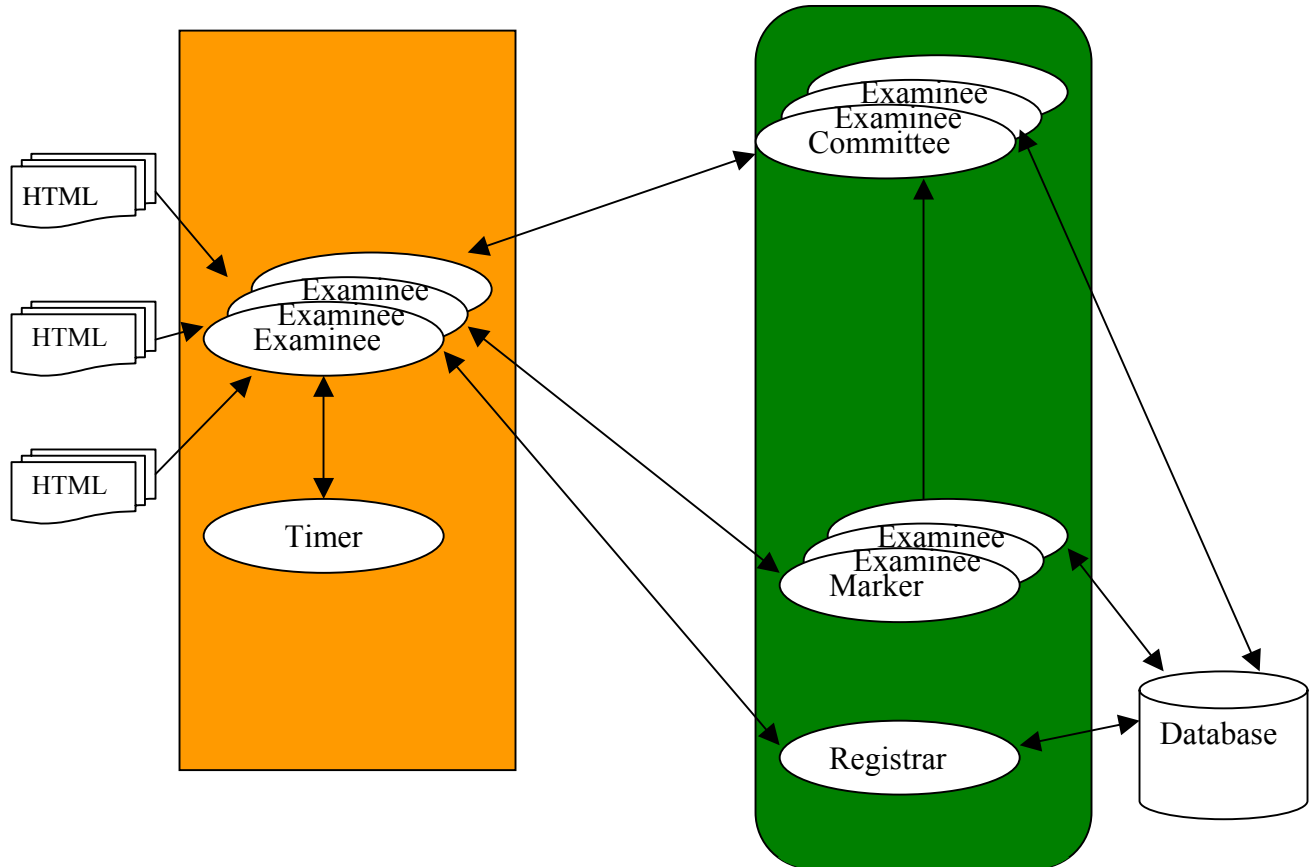
### 3.3.5. Definition of Protocol Associated with the Role Registrar



### 3.4. System Architecture

In this system, HTML pages via the web browser represent the entire client interfaces. User could submit their request to Examinee agents, such as taking an examination or enquiry the history reports. In this system, the examinee agents,

markers and committee agents are multiply instantiated to balance the system loading. Any users have to register first, and then they could allow taking the examination after they got their user account from registrar.



**Figure3. 4.1 Agent System Architecture**

### 3.4.1. Agent Description

We divided the system into five agents, Examinee Agent, Marker agent, Timer Agent, Register Agent and Committee Agent. Their function as follows;

- Registrar agent

It provides the registrar and authentication service. It receives the request sent by the examinee through the Internet browser. The registrar agent waits for the request through the listener. After getting request, and relative information, the registrar agent either displays the username and password on the examinee's Internet browser or authenticates the examinee to begin examination.

- Examinee Agent

In our architecture the examinee agent handles the correspondence with all other agents. The Examinee Agent is responsible of providing information to the

Registrar Agent. Then the Examinee Agent can request the Committee Agent to select to examination questions package. Moreover the Examinee Agent communicates with the Marker Agent to mark the answer. Finally the Examinee Agent triggers the Timer Agent to begin to count time.

- **Marker Agent**

The Marker Agent communicates with the Examinee Agent. Once the examinee has specified his/her examination questions, the Marker Agent begins to mark the question answer of the examinee. After that the Marker Agent begins to communicate with the Committee Agent to change the difficult level according to the examinee's answer and provides report.

- **Committee Agent**

It takes the inquiry request from the Examinee Agent and communicates with the Marker Agent to provide the questions and report.

- **Timer agent**

At the name suggests this agent take the responsibility of counting the time. The Timer Agent only communicates with the Examinee Agent.

### **3.4.2. Agent Internal Architecture**

Common Components in the internal agent architecture:

- **Listener:**  
The Listener Component listens to the ports for any incoming requests from the other agents in the system
- **XML Parse:**  
Since the XML is used in this system for the communication format, The XML Parse parses the input parameters embodied in the XML format messages.
- **Core Functions:**  
By responding the input parameters abstracted from the XML messages, core functions make corresponding actions to generate the output contents.
- **Discovery:**  
The Discovery component provides the ability to the agent to find the appropriate agent provide the service it needs.

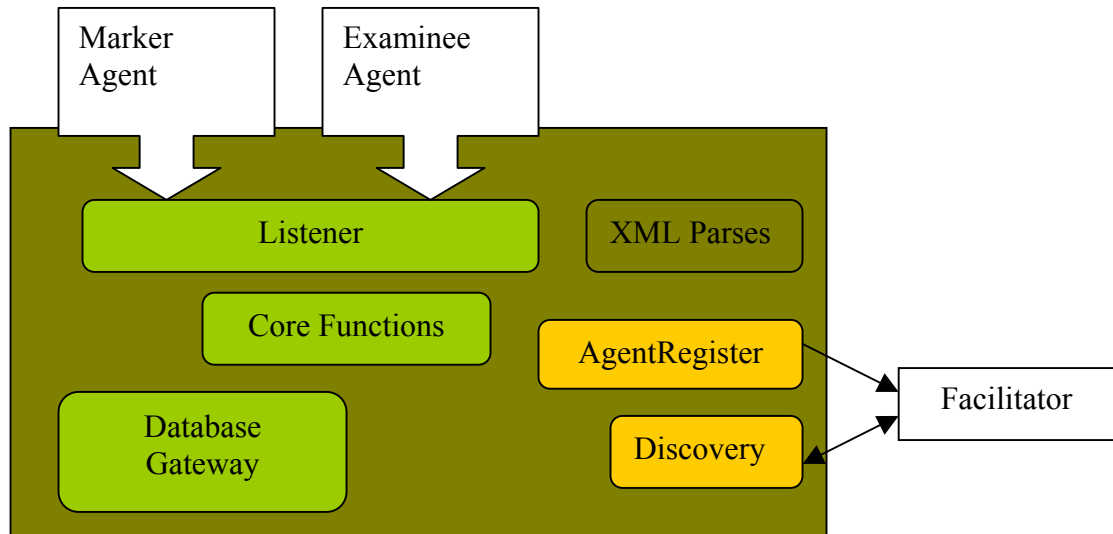
For the agents, which will have multiple instances in the system, such as Examinee, Marker and Committee agent, there have two more components in the agent internal architecture:

- **AgentRegister:**  
This component provides the registry service to facilitator in order to register the agent id and agent service to the agent service directory.

For the agents, which need sharing the knowledge by access central database, there has a database gateway component in the internal architecture:

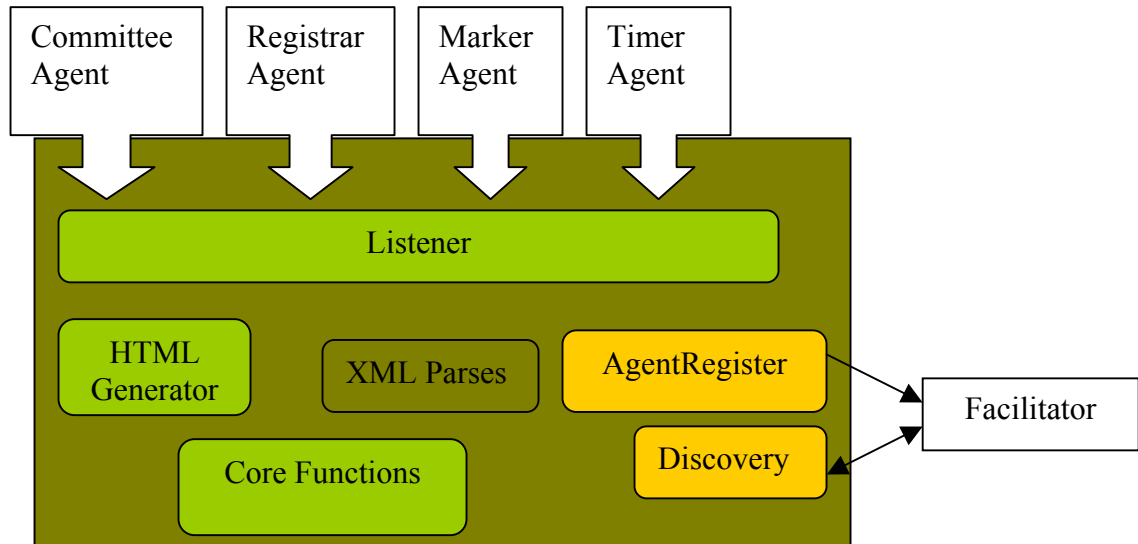
- Database gateway:**  
 This component provides the database access function for the agents.

Figure 3.4.2.1 shows the internal architecture for committee agent.



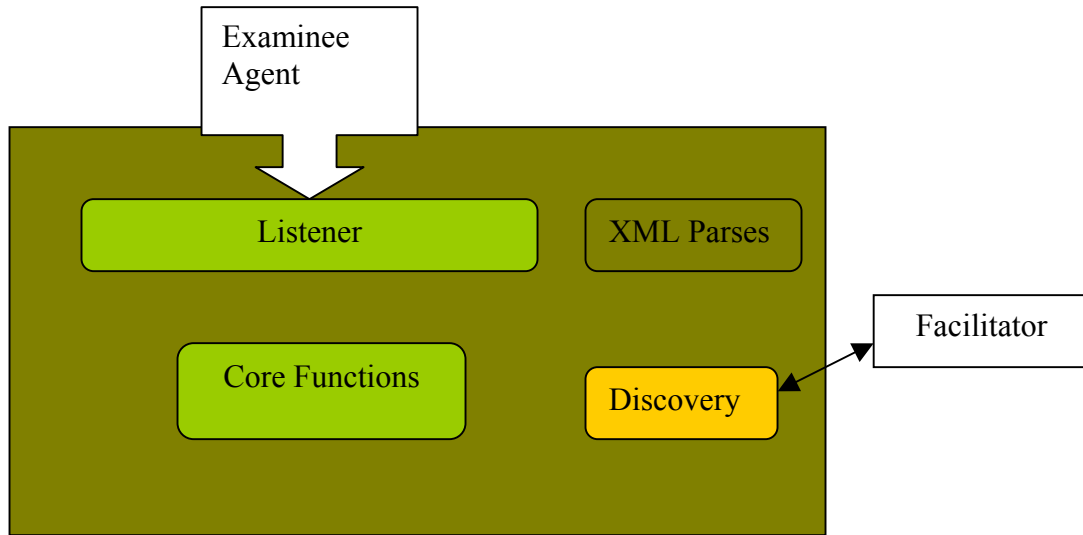
**Figure 3.4.2.1 Internal Architecture for Committee Agent**

Figure 3.4.2.2 shows the internal architecture for examinee agent.



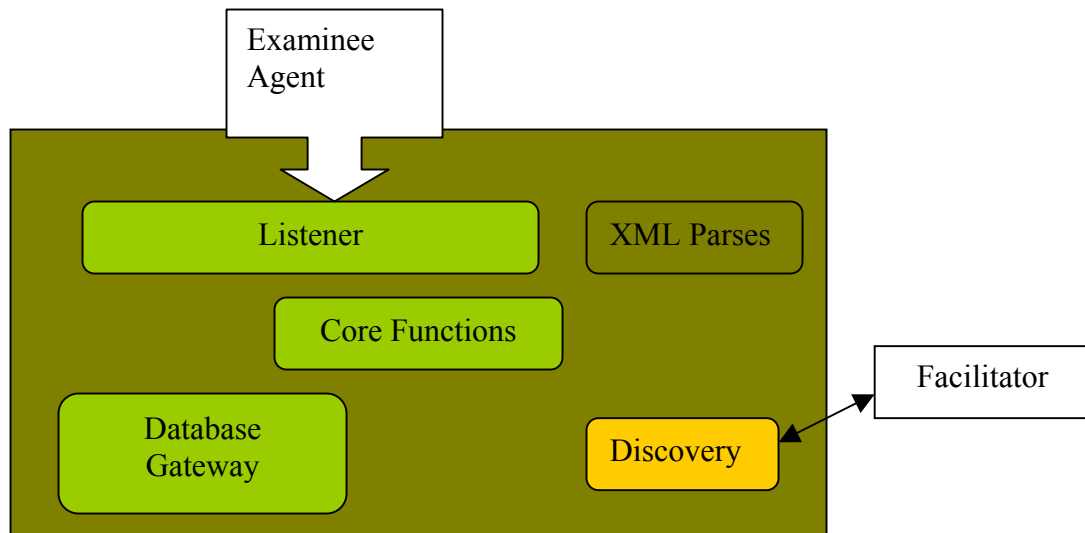
**Figure 3.4.2.2 Internal Architecture for Examinee Agent**

Figure 3.4.2.3 shows the internal architecture for timer agent.



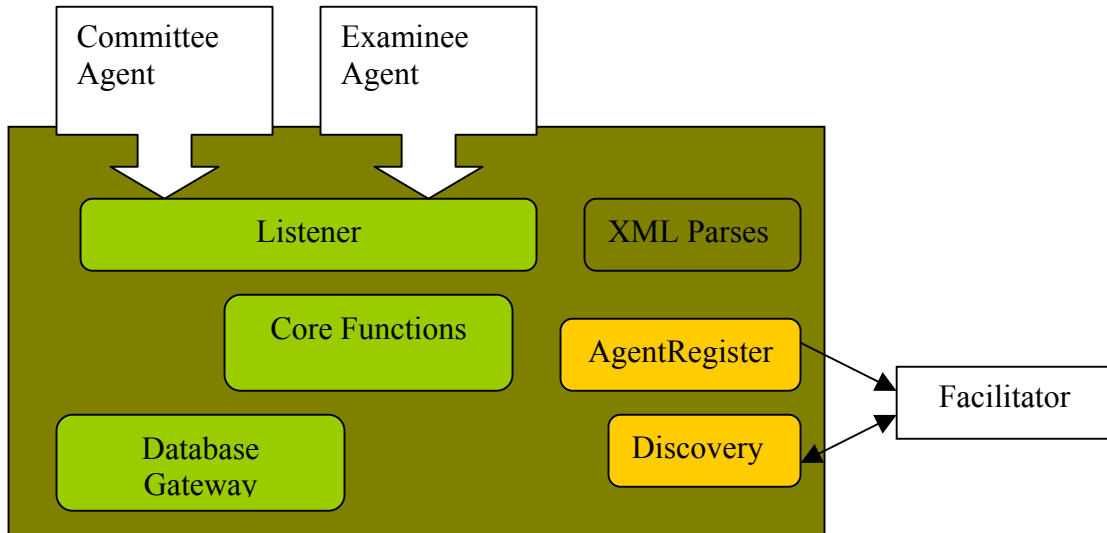
**Figure 3.4.2.2 Internal Architecture for Timer Agent**

Figure 3.4.2.4 shows internal architecture for registrar agent.



**Figure 3.4.2.4 Internal Architecture for Registrar Agent**

Figure 3.4.2.5 shows the internal architecture for marker agent.



**Figure 3.4.2.5 Internal Architecture for Marker Agent**

### 3.4.3. Technology Overview

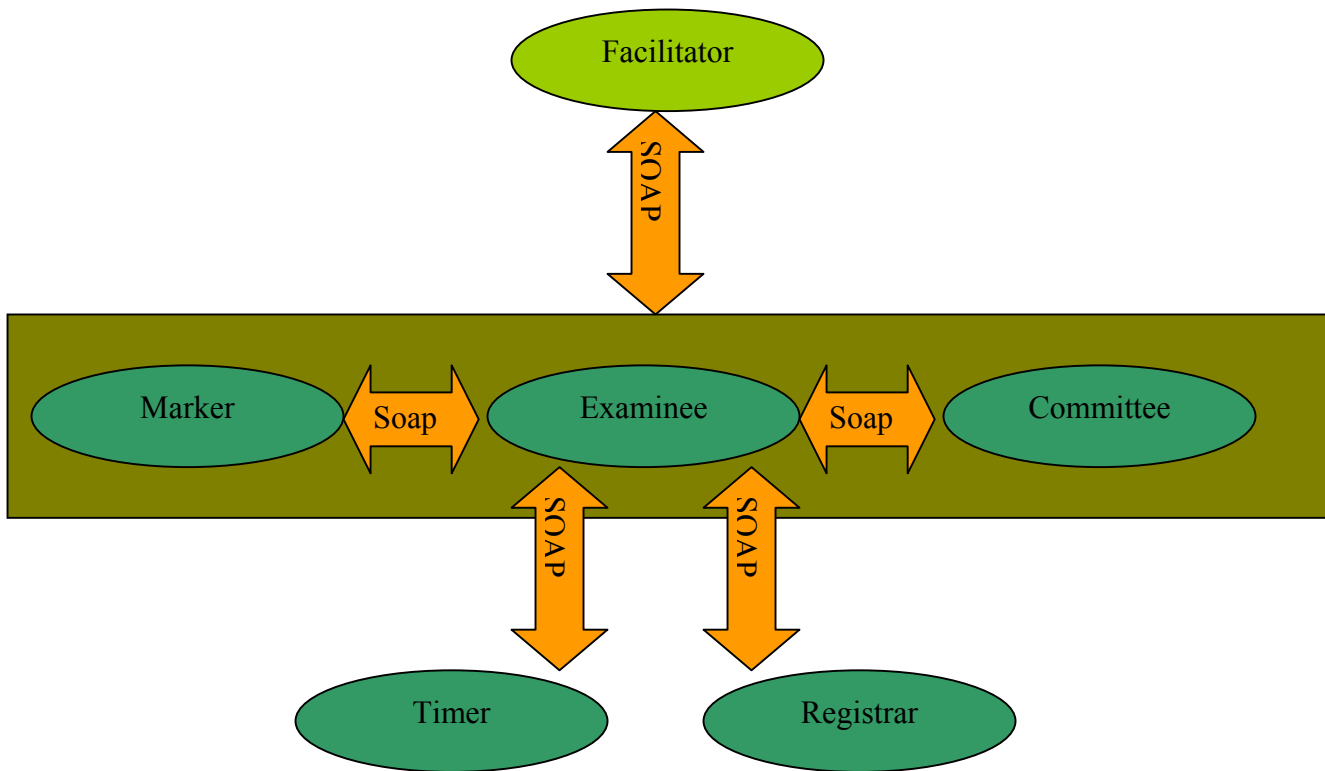
Universal Description, Discovery, and Integration (UDDI) with Web Service Description Language (WSDL) will be used in our system as the service discovery approaches. As what we mentioned in the assumptions, the facilitator is predefined on the Internet. However, the disadvantage of adopting the existed facilitator on the Internet is that it could not implement central control to provide the best service provider. For example, the committee agents provide the service to select the questions. However, we have multiple identical committee agents provide the same service. By now, since we use the standard facilitator, it only could provide the list who are providing this kind of service rather than give the specific agent based on not only services but also the costs to provide this service (such as the current working loading, network traffic). Therefore, we consider implement the customized facilitator in the next delivery system to provide the better discovery services. All the information is stored in a central database system. The committee, marker and registrar have the rights to share the knowledge in the central database.

#### 3.4.3.1. Communication Protocol: SOAP

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how

to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols.

However, we will use SOAP as a high-level communication protocol in our system to encode the HTTP header and an XML, which we will use in our system as the message format. The purpose of using in our system SOAP is to enable rich and automated agents based on a shared and open Web infrastructure. Since we consider we will implement our system via the Internet. And SOAP and XML could be implement by both Microsoft .NET Framework and J2EE, which are more popularity in terms of existing implementation environment. So we decided use SOAP as our communication protocol for this agent-based online examination system. All the inter agent communication will based on the SOAP and use XML as the message format in our system.

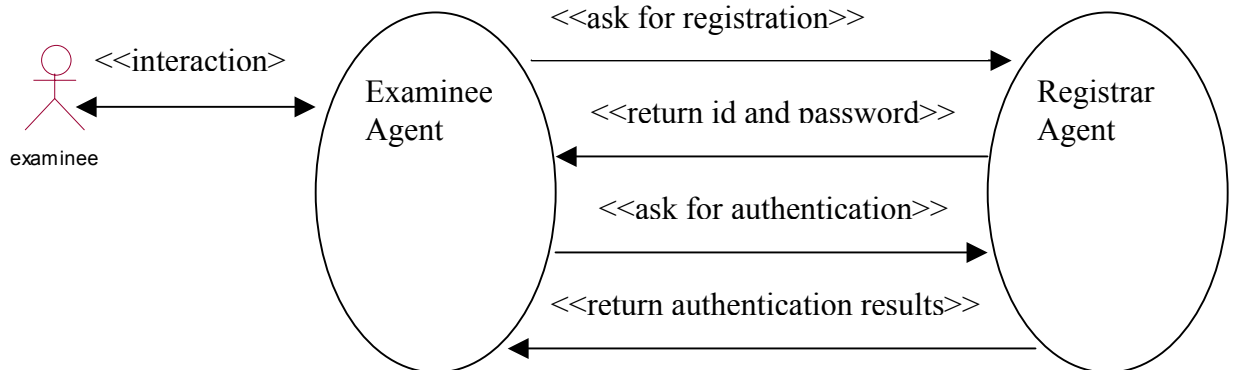


**Figure3.4.3.1.1Communication protocol: SOAP**

## 4. Detailed Design Document

### 4.1. Use Case

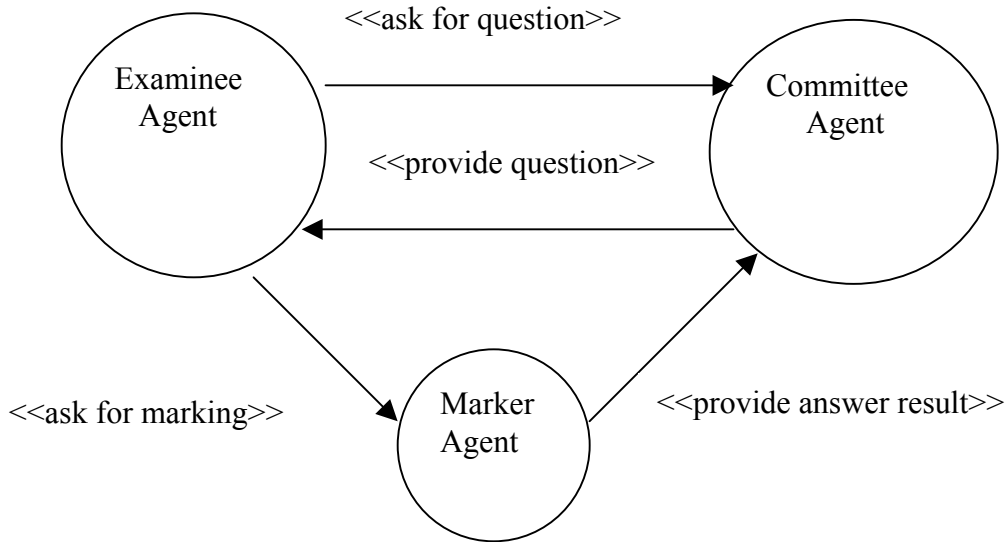
#### 4.1.1. Registrar Agent



#### Use case definition: Registrar Agent

Brief Description	The examinee agent uses this use case to request register and authentication service.
Preconditions:	Before being provided authentication service, examinee should register first.
Post conditions:	After authentication, examinee can choose taking exam or getting report.
<b>Process steps</b>	
1	Examinee makes request for registration
2	Registrar agent return id and password to examinee
3	Examinee makes request for authentication
4	Registrar agent return authentication results

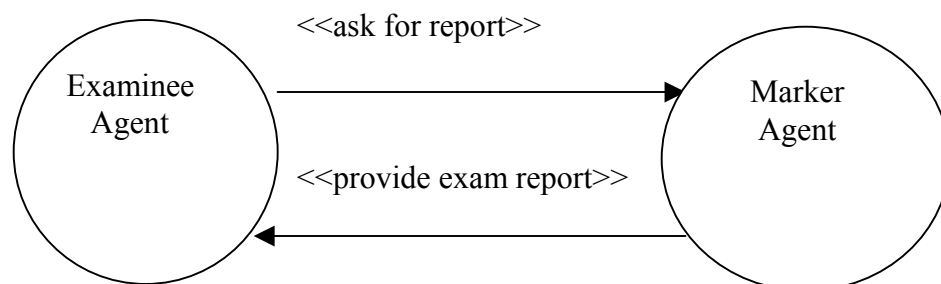
### 4.1.2. Committee Agent



#### Use case definition: Committee Agent

Brief Description	The examinee agent uses this use case to request question.
Preconditions:	Before being provided question, examinee should pass the authentication.
Post conditions:	After getting questions, examinee can take the exam.
<b>Process steps</b>	
1	Examinee makes request for first question.
2	Committee agent provide first question.
3	Examinee makes request for marking.
4	Marker agent provides answer result to Committee agent.
5	Committee agent adjust next question's difficulty level according to the previous question's result
6	Committee agent provides following question to Examinee.

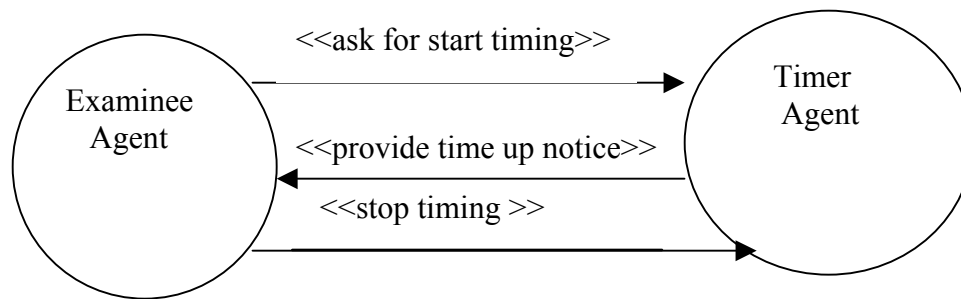
### 4.1.3. Marker Agent



**Use case definition: Marker Agent**

Brief Description	The examinee agent uses this use case to request exam report.
Preconditions:	Before being provided question, examinee should have finished the exam.
Post conditions:	N/A
<b>Process steps</b>	
1	Examinee makes request for exam report.
2	Marker agent generates the exam report for examinee.
3	Marker agent provides exam report to the examinee.

**4.1.4. Timer Agent**



**Use case definition: Timer Agent**

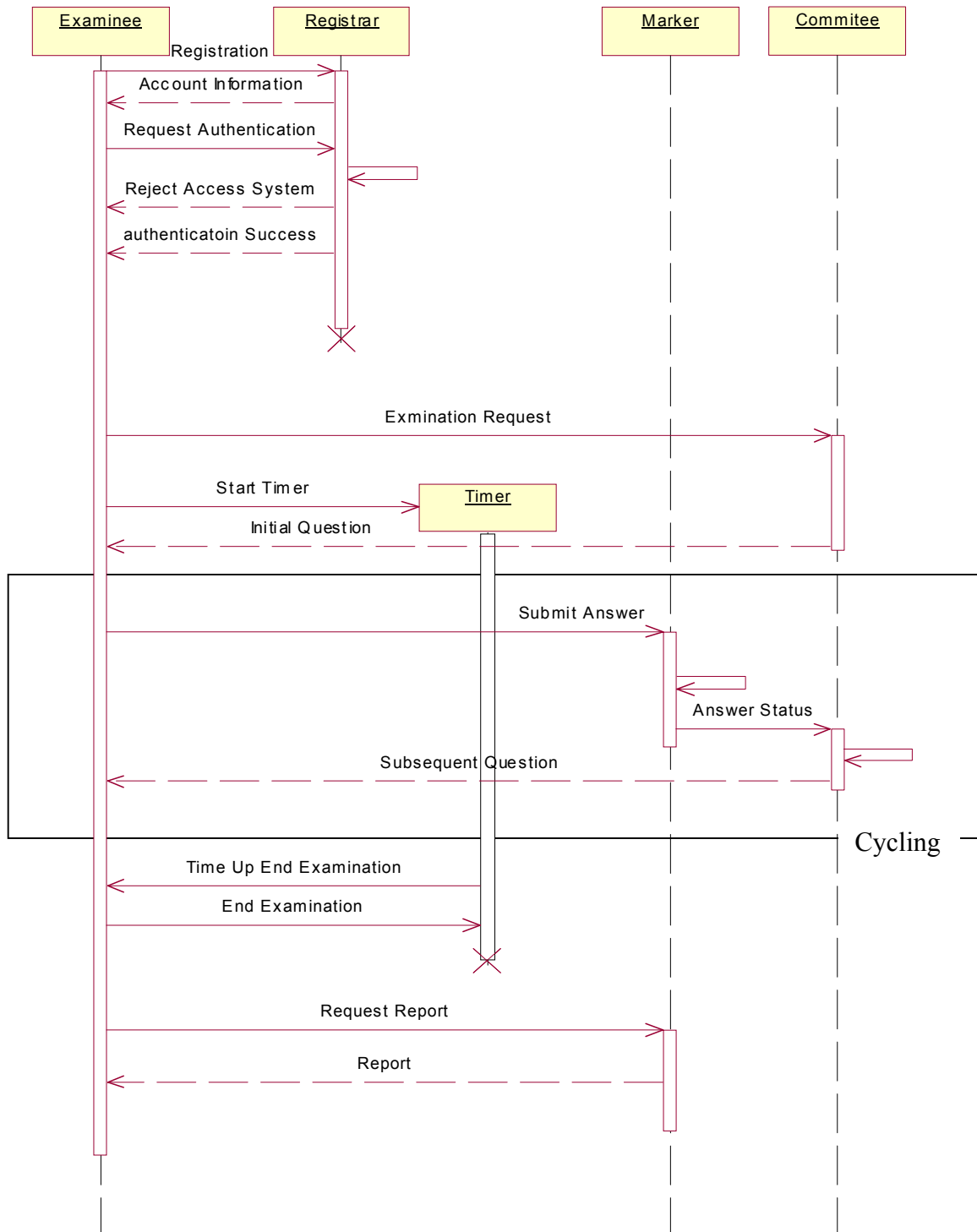
Brief Description	The examinee agent uses this use case to request timing service.
Preconditions:	Before being provided question, examinee should have received the first question.
Post conditions:	N/A
<b>Process steps</b>	
1	Examinee makes request for stat timing service.
2	Timer agent provides the examinee time up notice.
2	Examinee ask timer agent to stop timing.

## 4.2. Detailed Design

### 4.2.1. Overall Communication Sequence Chart

Figure 4.2.1.1 is Overall Sequence Chart represented by UML format. There have some features in this system:

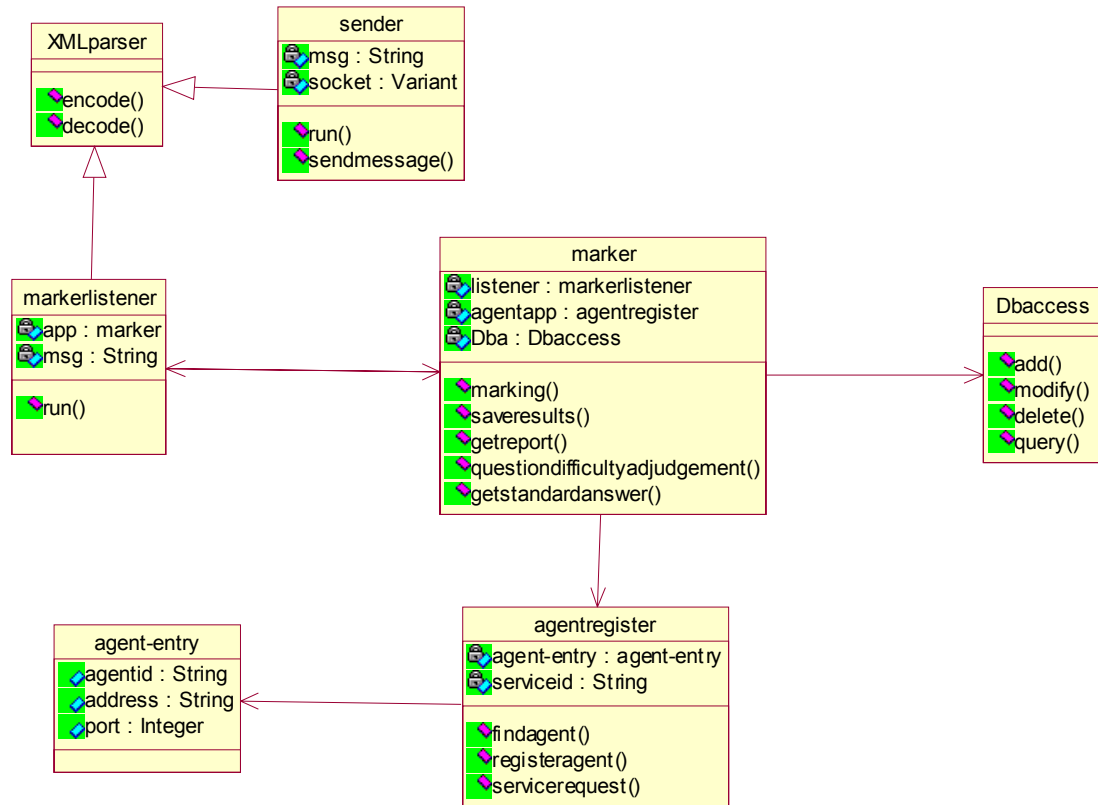
- Register Agent Life Cycle finished once the Examinee get the authentication failed or successful.
- The initial question is obtained by the communication that happened directly between examinee and committee. However, the subsequent questions are given by involving three agents, examinee, marker and committee.
- The life cycle of marker and committee finished by giving one questions since the examinee may find another pair marker and committee in the next question request.



**Figure 4.2.1.1 Overall Communication Sequence Chart**

### 4.2.1.1 Detailed design of Marker Agent

Figure 4.2.1.1.1 shows the class diagram of the Marker Agent.



**Figure 4.2.1.1.1 Marker Agent Class Diagram**

- Class marker: All the core functionality of the Marker agent is performed by the marker class. Its attributes include: listener, agentapp and Db. Its method include: marking, saveresults, getreport, questiondifficultyadjudgement and getstandardanswer.
- Class Dbaccess: Dbaccess class is used to interact with the OEAS database system. It is associated with the marker class as an attribute of it. Its method include: add, modify, delete, and query.
- Class XMLparser: XMLparser class is used to parse the XML message. Its method include: encode, and decode.
- Class markerlistener: markerlistener class is used to listen at a certain port as a thread process and notify the marker class to perform corresponding action according to the incoming message. It can parse the XML message by inherit the XMLparser class. It is associated with the marker class as an attribute of it. Its attributes include: app, and msg. Its method includes: run.
- Class sender: sender class is used to send out messages. It can also parse the XML message by inherit the XMLparser class. Its attributes include: msg, and socket. Its method includes: run, and sendmessage.

- Class agentregister: agentregister class is used to register the agent to a facilitator and discover a certain service required by the agent. It is associated with the marker class as an attribute of it. Its attributes include: agent-entry, serviceid. Its methods includes: findagent, registeragent, and servicerequest.

Figure 4.2.1.1.2 shows the communication sequence chart of the Marker Agent.

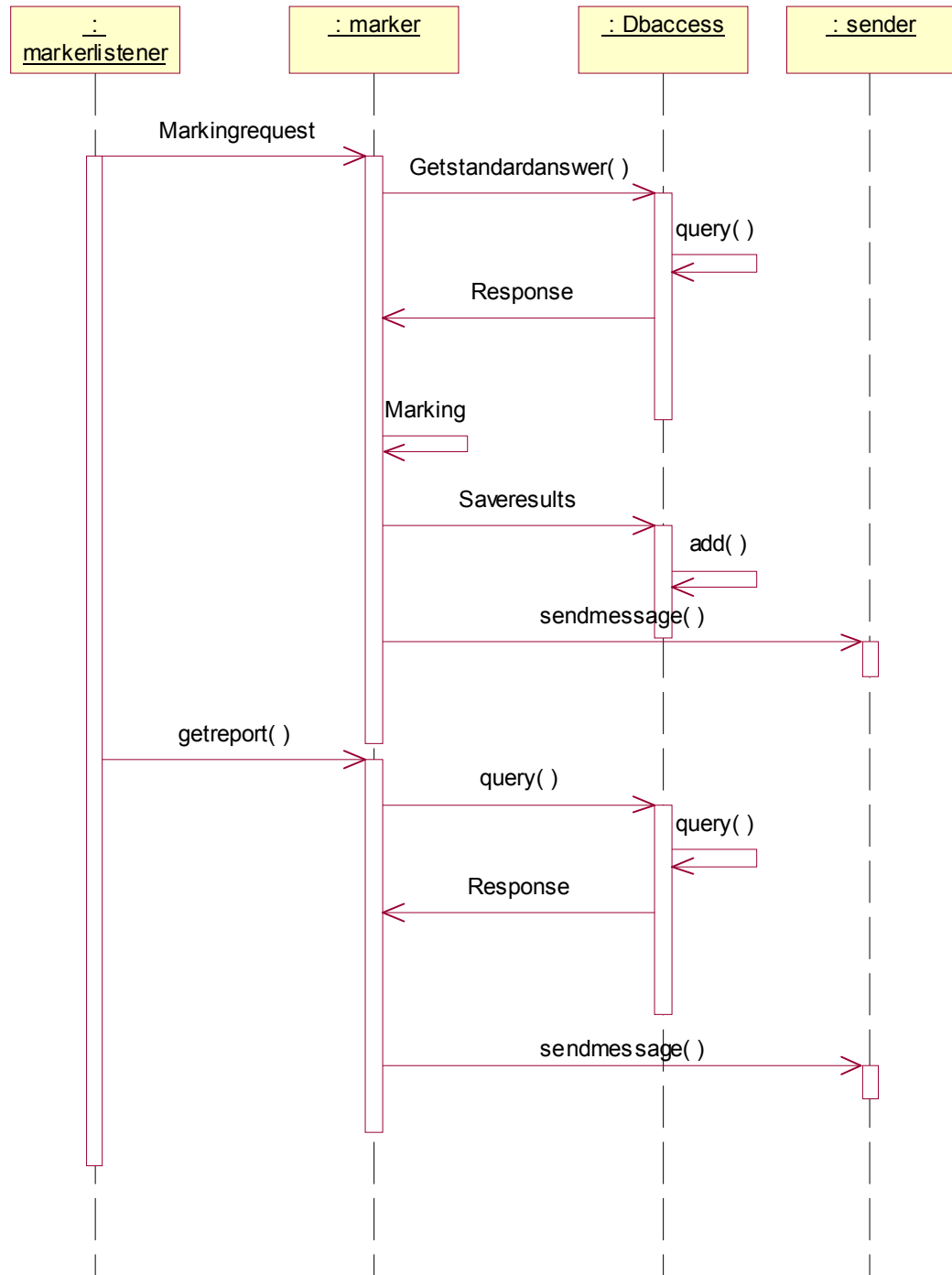
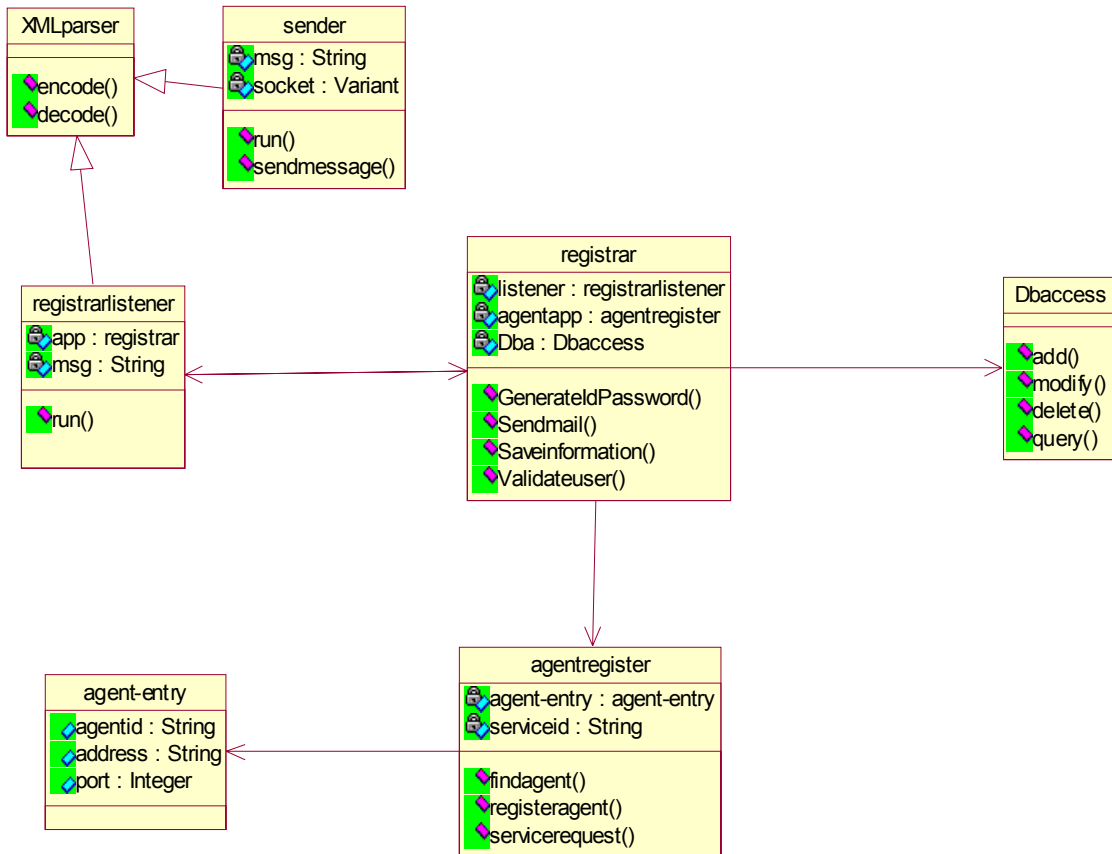


Figure 4.2.1.1.2 Marker Agent Communication Sequence Chart

#### 4.2.1.2 Registrar Agent

The following Figure 4.2.1.2.1 shows the class diagram of the Registrar Agent.

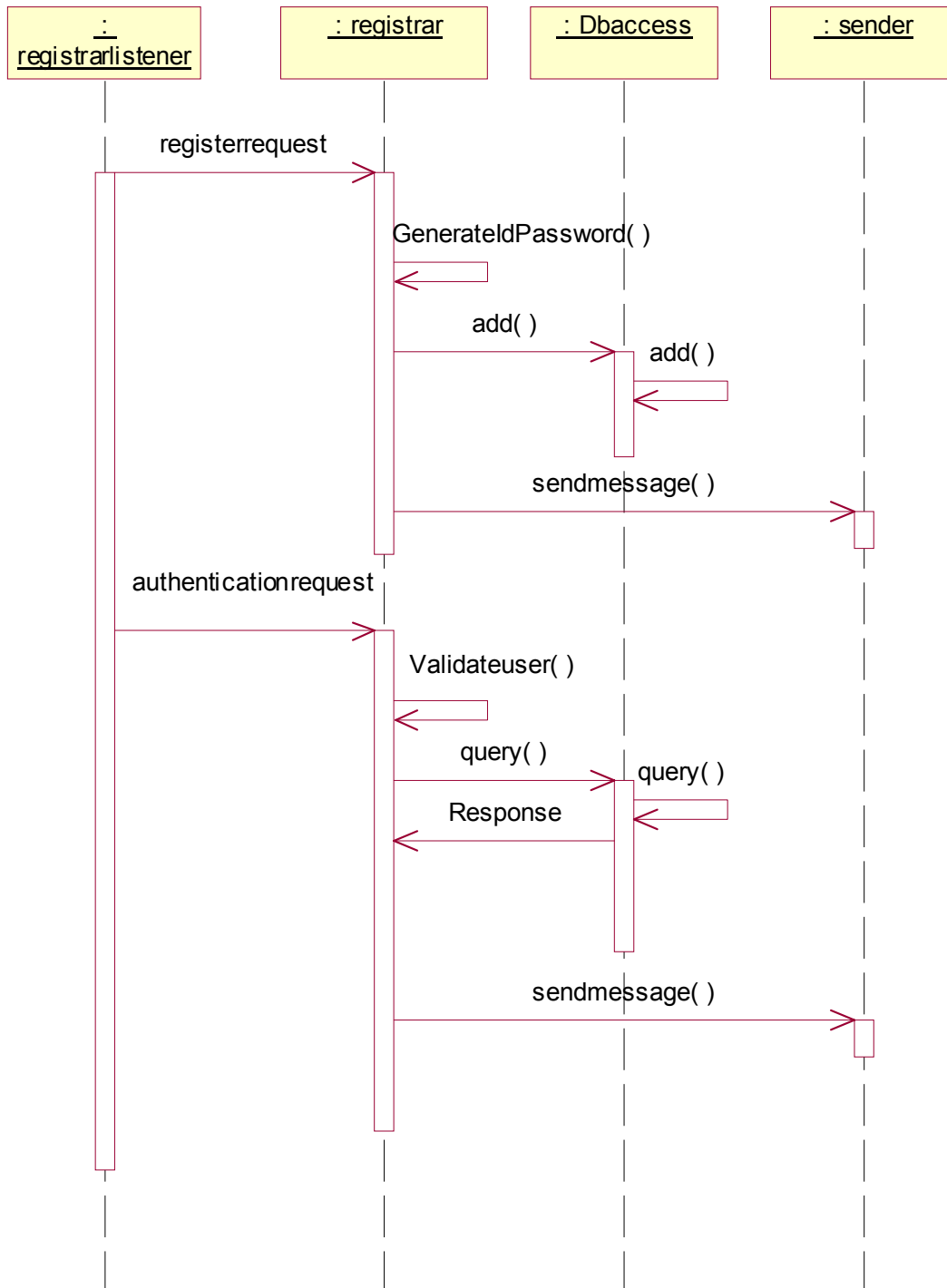


**Figure 4.2.1.2.1 Registrar Agent Class Diagram**

- Class registrar: All the core functionality of the registrar agent is performed by the registrar class. Its attributes include: listener, agentapp and Dbaccess. Its methods include: GenerateidPassword, sendmail, saveinformation and validateuser.
- Class Dbaccess: Dbaccess class is used to interact with the OEAS database system. It is associated with the registrar class as an attribute of it. Its methods include: add, modify, delete, and query.
- Class XMLparser: XMLparser class is used to parse the XML message. Its methods include: encode, and decode.
- Class registrarlistener: registrarlistener class is used to listen at a certain port as a thread process and notify the registrar class to perform corresponding action according to the incoming message. It can parse the XML message by inheriting the XMLparser class. It is associated with the registrar class as an attribute of it. Its attributes include: app, and msg. Its methods include: run.
- Class sender: sender class is used to send out messages. It can also parse the XML message by inheriting the XMLparser class. Its attributes include: msg, and socket. Its methods include: run, and sendmessage.
- Class agentregister: agentregister class is used to register the agent to a facilitator and discover a certain service required by the agent. It is associated

with the registrar class as an attribute of it. Its attributes include: agent-entry, serviceid. Its methods includes: findagent, registeragent, and servicerequest.

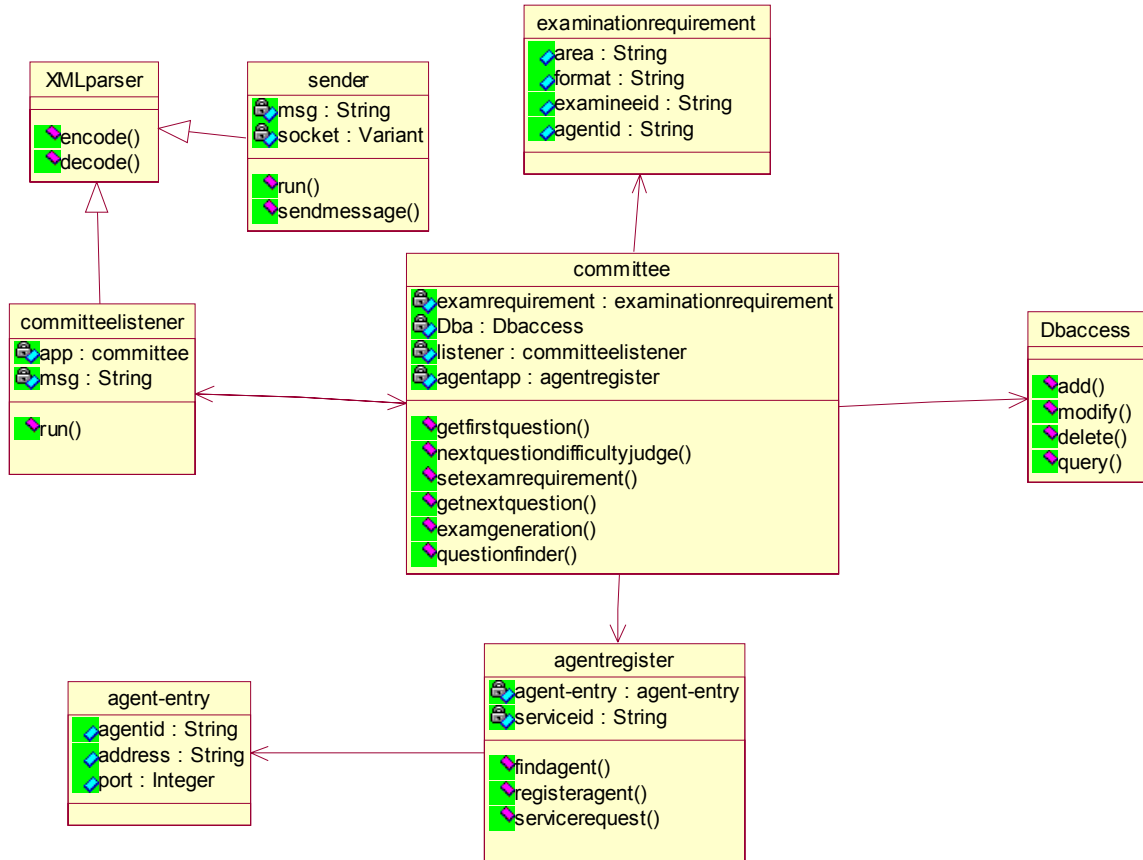
Figure 4.2.1.2.2 shows the communication sequence chart of the Registrar Agent.



**Figure 4.2.1.2.2 Registrar Agent Communication Sequence Chart**

### 4.2.1.3 Committee Agent

Figure 4.2.1.3.1 is the class diagram of the Committee Agent.

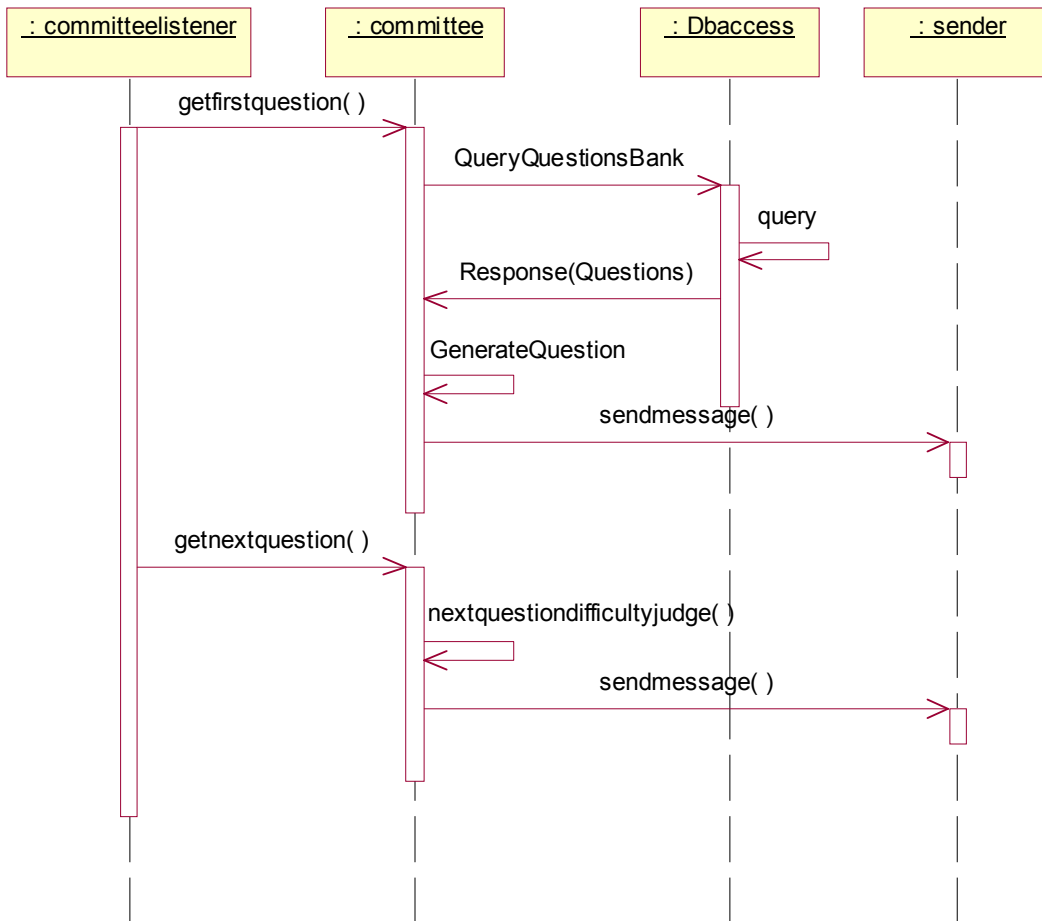


**Figure 4.2.1.3.1 Committee Agent Class Diagram**

- Class committee: All the core functionality of the committee agent is performed by the committee class. Its attributes include: listener, agentapp, Dbaccess and examrequirement. Its methods include: getfirstquestion, nextquestiondifficultyjudge, setexamrequirement, getnextquestion, examgeneration, and questionfinder.
- Class Dbaccess: Dbaccess class is used to interact with the OEAS database system. It is associated with the committee class as an attribute of it. Its method include: add, modify, delete, and query.
- Class XMLparser: XMLparser class is used to parse the XML message. Its method include: encode, and decode.
- Class committeelistener: committeelistener class is used to listen at a certain port as a thread process and notify the committee class to perform corresponding action according to the incoming message. It can parse the XML message by inherit the XMLparser class. It is associated with the committee class as an attribute of it. Its attributes include: app, and msg. Its method includes: run.

- Class sender: sender class is used to send out messages. It can also parse the XML message by inherit the XM:parser class. Its attributes include: msg, and socket. Its method includes: run, and sendmessage.
- Class agentregister: agentregister class is used to register the agent to a facilitator and discover a certain service required by the agent. It is associated with the committee class as an attribute of it. Its attributes include: agent-entry, serviceid. Its methods includes: findagent, registeragent, and servicerequest.

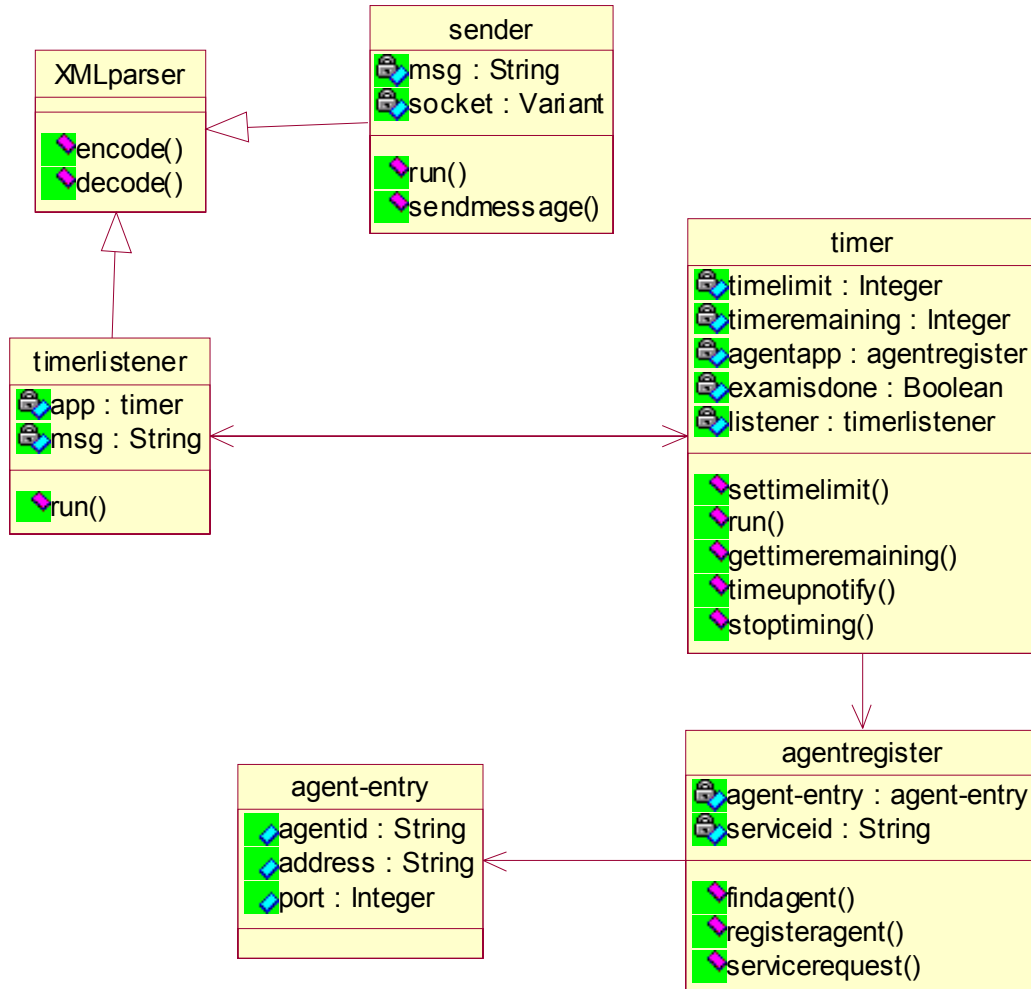
Figure 4.2.1.3.2 is the communication sequence chart of the Committee Agent.



**Figure 4.2.1.3.2 Committee Agent Communication Sequence Chart**

### 4.2.1.4 Timer Agent

Figure 4.2.1.4.1 shows the class diagram of Timer Agent.

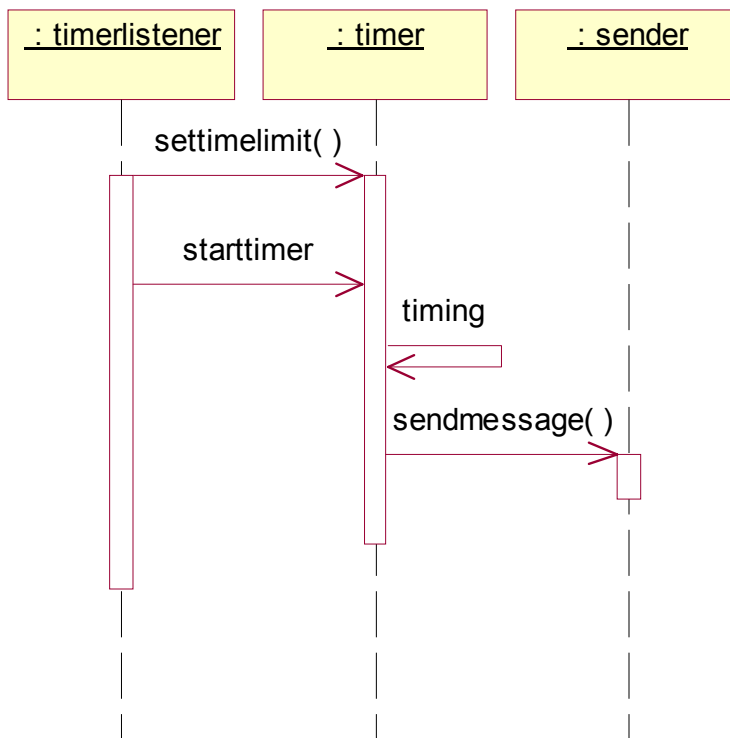


**Figure 4.2.1.4.1 Timer Agent Class Diagram**

- Class timer: All the core functionality of the timer agent is performed by the timer class. Its attributes include: listener, agentapp, timelimit, timerremaining, and examisdone. Its methods include: settimelimit, run, gettimerremaining, timeounotify, and stoptiming.
- Class XMLparser: XMLparser class is used to parse the XML message. Its method include: encode, and decode.
- Class timerlistener: timerlistener class is used to listen at a certain port as a thread process and notify the timer class to perform corresponding action according to the incoming message. It can parse the XML message by inherit the XMLparser class. It is associated with the timer class as an attribute of it. Its attributes include: app, and msg. Its method includes: run.

- Class sender: sender class is used to send out messages. It can also parse the XML message by inherit the XM: parser class. Its attributes include: msg, and socket. Its method includes: run, and sendmessage.
- Class agentregister: agentregister class is used to register the agent to a facilitator and discover a certain service required by the agent. It is associated with the timer class as an attribute of it. Its attributes include: agent-entry, serviceid. its methods includes: findagent, registeragent, and servicerequest.

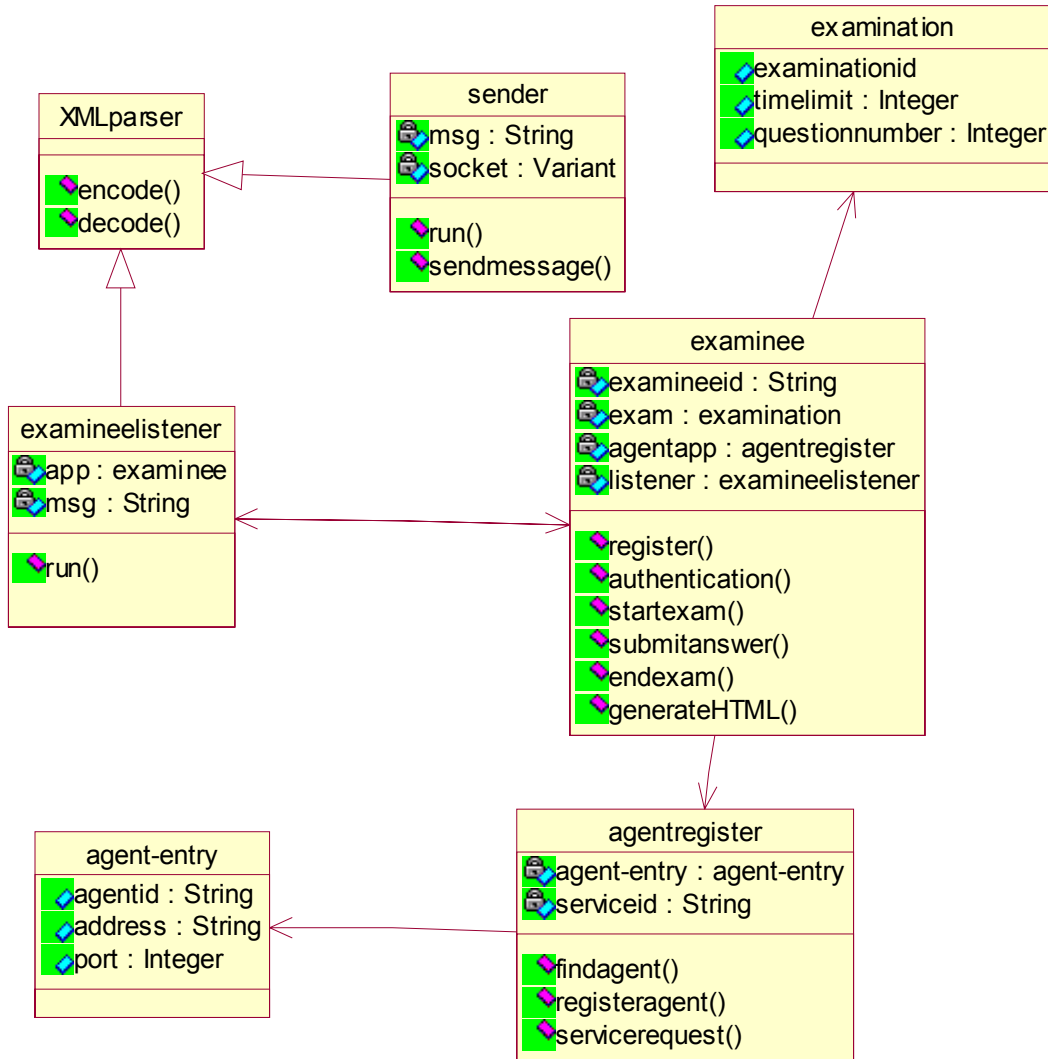
Figure 4.2.1.4.2 is the communication sequence chart of Timer Agent.



**Figure 4.2.1.4.2 Timer Agent Communication Sequence Chart**

### 4.2.1.5 Examinee Agent

Figure 4.2.1.5.1 shows the class diagram of Examinee Agent.



**Figure 4.2.1.5.1 Examinee Agent class diagram**

- Class examinee: All the core functionalities of the timer agent are performed by the examinee class. Its attributes include: listener, agentapp, exam, and examineeid. Its methods include: register, authentication, startexam, submitanswer, endexam, and generateHTML.
- Class XMLparser: XMLparser class is used to parse the XML message. Its method includes: encode, and decode.
- Class examineelister: examineelister class is used to listen at a certain port as a thread process and notify the examinee class to perform corresponding action according to the incoming message. It can parse the XML message by inherit the XMLparser class. It is associated with the

examinee class as an attribute of it. Its attributes include: app, and msg. Its method includes: run.

- Class sender: sender class is used to send out messages. It can also parse the XML message by inherit the XM: parser class. Its attributes include: msg, and socket. Its method includes: run, and sendmessage.
- Class agentregister: agentregister class is used to register the agent to a facilitator and discover a certain service required by the agent. It is associated with the examinee class as an attribute of it. its attributes include: agent-entry, serviceid. Its methods includes: findagent, registeragent, and servicerequest.

## 5. Data Specification

### 5.1.Data and/or Knowledge Sharing Specification

The Marker, Committee and Registrar agents share the central database. Figure 5.1.1 shows the relational table diagram of the OEAS database.

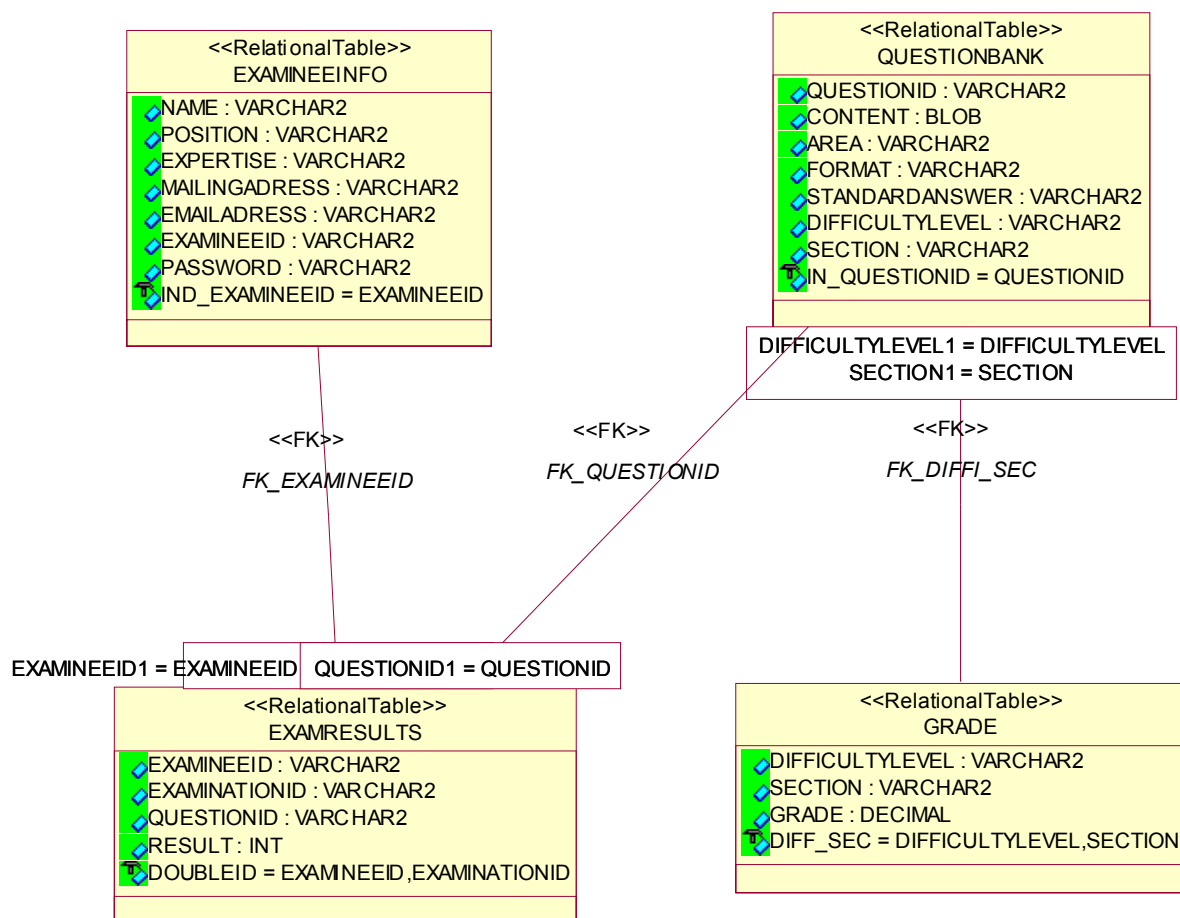


Figure 5.1.1 the relational table diagram of the OEAS database

## 5.2. Information Table

### 5.2.1. Examineeinfo Table

- The information of an examinee registered the system is stored in the Examineeinfo table. This table is accessed by the Registrar agent to get the examinee's information.

Field	Description	Type
Name	Examinee's name	Varchar2 (20)
position	Examinee's position	Varchar2 (10)
expertise	Examinee's expertise	Varchar2 (20)
mailingadress	Examinee's mailing address	Varchar2 (200)
Emailadress	Examinee's email address	Varchar2 (50)
examineeid	Examinee's identification id	Varchar2 (10)
password	Examinee's password	Varchar2 (10)

### 5.2.2. Questionbank Table

- The information about all the questions is stored in the Questionbank table. This table is also accessed by the Committee agent and the Marker agent to get the information of the question.

field	Description	Type
questionid	A question's identification id	Varchar2 (10)
content	The content of the question	BLOB
area	The area of the question (math, computer)	Varchar2 (20)
format	The format of the question (multiple choice, blankfilling)	Varchar2 (10)
standardanswer	The right answer of the question	Varchar2 (200)
difficultylevel	The difficulty level of the question (high, medium, low)	Varchar2 (10)
section	The subsection of the corresponding area (math: algebra, computer: operating system)	Varchar2 (10)

### 5.2.3. Examresult Table

- The results of each examination taken by the examinee are stored in the Examresults table. This table is also accessed by the Marker agent to generate the report and analyze the difficulty level of each question.

Field	Description	Type
examineeid	Examinee's identification id	Varchar2 (10)
examinationid	The identification id of a examination that is taken by examinee	Varchar2 (10)
questionid	A question's identification id	Varchar2 (10)
result	The result of a answer (1, right) (0, wrong)	int

### 5.2.4. Grade Table

- The marks for each section and difficulty level are stored in Grade table. This table is accessed by the Marker agent to compute marks for the examinee. For example: The mark for a question that is in the algebra section and high difficulty level is 10. In this way the marks can be given according to the results.

Field	Discription	Type
Difficultylevel	The difficulty level of the question (high, medium, low)	Varchar2 (10)
Section	The subsection of the corresponding area	Varchar2 (10)
Grade	The grade of a specific section and difficulty level	Decimal (5,2)

## 6. Inter-Agent Messages

As what we mentioned in Communication protocol section, we will use SOAP as the communication protocol between agents. All the agent will communicate and cooperated through the message exchanged based on XML message format, which could be understood by each agent in our system.

### Agent AgentRegister Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"? &gt; &lt;AgentRegister&gt;   &lt;Location&gt;IpAddress&lt;/Location&gt;   &lt;ListenerPort&gt;Integer&lt;/ListenerPort&gt;   &lt;Service&gt;String &lt;/Service&gt; &lt;/AgentRegister&gt;</pre>	Agent Register itself to Facilitator
Output Parameters	
<pre>&lt;?xml version="1.0"? &gt; &lt;AgentRegInfo&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt; &lt;/AgentRegInfo&gt;</pre>	Facilitator assign a AgentID to the agent

### Agent Discovery Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"? &gt; &lt;Discovery&gt;   &lt;Service&gt;String &lt;/Service&gt; &lt;/Register&gt;</pre>	Agent request an agent information to server its' request.
Output Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;AgentInformations&gt;   &lt;AgentInfo&gt;     &lt;AgentID&gt;String&lt;/AgentID&gt;     &lt;Location&gt;IpAddress&lt;/Location&gt;     &lt;ListenerPort&gt;Integer&lt;/ListenerPort&gt;   &lt;/AgentInfo&gt;   .   .   n   .   . &lt;/AgentInformations&gt;</pre>	Facilitator provide the agent lists available to server the particular requests

### Examinee Register Message

Message Format	Description
<b>Input Parameters</b>	
<pre>&lt;?xml version="1.0"? &gt; &lt;ExamineeRegister&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;FullName &gt;String &lt;/FullName&gt;   &lt;Position&gt;String &lt;/Position&gt;   &lt;Expertise&gt;String &lt;/Expertise&gt;   &lt;MailingAddress&gt;String &lt;/MailingAddress&gt;   &lt;EmailAddress&gt;String &lt;/EmailAddress&gt; &lt;/ExamineeRegister&gt;</pre>	Examinee's Registration Information.
<b>Output Parameters</b>	
<pre>&lt;?xml version="1.0"?&gt; &lt;AccountInformation&gt;   &lt;ExamineeID&gt;String&lt;/ExamineeID&gt;   &lt;Password&gt;String&lt;/Password&gt; &lt;/AccountInformation&gt;</pre>	Account's information

### Examinee Authentication Message

Message Format	Description
<b>Input Parameters</b>	
<pre>&lt;?xml version="1.0"?&gt; &lt;AccountInformation&gt;   &lt;ExamineeID&gt;String&lt;/ExamineeID&gt;   &lt;Password&gt;String&lt;/Password&gt; &lt;/AccountInformation&gt;</pre>	Account Information
<b>Output Parameters</b>	
<pre>&lt;?xml version="1.0"?&gt; &lt;Authentication&gt;   &lt;Result&gt;Boolean &lt;/Result&gt; &lt;/Authentication&gt;</pre>	Authentication result

### Examinee Question Request Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;QuestionRequest&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;ExamineeID&gt;String&lt;/ExamineeID&gt;   &lt;Area&gt;String &lt;/Area&gt; &lt;/QuestionRequest&gt;</pre>	Question Request
Output Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;QuestionInfo&gt;   &lt;ExaminationID&gt;String&lt;/ExaminationID&gt;   &lt;QuestionID&gt;String&lt;/QuestionID&gt;   &lt;QuestionContent&gt;Text&lt;/QuestionContent&gt; &lt;/QuestionInfo&gt;</pre>	Committee gives the first Question Information to examinee

### Examinee Mark Request Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;Answer&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;ExaminationID&gt;String&lt;ExaminationID&gt;   &lt;QuestionID&gt;String &lt;/QuestionID&gt;   &lt;Answer&gt;String &lt;/Answer&gt; &lt;/Answer&gt;</pre>	Agent request marker to mark the answer. <AgentID> is the examinee agent ID, which is needed when committee assigned the questions in next step.
Output Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;AnswerResult&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;ExaminationID&gt;String&lt;ExaminationID&gt;   &lt;QuestionID&gt;String&lt;/QuestionID&gt;   &lt;AnswerResult&gt;Boolean&lt;/AnswerResult&gt; &lt;/ AnswerResult &gt;</pre>	Marker agent gives this mark information to committee agent to generate the next questions.

### Examinee Start Timer Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;StartTimer&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;ExaminationID&gt;String &lt;/ExaminationID&gt; &lt;/StartTimer&gt;</pre>	Agent request timer agent to start information
Output Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;TimerStartInformation&gt;   &lt;TimeLeft&gt;Long&lt;/TimeLeft&gt; &lt;/ TimerStartInformation &gt;</pre>	Time left information

### Marker Question Request Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;AnswerResult&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;ExaminationID&gt;String&lt;ExaminationID&gt;   &lt;QuestionID&gt;String&lt;/QuestionID&gt;   &lt;AnswerResult&gt;Boolean&lt;/AnswerResult&gt; &lt;/ AnswerResult &gt;</pre>	Question answered status
Output Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;QuestionInfo&gt;   &lt;QuestionID&gt;String&lt;/QuestionID&gt;   &lt;QuestionContent&gt;Text&lt;/QuestionContent&gt; &lt;/QuestionInfo&gt;</pre>	The question given by Committee based on the answers status from Marker agent to examinee.

## Examinee Report Request Message

Message Format	Description
Input Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;ReportRequest&gt;   &lt;AgentID&gt;String&lt;/AgentID&gt;   &lt;ExaminationID&gt;String&lt;/ExaminationID&gt; &lt;/ReportRequest&gt;</pre>	Examinee requests report on an examination he/she has taken before.
Output Parameters	
<pre>&lt;?xml version="1.0"?&gt; &lt;ReportInformation&gt;    &lt;RightAnswerNumber&gt;Integer&lt;/RightAnswerNumber&gt;   &lt;WrongAnswerNumber&gt;Integer&lt;/WrongAnswerNumber&gt;   &lt;FinalGrade&gt;Integer&lt;/FinalGrade&gt; &lt;/ReportInformation&gt;</pre>	Examination Results

## 7. Conclusion

In this document, we have described the analysis and design of “On-line Exam System” using the methodology of agent-based systems. The key concepts in this design are roles, which have associated with them responsibilities, permission, activities, and protocols. Roles can interact with one another in certain institutionalized ways, which are defined in protocols of the respective roles.

First, in the analysis process, we divided the system into four agents; Registrar Agent, Committee Agent, Marker Agent, and Timer Agent according to their functions and inputs/outputs. Then according to these four agents, we defined five roles; Examinee Role, Registrar Role, Committee Role, Marker Role, and Timer Role, and their permission, responsibilities, and activities of each role. During analysis the interaction model, we defined a set of protocols. Using the protocol, we elaborated the roles model. During the design process, we aggregated roles into agent. In the data specification, the knowledge query and manipulation language formats are defined for this particular system. Then, the knowledge sharing and communication are analyzed. Finally, detailed design about each agent in this system is presented with the help of UML tools.

## 8. References

- Behrouz Homayoun Far, “Agent Oriented Software Engineering Agent-SE”
- Michael Wooldridge, Nicholas R. Jennings, and David Kinny,  
“A Methodology for Agent-Oriented Analysis and Design”
- Behrouz Homayoun FAR, Hassan Hajji, Shadan Saniepour, Sidi O.Soueina,  
Mahmoud M. Elkhoully, “Formalization of Organizational Intelligence for  
Multiagent System Design”
- Tim Finin, Yannis Labrou, James Mayfield, “KQML as An Agent  
Communication Language”
- Jurgen Lind, German Research Center for AI, “The Massive Development  
Method for Multiagent Systems”
- FIPA TC Architecture, “Foundation for Intelligent Physical Agents”