	Course Number: SENG 697	Course Name: Agent-Based Software Engineering
	Session: Fall, 2006	Group Members: Min Xu, Lizhe Ma
	Instructor: B. H. Far	Document Type: Project Report

**Multi-agent system for
Collaborative Software Design Assistant
(CSDA)**

Content

1.	System Specifications	3
1.1.	Business case.....	3
1.2.	System description	3
1.3.	Assumptions and Requirements	4
1.3.1.	Assumptions	4
1.3.2.	Requirements.....	4
1.4.	Wish Lists.....	5
2.	System Analysis	5
2.1.	System Structure.....	5
2.2.	The Role Model	7
2.2.1.	User Role.....	7
2.2.2.	UserHandler Role (UH).....	8
2.2.3.	ActivityManger Role (AM).....	9
2.2.4.	Design Role (Des)	10
2.2.5.	QualityControl Role (QC).....	10
2.2.6.	Documentation Role (Doc)	11
2.2.7.	Schedule Role (Sche).....	11
2.3.	Interaction Model.....	12
2.3.1.	Interactions for UserHandler Role	12
2.3.2.	Interactions for ActivityManager Role.....	14
2.3.3.	Interaction for Des Role	15
2.3.4.	Interactions for QC role.....	16
2.3.5.	Interactions for Doc role.....	18
2.3.6.	Interactions for Sche role.....	20
3.	System Design.....	22
3.1.	Agent Model	22
3.2.	Agent Internal Architecture	23
3.3.	Service Model.....	24
3.3.1.	Service of UserHandler role.....	24
3.3.2.	Service of MEA Role.....	24
3.3.3.	Service of Design Agent Role.....	25
3.3.4.	Service of QualityControl Role.....	26
3.3.5.	Service of Schedule Role	27
3.4.	Acquaintance Model.....	27
4.	Detailed Design.....	28
4.1.	Use Case definition	28
4.1.1.	User Case for PAA.....	28
4.1.2.	Use case for MEA.....	29
4.1.3.	Use case for DesA.....	30
4.1.4.	Use case for QCA.....	30
4.1.5.	User case for DocA.....	32
4.1.6.	Use case for SA.....	34
4.2.	Typical Data Specification	35
4.3.	Communication Specification	38
4.3.1.	Typical Inter-Agents Messages.....	38
4.3.2.	Other Inter-Agent Messages.....	40
5.	Conclusions	44
6.	Acknowledge	44
7.	Reference	45

1. System Specifications

1.1. Business case

Large projects involve a few teams working together. The team is usually consisted of designer team, quality control team, documentation team and manager.

The designers have to synchronize the code after the previous code is updated, to notify the quality control team and manager, to fix the bugs depending on the feedback from the quality control team and to speed up the development or not depending on the feedback from the manager.

The quality control team has to get the right version code from source control, test the code and give the feed back to designers.

The documentation team has to contact the quality control team to get the well tested code, and do the documentation.

The manager has to set the schedule for the whole project, and supervise the process of each team

From previous analysis, we can get that there are lots of synchronization and communication between different teams. The synchronization and communication between teams are the main problems.

In this case, the intelligent agents can be used to synchronize and communicate between different teams acting as assistant agents.

1.2. System description

The purpose of this collaborative software design assistant (CSDA) system is to release the work loads for the software design and development teams, so they can concentrate on the main tasks.

The system consists of personal assistant agents, which are located in the client side, message exchange agents (MEA), design agents (DesA), quality control agents (QCA), scheduler agents (SA) and document agents (DocA). Except PAA, all the agents are located in the server side and compose the whole management system. The system assists all the team members. Team member sends tasks to the system. The system identifies the type of the tasks, and forwards the tasks to the corresponding agent. These tasks are the goals of the agent. Every agent has its own knowledge base and is independent with other agents. It acts on its own behalf and it might interact with other agents to achieve the goal, and then return the result to the team member.

The agents use the blackboard to communicate with each other and exchange message through the message exchange agents which acts like a "yellow page"

1.3. Assumptions and Requirements

1.3.1. Assumptions

The collaborative software design assistant (CSDA) system can identify the type of each software design and development team member.

Individual agent in the CSDS system has its own responsibility and knowledge.

- The design agent is used to serve the requests from designer team
- The quality control agent is used to serve the requests from quality control team
- The scheduler agent is used to serve the requests from manager
- The document agent is used to serve the request from document team
- The message exchange agent is used to exchange the message for types of agents
- The blackboard is used to communicate between agents.
- The database is used to store all the information for all the tasks.

The developing engineers should post integrated module source code into this system. They should provide enough information for the system to tell its name, version and authors.

The test result provided by quality control team should be a Boolean variable, true or false.

The documentation provided by the documentation team should be the final version. Once it is posed, it will be integrated with the other parts.

1.3.2. Requirements

The agent corresponding each team should be able to learn new knowledge if the existing knowledge is not suitable for the tasks.

The knowledge base should be able to be updated after the agents learn new knowledge.

The agent should be able to use blackboard to communicate with other and use MEA to exchange the message, and get the information from the database.

The system should be able to identify the type of tasks, so it can forward the tasks to the proper agents.

The design agents can synchronize the code after the code is updated, and notify the agents of quality control team and the agents of manager. It can also get the feedback from quality control agents. So the CSDA system can release the work load for the designer.

The quality control agent can get the right version code from source control specified by

the quality control team member, provide the preferred test criteria, it can also learn new test criteria specified by the quality control team member.

The documentation agent can get the right version code from source control specified by the documentation team member, and provide the proper documentation template. It can also learn new documentation template specified by the documentation team member.

1.4. Wish Lists

The following items are wished to be added in the future CSDA system design

- The CSDA system allows the different users to modify the results online.
- If the schedule is not satisfied, there should a mechanism to send message to the delayed portion to reminding their schedule is not keeping up with the whole system.
- Set up the instant message communication system for all the participants to discuss or exchange opinions.

2. System Analysis

The objective of the analysis is to develop an understanding of the whole system and architecture. We view the system as a collection of roles. In this project, we adopt the GAIA as the basic analysis and design principles to fulfill the whole system, thus the organization model is comprised of two further models: the roles model and the interaction model.

2.1. System Structure

According to the requirement and use cases, the Software Design Assistant system showed in figure 2-1 involves the several agents and some resources:

Personal Assistant Agent (PAA): it will directly interact with users. There are four types of users to use this system. (1) Develop Engineers will use this system to post implemented modules and expect the get some feedbacks, such as the module is passed or not in the testing process. If not, they need to re-modify the source code accordingly. (2) Test Engineers will use this system to evaluate the posted modules. They expect the system return the suggested test criteria as well as the source code based on this module. They can adopt the criteria or modify them according to their own experiences. Then they need to input the test result as well so that the documentation team can take certain action accordingly. (3) Document team can use this system to generate the document for specific modules. (4) Managers can use this system to oversee the whole process and expect to get the prediction of the time schedule from the system based on current implementation status.

Design Agent (DesA): the main function of this agent is to deal with the request from

develop engineers, including post source code, fetch source code, inform Quality control agent to start with test process and updates the blackboard.

Quality Control Agent (QCA): the main function of this agent is to provide the test criteria after searching the knowledgebase. It is a kind of intelligent agents. Based on the information of the posted module, it can search the corresponding knowledge in the knowledgebase and return the preferred test criteria to PAA.

Documentation Agent (DocA): this agent is used to generate the documentation template according the information of the posted modules which have passed the test procedure. It is a kind of intelligent agents. Based on the information of the posted module, it can search the corresponding knowledge in the knowledgebase and return the preferred documentation template to PAA.

Schedule Agent (SA): this agent is going to assist the manager to oversee the whole design process and provide the time prediction based on the current status. It will inform the manager if the schedule is ahead of the plan, or on time or delayed. It is a kind of intelligent agents. Only when this module has passed the test and been documented correctly, the prediction process will be invoked to generate the schedule and return the result to manager or published in the blackboard.

Message Exchange Agent (MEA): it is used to exchange the message between the agents. It is a kind of agent to provide directory-service, which is to provide a location where agents register directory-entries, and the message-transport-service.

Knowledgebase (KB): this resource is used to archive the knowledge as well as some experiences related with the test criteria, document template and the schedule knowledge. Users can provide the feedbacks about this knowledge and then the knowledgebase will be updated with the new information and experiences.

Blackboard (BB): Blackboard is used to share the information for all the related agents. It directly reads and writes information from the database and provides the public allocations for agents to fetch the useful status values.

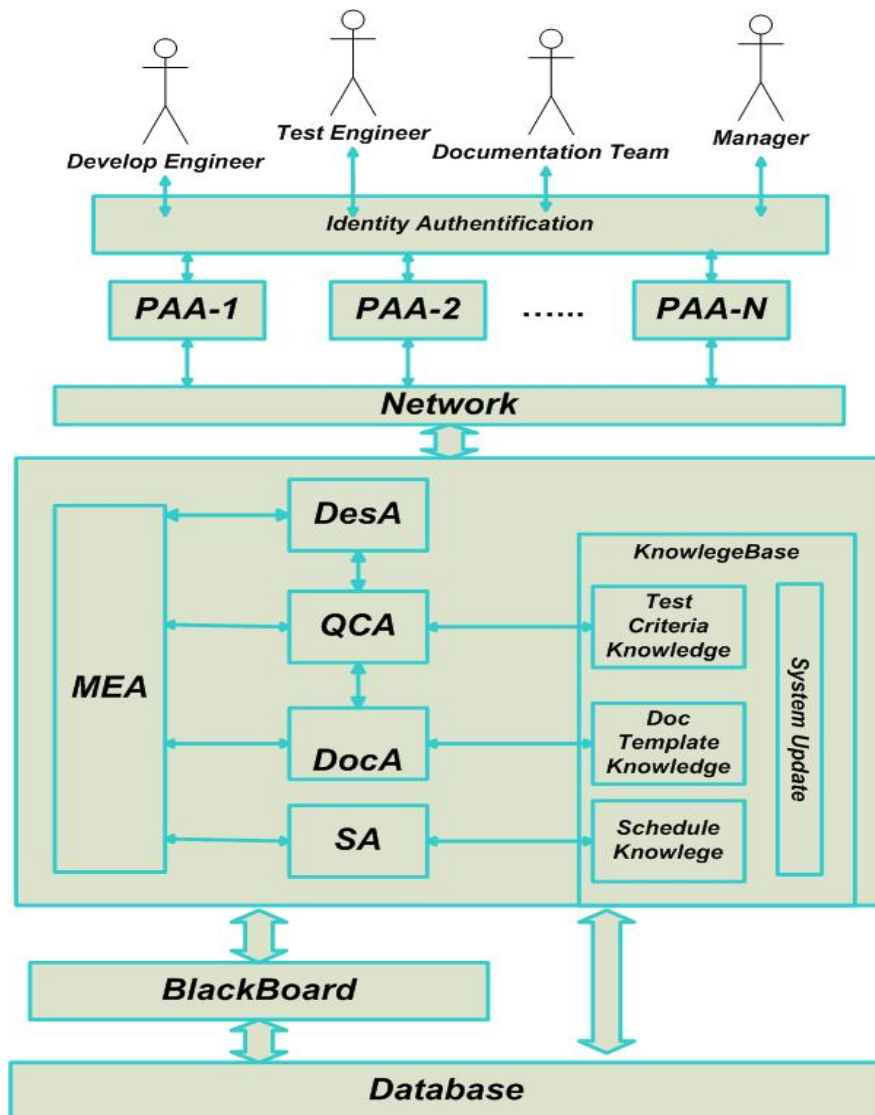


Figure 2-1: Software Design Assistant System Structure

2.2. The Role Model

In this model, we define the key roles in the system.

2.2.1. User Role

This role is to represent the users' behaviors who request the assistances from this system.

Role Schema:	User
Decription:	

	Designers post the new modules or the modified modules to the system, or Test Engineer and documentation team want to test or document the posted modules. Manager can also request to get the schedule prediction based on current statuses.
Protocols and Activities:	
	ProvideModule, ProvideTestResult, ModifyTestCriteria, ProvideDoc, ModifyDocTemplate, ModifySche, FetchSche.
Permissions:	
generate	newModule
	TestResult, TestCriModify, DocResult, DocTempModify, ScheModify
Responsibilities:	
Liveness:	User=UserActivity
	UserActivity=(ProvideModule ProvideTestResult ModifyTestCriteria ProvideDoc ModifyDocTemplate FetchSche ModifySche)+
Safety:	TRUE

2.2.2. UserHandler Role (UH)

PAA behavior falls into two distinct roles. One acting as an interface to the users, it is named as UserHandler; the other one is to oversee the process inside the organization, it is named as ActivityManager.

UserHandler is focusing to communicate with users. It receives module source code from Developer, or testing result or test criteria from testing engineer, or documentation or document template from Doc team. It also passes the request to generate the current schedule. It assists to oversee the whole process to ensure the testing criteria, documentation template and the schedule result is returned.

Role Schema:	UserHandler
Description:	
	Receives module source code from Developer, or testing result or modification from testing engineer, or Documentation as well as the feedback of the doc template from Doc Team or the schedule requests from managers and oversees process to ensure the testing criteria, documentation template and the schedule result is returned.
Protocols and Activities:	
	AwaitRequest, GetModule, GetTestResult, GetTestCri, GetDoc, GetDocTemp, GetSche, GetScheRequest, InformUser
Permissions:	
Read:	supplied ModuleDetails
	supplied test info.
	supplied doc info
	supplied sche info
Generate:	ModuleStatus, ModuleTemp, ScheTemp, ScheRequired
	TestTemp, DocTemp, CriteriaTemp, DocTempTemp

Responsibilities:	
Liveness:	UserHandler=(AwaitRequest.GetAction)w InformUser
	GetAction=(GetModule GetTestResult GetTestCri GetDoc GetDocTemp GetSche GetScheRequest)
Safety:	InformUser only when "TestReqToUser" or "DocReqToUser" or "ScheToUser" is True

2.2.3. ActivityManger Role (AM)

This role is the key point for PAA. It interacts with all the agents to deal with the requests and generate the useful information for users.

Role Schema:	ActivityManager
Description:	
	Publish the basic information provided by UserHandler to the blackboard, inform Quality Control role to search for test criteria if the qualified module is posted, inform Document role to search for template if testing result is posted or inform Schedule role to generate schedule. Or add new test criteria or new doc template or new schedule to KB.
Protocols and Activities:	
	Inform Des, InformQC, InformDocA, InformSA
Permissions:	
Read:	supplied ModuleDetails
	ModuleStatus
	TestCri, DocTemp, ScheResult
	TestResult, DocInfo
Change:	ModuleStatus
Generate:	ModuleStatus //if new module is posted.
	ModuleRequest, TestRequest, DocRequest, ScheRequest
Responsibilities:	
Liveness:	ActivityManager=ActivityResponse
	ActivityResponse=ModuleResp TestResp DocResp ScheResp
	ModuleResp=(InformDes)
	TestResp=(InformQC)
	DocResp=(InformDoc)
	ScheResp=(InformSA)
Safety:	Post and Add will be done only when PostRequest or AddRequest is true.

2.2.4. Design Role (Des)

It is used to deal with the requests to post, update module, fetch module status and source code. The main job of this role is to connect with AM and deal with all the requests and responses related with the develop engineers.

Role Schema:	Design
Description:	
	Receives post request from ActivityManager, posts or updates module to database and returns related information to AM, updates blackboard, and informs Quality Control role if the module is ready for testing.
Protocols and Activities:	
	CheckModule, GetModule, PostModule, InformQC, UpdateBB
Permissions:	
Read:	ModuleSource
Generate:	ModuleInfo, TestReq
Responsibilities:	
Liveness:	Design= CheckModule.PostModule.UpdateBB.InformQC
Safety	InformationAvailable

2.2.5. QualityControl Role (QC)

It is used to deal with the requests to get the test criteria intelligently. The main job of this role is to provide the preferred test criteria based on the analysis for the posted module. It will search for the related knowledge from the knowledgebase. If the test engineers are not satisfied with this criteria, they can modified it or define their own test criteria completely. The system can learn the new knowledge about the test criteria based on the target module and update the knowledgebase.

Role Schema:	Quality Control
Description:	
	Receives test request from ActivityManager, searches for test criteria from KB and returns it to Users, receives test result and updates the database, receives the modified test criteria from users and updates Knowledgebase, and inform the documentation role to do the document if the test is fully passed.
Protocols and Activities:	
	CheckModule, GetModule, GetCri, InformUserToTest, GetTestResult, UpdateBB, InformUserR, RecTestResult, PostTestResult, RecTestCri, AddCri
Permissions:	
Read:	ModuleSource, TestRequest
Generate:	ModuleInfo, TestCri, TestReqToUser, ReTestReq
Responsibilities:	
Liveness:	QualityControl=TestReq RecTest RecTestCri TestReq=CheckModule.GetModule.GetCri.InformUserToTest RecTest=RecTestResult.PostRestResult.UpdateBB.(InformDoc InformUserR)

	RecTestCri=ReciTestCri.AddCri
Safety	InformationAvailable

2.2.6. Documentation Role (Doc)

It is used to deal with the documentation request to get the document template intelligently. The main job of this role is to generate the document template for the target module, which has been passed in the test process. If the documentation team is not satisfied with this template, they can create their own template for this module. The system is able to learn the new information and update the knowledgebase.

Role Schema:	Documentation
Description:	
	Receives documentation request from ActivityManager, searches for Doc Template for specfic Module and returns it to users, receives the document result or document template from users, and posts the document into database or updates the knowledgebase.
Protocols and Activities:	
	CheckModule, CheckTestResult, GetTempt, InformUserToDoc, RecDoc, PostDoc, UpdateBB, RecDocTemp, AddTemp
Permissions:	
Read:	ModuleSource, DocRequest, TestStatus
Generate:	ModuleInforD, DocTemp
	DocReqToUser
Responsibilities:	
Liveness:	Documentation=(CheckTestResult.DocResponse) RecDoc RecDocTemp
	DocReponse=(CheckModule.GetTemp.InformUserToDoc) RecDoc=RecDoc.PostDoc.UpdateBB RecDocTemp=RecDocTemp.AddTemp
Safety	TestResult=Reject-->Doc=nil
	InformationAvailable

2.2.7. Schedule Role (Sche)

It is used to deal with the request from managers who want to know about the current status of this project and get the prediction of the time schedule. The main job of this role to predict the schedule of this project based on the status of all the modules, which includes the information such as the module test result, the module documentation result, etc. after interact with the knowledgebase, it could provide the prediction result for the requesters. If manager wants to change the schedule based on some new requirements, they can modified the schedule and the knowledgebase is able to learn the information and update the whole knowledge system.

Role Schema:	Schedule
Description:	

	Received the ScheduleRequest from ActivityManager, predicts the time schedule according to current status, returns the result to the Users and updates the blackboard
Protocols and Activities:	
	GenerateSche, InformUserSche, RecScheInfo, PostScheInfo, AddScheInfo
Permissions:	
Read:	ModuleStatus //from blackboard
	ScheRequest
Generate:	ModuleInfoS, ScheInfo, ScheToUser
Responsibilities:	
Liveness:	Schedule=ScheResponse RecSche
	ScheResponse=(GenerateSche.InformUserSche) RecSche=RecScheInfo.PostScheInfo.AddScheInfo
Safety:	InformationAvailable

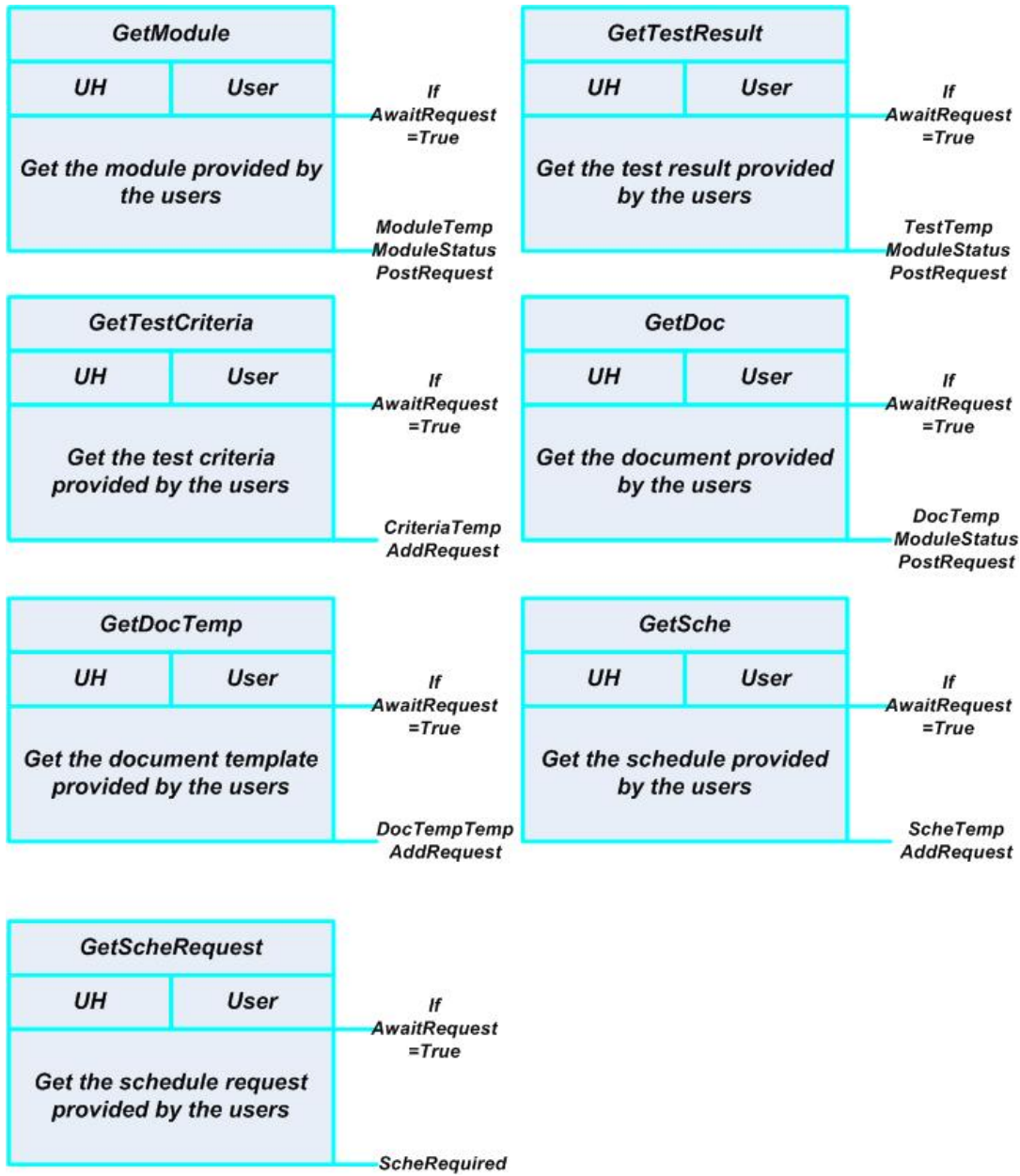
2.3. Interaction Model

In this section, we define the associated interaction models for these roles.

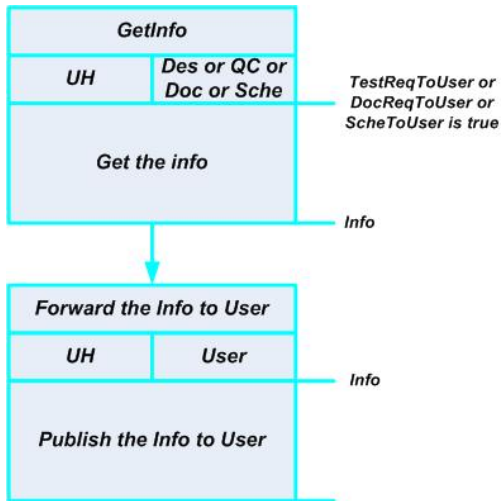
2.3.1. Interactions for UserHandler Role

This role interacts with User role to get users' requests.

Protocols related with Get

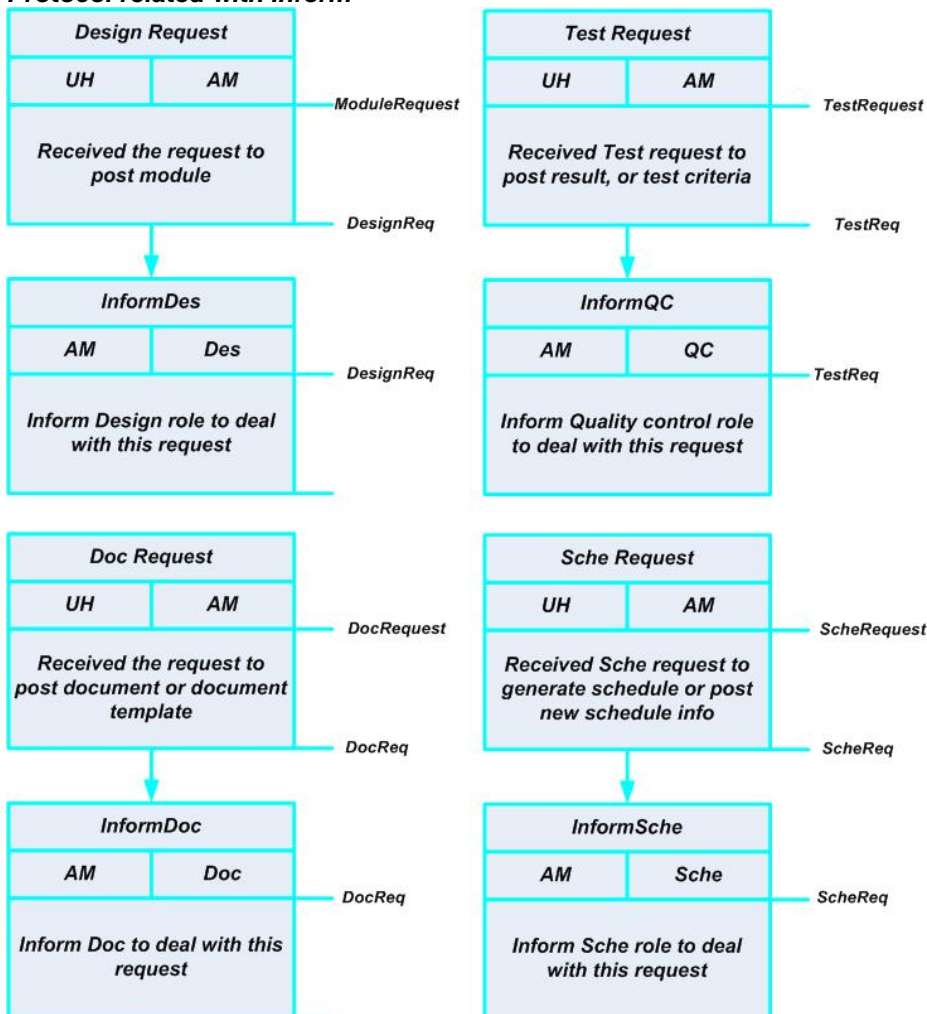


InformUser



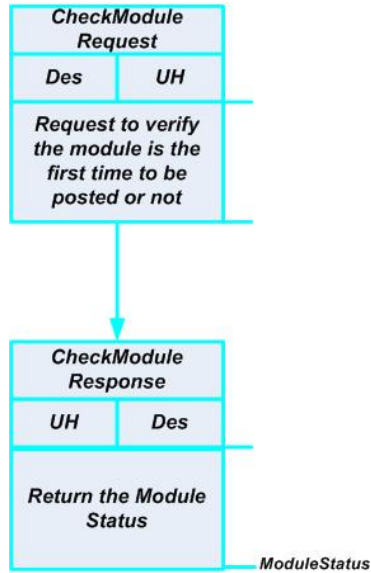
2.3.2. Interactions for ActivityManager Role

Protocol related with Inform

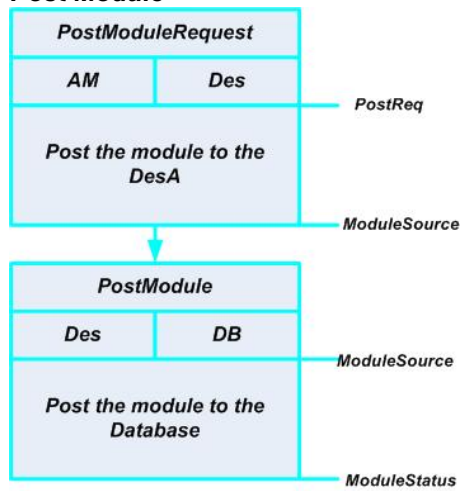


2.3.3. Interaction for Des Role

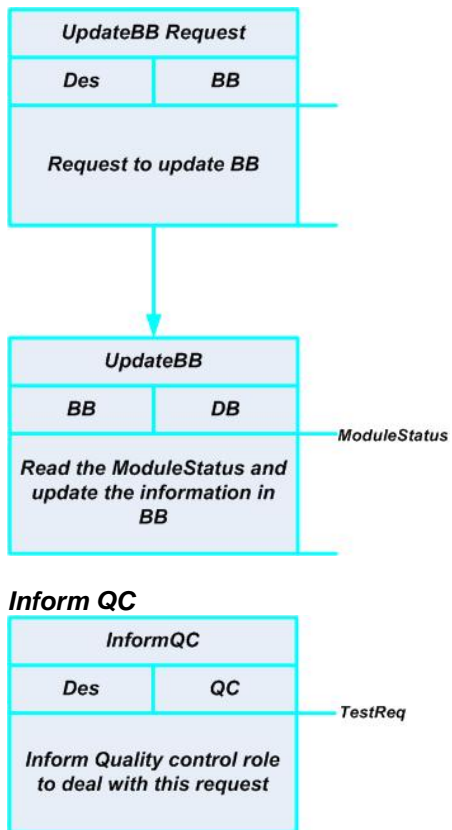
CheckModule: In order to tell the posted module is the new module or it is the modified version.



Post Module

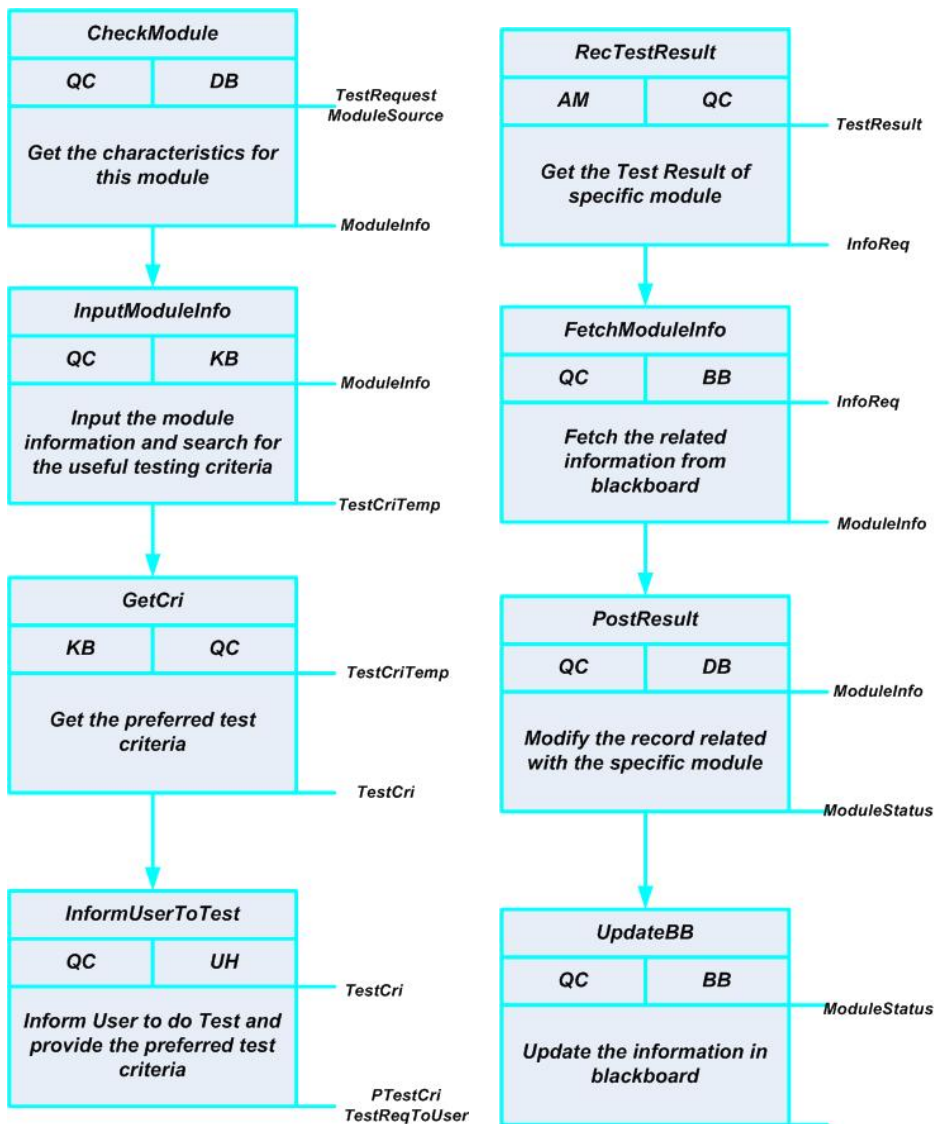


Update Blackboard

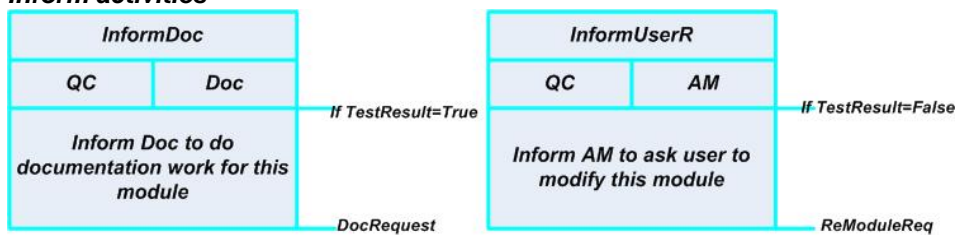


2.3.4. Interactions for QC role

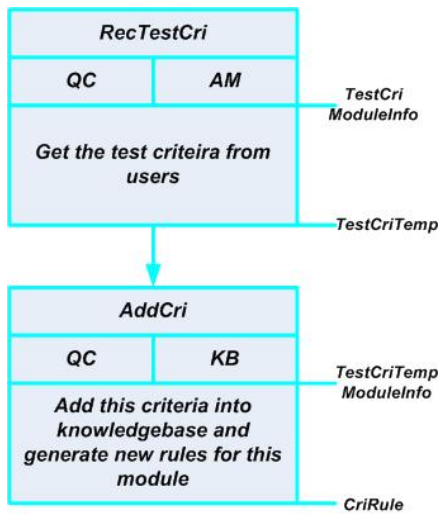
Test Request, include CheckModule, GetModule, GetCri, InformUserToTest; RecTest includes the activity such as RecTestResult, PostRest, UpdateBB.



Inform activities

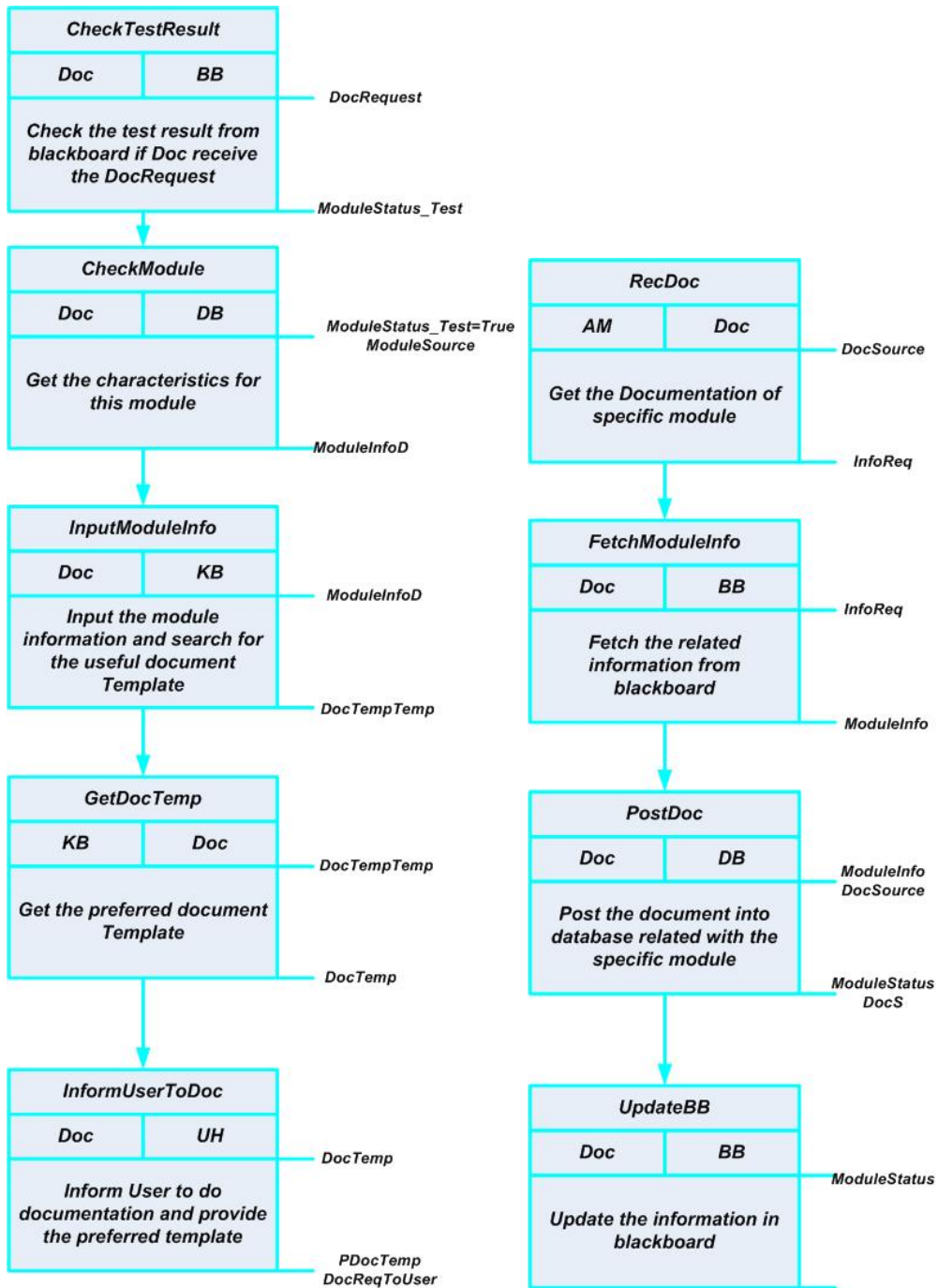


Receive Test Criteria and update the knowledge rules in knowledgebase

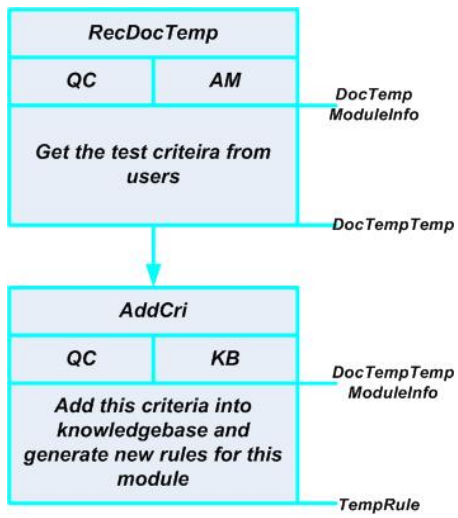


2.3.5. Interactions for Doc role

Document Request: including the CheckModule, GetTemp, InformUserToDoc; Receive documentation and post to database.

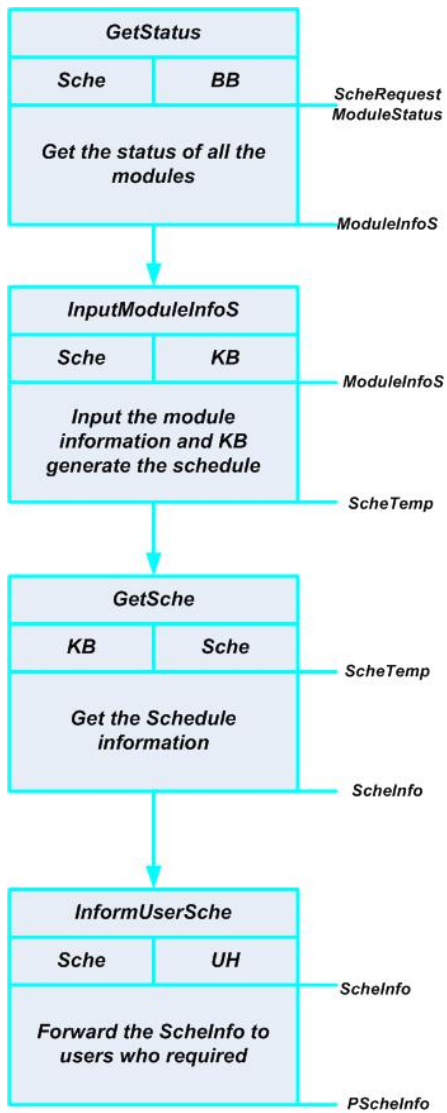


Receive document template and add it into knowledgebase

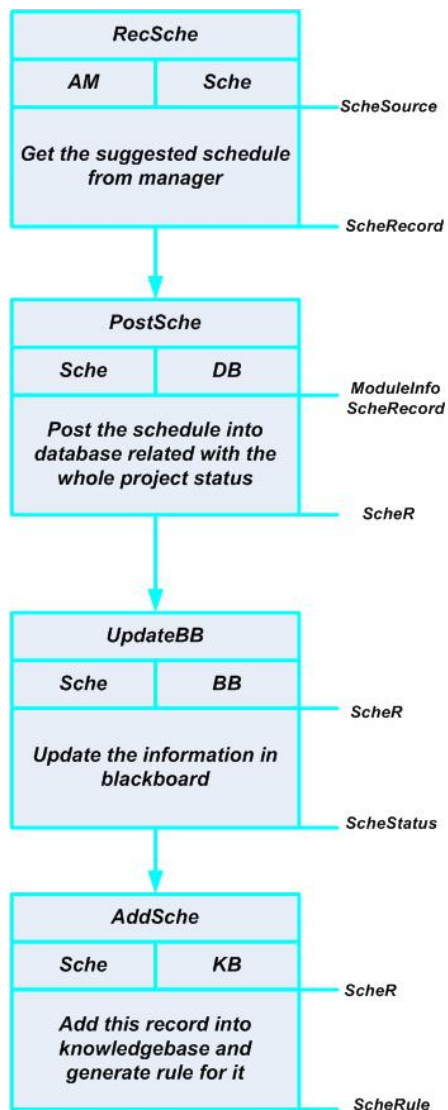


2.3.6. Interactions for Sche role

Schedule Response: including GenerateSche and InformUserSche



Receive suggested Schedule from manager. including *RecScheInfo*, *PostScheInfo*, and *AddScheInfo*



3. System Design

The design model consists of the agent model, service model and acquaintance model based on GAIA methodology. We also introduce the communication specification and data specification, as well as the use case design in next section “detailed design”.

3.1. Agent Model

The purpose of the agent model is to document the various agent types that will be used in the system under development, and the agent instances that will realize these agent types at run-time.

For most cases, there is a one-to-one dependence between roles and the agent types. However, for Personal Assistant Agent, there are two roles in this agent type. It is because the UserHandler role and ActivityManager role are of high degree of independency that we integrate them together into the PAA type. In the same time, we also define a MessageExchangeAgent (MEA) to deal with part of the ActivityManager roles, by which communicate between all agents. MEA is used to provide Directory-Service and Message-Transport-Service for all the other agents, based on the FIPA Abstract Architecture Specification (SC00001L).



Figure 3-1 Agent Model

3.2. Agent Internal Architecture

The Agent consists of the following parts: Listener, Processor, Interpreter, Discovery and Rendering. It is displayed in figure 3-2.

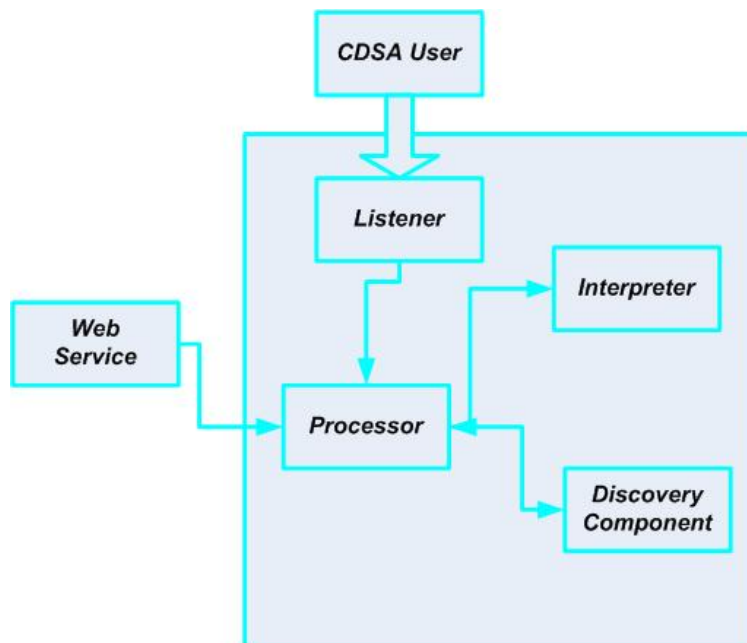


Figure 3-2: Agent Internal Architecture

The Listener is used to listen to a port for any incoming request from CDSA application.

The Interpreter parses and interprets the XML message. We assume that all the agents

have agreed on a Document Type Definition.

The Processor receives an XML document as input. It uses the Interpreter to parse the document and calls the specific function to run a process.

The Discovery component is used to provide the service discovery base-service.

3.3. Service Model

The service model identifies the main services that will be associated with each agent type based on GAIA methodology.

3.3.1. Service of UserHandler role

Wait request

Input: Null

Output: UserHandler are available

Pre-condition: true

Post-condition: true

Obtain requests

Input: request in {get module, get test result, get test criteria, get document, get document template, get schedule, update module, update criteria, update document template, upload module, upload test result, upload document}

Output: request is received by UserHandler.

Pre-condition: true

Post-condition: true

Send requests to MEA

Input: request in {get module, get test result, get test criteria, get document, get document template, get schedule, update module, update criteria, update document template, upload module, upload test result, upload document}

Output: request is sent to MEA and MEA receives the request

Pre-condition: MEA is available

Prot-condition: true

Inform user

Input: Null

Output: request result in {model, test result, test criteria, document, template, schedule}

Pre-condition: MEA return result to UserHanler

Post-condition: true

3.3.2. Service of MEA Role

Receive Request

Input: request in request in {get module, get test result, get test criteria, get document, get document template, get schedule, update module, update criteria, update document template, upload module, upload test result, upload document}

Output: Request is received by MEA

Pre-condition: MEA is available.

Pos-condition: true

Send request to QCA

Input: request in {get test result, get test criteria, update test criteria, upload test result}

Output: Request is received by QCA

Pre-condition: QCA is available

Post-condition: true

Send request to DocumentA

Input: request in {get document, get document template, update document template, upload document}

Output: Request is received by DocumentA

Pre-condition: DocumentA is available

Post-condition: true

Send request to SA

Input: request is getting schedule

Output: Request is received by SA

Pre-condition: SA is available

Post-condition: true

Send request to DesignA

Input: Request in {get module, update module, upload module}

Output: Request is received by DesignA

Pre-condition: DesignA is available

Post-condition: true

Return request result to PAA

Input: NULL

Output: request result in {model, test result, test criteria, document, template, schedule}

Pre-condition: Agent in {QCA, DocumentA, DesignA, SA} return result to MEA

Post-condition: NULL

3.3.3. Service of Design Agent Role

Get module from DB

Input: Module name and version

Output: requested module

Pre-request: DesignA is available, and module name and version are right

Post-request: requested module is not NULL

Update blackboard

Input: Module name and version

Output: blackboard is updated

Pre-request: DesignA is available

Post-request: true

Upload module into DB

Input: Module name and version

Output: module is uploaded into DB and blackboard is updated

Pre-condition: DesignA is available, module names and versions are right, and the uploaded module is not NULL

Post-condition: true

Update module

Input: New module name and version, old module name and version

Output: Old module is updated and blackboard is updated

Pre-condition: DesignA is available, module names and versions are right and the new module is different from the old one

Post-condition: true

Return module to MEA

Input: NULL

Output: requested module

Pre-condition: DesignA is available

Post-condition: returned module is not NULL

3.3.4. Service of QualityControl Role

Get test result

Input: test name and version

Output: requested test result

Pre-condition: QCA is available, and test name and version are right

Post-condition: requested test result is not NULL

Upload test result

Input: test name and version

Output: test result is uploaded into DB

Pre-condition: QCA is available, test name and version are right and the uploaded test result is no NULL

Post-condition: true

Update test result

Input: new test name and version, old test name and version

Output: test result is updated

Pre-condition: QCA is available, test names and versions are right, and the new test result is different from the old one

Post-condition: true

Get test criteria

Input: test criteria name and version

Output: requested test criteria

Pre-condition: QCA is available, test criteria name and version is right

Post-condition: requested test criteria is not NULL

Upload test criteria

Input: test criteria name and version

Output: test criteria is uploaded into DB

Pre-condition: QCA is available, test criteria name and version are right, and uploaded test criteria is not null

Post-condition: true

Update test criteria

Input: new test criteria name and version, old test criteria name and version

Output: requested test criteria is updated

Pre-condition: QCA is available, test criteria names and versions are right, and the new test criteria is different from the old one

Post-condition: true

Update the blackboard

Input: input in {test result name and version, test criteria name and version}

Output: blackboard is updated

Pre-condition: QCA is available.

Post-condition: true

Return test result to MEA

Input: NULL

Output: test result

Pre-condition: QCA is available

Post-condition: test result is not NULL

Return test criteria to MEA

Input: NULL

Output: test criteria

Pre-condition: QCA is available

Post-condition: test criteria is not NULL

3.3.5. *Service of Schedule Role*

Get schedule

Input: requested schedule

Output: Schedule

Pre-condition: SA is available

Post-condition: Returned schedule is not NULL

Upload schedule

Input: new schedule name

Output: new schedule is uploaded into database

Pre-condition: SA is available, new schedule name is right, and new schedule is not NULL

Post-condition: true

Update schedule

Input: new schedule name, old schedule name

Output: old schedule is updated

Pre-condition: SA is available, schedule names are right, and old schedule is different from new schedule

Post-condition: true

Update blackboard

Input: schedule name

Output: blackboard is updated

Pre-condition: SA is available and schedule is not NULL

Post-condition: true

Return schedule to MEA

Input: NULL

Output: schedule

Pre-condition: SA is available

Post-condition: schedule is not NULL

3.4. Acquaintance Model

The acquaintance model defines the communication links that exist between agent types. They do not define what messages are sent or when messages are sent. They simply indicate that communication pathways exist. The purpose of an acquaintance model is to

identity any potential communication bottlenecks, which may cause problems at run-time. Through this model, it may be found necessary to revisit the analysis stage and rework system design if there are any issues exposed.

MEA provide the dictionary service to all agent types. For example, if PAA want to communicate with DesA, it will first send message to MEA, MEA search for the DesA information such as the location, address, etc., and then send back to PAA. After receiving the information, PAA contacts DesA directly and exchange information as requirement.

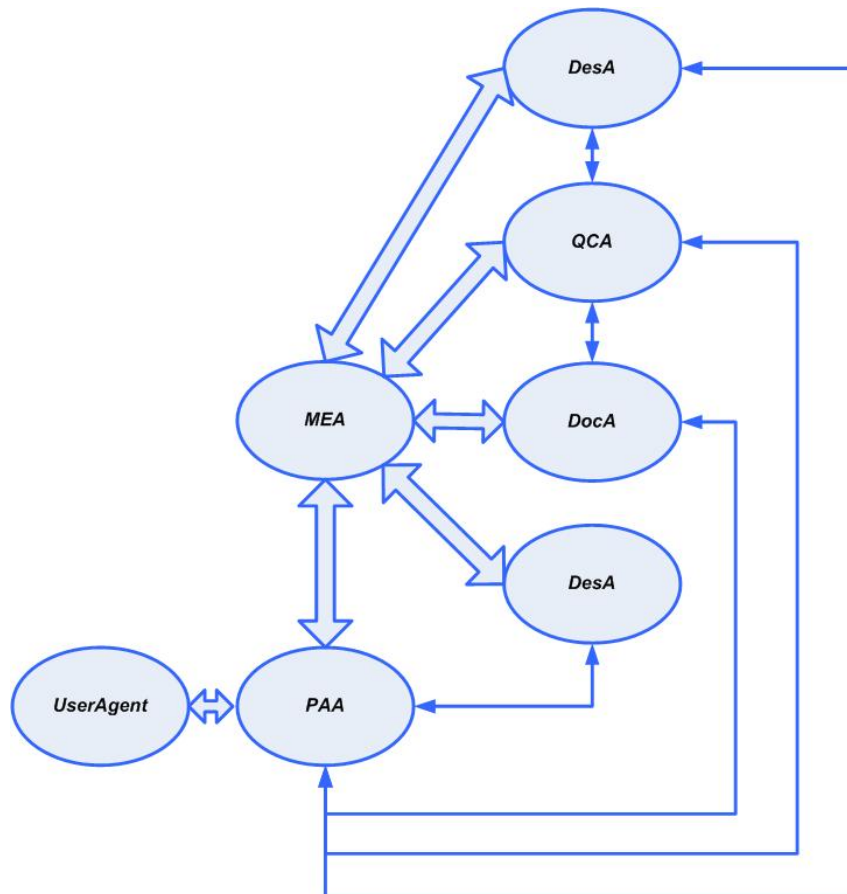


Figure 3-3 Acquaintance Model

4. Detailed Design

4.1. Use Case definition

4.1.1. User Case for PAA

PAA requests service, services includes getting module, getting test result, getting test criteria, getting document, getting document template, getting schedule, updating knowledge base, upload module, uploads test result, updates test result, uploads

document, updates document.

Brief Description:	The actor uses this use case to request assistance	
Pre-condition:	PAA is available	
Post-condition:	The result is returned to the actor if all the rules are successfully met	
Process Steps:		
1	Actor requests to PAA to ask assistance	
2	PAA receives the request and send request to MEA	
3	MEA processes the request and returns the result to PAA	
4	PAA returns the result to actor	
Exceptions		
2a	PAA is not available	Error message is generated stating that the system is not accessible, use case terminates
3a	MEA is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationships:		
Initiating	Actor	
Collaborating	PAA, MEA	
Data Requirements		
Data Required	Data request for PAA:	
	Requested Agent type	
	detailed request	

4.1.2. Use case for MEA

MEA forwards service, services includes getting module, getting test result, getting test criteria, getting document, getting document template, getting schedule, updating knowledge base, upload module, uploads test result, updates test result, uploads document, updates document.

Brief Description:	PAA use this use case to forward the requests to the right agent and return the result to PAA	
Pre-condition:	PAA is available and the corresponding agent is available too	
Pos-condition:	The result is returned to the PAA if all the rules are successfully met	
Process steps:		
1	MEA receives the forwarded service	
2	MEA detects the type of request and send requests to corresponding agent	
3	The agent processes the request and returns the result to MEA	
4	MEA returns the result to PAA	
Exceptions:		
1a	MEA is not available	Error message is generated stating that the system is not accessible, use case terminates
2a	Agent is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationships:		
Initiating	PAA	
Collaborating	MEA, QCA, DesignA, DocumentA and SA	

Data Requirements	
Data Required	Data required for MEA
	Requested Agent type
	Requested assistance

4.1.3. Use case for DesA

Design Agent response the request to post module from users, as well as the requests of fetch module source code and module status.

Brief Description:	DesignA receives the request from MEA which is getting the module from DB, updating module or uploading the module into the DB
Pre-condition:	DesignA is available
Post-condition:	The result is returned to the MEA if all the rules are successfully met
Process steps:	
1	DesignA receives the request from MEA
2	DesignA gets the module from DB if the request is getting the module from DB, updates the module if the request is updating module or upload the module into the DB if the request is uploading the module into the DB
3	DesignA updates the blackboard
4	DesignA returns the result to MEA, the result is the requested module and the status if the module is uploaded or updated
Exceptions:	
1a	DesignA is not available Error message is generated stating that the system is not accessible, use case terminates
Relationship:	
Initiating	DesignA
Collaborating	MEA
Data Requirements	
Data Required	Data required for DesignA
	module if the request is uploading the module , or new module and old module if the request is updating the module

4.1.4. Use case for QCA

QCA is used to deal with the request related with the testing process. Users will use this agent to post test result, get test criteria and test result.

(1) QCA with Test Result

Brief description:	QCA receives the request and gets, updates or uploads the test result.
Pre-condition:	QCA is available
Post-condition:	The result is returned to the MEA if all the rules are successfully met
Process steps:	
1	QCA receives the request from MEA

2	QCA gets the test result from DB if the request is getting test result from DB, updates the test result if the request is updating test result or uploads the test result into the DB if the request is uploading test result	
3	QCA updates the blackboard and module status	
4	QCA return the result to MEA, the result is test criteria if the request is getting test criteria or the status if the request is updating knowledge base	
Exceptions		
1a	QCA is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationship:		
Initiating	QCA	
Collaborating:	MEA	
Data Requirement		
Data Required:	Data required for QCA	
	test criteria if the request is updating knowledge base	

(2) QCA with Test Criteria

Brief description:	QCA receives the request and return the test criteria from the knowledge base or updates the knowledge base	
Pre-condition:	QCA is available	
Post-conditon:	The result is returned to the MEA if all the rules are successfully met	
Process steps:		
1	QCA receives the request from MEA	
2	QCA gets module from DB	
3	QCA generates the test criteria from knowledge base depending on the module if the request is getting test criteria or updates the knowledge base using the module and the new test criteria if the request is updating the knowledge base	
4	QCA updates the blackboard and module status	
5	QCA return the result to MEA, the result is test criteria if the request is getting test criteria or the status if the request is updating knowledge base	
Exceptions		
1a	QCA is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationship:		
Initiating	QCA	
Collaborating:	MEA	
Data Requirements		
Data Required:	Data required for QCA	
	test criteria if the request is updating knowledge base	

Brief description:	QCA receives the message from DesignA and return the test criteria from the knowledge base	
Pre-condition:	QCA is available	
Post-conditon:	The result is returned to the MEA if all the rules are successfully met	

Process steps:	
1	QCA receives the request from MEA
2	QCA gets module from DB
3	QCA generates the test criteria from knowledge base depending on the module
4	QCA updates the blackboard
5	QCA return the result to MEA, the result is test criteria
Exceptions	
1a	QCA is not available Error message is generated stating that the system is not accessible, use case terminates
Relationship:	
Initiating	QCA
Collaborating:	DesignA
Data Requirements	
Data Required:	Data required for QCA Message from DesignA

4.1.5. User case for DocA

DocA is used to deal with the request related with the documentation process. User will use this agent to post documentation, get document template and provide modified template as well.

(1) DocA with Document Result

Brief description:	DocumentA receives the request and gets, updates or uploads the document
Pre-condition:	DocumentA is available
Post-condition:	The result is returned to the MEA if all the rules are successfully met
Process steps:	
1	DocumentA receives the request from MEA
2	DocumentA gets the document from DB if the request is getting document from DB, updates the document if the request is updating document or uploads document into the DB if the request is uploading the document into DB
3	DocumentA updates the blackboard and module status
4	DocumentA returns the result to MEA, the result is the document if the request is getting document or the status if the request is uploading or updating the document
Exceptions:	
1a	DocumentA is not available Error message is generated stating that the system is not accessible, use case terminates
Relationship:	
Initiating	DocumentA
Collaborating:	MEA
Data Requirements	
Data Required:	Data required for DocumentA Document if the request is uploading the test result , or new

	document and old document if the request is updating document
--	---

(2) Doc with Documentation Template

Brief description:	DocumentA receives the request and return the document template from the knowledge base or updates the the knowledge base	
Pre-condition:	DocumentA is available	
Post-conditon:	The result is returned to the MEA if all the rules are successfully met	
Process steps:		
1	DocumentA receives the request from MEA	
2	DocumentA gets module from DB	
3	DocumentA generates the document template from knowledge base depending on the module if the request is getting document template or updates the knowledge base using the module and new document template if the quest is updating the knowledge base	
4	DocumentA updates the blackboard	
5	DocumentA returns the result to MEA, the result is document template if the request is getting document template or the status if the request is updating knowledge base	
Exceptions		
1a	DocumentA is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationship:		
Initiating	DocumentA	
Collaborating:	MEA	
Data Requirements		
Data Required:	Data required for DocumentA	
	document template if the request is updating knowledge base	

Brief description:	DocumentA receives the message from QCA and return the document template from the knowledge base	
Pre-condition:	DocumentA is available	
Post-conditon:	The result is returned to the MEA if all the rules are successfully met	
Process steps:		
1	DocumentA receives the request from MEA	
2	DocumentA gets module from DB	
3	DocumentA generates the document template from knowledge base depending on the module	
4	DocumentA updates the blackboard	
5	DocumentA returns the result to MEA, the result is document template if the request is getting document template or the status if the request is updating knowledge base	
Exceptions		
1a	DocumentA is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationship:		
Initiating	DocumentA	
Collaborating:	QCA	

Data Requirements	
Data Required:	Data required for DocumentA
	Message from QCA

4.1.6. Use case for SA

SA is used to deal with request related with Schedule providing. Users will use this agent to get the schedule for current status, or managers have the right to modify the schedule and post into the system.

(1) SA with Database

Brief description:	SA uploads the schedule into the database
Pre-condition	SA is available
Post-condition	The result is returned to MEA if all the rules are successfully met
Process steps:	
1	SA receives the request from MEA
2	SA loads the schedule into DB
3	SA updates the blackboard
4	SA returns the result to MEA, the result is the status if the schedule is successfully loaded or not
Exceptions	
1a	SA is not available Error message is generated stating that the system is not accessible, use case terminates
Relationship:	
Initiating	SA
Collaborating:	MEA
Data Requirements	
Data Required:	Data required for SA
	schedule

(2) SA with Knowledgebase

Brief description:	SA receives the request from the MEA, SA generates the schedule if the request is getting the schedule or updates the knowledge base if the request is updating the knowledge base
Pre-condition:	SA is available
Post-condition:	The result is returned to MEA if all the rules are successfully met
Process steps:	
1	SA receives the request from MEA, the request is getting the schedule or updates the knowledge base
2	SA get module status from DB if the request is getting schedule
3	SA generates the schedule depending on the module status and stores it into the DB if the request is getting schedule or updates the knowledge if the request is updating the knowledge base using the new schedule and the module status
4	SA updates the blackboard
5	SA return the result to MEA, the result is the schedule if the request is getting the schedule or the status if the request is updating the knowledge base

Exceptions		
1a	SA is not available	Error message is generated stating that the system is not accessible, use case terminates
Relationship:		
Initiating	SA	
Collaborating:	MEA	
Data Requirements		
Data Required:	Data required for SA	
	schedule if the request is updating the knowledge base	

4.2. Typical Data Specification

ModuleInfo:

It saves the information for each module which has been posted into the system, which is used for intelligent agents such as the QCA, DocA to capture related attributes and search for the best preference from knowledgebase.

Field	Description	Type
Module_ID	The unique numeric identifier of the product	Long
Module_Au	the author who composes this module	String
Module_V	Module version	Long
Line_Code	the total number of lines for the given module	Long
Line_Excu	the number of lines for executable code	Long
Design_Complexity	The design complexity of a module. it is the metric defined by software engineering domain which used to evaluate the module	long
Essential_Complexity	The essential complexity of a module.it is the metric defined by software engineering domain which used to evaluate the module	long
Halstead_Difficulty	The halstead difficulty metric of a module.it is the metric defined by software engineering domain which used to evaluate the module	long
Major_Fun	1. Server related/2. Agent Related/3. database related	Integer
Design_Lang	1. C/C++; 2. Java; 3. Perl; 4. Python; 5. others	Integer
New	true=first time to post; false=modified version	boolean
Reserved1	reserved field	null
Reserved2	reserved field	null

ModuleStatus:

It is used to record the status for each module, including the testing result, documentation result, etc.

Field	Description	Type
Module_ID	The unique numeric identifier of the product	Long
Module_Au	the author who composes this module	String

Module_V	Module version	Long
TestResult	True=pass; False=failed	Boolean
DocResult	True=doc has done; False=not yet	Boolean
Reserved1	reserved field	null
Reserved2	reserved field	null

TestCri

It is provided by the knowledgebase system after analyzing the attributes of the given module, and then returns back the test criteria for testing engineers.

Field	Description	Type
Module_ID	The unique numeric identifier of the product	Long
Module_Au	the author who composes this module	String
Module_V	Module version	Long
Unit_T	Unit Test, 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Func_T	Functionality test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Neg_T	Negative Test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
CC_T	Code Coverage Test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
PC_T	Path Coverage Test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
RB_T	Risk-Based Test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
RI_T	Reginal Impact Test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
OEU_T	Objective Evidence of Unit Test; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Reserved1	reserved field	null
Reserved2	reserved field	null

DocTemp:

It is provided by the knowledgebase system after analyzing the attributes of the given module, and then returns back the documentation template for document engineers.

Field	Description	Type
Module_ID	The unique numeric identifier of the product	Long
Module_Au	the author who composes this module	String

Module_V	Module version	Long
Rel_Intro	Introduce relationships between the subcomponents; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Inter_Intro	Introduce interactions between the subcomponents; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
High_Intro	gain a high-level understanding of the entire system; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Sep_Intro	describe them in a separate subsection of this section; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Ref_Intro	If this component is very large and/or complex, document its design in a separate document and simply including a reference to it in this section; 0=no need to test, 1=less important, 2=a little important, 3=important, 4=very important	Integer
Reserved1	reserved field	null
Reserved2	reserved field	null

SchInfo:

It is generated by the system to response the Schedule request.

Field	Description	Type
Num_M	How many modules have been posted successfully	Long
Num_T	How many modules have been passed in the test process	Long
Num_D	How many modules have been documented	Long
Sta_D	Status for design team; 0=delay much, 1=delay, 2=on time, 3=ahead of time , 4=ahead of time much	Integer
Sta_T	Status for test team; 0=delay much, 1=delay, 2=on time, 3=ahead of time , 4=ahead of time much	Integer
Sta_Doc	Status for documentation team; 0=delay much, 1=delay, 2=on time, 3=ahead of time , 4=ahead of time much	Integer
Reserved1	reserved field	null
Reserved2	reserved field	null

4.3. Communication Specification

We choose SOAP as the communication protocol. Simple Object Access Protocol (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:

Envelop: it defines a framework used to describe what is in a message and how to process it.

Encoding Rules: it is used for expressing instances of application defined data types

Convention: it is used for representing remote procedure calls and responses.

SOAP is proposed to encode an HTTP header and an XML message, which helps CSDA agents to call and exchange information to each other.

4.3.1. Typical Inter-Agents Messages

The input and output parameters of each function introduced has an XML format. These XML documents map the data structure defined in the Data Dictionary document.

Here we generate the typical Inter-Agents messages for the main functions: GetTestCri, GetDocT, and GetSche.

GetTestCri:

Input: TestCriRequest

```
<TestCriRequest>
  <Header>
    <MessageType>TestCriRequest</MessageType>
  </Header>
  <Body>
    <ModuleInfo>
      <Module_ID>Module_ID</Module_ID>
      <Module_Au>Module_Au</Module_Au>
      <Module_V>Module_V</Module_V>
      <Line_Code>Line_Code</Line_Code>
      <Line_Excus>Line_Excus</Line_Excus>
      <Design_Complexity>Design_Complexity</Design_Complexity>
      <Essential_Complexity>Essential_Complexity</Essential_Complexity>
      <Halstead_Difficulty>Halstead_Difficulty</Halstead_Difficulty>
      <Major_Fun>Major_Fun</Major_Fun>
      <Design_Lang>Design_Lang</Design_Lang>
      <New>New</New>
    </ModuleInfo>
  </Body>
</TestCriRequest>
```

Output: TestCriResponse

```
<TestCriResponse>
  <Header>
    <MessageType>TestCriResponse</MessageType>
  </Header>
  <Body>
    <TestCri>
      <Module_ID>Module_ID</Module_ID>
      <Module_Au>Module_Au</Module_Au>
      <Module_V>Module_V</Module_V>
      <Unit_T>Unit_T</Unit_T>
    </TestCri>
  </Body>
</TestCriResponse>
```

```

    <Func_T>Func_T</Func_T>
    <Neg_T>Neg_T</Neg_T>
    <CC_T>CC_T</CC_T>
    <PC_T>PC_T</PC_T>
    <RB_T>RB_T</RB_T>
    <RI_T>RI_T</RI_T>
    <OEU_T>OEU_T</OEU_T>
  </TestCri>
</Body>
</TestCriResponse>

```

GetDocTemp:

Input: DocTempRequest

```

<DocTempRequest>
  <Header>
    <MessageType>DocTempRequest</MessageType>
  </Header>
  <Body>
    <ModuleInfo>
      <Module_ID>Module_ID</Module_ID>
      <Module_Au>Module_Au</Module_Au>
      <Module_V>Module_V</Module_V>
      <Line_Code>Line_Code</Line_Code>
      <Line_Excus>Line_Excus</Line_Excus>
      <Design_Complexity>Design_Complexity</Design_Complexity>
      <Essential_Complexity>Essential_Complexity</Essential_Complexity>
      <Halstead_Difficulty>Halstead_Difficulty</Halstead_Difficulty>
      <Major_Fun>Major_Fun</Major_Fun>
      <Design_Lang>Design_Lang</Design_Lang>
      <New>New</New>
    </ModuleInfo>
  </Body>
</DocTempRequest>

```

Output: DocTempResponse

```

<DocTempResponse>
  <Header>
    <MessageType>DocTempResponse</MessageType>
  </Header>
  <Body>
    <DocTemp>
      <Module_ID>Module_ID</Module_ID>
      <Module_Au>Module_Au</Module_Au>
      <Module_V>Module_V</Module_V>
      <Rel_Intro>Rel_Intro</Rel_Intro>
      <Inter_Intro>Inter_Intro</Inter_Intro>
      <High_Intro>High_Intro</High_Intro>
      <Sep_Intro>Sep_Intro</Sep_Intro>
      <Ref_Intro>Ref_Intro</Ref_Intro>
    </DocTemp>
  </Body>
</DocTempResponse>

```

GetSche:

Input: ScheRequest

```

<ScheRequest>

```

```

<Header>
  <MessageType>ScheRequest</MessageType>
</Header>
<Body>
  <ProjectInfo>
    <Module_Num>Module_Num</Module_Num>
    <P_Time_Design>P_Time_Design</P_Time_Design>
    <P_Time_Test>P_Time_Test</P_Time_Test>
    <P_Time_Doc>P_Time_Doc</P_Time_Doc>
    <P_Time_Project>P_Time_Project</P_Time_Project>
  </ProjectInfo>
</Body>
</ScheRequest>
Output: ScheResponse
<ScheResponse>
  <Header>
    <MessageType>ScheResponse</MessageType>
  </Header>
  <Body>
    <SchelInfo>
      <Num_M>Num_M</Num_M>
      <Num_T>Num_T</Num_T>
      <Num_D>Num_D</Num_D>
      <Sta_D>Sta_D</Sta_D>
      <Sta_T>Sta_T</Sta_T>
      <Sta_Doc>Sta_Doc</Sta_Doc>
    </SchelInfo>
  </Body>
</ScheResponse>

```

4.3.2. Other Inter-Agent Messages

Get module	
Input parameters	Description
<pre> <module> <moduleID>Long</moduleID> <moduleV>Long</moduleV> </module> </pre>	Module related information
Output parameters	Description

<pre> <module> <moduleID>Long</moduleID> <moduleAu>String</moduleAu> <moduleV>Long</moduleV> <LineCode>Long</LineCode> <LineExcu>Long</LineExcu> <DesignComplexity>Long</DesignComplexity> <EssentialComplexity>Long</EssentialComplexity> <HalsteadDifficulty>Long</HalsteadDifficulty> <MajorFun>Integer</MajorFun> <DesignLang>Integer</DesignLang> <New>Boolean</New> <Reserved1>String</Reserved1> </module> </pre>	List the requested module, and reserved field stores the module
--	---

Upload module	
Input parameters	Description
<pre> <module> <moduleID>Long</moduleID> <moduleAu>String</moduleAu> <moduleV>Long</moduleV> <LineCode>Long</LineCode> <LineExcu>Long</LineExcu> <DesignComplexity>Long</DesignComplexity> <EssentialComplexity>Long</EssentialComplexity> <HalsteadDifficulty>Long</HalsteadDifficulty> <MajorFun>Integer</MajorFun> <DesignLang>Integer</DesignLang> <New>Boolean</New> <Reserved1>String</Reserved1> </module> </pre>	Module related information, reserved field stores the module
Output parameter	Description
<pre> <status> <moduleID>Long</moduleID> <moduleAu>String</moduleAu> <moduleV>Long</moduleV> <testResult>Boolean</testResult> <docResult>Boolean</docResult> </status> </pre>	module status related information

Get test result	
Input parameters	Description
<pre> <test> <moduleID>Long</moduleID> <moduleV>Long</moduleV> </test> </pre>	test related informaiton
Output parameters	Description

<pre><testResult> <moduleID>Long</moduleID> <moduleV>Long</moduleV> <testRes>Boolean</testRes> </testResult></pre>	test result related information
--	---------------------------------

upload test result	
Input parameters	Description
<pre><testResult> <moduleID>Long</moduleID> <moduleV>Long</moduleV> <testRes>Boolean</testRes> </testResult></pre>	test result related information
Output parameters	Description
<pre><returnResult> <moduleID>Long</moduleID> <moduleV>Long</moduleV> </returnResult></pre>	test result related information

update test criteria	
Input parameters	Description
<pre><testCriterias> <newTestCriteria> <moduleID>Long</moduleID> <moduleAu>String</moduleAu> <moduleV>Long</moduleV> <UnitT>Integer</UnitT> <FuncT>Integer</FuncT> <NegT>Integer</NegT> <CCT>Integer</CCT> <PCT>Integer</PCT> <RBT>Integer</RBT> <RIT>Integer</RIT> <OEUT>Integer</OEUT> </newTestCriteria> <oldTestCriteria> <moduleID>Long</moduleID> <moduleAu>String</moduleAu> <moduleV>Long</moduleV> <UnitT>Integer</UnitT> <FuncT>Integer</FuncT> <NegT>Integer</NegT> <CCT>Integer</CCT> <PCT>Integer</PCT> <RBT>Integer</RBT> <RIT>Integer</RIT> <OEUT>Integer</OEUT> </oldTestCriteria> </testCriterias></pre>	test criteria related information
Output parameters	Description

<pre><testCriteria> <moduleID>Long</moduleID> <moduleAu>String</moduleAu> <moduleV>Long</moduleV> </testCriteria></pre>	test criteria related information
---	-----------------------------------

Update document	
Input parameters	Description
<pre><documents> <newDocument> <moduleID>Long</moduleID> <moduleV>Long</moduleV> <documentRes>Boolean</documentRes> <Reserved1>String</Reserved1> </newDocument> <oldDocument> <moduleID>Long</moduleID> <moduleV>Long</moduleV> <documentRes>Boolean</documentRes> <Reserved1>String</Reserved1> </oldDocument> </documents></pre>	document related information, reserved field stores the new and old documents
Output parameters	Description
<pre><document> <moduleID>Long</moduleID> <moduleV>Long</moduleV> </document></pre>	document related information

update document template	
Input parameters	Description
<pre><documentTemplates> <newDocumentTemplate> <moduleID>Long</moduleID> <moduleV>Long</moduleV> <rellIntro>Integer</rellIntro> <interIntro>Integer</interIntro> <highIntro>Integer</highIntro> <seplIntro>Integer</seplIntro> <refIntro>Integer</refIntro> </newDocumentTemplate> <oldDocumentTemplate> <moduleID>Long</moduleID> <moduleV>Long</moduleV> <rellIntro>Integer</rellIntro> <interIntro>Integer</interIntro> <highIntro>Integer</highIntro> <seplIntro>Integer</seplIntro> <refIntro>Integer</refIntro> </oldDocumentTemplate> </documentTemplates></pre>	document template related information

Output parameters	Description
<pre><documentTemplate> <moduleID>Long</moduleID> <moduleV>Long</moduleV> </documentTemplate></pre>	document template related information

Upload schedule	
Input parameters	Description
<pre><schedule> <scheduleID>Long</scheduleID> <numM>Long</numM> <numT>Long</numT> <numD>Integer</numD> <staDe>Integer</staDe> <staT>Integer</staT> <staDo>Integer</staDo> </schedule></pre>	schedule related information
Output parameters	
<pre><schedule> <scheduleID>Long</scheduleID> </schedule></pre>	schedule related information

5. Conclusions

The development of this project is based on the GAIA methodology for Agent-Oriented analysis and design. We found GAIA is simple, clear and easy to use. However, as for the design part, there are three models needed to be developed which are Agent model, Service model and Acquaintance model. The Service model and the Acquaintance model are a little repeated with the interaction model in the analysis phase. In the design part, the guideline of this methodology is not as good as that in the analysis part. We just added all the information we think it is useful to represent the whole system.

6. Acknowledge

Here we would like to thank Dr. Far for his second lecture, which help us to build up this project using the methodologies for agent-based modeling, analysis and design. We also want to thank all the participants who have given the presentations for the main agent-based methodologies in the last lecture in SENG 697. We have got many useful information and opinions from them.

7. Reference

- [1]. Dr. Behrouz H. Far, Agent-based software Engineering, handouts, University of Calgary, 2006
- [2]. SENG 697, Agent-based Software Engineering, Sample Project: Travel Agency System (TAS), University of Calgary, Fall, 2006-10-28
- [3]. M. Wooldridge, N.R. Jennings, D. Kinny: The Gaia Methodology for Agent-Oriented Analysis and Design, *Autonomous Agents and Multi-Agent Systems*, 3, 285-312, 2000.
- [4]. Ella Belisario, Creating XML files Tutorial, <http://www.eclipse.org/webtools/>, August 18, 2004.
- [5]. UML Distilled, Applying The Standard Object Modeling Language. Martin Fowler

