Software Engineering of Distributed Systems, KTH

Design an Agent Platform conformant to FIPA Specifications

Distributed AI and Intelligent Agents, Project 2007

Shanbo LI, Sike Huang

2007-12-10

Index

1. Complete design of your Agent Platform using GAIA AOSE Methodology

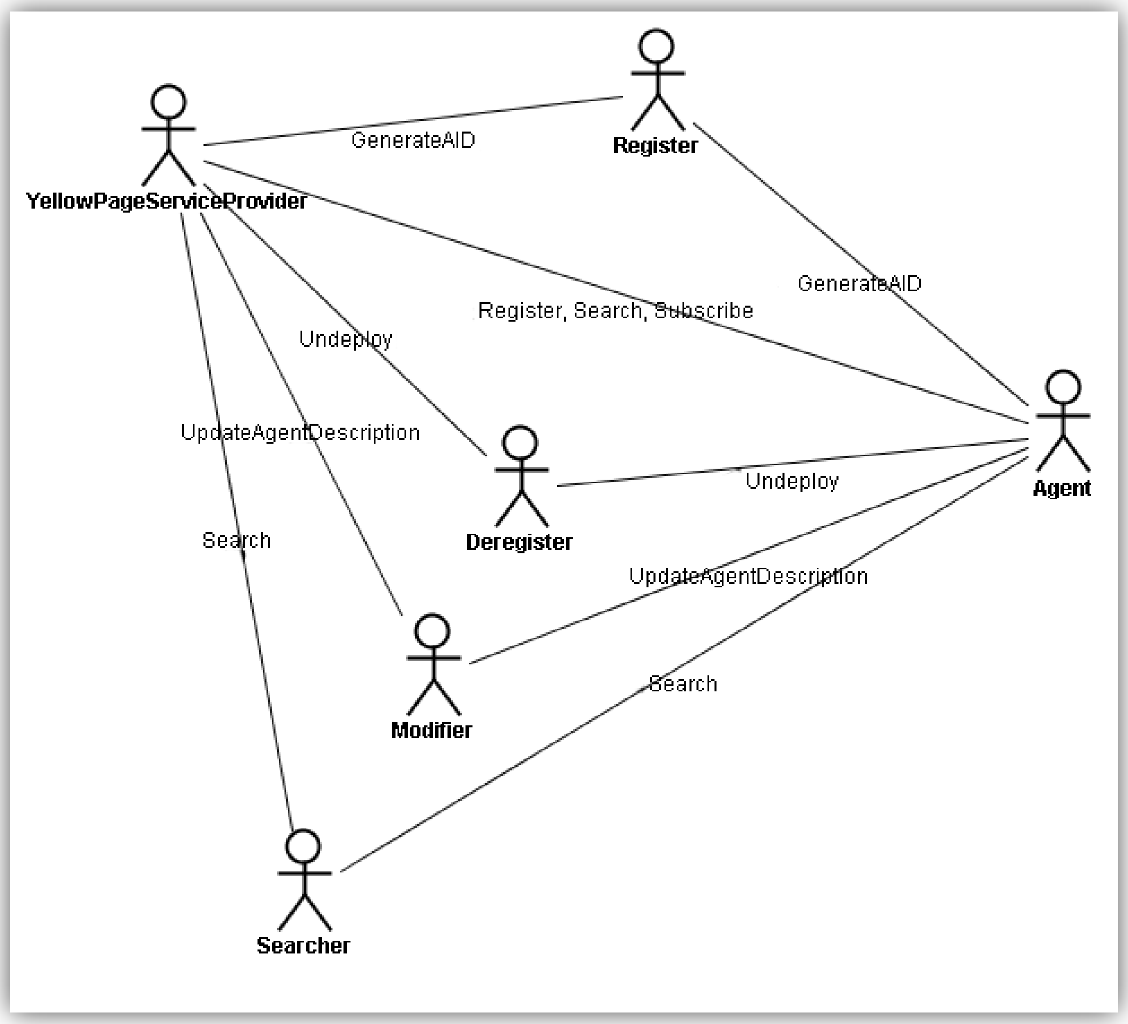
1.1 Role Model

Role	Register
Description	It acts as a register and gives AIDs to agents
Protocols and Activities	GenerateAID
Permissions	
reads	new AgentRequire
generates	AID
Responsibilities	
Liveness	GenerateAID = AgentLocalName@*hap_name*
Safety	true

Role	Modifier
Description	It acts as a modifier and edit agent description
Protocols and Activities	UpdateAgentDescription
Permissions	
reads	AID
generates	description
Responsibilities	
Liveness	newDescription = Description • update
Safety	true

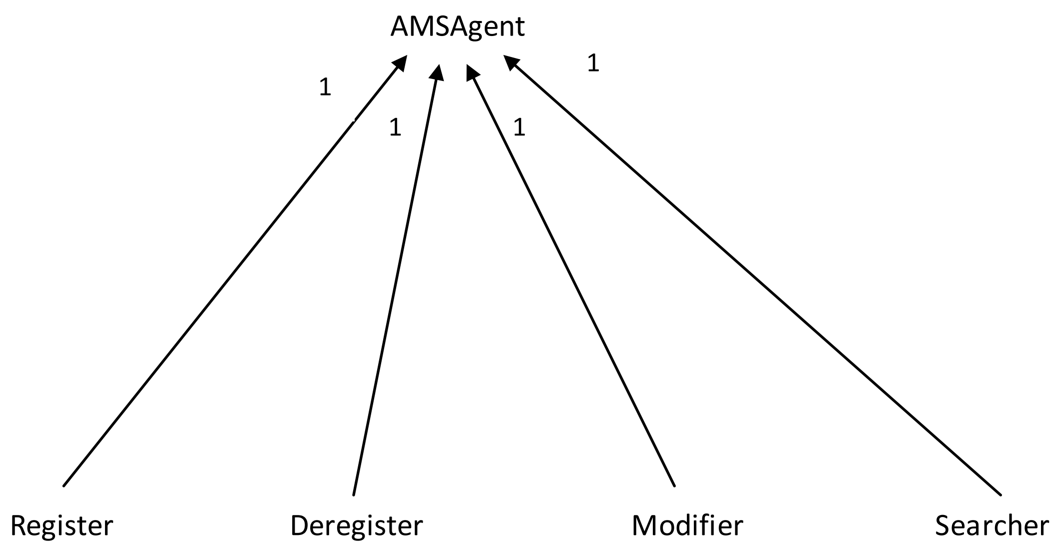
Role	YellopageServiceProvier
Description	It behaves like a yellow page
Protocols and Activities	Register, Search, Subscribe
Permissions	
reads	agentDescription
generates	searchResult
Responsibilities	
Liveness	GenerateResult = SearchRegistedAgent • filterAgentDescription
Safety	true

1.2 Interaction Model

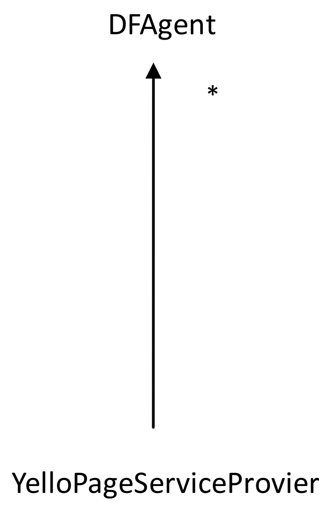


We introduce a normal agent besides DF and AMS, since an agent communicates with both AMS and DF. YellowPageServiceProvider obtains its AID from Register, and asks Modifier to update its description as Directory Facilitator. A normal agent in the agent system registers to the YellowPageServiceProvide, more it can search other agents based on description or subscribe the notification to certain descriptions, if a agent searches another agent using a AID, the call directly goes to the Searcher, the agent obtains its AID from Register and it can update its description via Modifier.

1.3 Agent Model



Agent Management System agent plays four different roles, register, deregister, modifier and searcher. Register registers an agent with its local name to agent platform, and gives the agent a global unique identifier. Deregister deregisters an agent and releases its global unique identifier assigned previously. Modifier modifies the agent description and agent service description. Searcher matches the requesting description to current stored ones.

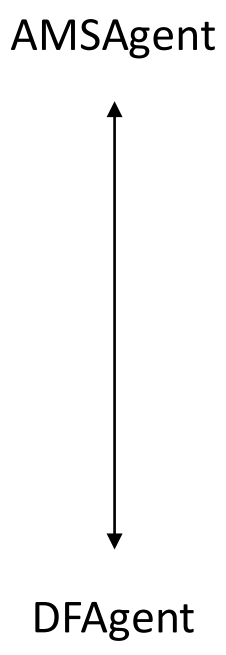


DFAgent plays only one role, which is the YellowPageServiceProvider.

1.4 Service Model

Service	Inputs	Outputs	Pre-condition	Post-condition
GenerateAID		*AID*	*no same localName*	*true*
UpdateAgentDescription	*AID*	*Description*	*AID ≠ nil*	*true*
Register	*AID*		*true*	*true*
Search	*ServiceDescription*	*SearchResult*	*true*	*true*
Subscribe	*ServiceDescription*	*SubscribeResult*	*true*	*true*

1.5 Acquaintance Model



AMSAgent and DFAgent negotiate with each other directly if necessary.

2. Represent agent interaction protocols in UML.

2.1 Level 1: Representing the Overall Protocol

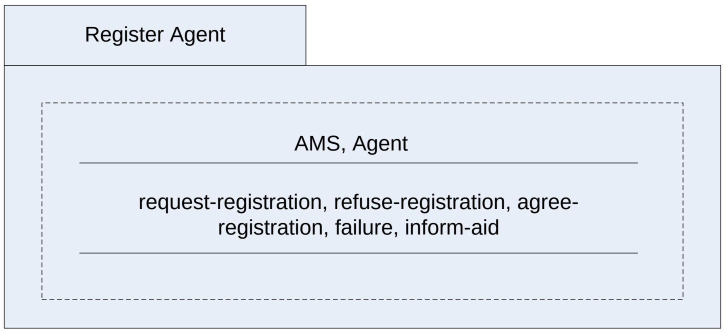
2.1.1. Templates

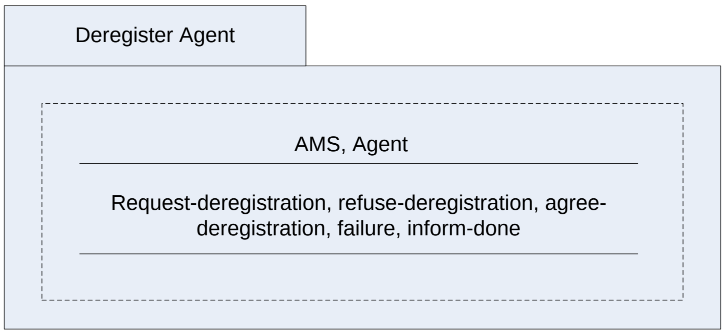
A template is a parameterized model element whose parameters are bound at model time. We consider four templates from FIPA interaction protocol specifications, which are:

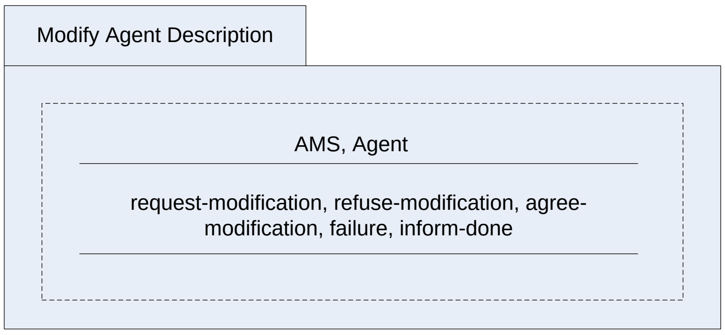
- FIPA Request Interaction Protocol
  - http://www.fipa.org/specs/fipa00026/SC00026H.html
  - The FIPA Request Interaction Protocol (IP) allows one agent to request another to perform some action. The Participant processes the request and makes a decision whether to accept or refuse the request. If a refuse decision is made, then "refused" becomes true and the Participant communicates a refuse. Otherwise, "agreed" becomes true.
- FIPA Subscribe Interaction Protocol
  - http://www.fipa.org/specs/fipa00035/SC00035H.html
  - The Initiator begins the interaction with a subscribe message containing the reference of the objects in which they are interested. The Participant processes the subscribe message and makes a decision whether to accept or refuse the query request. If the Participant makes a refuse decision, then "refused" becomes true and the Participant communicates a refuse. Otherwise, "agreed" becomes true.
- FIPA Query Interaction Protocol
  - http://www.fipa.org/specs/fipa00027/SC00027H.html
  - The Initiator requests the Participant to perform some kind of inform action using one of two query communicative acts, query-if or query-ref (see [FIPA00037]). The query-if communication is used when the Initiator wants to query whether a particular proposition is true or false and the query-ref communication is used when the Initiator wants to query for some identified objects. The Participant processes the query-if or query-ref and makes a decision whether to accept or refuse the query request. If the Participant makes a refuse decision, then "refused" becomes true and the Participant communicates a refuse. Otherwise, "agreed" becomes true.
- FIPA Propose Interaction Protocol
  - http://www.fipa.org/specs/fipa00036/SC00036H.html
  - The Initiator sends a propose message to the Participant indicating that it will perform some action if the Participant agrees. The Participant responds by either accepting or rejecting the proposal, communicating this with the accept-proposal or reject-proposal communicative act, accordingly. Completion of this IP with an accept-proposal act (see [FIPA00037]) would typically be followed by the performance by the Initiator of the proposed action and then the return of a status response.

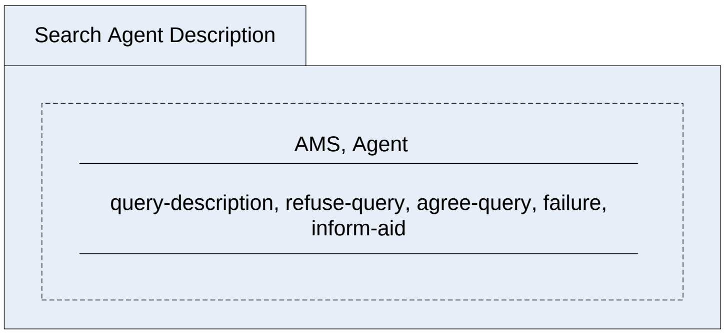
2.2.2. Packages

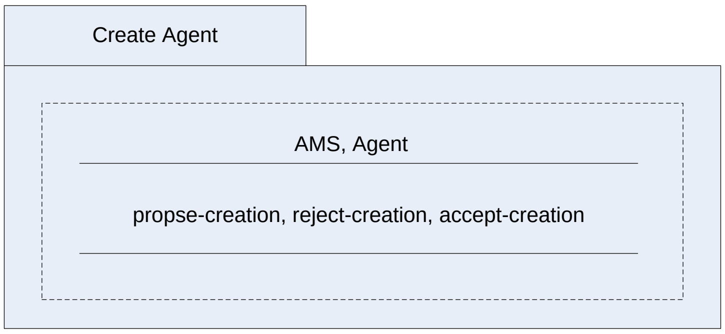
Packages for Agent Management System

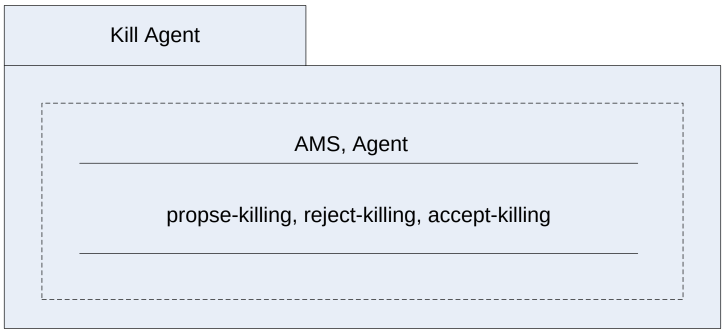

Register Agent
AMS, Agent
request-registration, refuse-registration, agree-registration, failure, inform-aid


Deregister Agent
AMS, Agent
Request-deregistration, refuse-deregistration, agree-deregistration, failure, inform-done

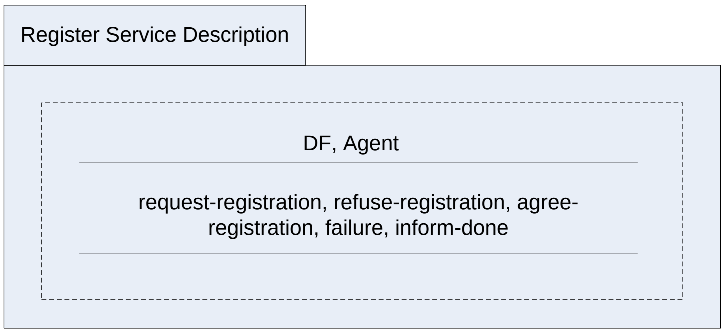

Modify Agent Description
AMS, Agent
request-modification, refuse-modification, agree-modification, failure, inform-done

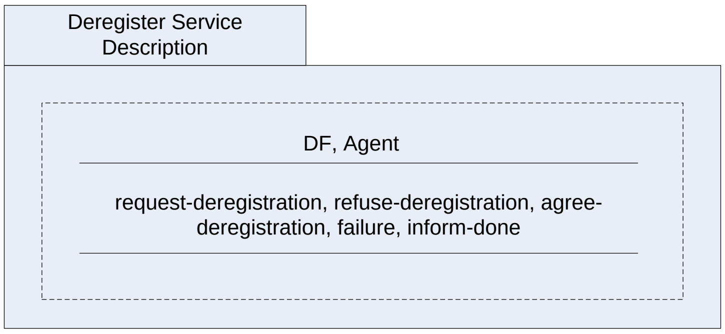

Search Agent Description
AMS, Agent
query-description, refuse-query, agree-query, failure, inform-aid

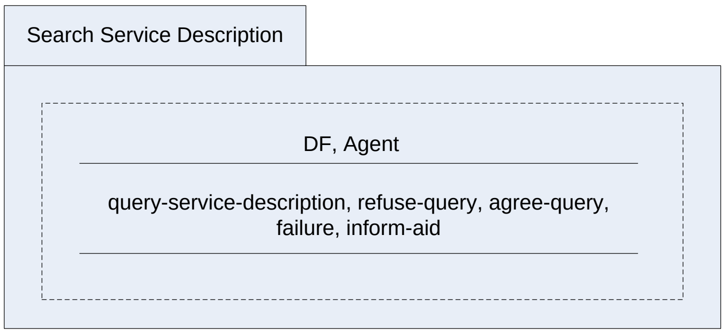


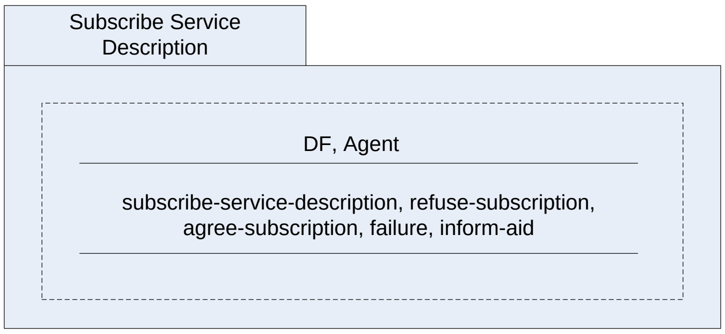


Packages for Directory Facilitator


Register Service Description
DF, Agent
request-registration, refuse-registration, agree-registration, failure, inform-done

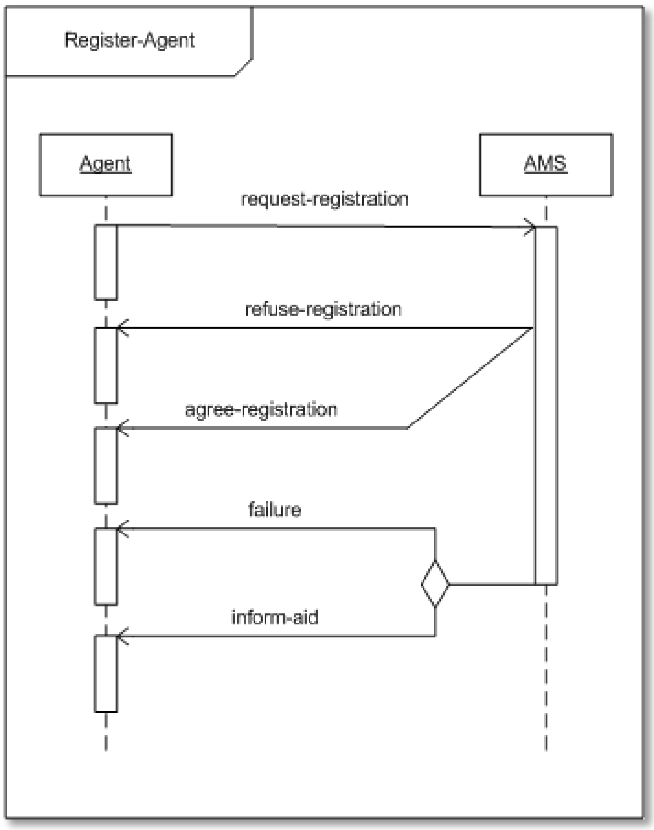

Deregister Service Description
DF, Agent
request-deregistration, refuse-deregistration, agree-deregistration, failure, inform-done


Search Service Description
DF, Agent
query-service-description, refuse-query, agree-query, failure, inform-aid

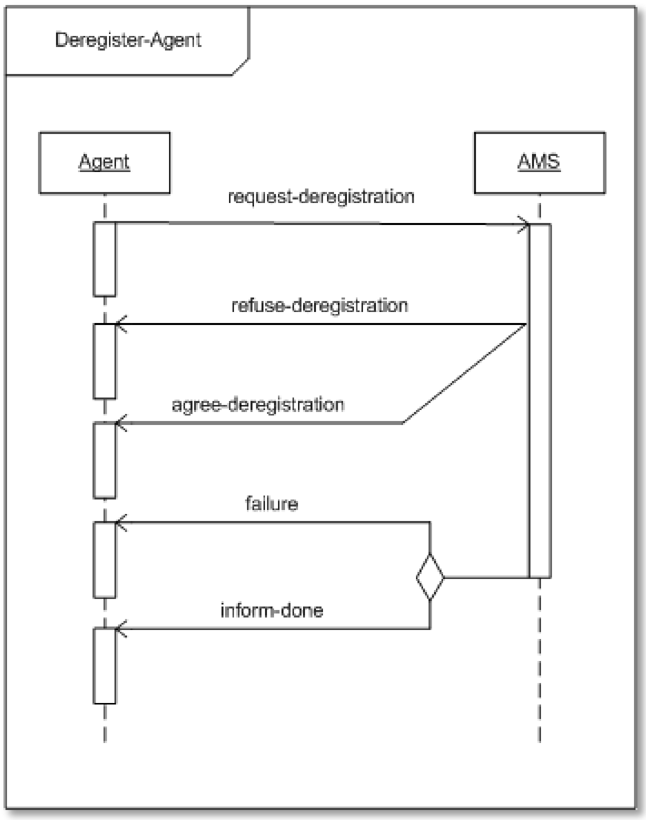

Subscribe Service Description
DF, Agent
subscribe-service-description, refuse-subscription, agree-subscription, failure, inform-aid

2.2 Level 2: Representing Interactions among agents

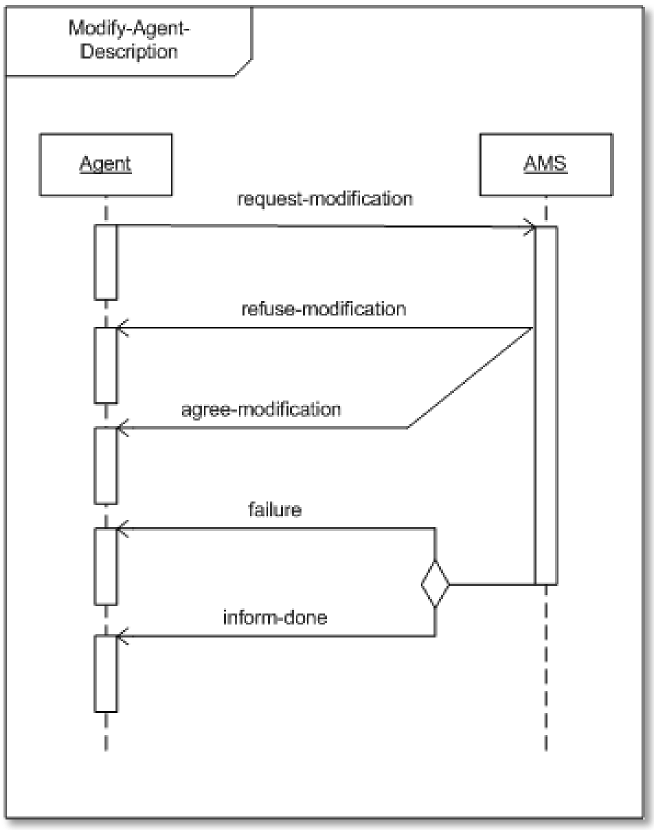
Interaction between AMS and Agent



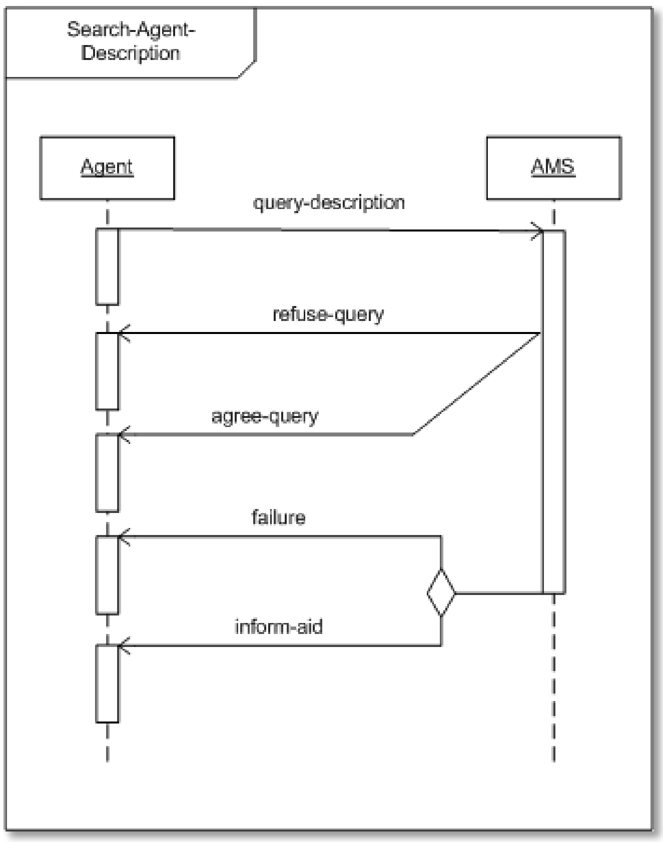
AMS: Register-Agent



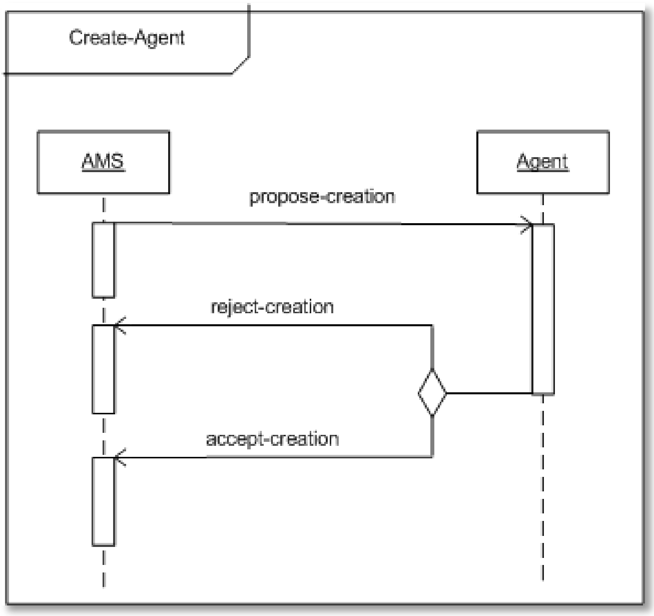
AMS: Deregister-Agent



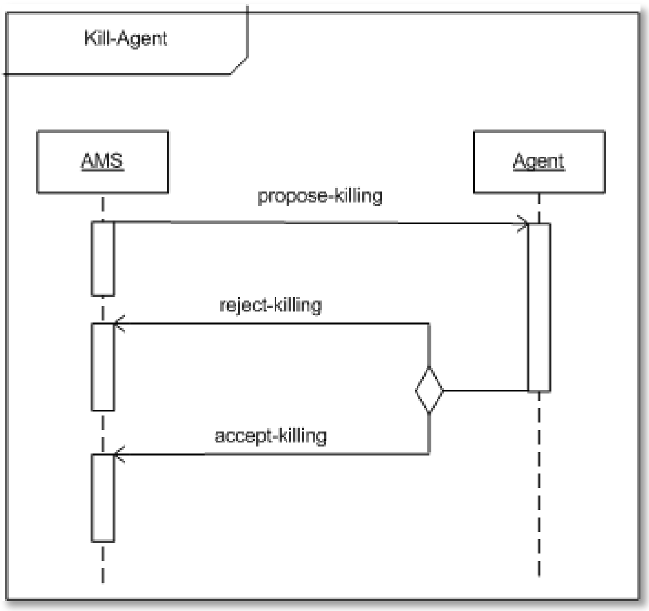
AMS: Modify-Agent-Description



AMS: Search-Agent-Description

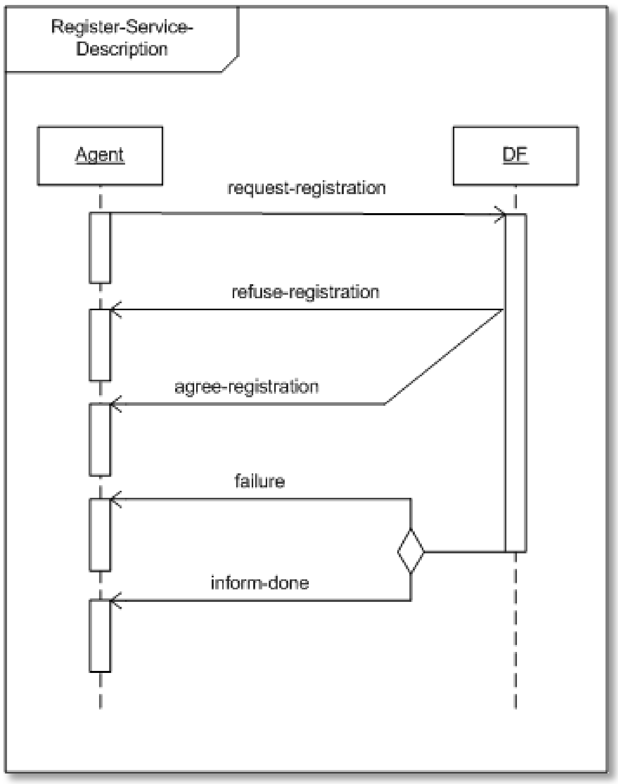


AMS: Create-Agent

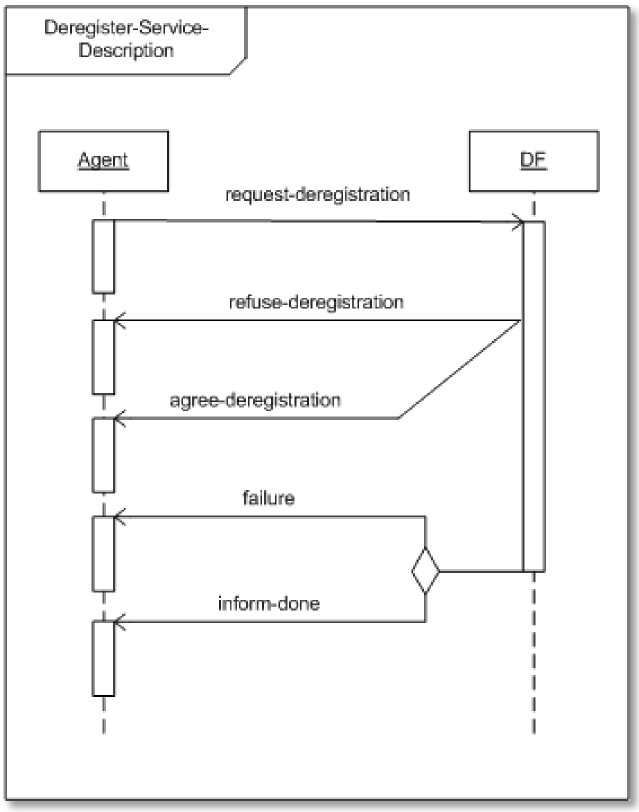


AMS: Kill-Agent

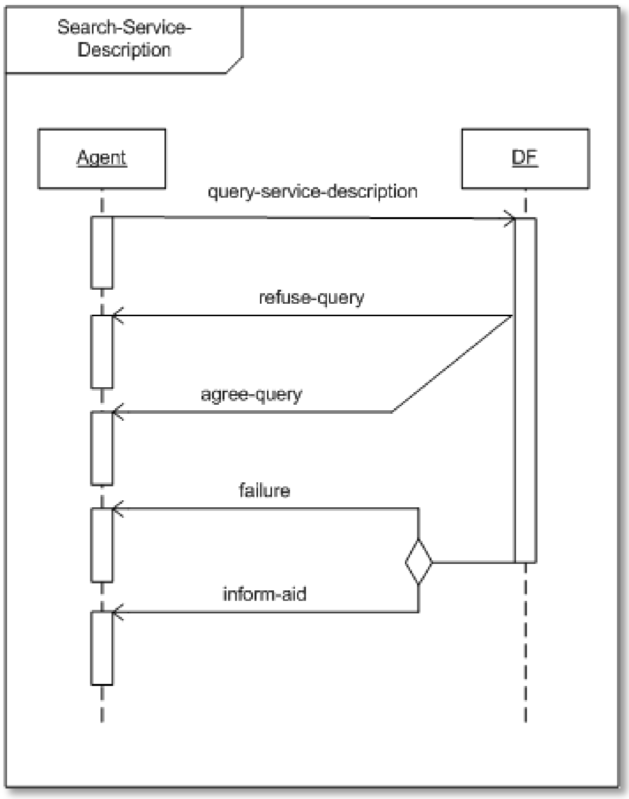
Interaction between DF and Agent



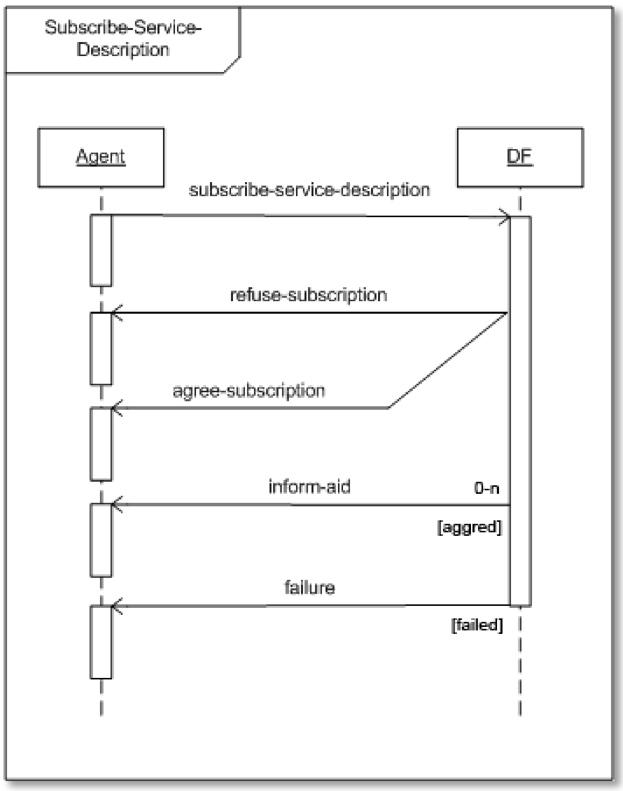
DF: Register-Service-Description



DF: Deregister-Service-Description

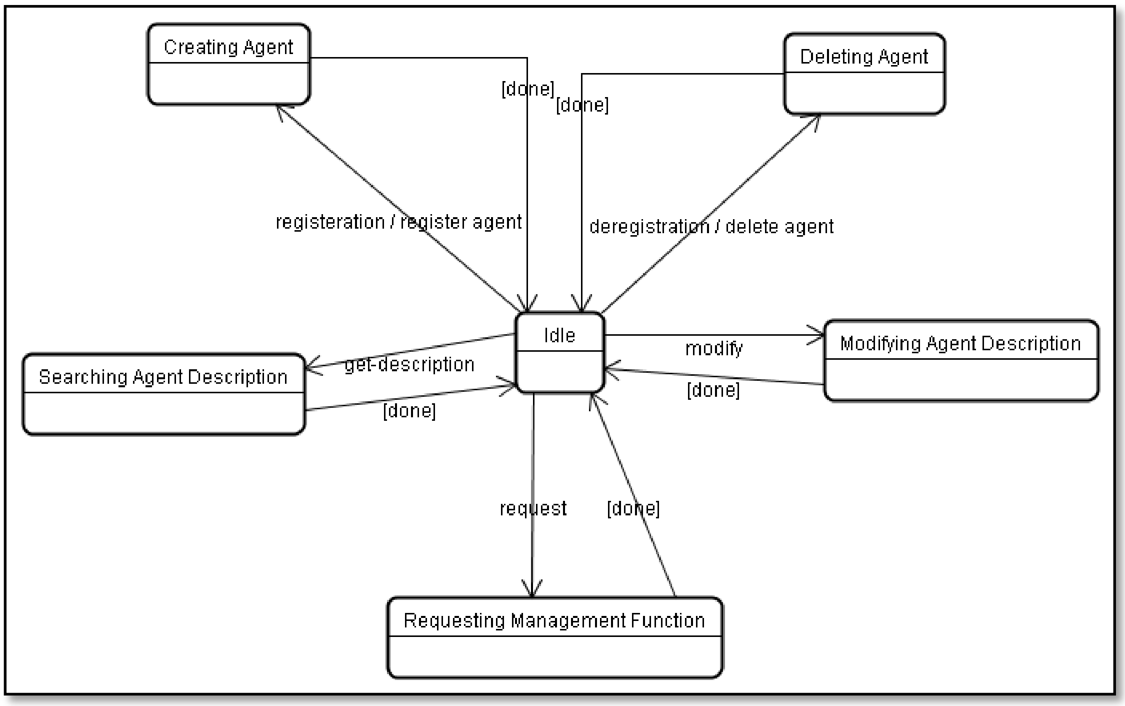


DF: Serach-Service-Description

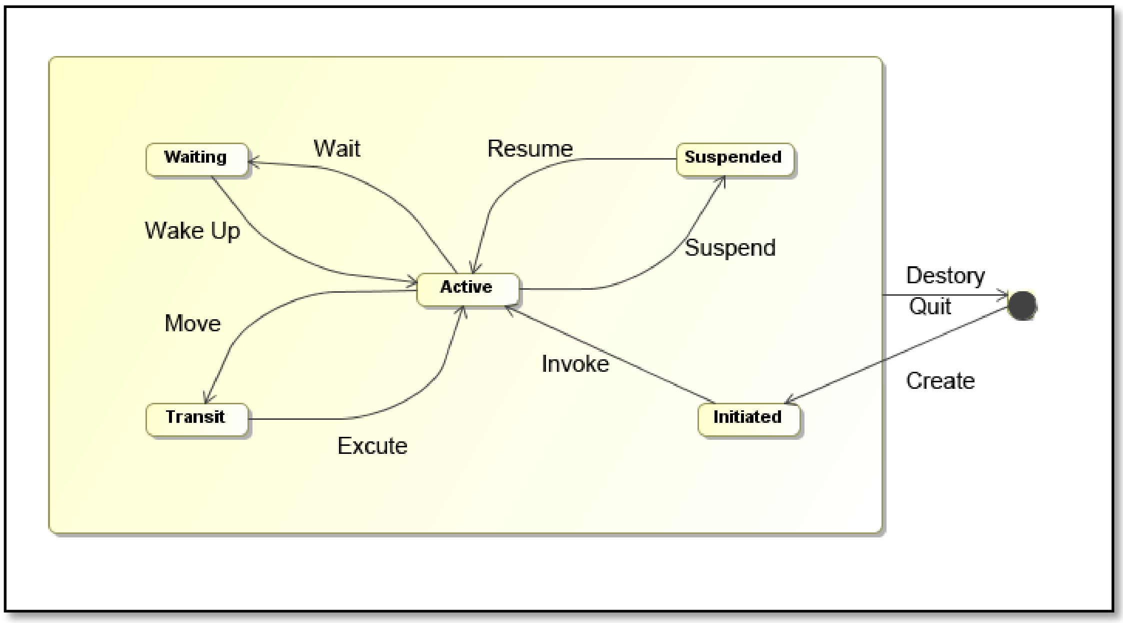


DF: Subscribe-Service-Description

2.2 Level 2: AMS agent representation using State-chart diagrams


Creating Agent
Deleting Agent
[done]
[done]
registeration / register agent
deregistration / delete agent
Idle
modify
Modifying Agent Description
[done]
Searching Agent Description
get-description
[done]
request
[done]
Requesting Management Function

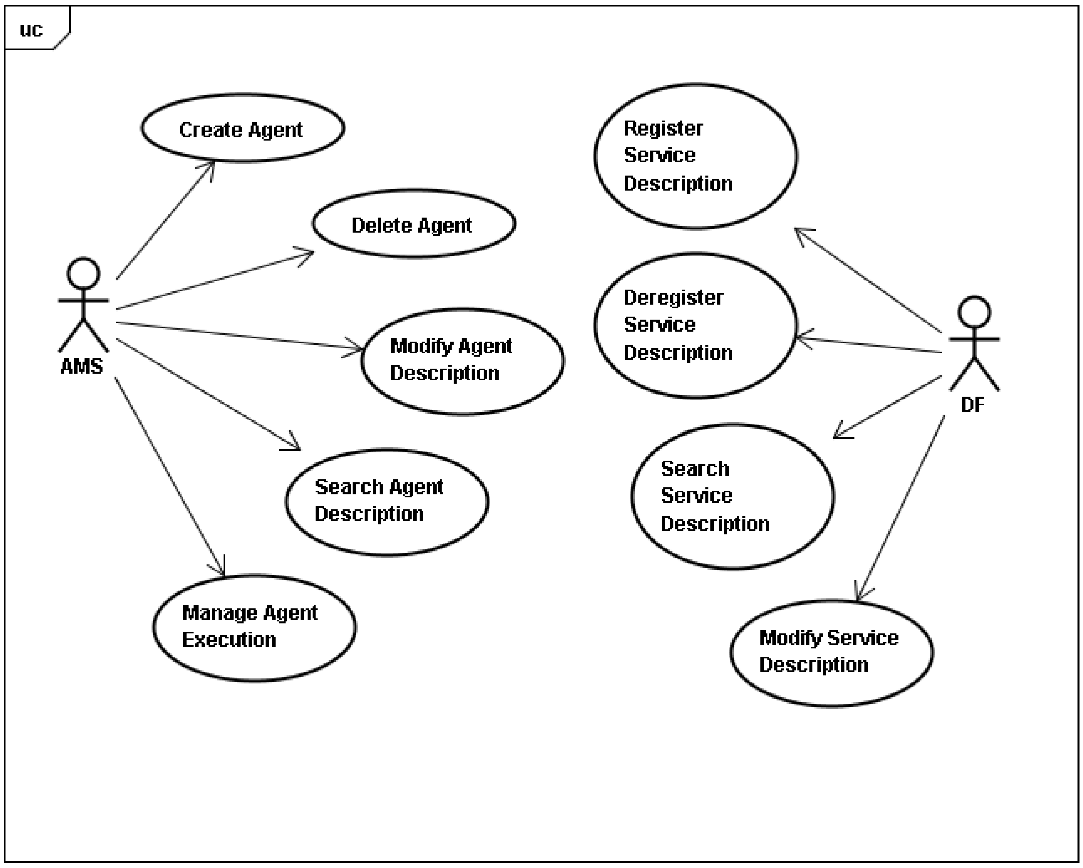
One of the most important state AMS agent has is creating agent, where each agent registers with AMS in order to obtain an AID which is then retained by AMS as a directory of all agents present within the Agent Platform and their current state. Agent descriptions can be modified under restriction of authorization by the AMS. Deregistration causes AMS into the state of deleting agent, where the AID of that agent is removed and can be made available to other agents who should request it. Agent descriptions are maintained in the state of modifying agent description and retrieved by requesting the action get-description. And The AMS can request that an agent performs a specific management function, such as to terminate its execution, and has the authority to enforce the operation if the request is ignored.



Since AMS itself is an agent, it can be in one of several states, according to Agent Platform Life Cycle in FIFA specification. The states are initiated – the agent object is built; active – the agent object is registered; suspended – the agent object is stopped; waiting – the agent object is blocked; deleted – the agent is dead and transit – agent is migrating to the new location.

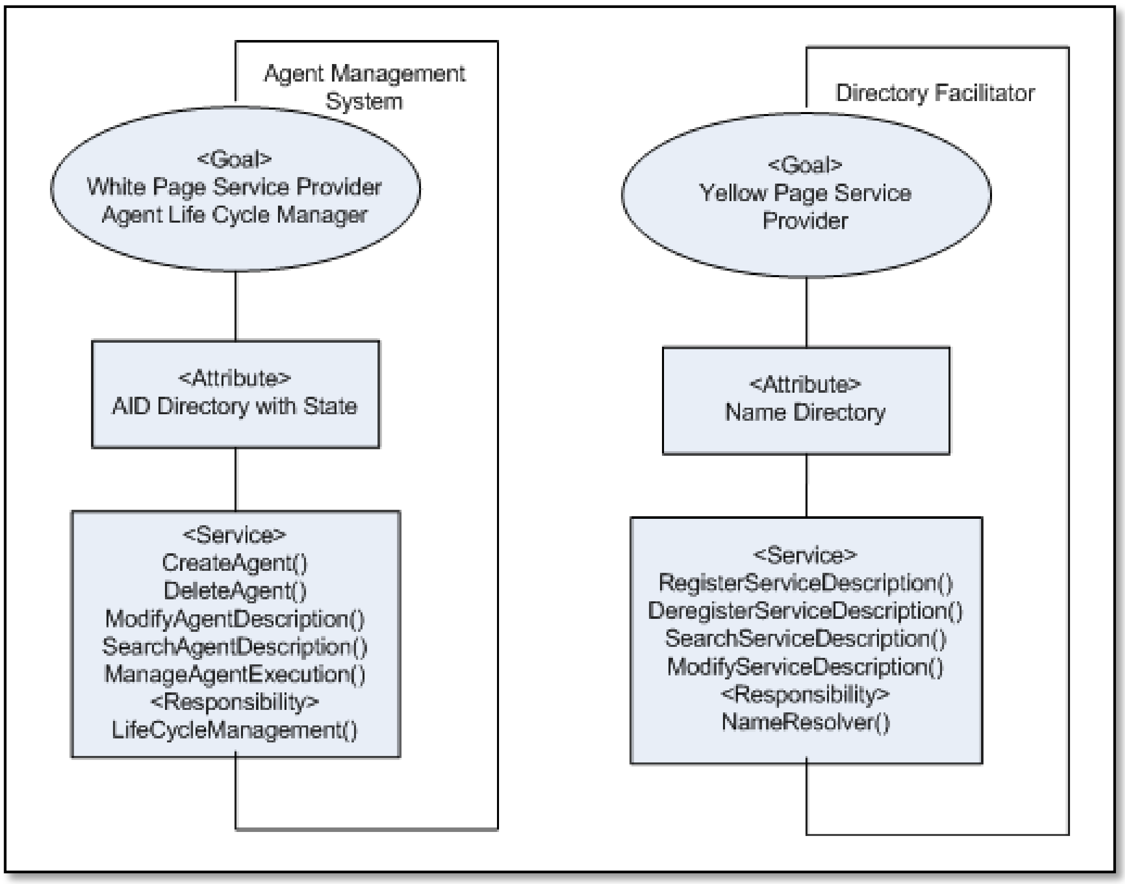
3. Use role-based modeling method to model FIPA specified roles for Agent Platform

3.1. Capture Use Cases


uc
Create Agent
Delete Agent
Modify Agent Description
Search Agent Description
Manage Agent Execution
AMS
Register Service Description
Deregister Service Description
Search Service Description
Modify Service Description
DF

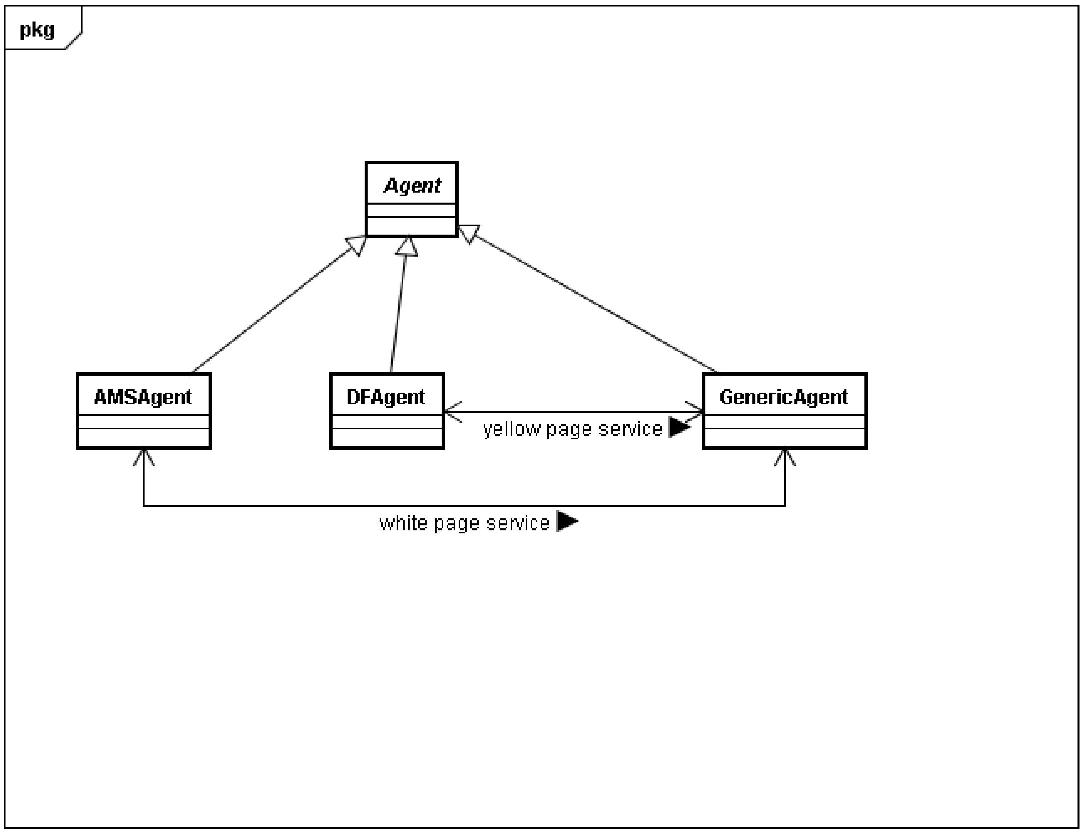
We consider two actors, Agent Management System and Directory Facilitator. AMS includes five use cases: Create Agent and Delete Agent take care of registering and deregistering agent in the agent platform; Agent descriptions are maintained by Use case Modify Agent Description; Agent descriptions can be returned by Search Agent Description; Manage Agent Execution enforces specific management function on an agent. On the other hand, DF keeps track of all service description by four different use cases as listed in the figure.

3.2. Identify Roles


Agent Management System
<Goal> White Page Service Provider Agent Life Cycle Manager
<Attribute> AID Directory with State
<Service> CreateAgent() DeleteAgent() ModifyAgentDescription() SearchAgentDescription() ManageAgentExecution() <Responsibility> LifeCycleManagement()
Directory Facilitator
<Goal> Yellow Page Service Provider
<Attribute> Name Directory
<Service> RegisterServiceDescription() DeregisterServiceDescription() SearchServiceDescription() ModifyServiceDescription() <Responsibility> NameResolver()

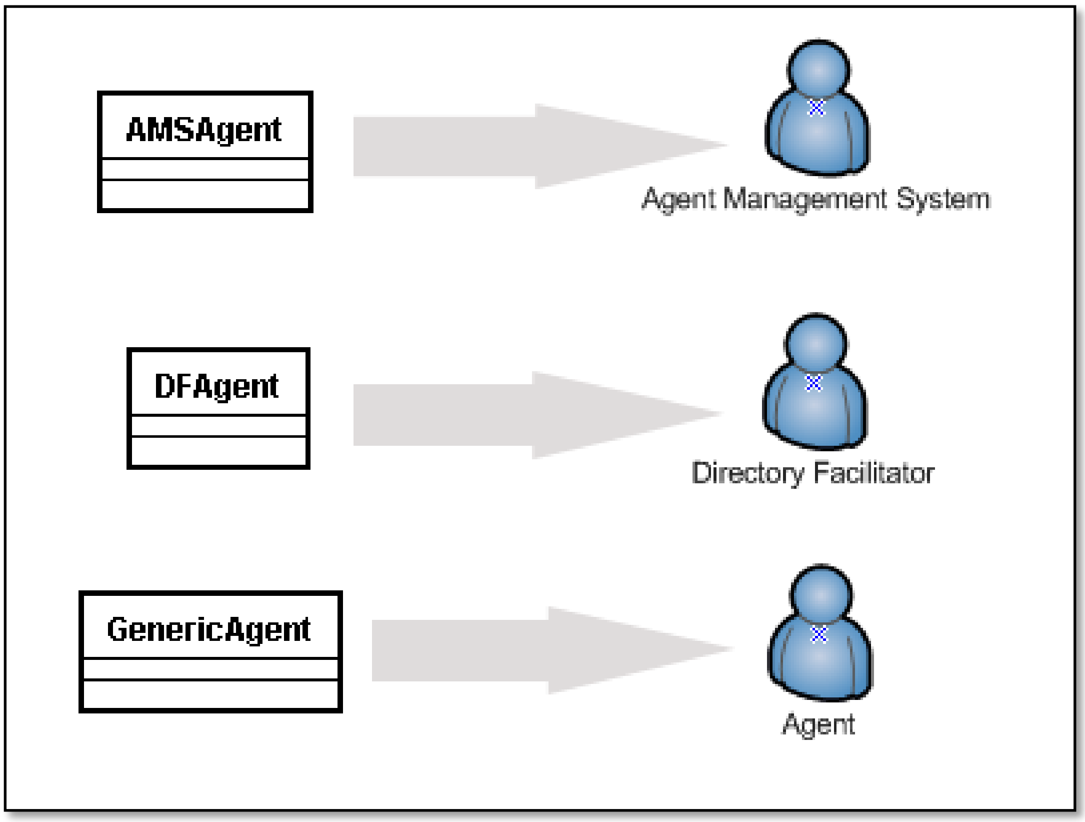
Two roles are identified from the use cases, Agent Management System and Directory Facilitator. Agent Management System provides white pager service and manages the agent life cycle, while Directory Facilitator offers yellow page service and resolves the naming.

3.3 Construct Role Organizations



We introduce a new role called Agent, which is the superset. AMSAgent and DFAgent are two specialized roles inherited from Agent, and GenericAgent is a concrete agent deduced from Agent, which takes other responsibilities other than AMS and DF.

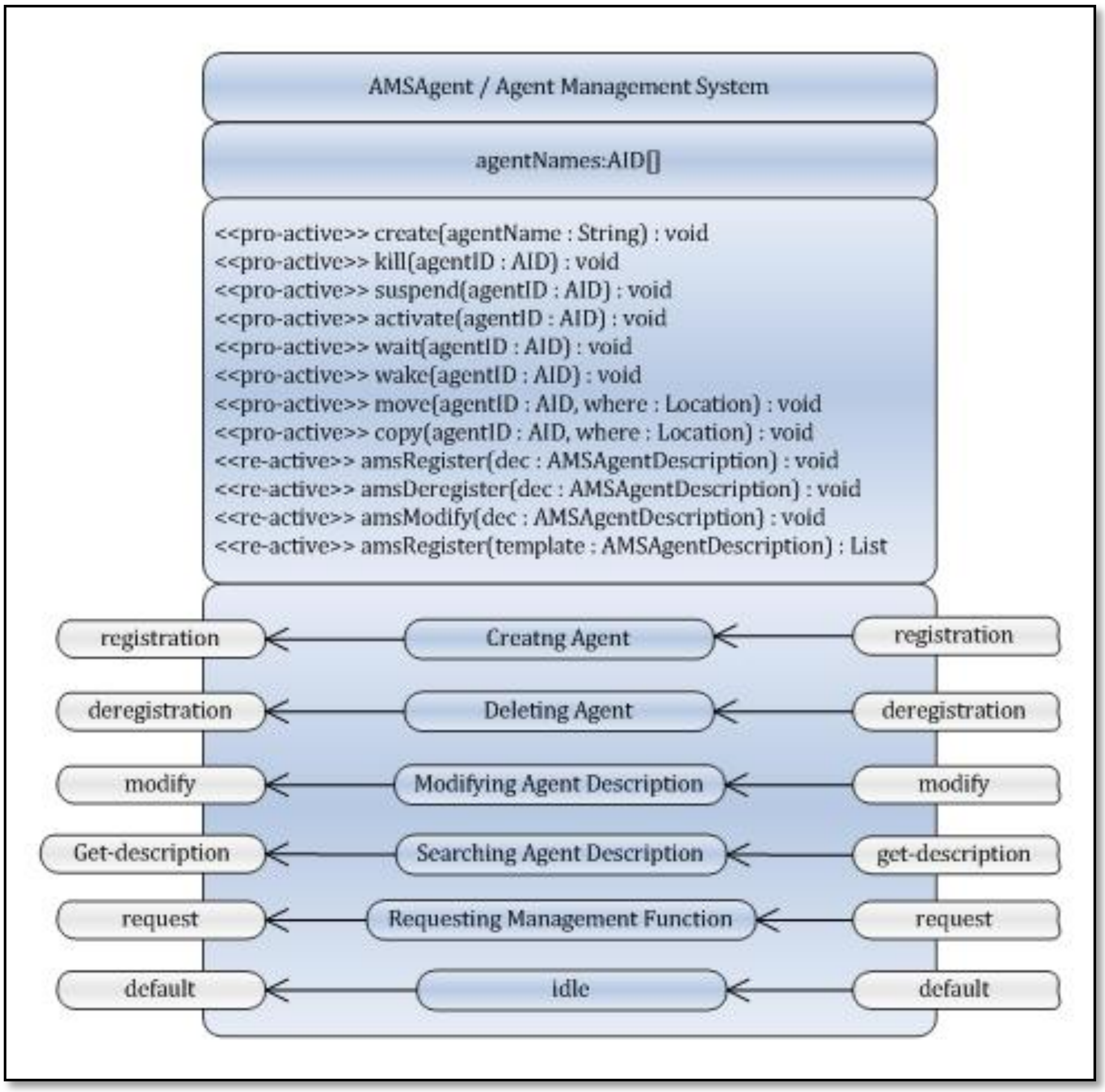
3.4 Bind Roles to Agents



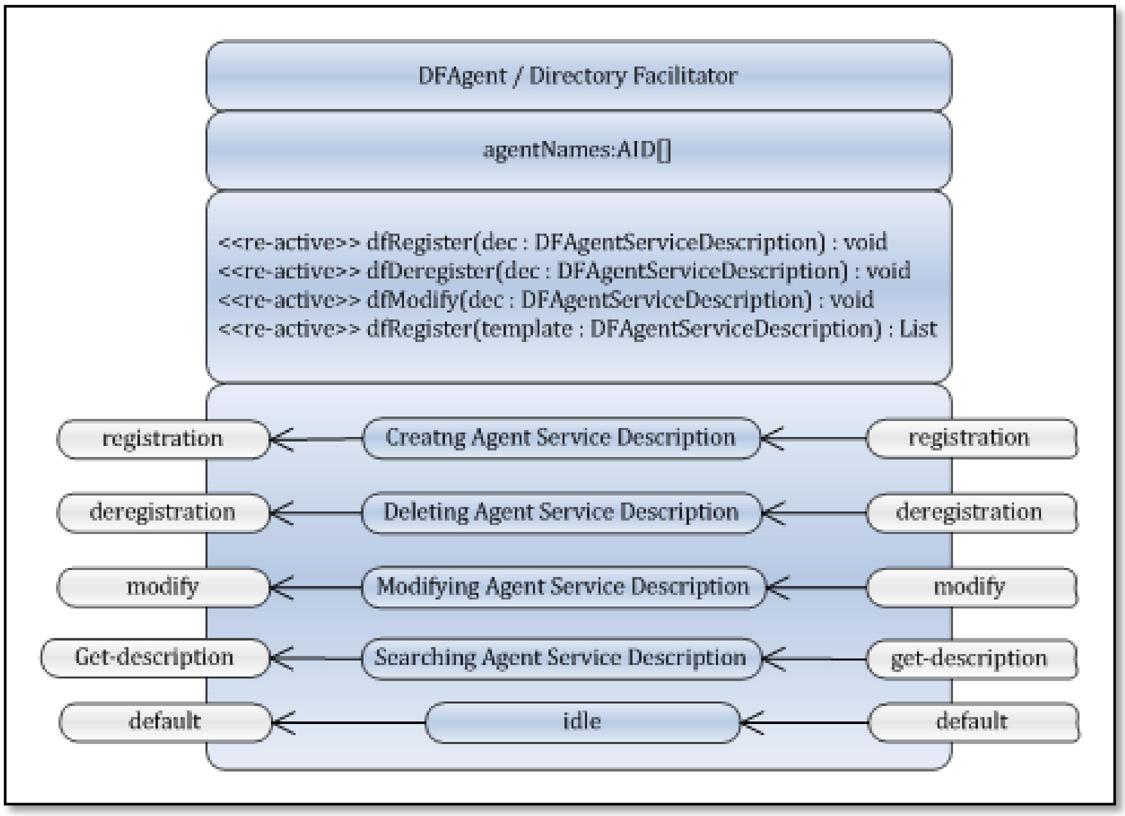
AMSAgent plays the role of Agent Mangement System; DFAgent plays the role of Directory Facilitator; GenericAgent plays the role of Agent.

3.5 Class Design

AMSAgent / Agent Management System


AMSAgent / Agent Management System
agentNames:AID[]
<<pro-active>> create(agentName : String) : void
<<pro-active>> kill(agentID : AID) : void
<<pro-active>> suspend(agentID : AID) : void
<<pro-active>> activate(agentID : AID) : void
<<pro-active>> wait(agentID : AID) : void
<<pro-active>> wake(agentID : AID) : void
<<pro-active>> move(agentID : AID, where : Location) : void
<<pro-active>> copy(agentID : AID, where : Location) : void
<<re-active>> amsRegister(dec : AMSAgentDescription) : void
<<re-active>> amsDeregister(dec : AMSAgentDescription) : void
<<re-active>> amsModify(dec : AMSAgentDescription) : void
<<re-active>> amsRegister(template : AMSAgentDescription) : List
registration
Creatng Agent
registration
deregistration
Deleting Agent
deregistration
modify
Modifying Agent Description
modify
Get-description
Searching Agent Description
get-description
request
Requesting Management Function
request
default
idle
default

DFAgent / Directory Facilitator

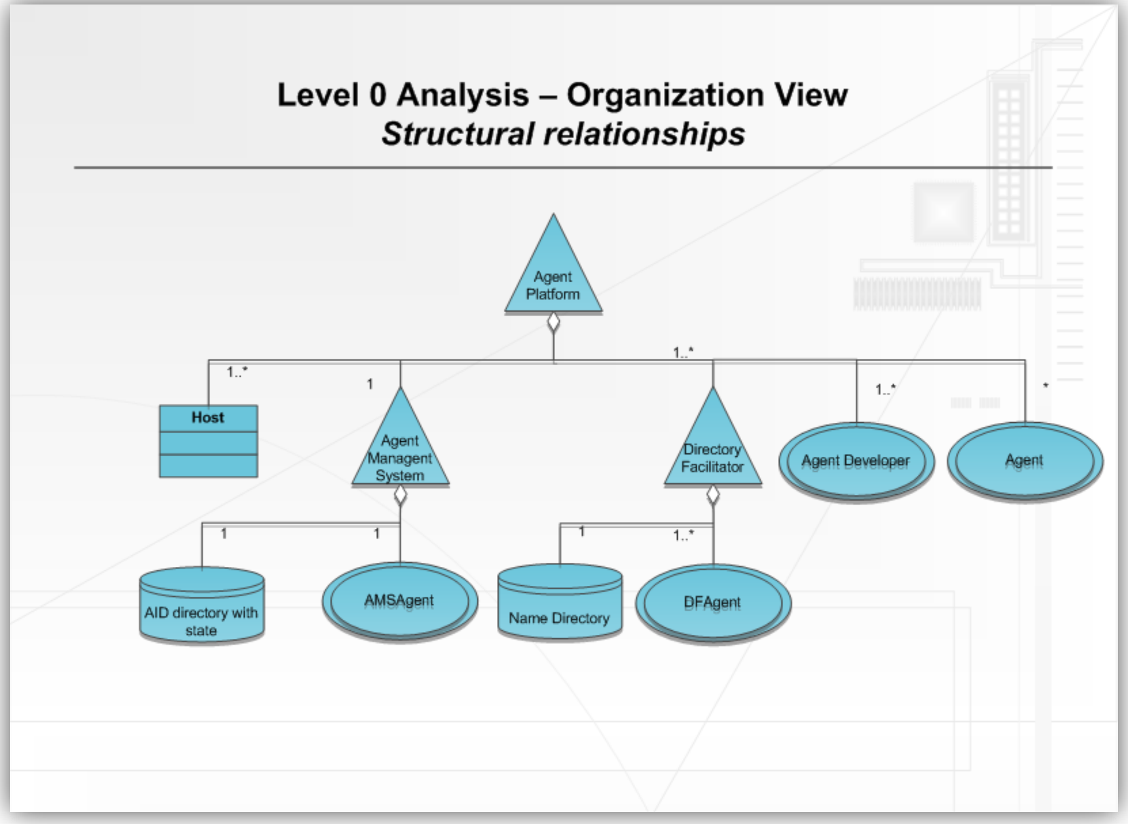

DFAgent / Directory Facilitator
agentNames:AID[]
<<re-active>> dfRegister(dec : DFAgentServiceDescription) : void
<<re-active>> dfDeregister(dec : DFAgentServiceDescription) : void
<<re-active>> dfModify(dec : DFAgentServiceDescription) : void
<<re-active>> dfRegister(template : DFAgentServiceDescription) : List
registration
Creatng Agent Service Description
registration
deregistration
Deleting Agent Service Description
deregistration
modify
Modifying Agent Service Description
modify
Get-description
Searching Agent Service Description
get-description
default
idle
default

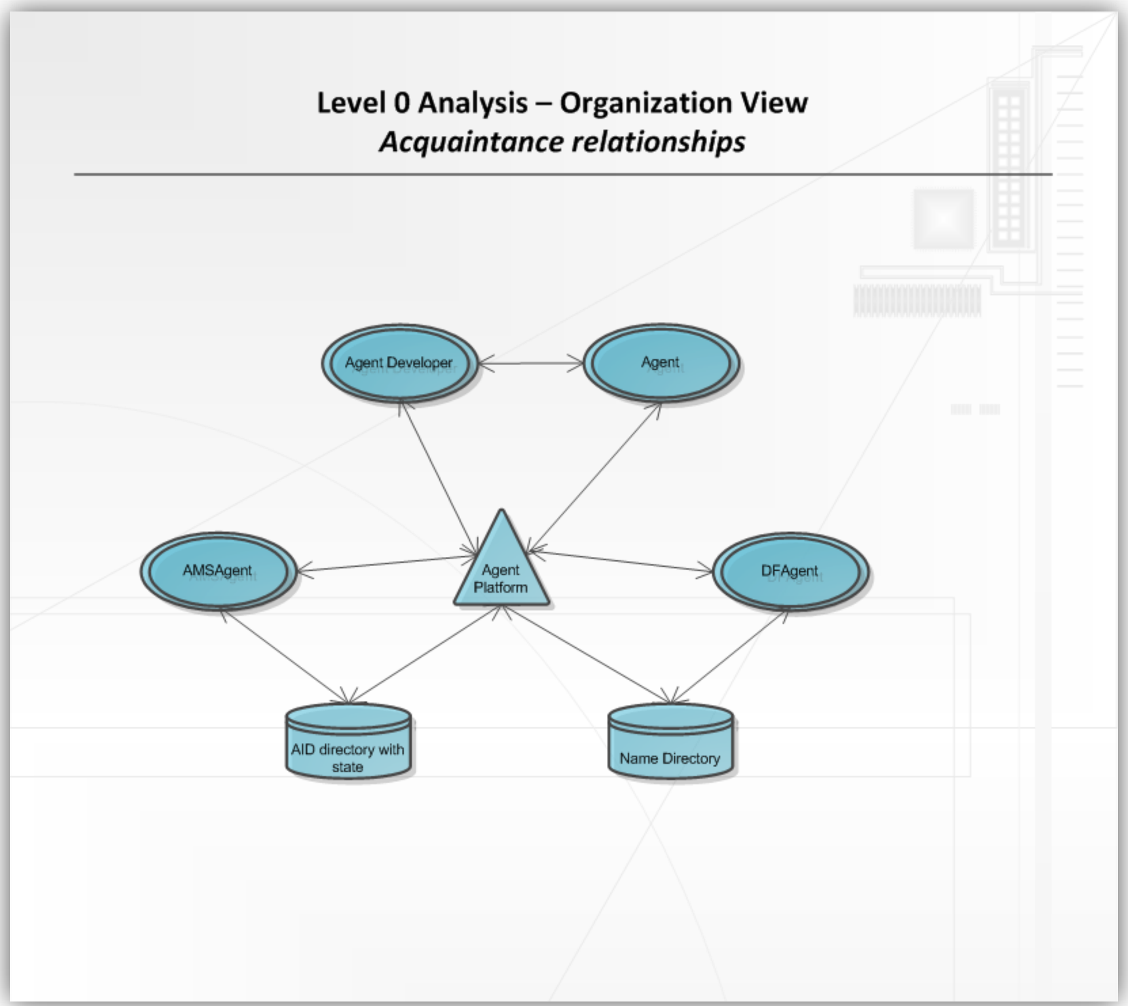
4. Re-modeling the above using MESSAGE UML.

4.1 Level 0 Analysis

4.1.1 Organization view

The *Organization View* shows ConcreteEntities (Agents, Organisations, Roles, Resources) in the system and its environment and coarse-grained relationships between them (aggregation, power, and acquaintance relationships). An acquaintance relationship indicates the existence of at least one Interaction involving the entities concerned.





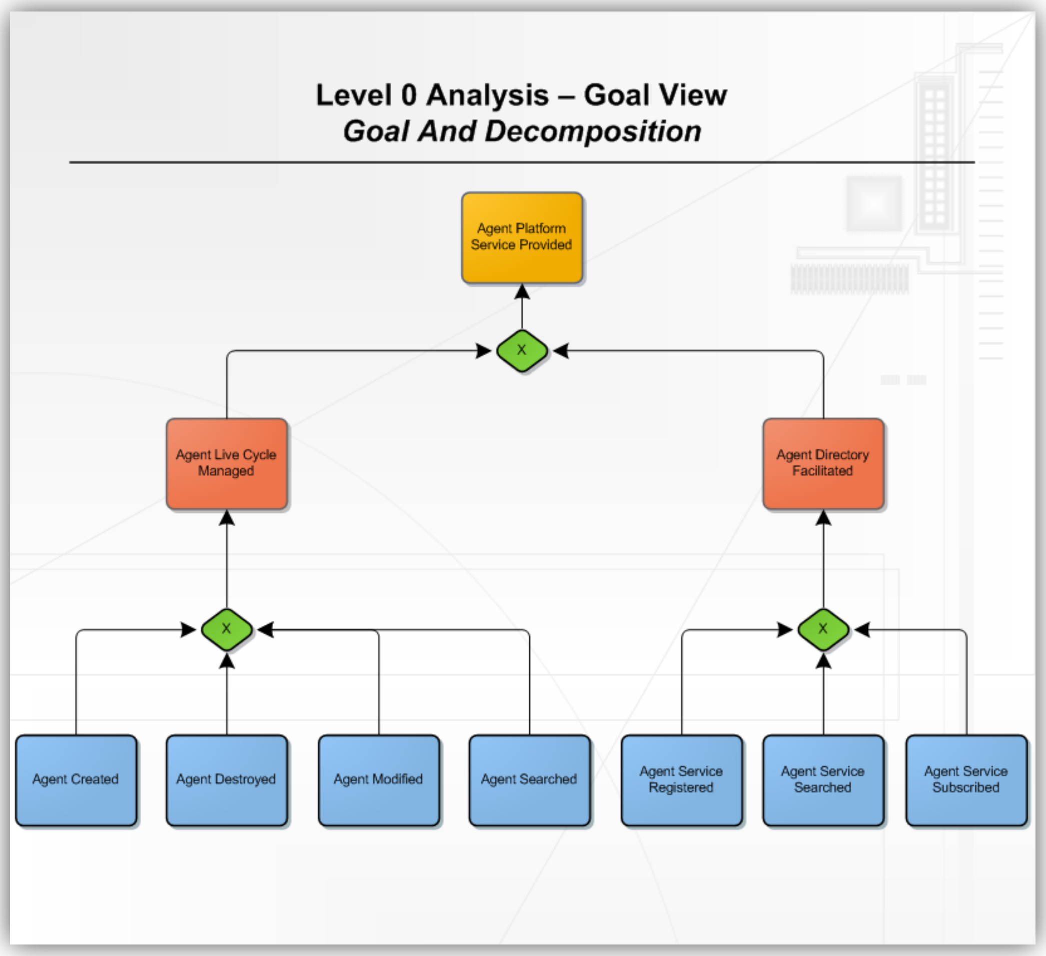
Level 0 Analysis – Organization View

Acquaintance relationships

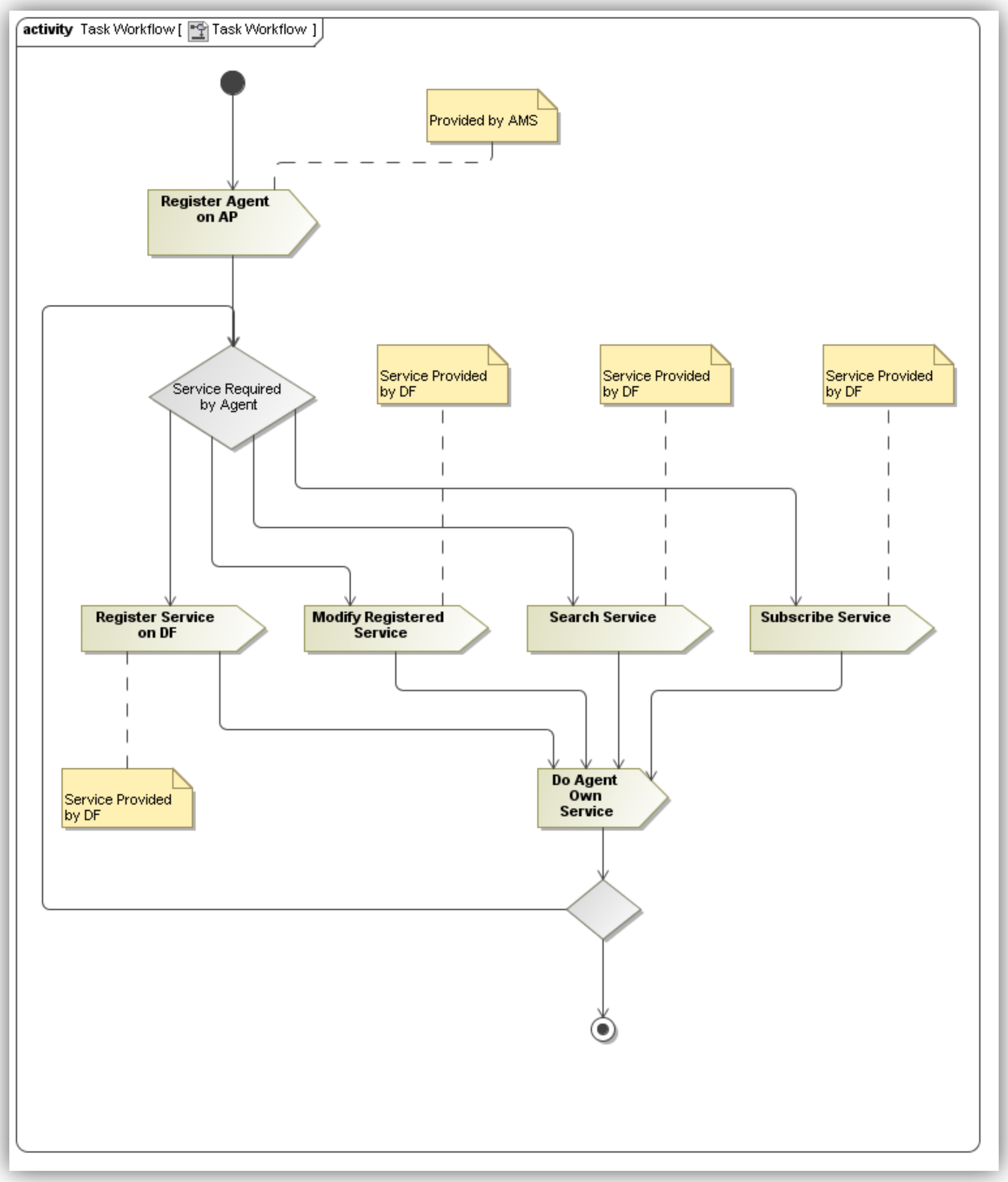
4.1.2 Goal/Task view

This shows Goals, Tasks, Situationsand the dependencies among them. Goals and Tasks both have attributes of type Situation, so that they can be linked by logical dependencies to form graphs that show e.g. decomposition of high-level Goals into sub-goals, and how Tasks can be performed to achieve Goals. Graphs showing temporal dependencies can also be drawn, and we have found UML Activity Diagram notation useful here.

4.1.2.1 Goal View

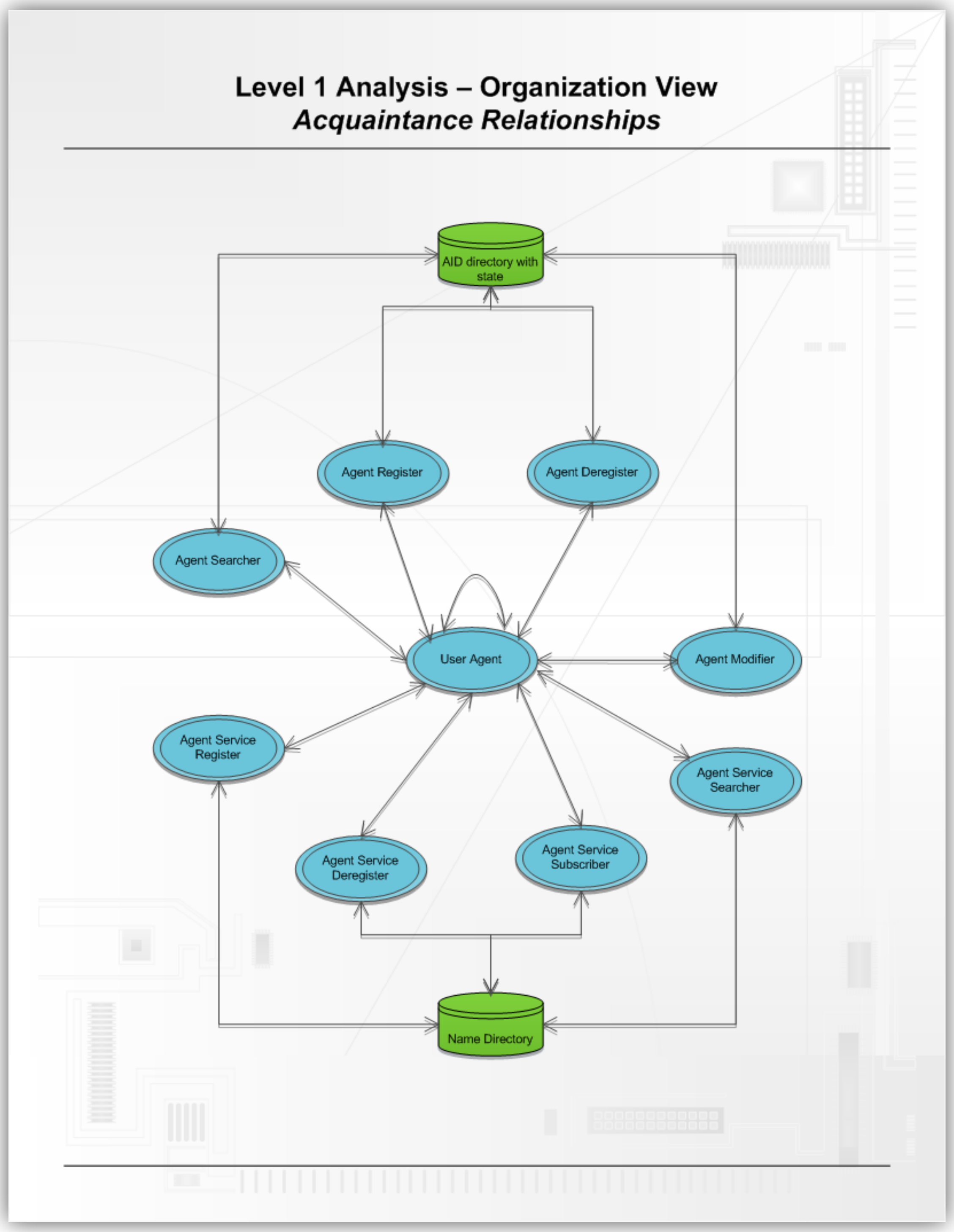


4.1.2.2 Task View



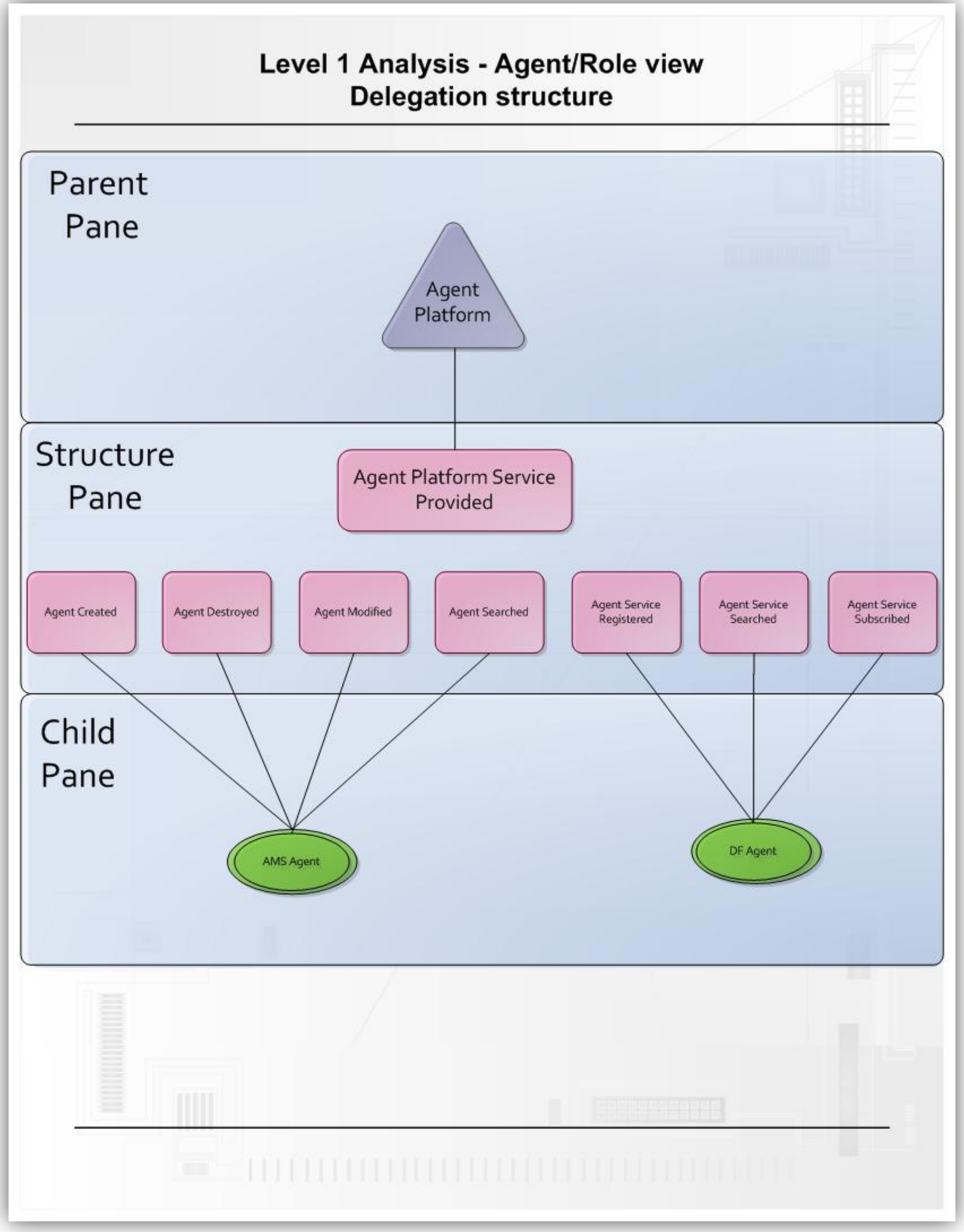
4.2 Level 1 Analysis

4.2.1 Organization view



4.2.2 Agent/Role view

This focuses on the individual Agents and Roles. For each agent/role it uses schemata supported by diagrams to its characteristics such as what Goals it is responsible for, what events it needs to sense, what resources it controls, what Tasks it knows how to perform, 'behaviour rules', etc.

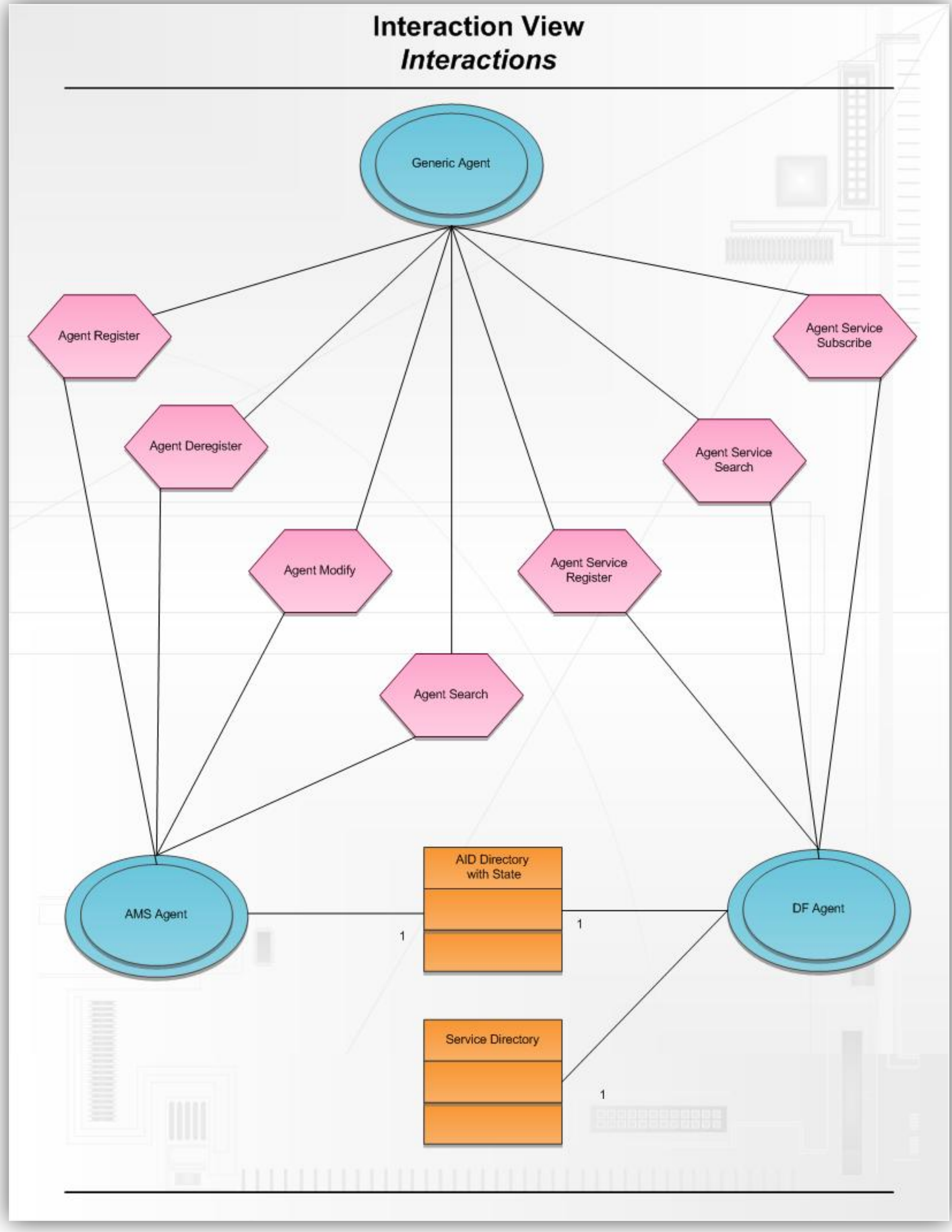


Role Schema	Agent Management System
Goals	AgentCreate AgentDestroy AgentModify AgentSearch
Capability	Some management capabilities are required, to supervise and control other agent.
Knowledge, Beliefs	Agent AID and Agent State.
Agent requirements	The role will be played by AMS Agent

Role Schema	Directory Facilitator
Goals	AgentServiceRegister AgentServiceSearch AgentServiceSubscribe
Capability	The yellow page service capability is required, to search agent service and manage service subscriptions.
Knowledge, Beliefs	Agent AID and Agent Services.
Agent requirements	The role will be played by DF Agent

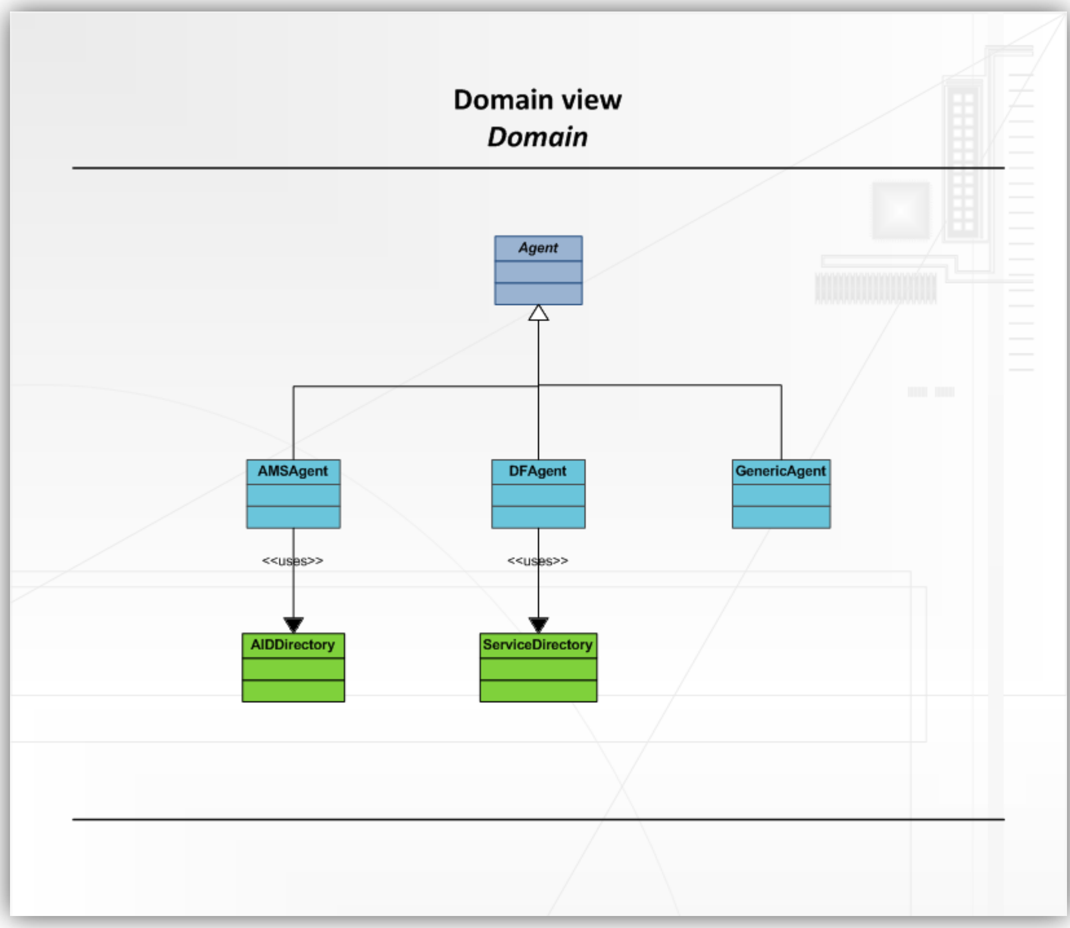
4.2.3 Interaction view

For each interaction among agents/roles, shows the initiator, the collaborators, the motivator (generally a goal the initiator is responsible for), the relevant information supplied/achieved by each participant, the events that trigger the interaction, other relevant effects of the interaction (e.g. an agent becomes responsible for a new goal). Larger chains of interaction across the system (e.g. corresponding to uses cases) can also be considered.

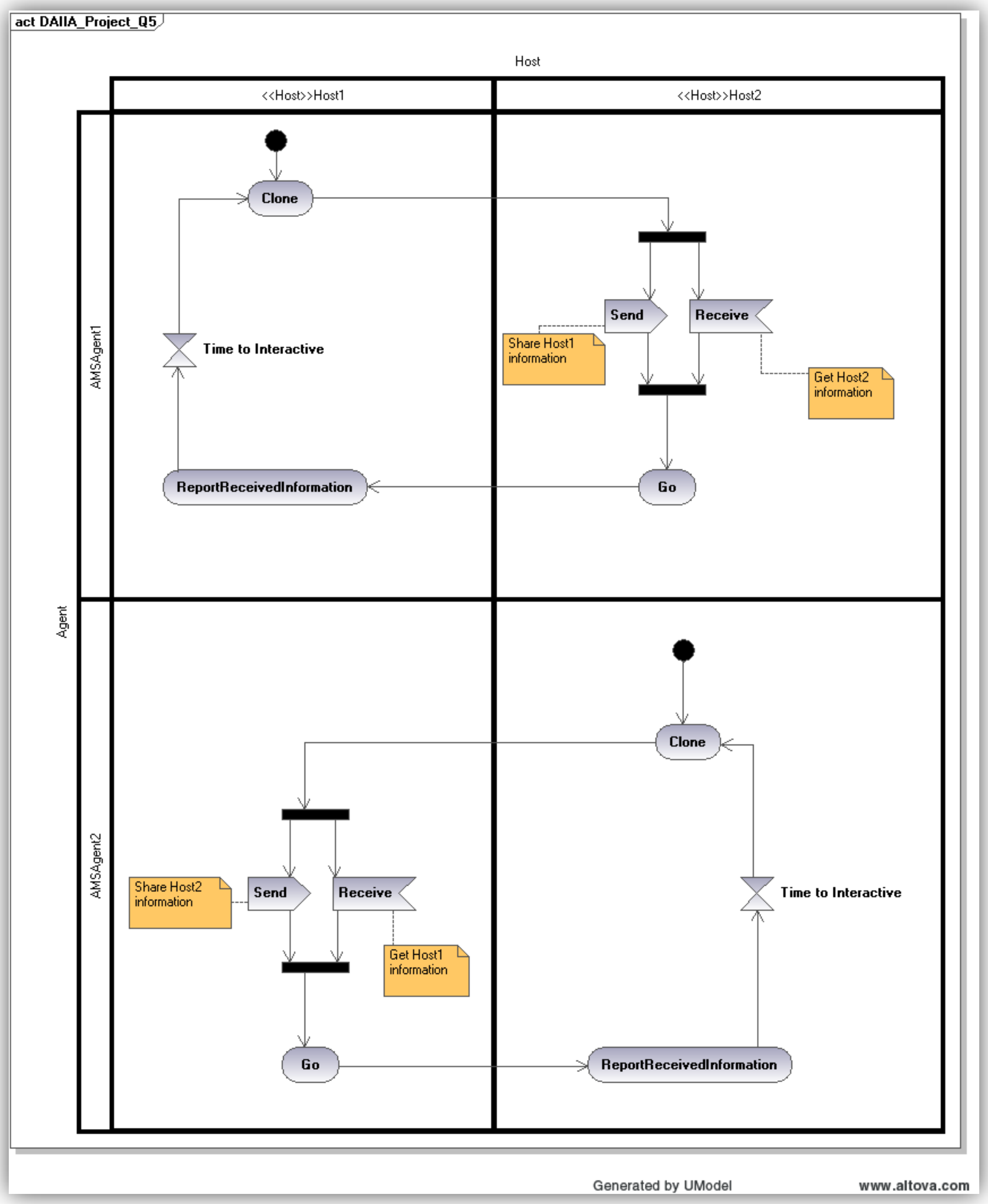


4.2.4 Domain view

Shows the domain specific concepts and relations that are relevant for the system under development.



5. Model mobility in UML 2.0 activity diagrams.



As shown on the figure above, both Hosts have their own AMS agent. When they start, they will make a clone of themselves on each other host. The cloned AMS agent takes the information of their original host. And when they will share their information as well as get the associate host information. After that, the cloned agent will move back to the original host and report received information. A timeout is used to control the AMS to clone itself again to the other host so they can keep an active state of each other host.