|  |
| --- |
| 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 |

## Complete design of your Agent Platform using GAIA AOSE Methodology

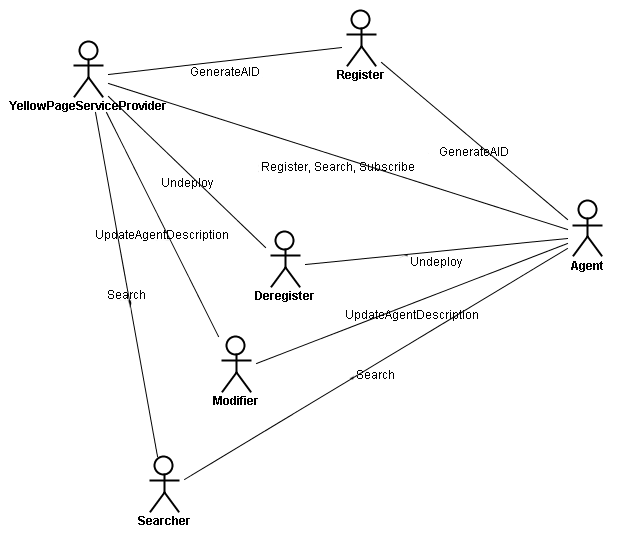
### Role Model

|  |  |  |
| --- | --- | --- |
| 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 |

### 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.

### Agent Model

AMSAgent

1

1

1

1

Register Deregister Modifier Searcher

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

\*

YelloPageServiceProvier

DFAgent plays only one role, which is the YellowPageServiceProvider.

### 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* |

### Acquaintance Model

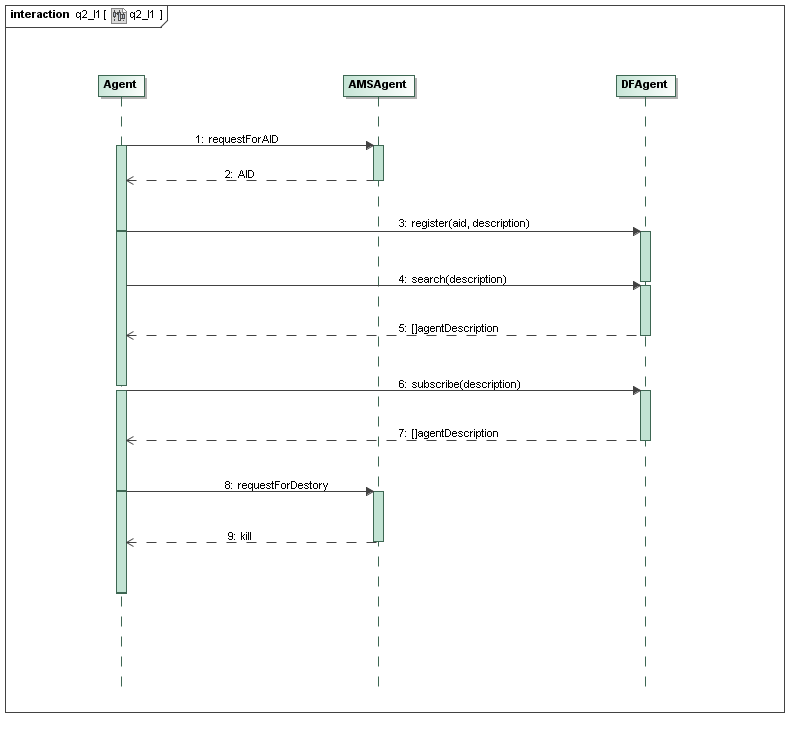
AMSAgent

DFAgent

AMSAgent and DFAgent negotiate with each other directly if necessary.

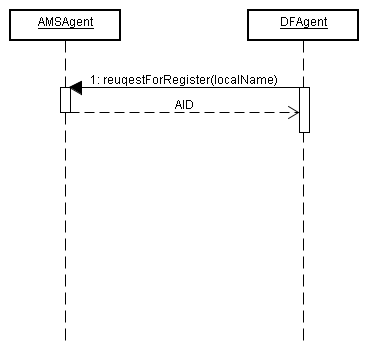
## Represent agent interaction protocols in UML.

### Level 1: Representing the Overall Protocol



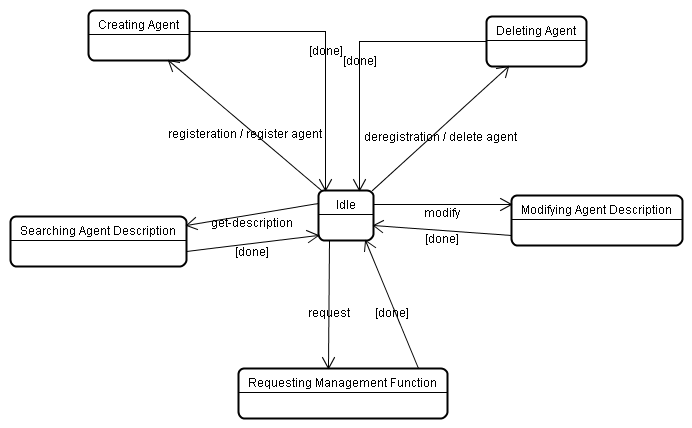
First and foremost, agent obtains its AID from AMSAgent, and then it can optionally register itself at DFAgent providing its AID and description. Agent can search other agents from DFAgent based on descriptions, it can also subscribe to notification by DFAgent if any matching description is found. At last, it’s AMSAgent’s job to kill Agent, if such Agent terminates its execution.

### 2.2 Level 2: Representing Interactions among agents

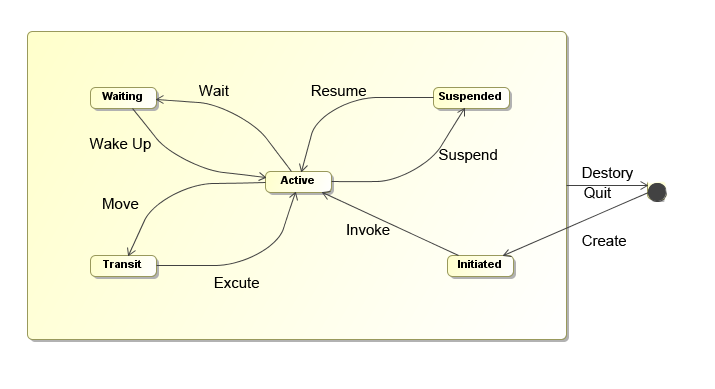


We consider two significant functional agents, AMSAgent and DFAgent in the agent platform (though DFAgent is an optional component of agent platform). AMSAgent is the agent who maintains the life cycles of all agents in the platform, and give each agent a global unique identifier. DFAgent acts a yellow page, which keeps a table of AID-description pair. However, DFAgent first has to obtain an AID for itself from AMSAgent, so that other agents are able to find Directory Facilitator.

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



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.



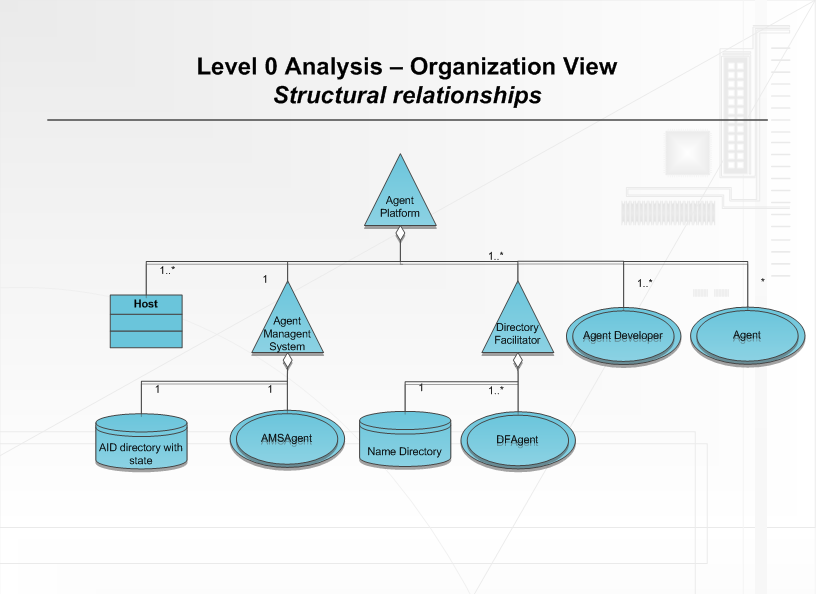
Since AMS itself is an agent, it can be in one of several states, according to Agent Platform Life Cycle in FIFA specification. The states are initiated – the agent object is built; active – the agent object is registered; suspended – the agent object is stopped; waiting – the agent object is blocked;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

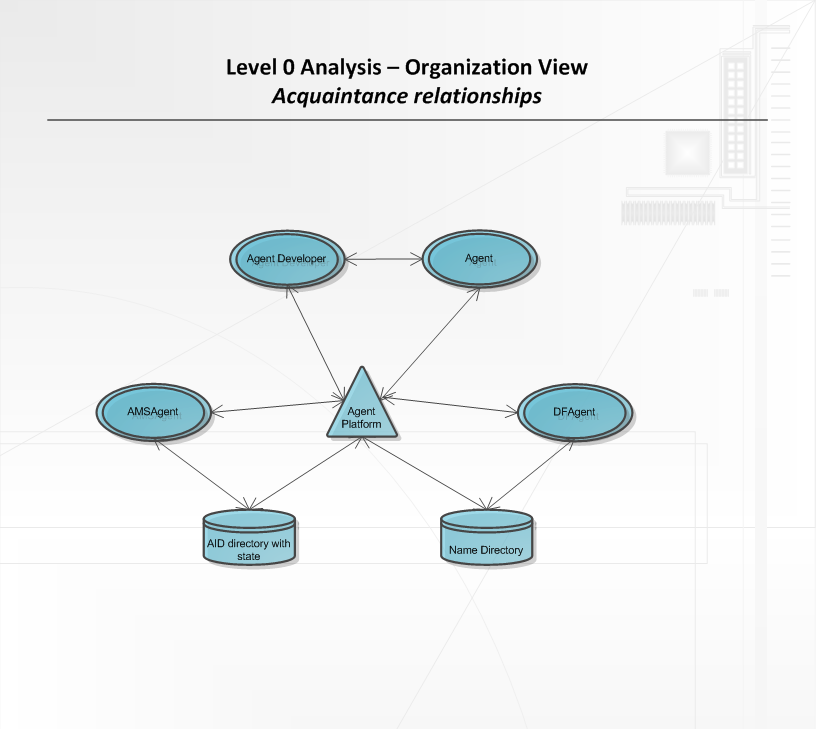
## Use role-based modeling method to model FIPA specified roles for Agent Platform

## Re-modeling the above using MESSAGE UML.

### 4.1 Level 0 Analysis

#### 4.1.1 Organization view





#### 4.1.2 Goal/Task view

### 4.2 Level 1 Analysis

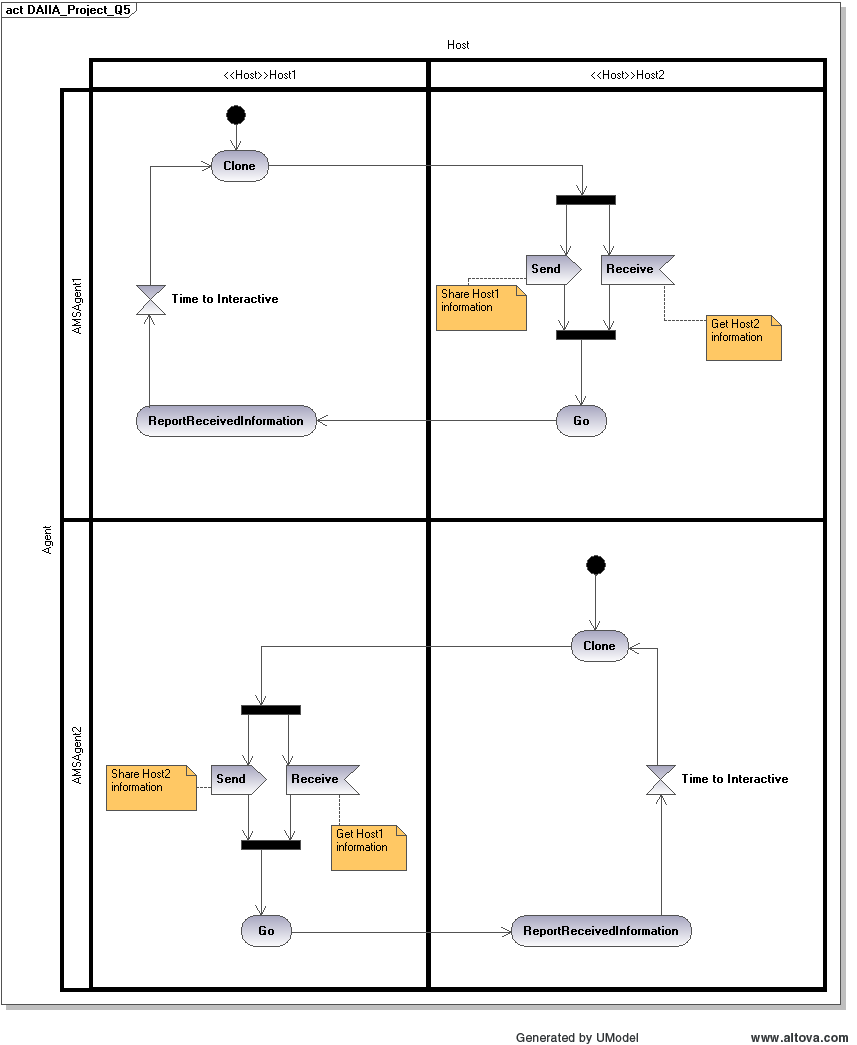
#### 4.2.1 Organization view

#### 4.2.2 Agent/Role view

#### 4.2.3 Interaction view

#### 4.2.4 Domain view

## Model mobility in UML 2.0 activity diagrams.

****

As shown on the figure above, both Host 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 a active state of each other host.