Royal Institute of Technology

MSc. Software Engineering of Distributed Systems

ID2209 Distributed Artificial Intelligence and Intelligent Agents

Homework 1

Andrei Shumanski
andreish@kth.se
00460707761992

Trigonakis Vasileios
vtri@kth.se
00460707694420

Stockholm 2009

TABLE OF CONTENTS

Question 1 4

Answer 4

JADE 4

TRYLLIAN ADK 5

Question 2 6

Answer 6

Jade 6

TRYLLIAN ADK 6

Question 3 7

Answer 7

Question 4 9

Answer 9

Jade 9

Tryllian ADK 10

Question 5 11

Answer 11

QUESTION 1

Architecture of Platform

ANSWER

JADE

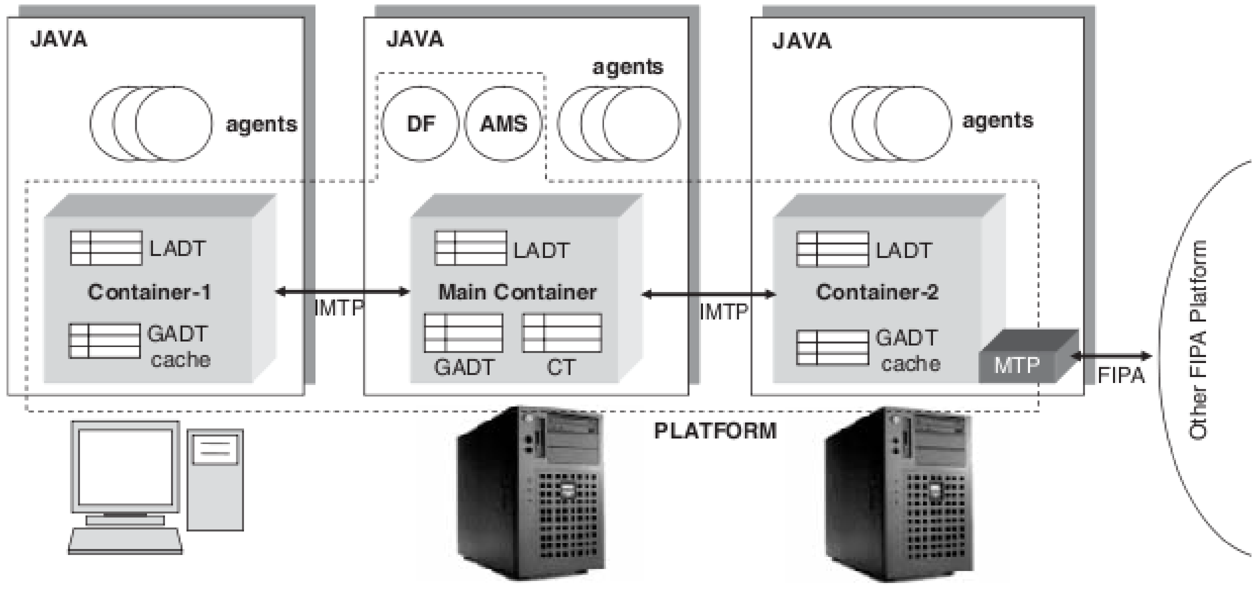


Figure 3.1 Relationship between the main architectural elements

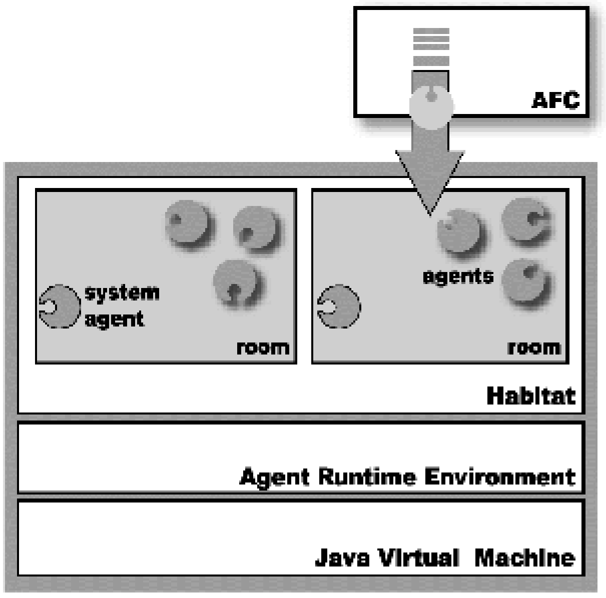
A JADE platform is com posed of agent containers that can be distributed over the network. Agents live in containers which are the Java process that provides the JADE run-time and all the services needed for hosting and executing agents. There is a special container, called the main container, which represents the bootstrap point of a platform: it is the first container to be launched and all other containers must join to a main container by registering with it. The programmer identifies containers by simply using a logical name; by default the main container is named 'Main Container' while the others are named 'Container-1', 'Container-2', etc. Command-line options are available to override default names. As a bootstrap point, the main container has the following special responsibilities:

- Managing the container table (CT), which is the registry of the object references and transport addresses of all container nodes composing the platform;

- Managing the global agent descriptor table (GADT), which is the registry of all agents present in the platform, including their current status and location;

- Hosting the AMS and the DF, the two special agents that provide the agent management and white page service, and the default yellow page service of the platform, respectively.

TRYLLIAN ADK



Two of the most important components of the AGENT RUNTIME ENVIRONMENT (ADK) are the AGENT FOUNDATION CLASSES (AFC) and the AGENT RUNTIME ENVIRONMENT (ARE). Most developers will interact with the AFC to construct their agents. The AFC encapsulates all the functionalities of the ARE and therefore you will normally not need to use the ARE directly. When you want create functionalities that are not provided by the AFC, you will then directly interact with the ARE.

The Tryllian ADK allows application programmers to define all of the components that are required to build an agent-based application. The AFC is the interface layer and contains all interfaces and classes needed to interact with the agent-based elements. All of the API's that are seen by the developer can also be found here. An important part of understanding the development kit is the use of the communication protocols. The ADK provides some easy to use tasks that make the actual language used extremely simple.

QUESTION 2

Services provided by Platform

ANSWER

JADE

Jade platform provides the following services:

Message Transport Service (MTS): The MTS is a service provided by an AP to transport FIPA-ACL messages between agents on any given AP and between agents on different APs. Messages are providing a transport envelope that comprises the set of parameters detailing, for example, to whom the message is to be sent.

Directory Facilitator (DF) : The DF is an optional component of an AP providing yellow pages services to other agents. It maintains an accurate, complete and timely list of agents and must provide the most current information about agents in its directory on a non-discriminatory basis to all authorized agents. An AP may support any number of DFs which may register with one another to form federations.

Agent Management System (AMS): The AMS is a mandatory component of an AP and is responsible for managing the operation of an AP, such as the creation and deletion of agents, and overseeing the migration of agents to and from the AP.

TRYLLIAN ADK

The most important system agents that provide services to user-agents are:

• Habitat agent : This agent allows communication with the entire habitat and also knows the addresses of all other SA

• Room agent : The room agent is the primary point of interaction for any agent. Agents can query this agent for information on their environment (rooms, other agents etc.).

• Transporter agent : The transporter agent is the point of interaction for an agent that wants to move to another location.

QUESTION 3

Comparison of implementation of a simple and a complex scenario same as task2 (i.e. Service Implementation, Service Registration, and Service Discovery)

ANSWER

In Tryllian ADK habitat is analogue of container in Jade. The building blocks of the habitat are agents. The XML habitat description files allow you to specify which of these components you want to create. There is only one habitat per VM. Habitat is merely a word for the agent container: a collection of agents that provides services.

The XML file for the Hello World example is as follows:

```
<!DOCTYPE habitat SYSTEM "http://www.tryllian.com/xml/habitat_1_1.dtd">

<habitat>
  <agent dna="scenarios/agents/helloworld.dna"/>
</habitat>
```

The starting point of every agent is a Java class that extends the Agent interface from the tryllian.afc.agent package. It's similar to jade.core.Agent class in Jade.

The code of the HelloWorldAgent:

```
package tryllian.scenarios.helloworld;
import tryllian.afc.agent.Agent;
import tryllian.afc.task.standard.LogTask;
import tryllian.afc.task.standard.PeriodicalTask;

public class HelloWorldAgent extends Agent
                             implements tryllian.are.TransientAgent
{
    public HelloWorldAgent(String[] args) {
        // Set name if specified in habitat xml file
        if (args != null && args.length > 0) {
            setName(args[0]);
        }
        else {
            setName("hello-agent");
        }
        addTask(new PeriodicalTask(new LogTask("Hello World!")), 30);
    }
}
```

The structure of the agent is similar to the agent in Jade. We also extend Agent class. But in Trillian ADK initialization is performed inside constructor and we add tasks, not behaviours like in Jade. Periodical task is analog of TickerBehaviour in Jade.

In Trillian ADK we need to pack agents into DNA files, in Jade we simply use class files. The DNA file is a file that contains:

- The jar file of your agent
- The optional libraries that your agent uses
- A file descriptor that contains the location of the class file of your agent in the jar file and the location of the libraries needed
- The signature of the agent with your private key for authentication and versioning

The call to ANT is wrapped in a shell script or a batch file, depending upon the host OS. This makes recreating agents very simple: to compile the Hello World example and create the agent, type:

```
build helloworld
```

The build process consists of the following steps:

1. Compiling the Java classes.
2. Packaging the class files in one or more jar file.
3. Creating the agent dna file (combining a descriptor file and jar files together).

After building, a file called helloworld.dna will be built from your new code in the agents directory. To run your agent, type

```
habitat helloworld.xml
```

In Jade we just print two lines:

```
javac HelloAgent.java
java jade.Boot fred:HelloAgent
```

To register registers the agent with the specified service name in Tryllian we can use RegisterTask class. This isanalogue of DFService.register in Jade.

To find agents in Tryllian ADK we can use tryllian.afc.task.find.Directory interface:

- FindAgentNamesTask retrieves the names of all agents in a habitat.
- FindAgentsTask retrieves the addresses of all agents in a habitat.
- FindHabitatServiceTask retrieves the address of the Habitat Service Agent.
- JNDIResolveTask resolves a jndi context-name.
- QueryAgentNameTask asks an agent for its name.
- TrackAgentsTask is notified when an agent arrives or departs.
- getServiceNames returns a set of all currently registered services.

The Directory interface is analogue to DFService.search in Jade.

QUESTION 4

List some notable projects which used that platform.

ANSWER

JADE

LEAP project

During the 2000 – 2002 time span, thanks to the valuable contribution of the Consortium partners - Motorola (coordinator), ADAC, Broadcom, BT, Telecom Italia, University of Parma, Siemens – the project accomplished the relevant goal of developing a new lightweight runtime environment for JADE that can be both exploited in the Internet and in the wireless context. The LEAP extension allows the deployment of JADE on J2ME devices, and open the doors for mobile applications based on the P2P intelligent agent approach.

TeSCHeT

Timeframe:

2002-2005

Participants:

Telecom Italia, Engineering Ingegneria Informatica S.p.A, Isufi

Goal:

The project intends to study and develop a platform-demonstrator, characterized by "open source" standards and technologies, and based on the JADE platform, enabling the treatment and the organization of the tourist-cultural information and the creation of spontaneous services networks

PRIMO

Timeframe:

2002-2005

Participants:

CERCOM - POLITECNICO DI TORINO, DEPARTMENT OF INFORMATION ENGINEERING OF THE UNIVERSITY OF PISA, DEPARTMENT OF INFORMATION ENGINEERING OF THE UNIVERSITY OF PADOVA, ISTITUTO SUPERIORE MARIO BOELLA, DEPARTMENT OF SCIENCE AND TECHNIQUE OF COMMUNICATIONS AND INFORMATION (INFOCOM) OF THE UNIVERSITY OF ROMA "LA SAPIENZA", DEPARTMENT OF ELECTRONIC ENGINEERING AND INFORMATION (DIEI) OF THE UNIVERSITÀ OF PERUGIA, DEPARTMENT OF ELECTRIC ENGINNERING (DIE) OF THE UNIVERSITY OF PALERMO¸ DEPARTMENT OF ELECTRIC ENGINNERING (DIE) OF THE UNIVERSITY OF ROMA “TOR VERGATA”, STMICROELECTRONICS, TELECOM ITALIA LAB, MARCONI SELENIA COMMUNICATIONS

Goal:

The PRIMO project deals with the design, the development, and the experimental validation of base stations and user terminals for wideband wireless communications systems able to cope with those reconfigurability and interoperability characteristics required by the next generation mobile communication systems.

E-Commerce Agent Platform (E-CAP)
Timeframe:
2006-2008
Participants:
Systems Research Institute of Polish Academy of Sciences
Goal:
Comprehensive agent-based e-commerce system realizing price negotations and combining: (a) adaptability and intelligence (change strategy according to history of purchases), (b) mobility (based on transfer of negotation modules and agents). Project is supported by a Marie Curie IRG Grant

TRYLLIAN ADK

WORKSTATION MANAGEMENT

At a big five consulting firm, all of the employees have administrator rights at their workstations. As a result of this each person can install software on his or her workstation. When this software is not licensed with the company it can result in license infractions.

Through the use of Tryllian's workstation agent application this type of scenario can be avoided. Not only can Tryllian's workstation application detect certain software but it can also perform diagnostic duties such as updates on software that is allowed on each employee's workstation. All of the components that make up the application are put into use on a closed network, with the possibility of being extended beyond the firewall for home workers and consultants.

AGENT-BASED SCHEDULING

Through the use of ADK building blocks an agent application has been created to integrate and simplify the communication and work process in the Rotterdam harbor.

In this process four departments are involved: forwarding, the customs department, the transport services and the International Transport Service (ETT). The designed and implemented workflow architecture demonstrates how agent applications can be used to integrate and create efficiency in complicated situations.

The situation is as follows; containers are continually arriving in the Rotterdam harbor and need to be checked by the customs department and then scheduled for transport to their final destinations. This is done through a series of communications by different departments throughout the harbor by fax and telephone. This is an outdated and slow procedure. By using agent applications all of the involved parties were able to integrate their actions and have the results returned to them by agents. By removing overlapping and confusing communication this not only speeded up the process, but created time frames where other duties could be performed.

QUESTION 5

Your personal opinion/judgment about the platform as compared to JADE.

ANSWER

From our short experience with these platforms, they are similar to one another. We experienced the following differences between the two platforms:

- Agent Oriented Programming (JADE) vs Task Oriented Programming (Tryllian). From our experience, the tasks that Tryllian introduces are similar (or better identical) to the behaviours that JADE has.
- You can start an agent in JADE by just specifying the parameters (if any), while in Tryllian you need to make an XML descriptor of the agent.
- The Containers in JADE are automatically created, while in Tryllian you create the Habitats (analogous with Containers) by specifying an XML descriptor.
- Tryllian has a GUI tool for deploying agents and a tool like RMA of JADE but with much more less features.

Unfortunately, we did not have enough time to work with Tryllian, but from our small experience we can conclude that we would prefer to work on JADE platform rather than on Tryllian one. The two important advantages of JADE over Tryllian are:

1. Simplicity. Even if they have almost identical logic in programming (JAVA part), the creation and deployment of new agents, is significantly simpler in JADE. There is no need to create any extra files than the agent code itself. Of course, in bigger projects, the Tryllian approach, with the ability to manage the Habitats may be preferable.
2. Robustness of tools. The JADE platform provides very useful GUI tools for deploying, monitoring and debugging the application. The GUI tools of Tryllian look more preliminary.