# Task5

<http://jasss.soc.surrey.ac.uk/18/1/11.html>

chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/http://kio.pg.gda.pl/iag/download/leszczyna04evaluation.pdf

In this task we want to compare two Agent Platforms with JADE, the one that we used for our previous homework. We consider FIPA-OS and JACK.

## 1. Architecture of platform

### JADE

JADE is a distributed agent platform, which has a container for each host where you are running the agents. Each platform must have a parent container that has two special agents called AMS and DF:

* the DF (Directory Facilitator) provides a directory which announces which agents are available on the platform.
* the AMS (Agent Management System) controls the platform. It is the only one who can create and destroy other agents, destroy containers and stop the platform.

### JACK

JACK applications consist of a collection of autonomous agents that take input from the environment and communicate with other agents. This provides system builders with a very powerful form of encapsulation. Each agent is defined in terms of its:

* goals, which must be pursued by the agent,
* knowledge and beliefs,
* social capability, that means that the agent can react to events and communicate with other agents in the system.

Then the agent performs its function autonomously within the environment it is embedded in.

Multi-agent systems can be written in Java code and the JACK plan language in a standard IDE, although the platform provides an agent-centric IDE called the JACK Development Environment or JDE.

## 2. Services provided by the platform

### JADE

The *Agent* class is a super class which allows the users to create JADE agents. To create an agent, one needs to inherit directly from *Agent*. Normally, each agent recorder several services which they should be implemented by one or more behaviors.

This class provides methods to perform the basic tasks of the agents as:

* Pass messages by objects ACLMessage, with pattern matching
* Support the life cycle of an agent
* Plan and execute multiple activities at the same time

The behavior defines the actions under a given event. This behavior of the agent is defined in the method setup using the method *addBehaviour*. The different behaviors that the agent will adopt are defined from the abstract class *Behaviour*. The class *Behaviour* contains the abstract methods:

* action (): Is executed when the action takes place.
* done (): Is executed at the end of the performance.

A user can override the methods onStart () and OnEnd () property.

### JACK

Compliance with FIPA specifications provides some specific advantages that could be proven useful or even crucial for some cases. For instance, being compliant to FIPA guarantees the system's architecture and performance due to the powerful and well-tested protocols that enable agent cooperation and interoperability. Hence, FIPA standards are needed whenever dynamic, flexible and reconfigurable cases are studied. Moreover, FIPA specifications are popular in agent-based research and, thus, there is plenty of knowledge and support in the research community. Yet, it is up to the researches to decide if they need FIPA compliance or not and to what extent for their study. JADE, JACK and FIPA-OS are full compliant to FIPA.

JACK: The JDE provides graphical tools for writing plans, connecting plans to agents, managing inter-agent communication, as well as compiling and running. The JDE also provides graphical tools for debugging and tracing the execution of plans and inter-agent message passing.

There is no need to explicitly program the interactions of the whole application, rather, the interactions emerge as a by-product of the individual goals and capabilities of the constituent agents.