JADE

JAVA AGENT DEVELOPMENT FRAMEWORK

Index

Agent oriented programming

Agents are distributed and autonomous objects that interact solely through asynchronous messages:

- no agent will invoke another agent's method/ service,
- an agent <u>asks</u> another agent (through messages) to provide a service or execute a method.

Agent oriented programming

Implementations:

- JADE framework (decentralized, fault tolerant with some effort, used in this presentation)
- AgentBuilder;
- JESS;
- More Alternatives:
  - https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software

JADE

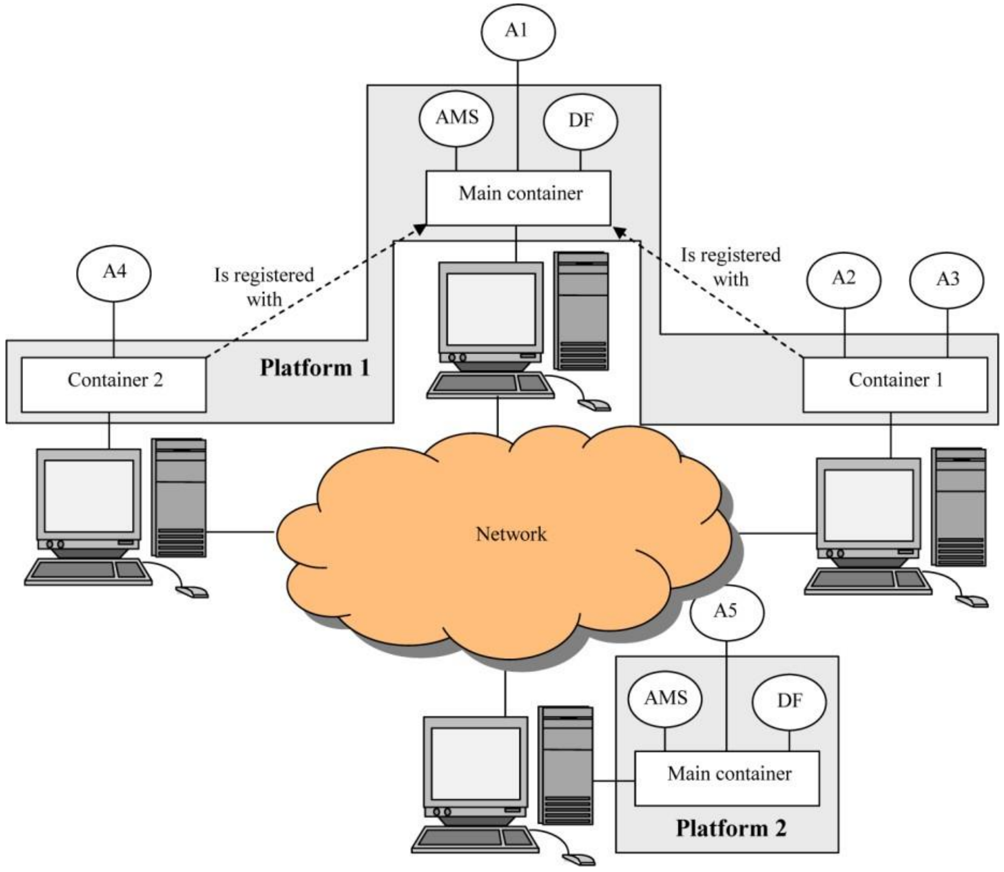
Multi-Agent Platform that enables the development of Multi-Agent Systems:

- Execution environment;
- Libraries for developers;
- Graphic tools for agent management;
- Third party add-on for extended functionality.

Jade

Platform

- Set of distributed containers



Jade

The AMS (Agent Management System) provides the naming service:

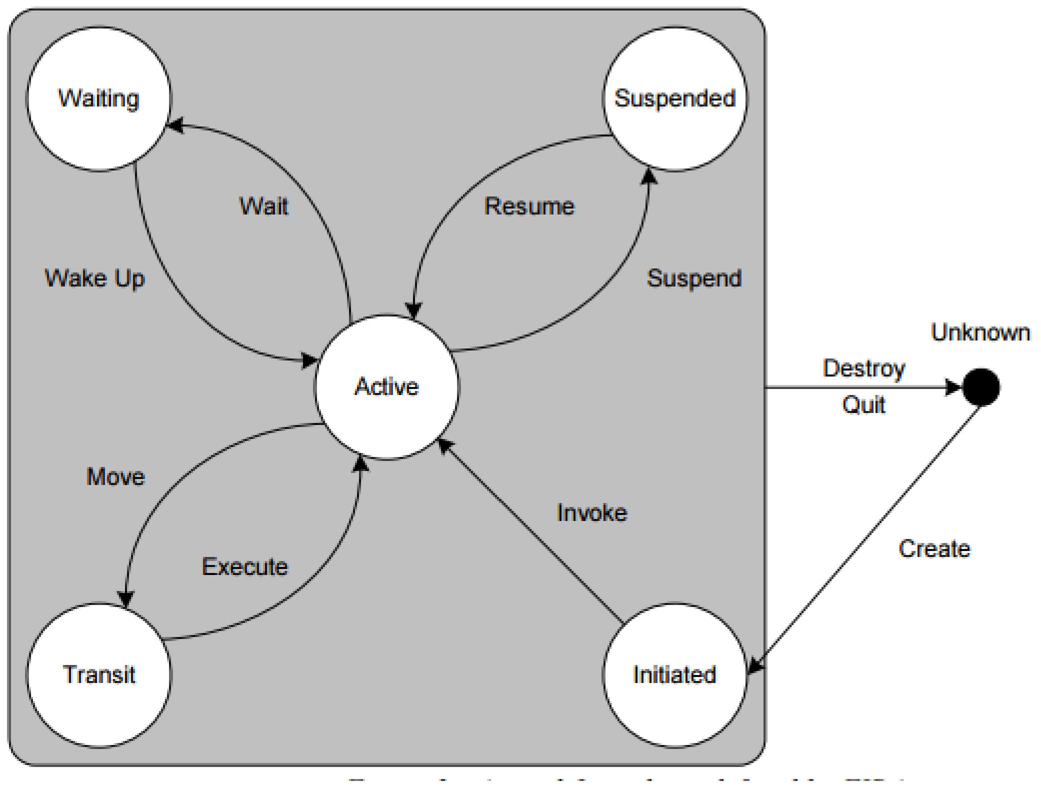
- ensures that each agent in the platform has a unique name;
- it is possible to create/kill agents on remote containers by requesting that to the AMS.

The DF (Directory Facilitator) provides a Yellow Pages service:

- Allows an agent to find other agents providing the services;
- Allows the registration of agent services

JADE Agents

Agent lifecycle



JADE Behaviours

Behaviours that can be added to JADE agents:

- SimpleBehaviour;
- Oneshot;
- Cyclic;
- FSM;
- Others…

```
CyclicBehaviour behav = new CyclicBehaviour(this)
{
   public void action(){
                  // actions…
   }
};
this.addBehaviour(behav) // adds behaviour to agent
```

JADE Behaviours

```
public class ReceiverAgent extends Agent {

  protected void setup() {
        //this is the entry point of the agent
        addBehaviour(new ReceiverBehaviourReceive());
    }

}
```

JADE Behaviours

```
Public class ReceiverBehaviourReceivePing extends SimpleBehaviour{
boolean finished = false;

  public void action() {   // this is the method that defines what the behaviour does
    ACLMessage msg = myAgent.receive();
    if (msg != null) {
        //sometimes the message queue might be empty for example the first time this behavior runs
        //add what you want to do when a message is received
    } else {
        block();
        //free the thread for other behaviors to use until a new message appears in the message queue
    }
  }

  @Override
  public boolean done() {
   //when this method returns true the behavior is no longer activated  even when a message is received
    return finished;
  }
}
// method action will run infinitely unless the block method is called
```

Jade Messages

Sending Messages :

- should be done inside agent behaviours

```
AID receiver = new AID();
receiver.setLocalName("receiver"); // receiver is the name of the target agent
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.setContent("HelloWorld");
msg.setConversationId(""+System.currentTimeMillis());
msg.addReceiver(receiver);
myAgent.send(msg);
```

Jade Messages

Receiving Messages:

- should be done inside agent behaviors

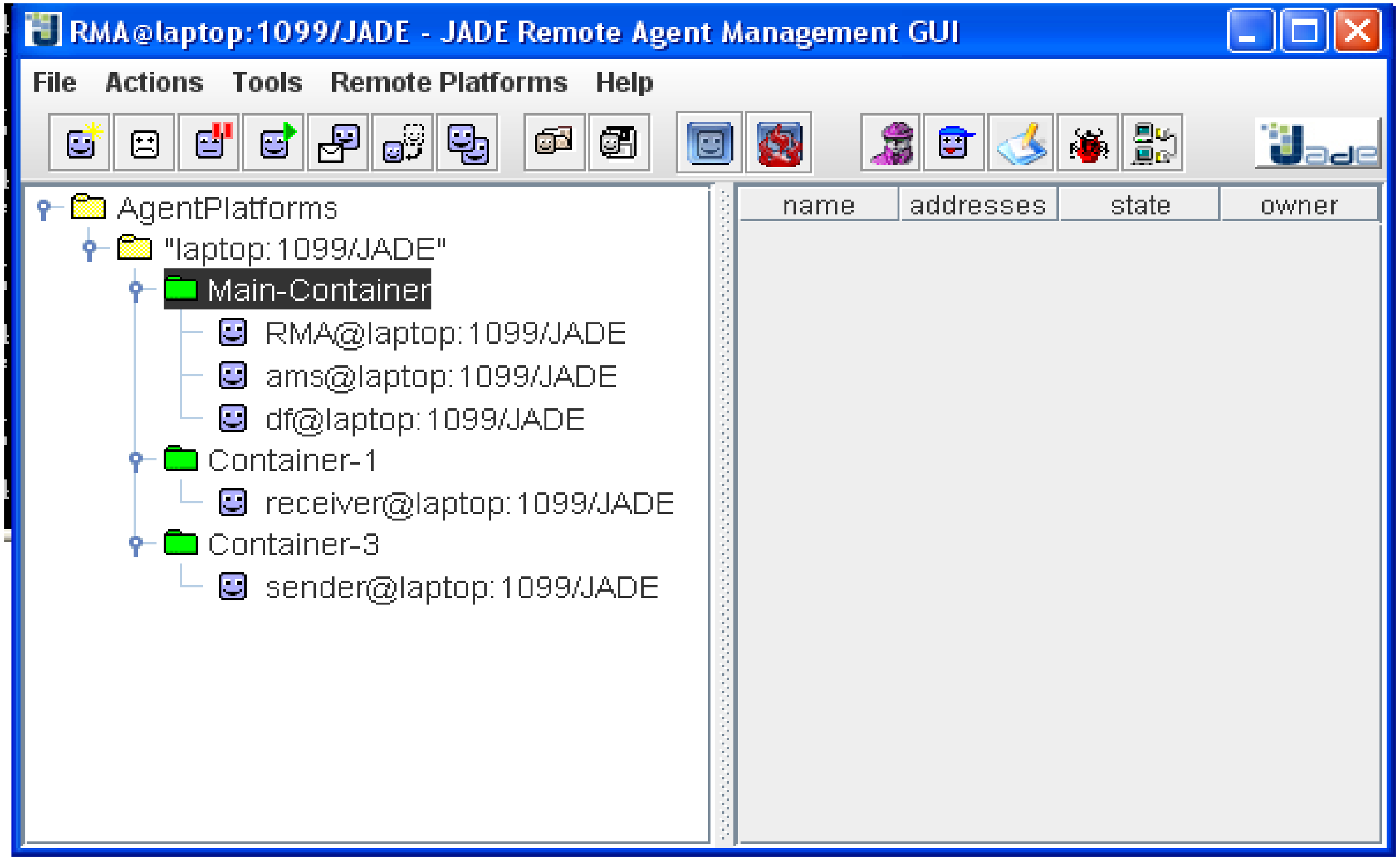
```
private ACLMessage msg = null;
if ((msg = myAgent.receive()) != null)
{
  System.out.println("==== New message received ===");
  System.out.println("= Sender                 :  "+msg.getSender().getLocalName());
  System.out.println("= Address :  "+msg.getSender().getAllAddresses().next());
  System.out.println("= Name : "+msg.getSender().getName());
  System.out.println("= Content : "+msg.getContent());
  System.out.println("= Time   : "+msg.getConversationId());
} else {
  block();
}
```

Jade Messages

Exercise:

- Develop the following agentes:
  - ReceiverAgent – prints every message received
  - HelloAgent – sends helloworld messges to agents

JADE Messages

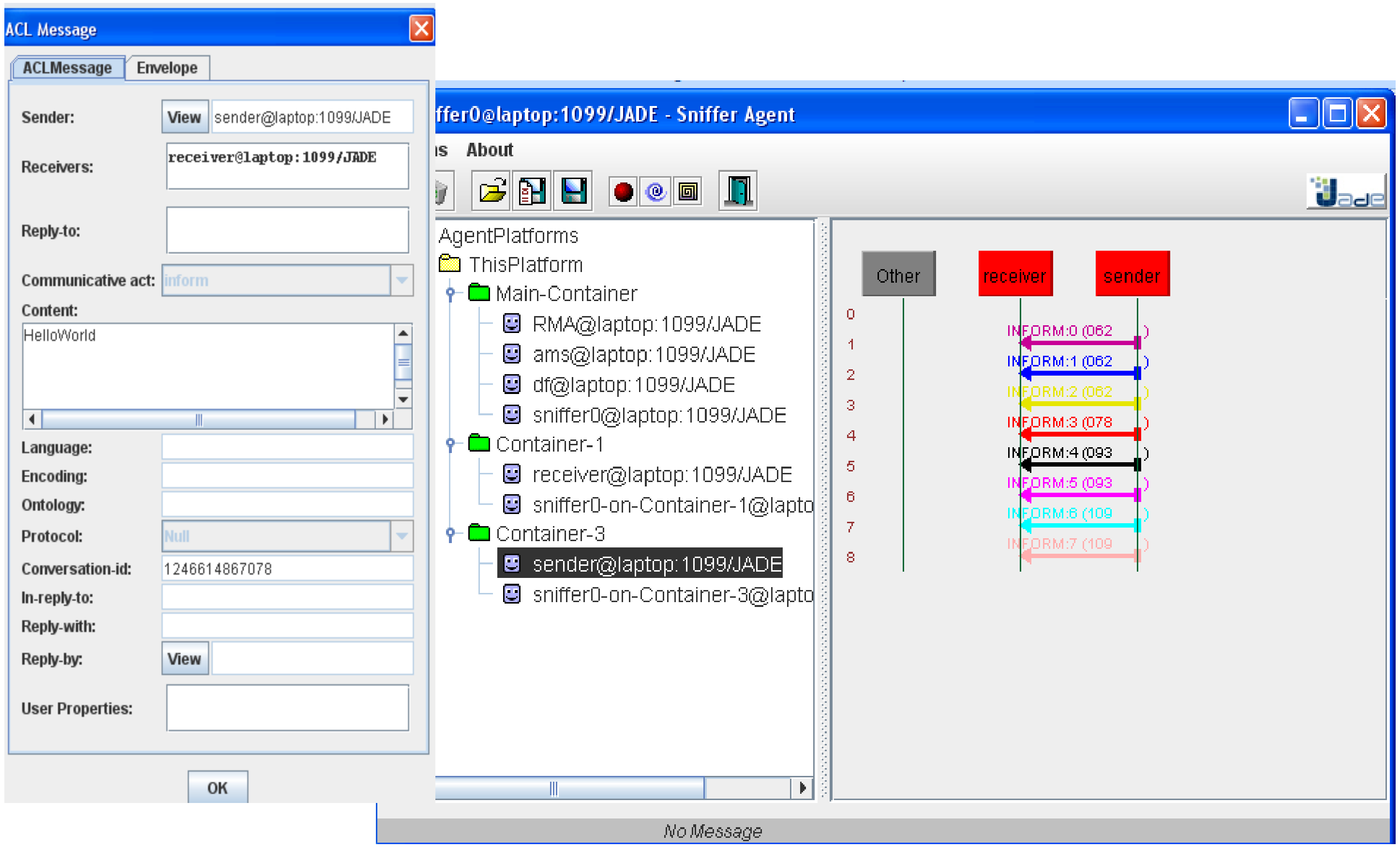

RMA@laptop:1099/JADE - JADE Remote Agent Management GUI

File Actions Tools Remote Platforms Help

AgentPlatforms
"laptop:1099/JADE"
Main-Container
RMA@laptop:1099/JADE
ams@laptop:1099/JADE
df@laptop:1099/JADE
Container-1
receiver@laptop:1099/JADE
Container-3
sender@laptop:1099/JADE

name	addresses	state	owner

JADE Messages


ACL Message

ACLMessage | Envelope

Sender:	sender@laptop:1099/JADE
Receivers:	receiver@laptop:1099/JADE
Reply-to:	
Communicative act:	inform
Content:	HelloWorld
Language:	
Encoding:	
Ontology:	
Protocol:	Null
Conversation-id:	1246614867078
In-reply-to:	
Reply-with:	
Reply-by:	
User Properties:	

ffer0@laptop:1099/JADE - Sniffer Agent

```
AgentPlatforms
  ThisPlatform
    Main-Container
      RMA@laptop:1099/JADE
      ams@laptop:1099/JADE
      df@laptop:1099/JADE
      sniffer0@laptop:1099/JADE
    Container-1
      receiver@laptop:1099/JADE
      sniffer0-on-Container-1@lapto
    Container-3
      sender@laptop:1099/JADE
      sniffer0-on-Container-3@lapto
```

JADE DF

Register Agent Services in DF

```
OneShotBehaviour notify = new OneShotBehaviour(this) {
      public void action() {
         DFAgentDescription dfd = new DFAgentDescription();
         dfd.setName(myAgent.getAID());
         for(String n: services){
              ServiceDescription sd = new ServiceDescription();
              sd.setName(myAgentName);
              sd.setType(n);
              dfd.addServices(sd);
         }
       try {
         DFAgentDescription list[] = DFService.search( myAgent, dfd );
              if ( list.length>0 ) {
              DFService.deregister(myAgent);
              }
              DFService.register(myAgent,dfd);
         }
       catch (FIPAException fe) {
         fe.printStackTrace();
       }
      }
};
this.addBehaviour(notify);
```

JADE DF

Unregister Agent Services in DF

```
OneShotBehaviour unNotify = new OneShotBehaviour(this) {
    public void action() {
        try {
            DFService.deregister(myAgent);
        } catch (FIPAException e) {
            e.printStackTrace();
        }
    }
};
this.addBehaviour(unNotify);
```

JADE DF

Subscribe Agents by their services

```
public void subscribe(final String[] services){
        DFAgentDescription template = new DFAgentDescription();
        for(String n: services){
           ServiceDescription sd = new ServiceDescription();
            sd.setType(n);
            template.addServices(sd);
        }
        Behaviour subscribe = new SubscriptionInitiator(this, DFService.createSubscriptionMessage(this,
getDefaultDF(), template, null)) {

            @Override
            protected void handleInform(ACLMessage inform) {
                try {
                    DFAgentDescription[] resultados = DFService.decodeNotification(inform.getContent());
                    handleDfNotification(resultados);
                } catch (FIPAException fe) {
                    fe.printStackTrace();
                }
            }
        };
        addBehaviour(subscribe);
        System.out.println("DF services subscrition: " + myAgentName);
      }
```

JADE DF

Exercise

- Update last exercise and use DF for service discovery.

JADE inside Java Applications

Launch the Jade from java code:

```
public class Main {
		Runtime rt;
		ContainerController container;

		public void initMainContainer(String host, String port) {
				this.rt = Runtime.instance();
				Profile prof = new ProfileImpl();
				prof.setParameter(Profile.MAIN_HOST, host);
				prof.setParameter(Profile.MAIN_PORT, port);
				prof.setParameter(Profile.MAIN, "true");
				prof.setParameter(Profile.GUI, "true");
				this.container = rt.createMainContainer(prof);
				rt.setCloseVM(true);
		}
		…
```

JADE inside Java Applications

Launch the Jade from java code:

```
…
 public void startAgentInPlatform(String name, String classpath){
   try {
     AgentController ac = container.createNewAgent(
                 name,
                 classpath,
                 new Object[0]);
     ac.start();
   } catch (Exception e) {
     e.printStackTrace();
   }
 }
…
```

JADE inside Java Applications

Launch the Jade from java code:

```
public static void main(String args[]) {
                Main mc = new Main();
                mc.initContainer("127.0.0.1", "1099");
mc.startAgentInPlatform("SensorName", “agentPackage.agentClass");
}
```

No need for run confgurations!

References

Bellifemine F., Caire G., Greenwood D., Developing Multi-Agent Systems with JADE, John Wiley & Sons, ISBN: 978-0470057476, 2007.

JADE: Java Agent DEvelopment framework

- http://jade.tilab.com/

JADE

JAVA AGENT DEVELOPMENT FRAMEWORK