

# Plate-forme JADE

## UE – SMA 2009

Java Agent Development Framework

<http://jade.tilab.com/doc/index.html>

Zach Lewkovicz

<http://www-poleia.lip6.fr/~lewkovicz/>

# Plate-formes SMA

$\neq$  *objectifs*  $\rightarrow$   $\neq$  *plate-formes*

- Rôle d'une *plate-forme*
  - Prendre en charge l'infrastructure
    - Gestion des threads agents
    - Transport des messages
    - Architecture distribuée
  - Faciliter la méthodologie SMA
    - Langage de programmation d'agents
    - Interface de visualisation
    - Primitives spécifiques

# Plate-formes

- **Jade** <http://jade.tilab.com/>
  - API Java – norme FIPA
- **Netlogo** <http://ccl.northwestern.edu/netlogo/>
  - réactif localisé 2D
- **Madkit** <http://www.madkit.org/>
  - modèle AGR en Java
- **VDL** <http://www-poleia.lip6.fr:8180/~sabouret/demos/index.html>
  - introspection
- **CLAIM** (LIP6)
  - mobilité

# JADE

- ***les spécifications FIPA définissent*** le modèle de référence d'une plate forme d'agents et l'ensemble des services qui doivent être fournis pour une réalisation inter opérable des systèmes multi-agent
- ***JADE (Java Agent DEvelopment framework)*** est un **logiciel environnemental** qui permet de construire des **systèmes d'agents** pour la gestion des ressources d'information sur le réseau, adapté aux **spécifications de FIPA**

# Structure

- **JADE** contient les bibliothèques **de Java (classes)** qui sont requises pour le développement des applications d'agents, ainsi que le **run-time** qui fournit les services de base
- Chaque **instance** de **JADE** run-time est appelée un **container** (puisque'il "contient" des agents)
- L'**ensemble de tous les containers** est appelé **plate-forme** et fournit une couche homogène qui cache complètement la complexité et la diversité du réseau des agents

Distributed application composed of a set of agents



**JADE LAYER**

Container

Container

Container

Container

**JADE**

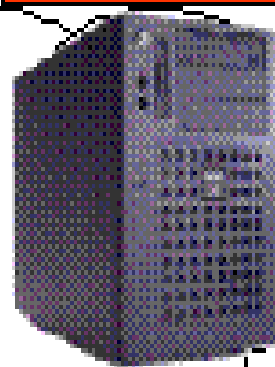
**JAVA VM LAYER**

**J2SE**

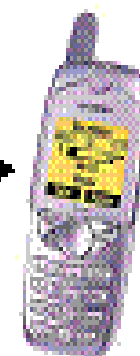
**J2EE**

**PersonalJav**

**CLDC**



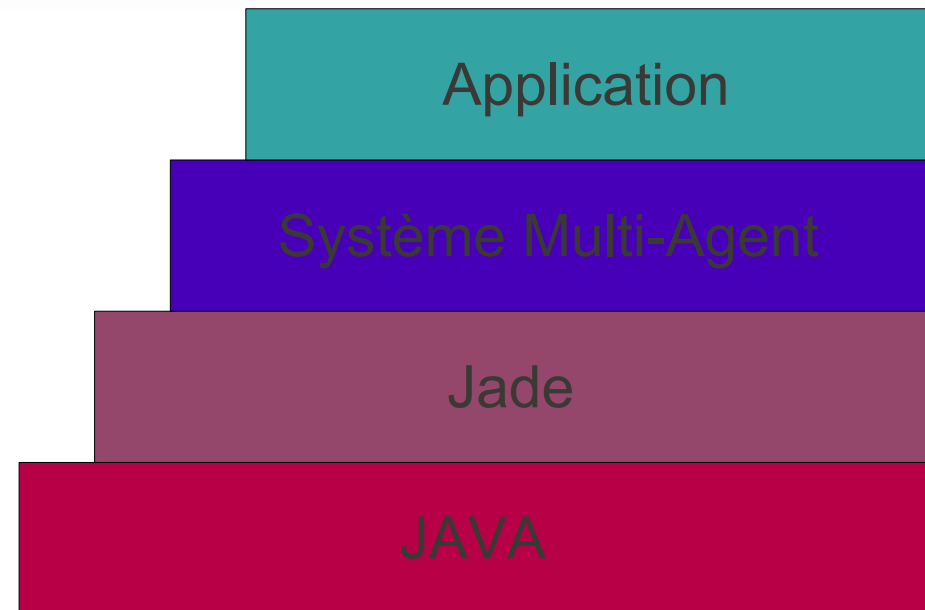
**Internet**



**Wireless environment**

# Jade

- Fondation privée
  - **TI Lab**, Motorola, *Whitestein*, *Profactor*, *FT*
- Open source (LGPL) <http://jade.tilab.com/>
- Plate-forme répartie (multi-hôtes)
  - Adressage
  - RMI
- API Java
- Norme FIPA
  - ACL, AMS/DF
- Tutoriel (Jade tutorial for beginners)



# Plan

- Principe général
- Agents
- Comportements
- Communication
- Protocoles
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques



# Plan

- **Principe général**
- **Agents**
- **Comportements**
- **Communication**
- **Protocoles**
- **Pages jaunes**
- **Mobilité**
- **Lancement de JADE**
- **Interfaces & outils graphiques**

# Principe général

- Plate-forme (distribuée)
  - Conteneurs
    - Agents
      - Comportements
  - 1 Conteneur principal : hôte+port connus
    - AMS : agent de nommage
    - DF : pages jaunes
  - N Conteneurs secondaires
    - Enregistrement
- Démarrer un conteneur
  - *Jade administrator guide* (paper&resources / online doc)

# Packages

```
import jade.core.Agent;  
import jade.core.AID;  
import jade.lang.acl.ACLMessage;  
import jade.lang.acl.MessageTemplate;  
import jade.core.behaviours.CyclicBehaviour;  
import jade.core.behaviours.OneShotBehaviour;  
import jade.core.behaviours.TickerBehaviour;  
...
```

# Plan

- Principe général
- **Agents**
- Comportements
- Communication
- Protocoles
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques

# Agents

- Classe *jade.core.Agent*
  - 1 Thread par agent
  - Méthodes *setup* (init) et *takeDown* (stop)
- Adressage : notion d'AID
  - Classe *jade.core.AID*
- Ensemble de *comportements*
  - Méthode *addBehaviour()*
- Envoi de *messages*
  - Méthodes *send()* et *receive()*
- Fin : méthode *doDelete()*

# Initialisation d'un Agent

```
protected void setup() {  
    addBehaviour(new TickerBehaviour(this, 4000) {  
        protected void onTick() {  
            mettreAJourUtilite();  
        }  
    });  
    addBehaviour(new myBehaviourA());  
    addBehaviour(new myBehaviourB());  
    addBehaviour(new myBehaviourC());  
    addBehaviour(new myBehaviourD());  
}
```

# États des Agents

/\* Les méthodes que l'agent exécute dans la vie quotidienne étant dans les états: etatA, etatB ou etatC \*/

```
addBehaviour(new TickerBehaviour(this, 4000) {  
    protected void onTick() {
```

```
        if (monEtat.equals("etatA")) {  
            methodeA();
```

```
        }
```

```
        if (monEtat.equals("etatB")) {  
            methodeB();
```

```
        }
```

```
        if (monEtat.equals("etatC")) {  
            methodeC();
```

```
        }));
```

```
}
```

# Terminaison d'un agent

```
/**  
 * La méthode prédéfinie par JADE qui est appelée quand un agent meurt  
 */  
protected void takeDown() {  
  
    String s = "Je suis mort : " + getLocalName();  
  
}
```



# Terminaison d'un agent

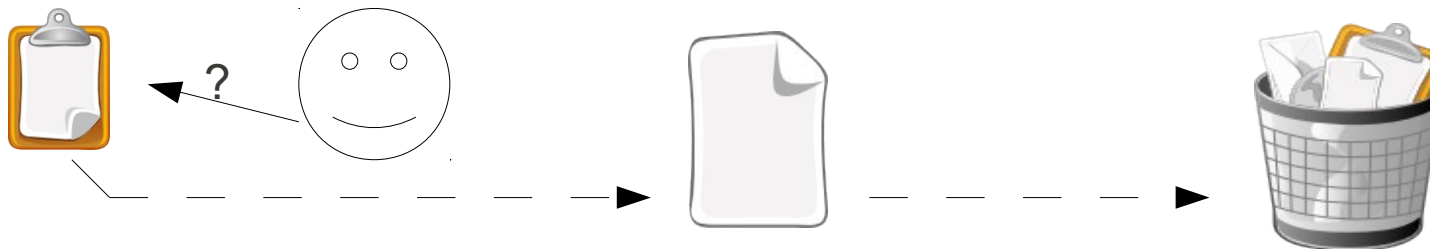
```
/**  
 * La méthode prédéfinie par JADE qui est appelée quand un agent meurt  
 */  
protected void takeDown() {  
    ACLMessage terminate = new ACLMessage(ACLMessage.REQUEST);  
    terminate.addReceiver(getAMS());  
    terminate.setOntology(jade.domain.JADEAgentManagement.JADE  
        ManagementOntology.SHUTDOWNPLATFORM);  
    send(terminate);  
}
```

# Plan

- Principe général
- Agents
- **Comportements**
- Communication
- Protocoles
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques

# Comportements

- Classe abstraite *jade.core.behaviours.Behaviour*
  - Méthodes *action()* et *done()* (redéfinition)
  - Méthode *block()* (voir messages)
- Agent : « pool » des comportements actifs
  - File (FIFO) avec remise
    1. Prendre un comportement
    2. L'exécuter (*action*) : non interrompue (pas Thread)
    3. Tester *done*
      - true → retirer du pool, false → remet à la fin du pool



# Comportements

- Sous-classes

- *OneShotBehaviour*

- *done* → *true*



- *CyclicBehaviour*

- *done* → *false*

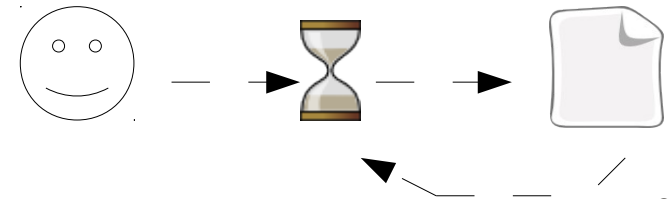
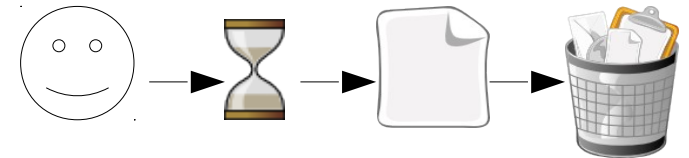


- *WakerBehaviour* (timeouts)

- méthode *handleElapsedTimeout*

- *TickedBehaviour* (timeouts répétés)

- méthode *onTick*



# Comportement itératif

- Séquence d'étapes
  - « done » seulement à la fin
- Entrelacement de comportements !
- Exemple

```
class PasAPas extends Behaviour {  
    private int etape = 0;  
    public void action() {  
        etape++;  
        switch(etape) {  
            ...  
        }  
    }  
    public boolean done() {  
        return (step>=5);  
    }  
}
```

# Comportement unique

```
public class Participation extends OneShotBehaviour {  
    public void action() {  
        String s = "Bonjour, Je suis " + getLocalName();  
    }  
}
```

# Enlever un comportement

```
public class Inscription extends CyclicBehaviour {  
    public void action() {  
        if (Accueil.estPremiereInscription()) {  
            removeBehaviour(premiereInscription);  
        }  
    }  
}
```

# Plan

- Principe général
- Agents
- Comportements
- **Communication**
- Protocoles
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques

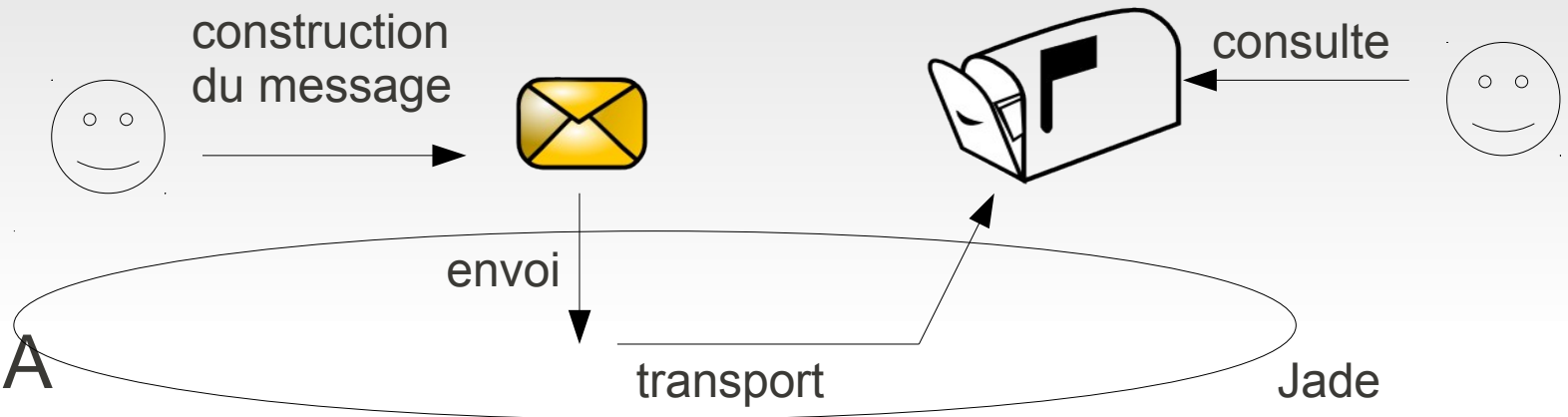


# Communication

- Les agents communiquent en échangeant des messages **asynchrones**
- Pour communiquer l'agent **ne fait qu'envoyer** un message à une destination
- Les agents sont identifiés par un **nom**
- Il n'y a pas de dépendance **temporelle** entre des agents communicants
- L'émetteur et le récepteur peuvent ne pas être **joignables** en même temps
- Le récepteur peut même **ne pas exister** ou ne pas être directement connu de l'émetteur, par exemple: « tous les agents intéressés par la musique » comme destination

# Messages

- Principe général



- Agent A

1. Construction du message

2. Envoi


- Agent B : regarder son courrier

*Dans des comportements*

# Message FIPA-ACL

- Émetteur {Agent ID}
- Destinataires {Agent ID}
- Performatif
  - REQUEST *Appel à l'ascenseur*
  - INFORM *Ascenseur arrivé*
  - QUERY\_IF *Est-porte ouverte ?*  
(entre autres)
- Contenu
  - + Langage ; + Ontologie
- Identifiants de conversation
  - conversation-id, reply-with, in-reply-to, reply-by

# Message Jade

- Classe *jade.lang.acl.ACLMessage*
  - Constructeur → performatif
  - *addReceiver()* → AID
  - *setContent()* → String
    - et autres setters (langage  contenu...) et getters
- Agents
  - *send()*
  - *receive* (non bloquant) → *null* si vide
    - Méthode *block()* dans les comportements
    - Filtrage de messages

# Envoi d'un ACLMessage

```
void appelAscenseur(Etage etage) {  
    ACLMessage msgAppelAscenseur = new  
    ACLMessage(ACLMessage.INFORM);  
    msgAppelAscenseur.addReceiver(new AID("ascenseur",  
    AID.ISLOCALNAME));  
    msgAppelAscenseur.setLanguage("English");  
    msgAppelAscenseur.setOntology("Aller à l'étage");  
    try {  
        msgAppelAscenseur.setContentObject(etage);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    send(msgAppelAscenseur);  
}
```

# Message filtrant

- Classe *jade.lang.acl.MessageTemplate*
  - Ensemble de « factory » statiques
    - *MatchContent*
    - *MatchPerformative*
    - *MatchOntology*
    - *etc...*
  - *Agent.receive(MessageTemplate)*

# Reception d'un ACLMessage

```
public class AllerEtage extends CyclicBehaviour {  
  
    MessageTemplate templateEtage = MessageTemplate.MatchOntology("Aller à l'étage");  
  
    public void action() {  
  
        ACLMessage msgAllerEtage = myAgent.receive(templateEtage);  
        Etage etage = null;  
        if (msgAllerEtage != null) {  
            try {  
                etage = (Etage) msgAllerEtage.getContentObject();  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
            allerALEtage(etage);  
        } else {  
            block();  
        }  
    }  
}
```

# Comportements (bis)

- Agent et comportement
    - Mise en veille si pas de comportement
  - Messages bloquants
    - Bloquer un comportement itératif
- Le comportement n'est plus éligible



jusqu'à :

- réception d'un message (par l'agent)
- timeout expire
- restart()

L'agent peut continuer de sélectionner d'autres comportements



# Message bloquant

- Pattern Java :

```
public void action() {  
    ACLMessage msg = myAgent.receive();  
    if (msg != null) {  
        // Message received. Process it  
        ...  
    }  
    else {  
        block();  
    }  
}
```

## Attention

- Méthode *blockingReceive()* de Agent  
→ Bloque **tout l'agent** !



# Plan

- Principe général
- Agents
- Comportements
- Communication
- **Protocoles**
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques

# Protocoles

- Comportement itératif
  - Étapes de réception et d'envoi
  - Exemple : CNet
    - Agent vendeur
      - send(CFP,article)
      - receive-block(OFFER,article)
      - send(ACCEPT) ou send(REFUSE)
      - ...
    - Agents acheteurs
      - receive(CFP,article)
      - send(OFFER)
      - receive-block(ACCEPT/REFUSE)
      - ...

# Protocoles

- Classes prédéfinies
  - Package *jade.proto*
  - Ex: *jade.proto.ContractNetInitiator*
    - Le comportement pour l'agent « vendeur »
- Définir un protocole
  - = Définir 1 comportement pour chaque rôle

# Plan

- Principe général
- Agents
- Comportements
- Communication
- Protocoles
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques

# Pages jaunes

- Description

  - jade.domain.FIPAAgentManagement.DFAgentDescription*

  - *ServiceDescription*

  - *Properties* (QoS)

- Directory Facilitator

  - Classe *jade.domain.DFService*

  - Méthode statique *register* → (Agent,Description)

    - Pensez à signaler le départ (*deregister* dans *takeDown*)

  - Méthode statique *search* → (Agent,Description)

    - Agent = l'agent qui cherche

    - Description = *DFAgentDescription* comme pattern



# Plan

- Principe général
- Agents
- Comportements
- Communication
- Protocoles
- Pages jaunes
- **Mobilité**
- Lancement de JADE
- Interfaces & outils graphiques

Distributed application composed of a set of agents



**JADE LAYER**

Container

Container

Container

Container

**JADE**

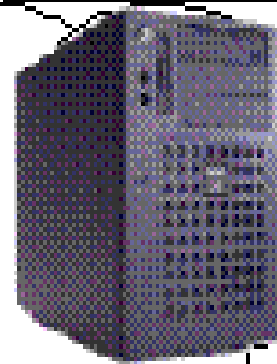
**JAVA VM LAYER**

**J2SE**

**J2EE**

**PersonalJav**

**CLDC**



**Internet**



**Wireless environment**



# Mobilitéé

`jade.domain.mobility`

- `beforeMove()`
- `doMove(jade.core.Location endroit)`
- `afterMove()`
  
- `beforeClone()`
- `doClone(jade.core.Location endroit, String nouveauNom)`
- `afterClone()`
  
- `jade.domain.mobility.MobilityOntology`

# Plan

- Principe général
- Agents
- Comportements
- Communication
- Protocoles
- Pages jaunes
- Mobilité
- **Lancement de JADE**
- Interfaces & outils graphiques

# Lancement de JADE

- `java jade.Boot [options] [AgentSpecifier list]`

# Lancement de JADE

- `java jade.Boot [options] [AgentSpecifieur list]`
- `java jade.Boot -gui`
  - `Ascenseur1:MyAscenseurAgent("Tokyo" 6)`
  - `Ascenseur2:MyAscenseurAgent("NewYork" 8)`
  - `Ascenseur3:MyAscenseurAgent("Londres" 12)`

# Plan

- Principe général
- Agents
- Comportements
- Communication
- Protocoles
- Pages jaunes
- Mobilité
- Lancement de JADE
- Interfaces & outils graphiques

sniffer0@AP:1099/JADE - Sniffer Agent

Actions About



AgentPlatforms

"AP:1099/JADE"

Main-Container

Container-1

Container-2

da0@AP:1099/JADE - DummyAgent

General Current message Queued message



ACLMessage Envelope

Sender: Set da0@AP:1099/JADE

Receivers:

Reply-to:

Communicative act: accept-p

Content:

log0@AP:1099/JADE - LogManagerAgent

Logger Name Set Level

global INFO

jade.content.lang.sl.SL0Ont... INFO

jade.content.lang.sl.SL1Ont... INFO

jade.content.lang.sl.SL2Ont... INFO

jade.content.lang.sl.SL0Ontol... INFO

jade.content.onto.BasicOntol... INFO

jade.content.onto.Ontology INFO

jade.content.onto.Serializabl... INFO

jade.content.schema.AgentA... INFO

jade.content.schema.Aggreg... INFO

jade.content.schema.Conce... INFO

jade.content.schema.Conte... INFO

jade.content.schema.Conte... INFO

jade.content.schema.IRESc... INFO

jade.content.schema.Object... INFO

jade.content.schema.Predic... INFO

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

File Actions Tools Remote Platforms Help



AgentPlatforms

"AP:1099/JADE"

Main-Container

df@AP:1099/JADE

RMA@AP:1099/JADE

ams@AP:1099/JADE

sniffer0@AP:1099/JADE

da0@AP:1099/JADE

Introspector0@AP:1099/JADE

Introspector0-on-Main-Contain

log0@AP:1099/JADE

Container-1

Container-2

name	addresses	state	owner
da0@AP:1...		active	NONE

Introspector0@AP:1099/JADE

File About

Main-Container

df@AP:1

RMA@A

da0@AP

sniffer0@

ams@AP

Introspe

AgentPlatforms."AP:1099/JADE"

ams@AP:1099/JADE

View State Debug

Current State

Active

Suspended

Idle

Waiting

Moving

Dead

Change State

Suspend

Wait

Wake Up

Kill

Incoming Messages

Pending Received

Incoming Messages -- Pending

Outgoing M

Pending Sent

Outgoing Message

Behaviours

AMSJadeAgentManagem

EventManager

AMSFipaAgentManagem

DeregisterToolBehaviour

RegisterToolBehaviour

Name: EventManager

Class: jade.domain.a

Kind: CyclicBehavior

# Outils graphiques (1)

- pour debugger -

- **L'Agent Dummy**

- inspecte les messages échangés
- valide des patterns des messages échangés par les agents
- fournit un support pour éditer, composer et envoyer des messages ACL

- **L'Agent Sniffer**

- permet de suivre les traces des messages échangés
- chaque message est suivi et affiché dans la fenêtre du sniffer

# Outils graphiques (2)

- pour debugger -

- **L'Agent Introspector**

- supervise et contrôle le cycle de vie des agents
- inspecte toutes les files des messages entrants et sortants
- contrôle la queue des comportements, en permettant l'exécution d'un comportement pas-à-pas

- **L'Agent Log Manager**

- suit les traces et mémorise les événements qui ont lieu durant la vie des systèmes multi-agents



**Merci, Questions ?**