Jade - The Java Agent Development Framework

O que é jade?

- É um *middleware* de agentes que implementa uma plataforma distribuída e um *framework* de desenvolvimento para SMA
- Desenvolvido pelo CSELT e Univ. Parma
- Completamente implementado em Java e Obedece às especificações da FIPA - *Foundation for Intelligent Physical Agents* (1997 / 2000)
- Utiliza classes já definidas em Java.

Serviços e requisitos

Serviços

- Gerenciador de projetos;
- Transporte de mensagens;
- Suporte às fases de desenvolvimento e depuração;
- Projetado para sustentar escalabilidade;

Requisitos

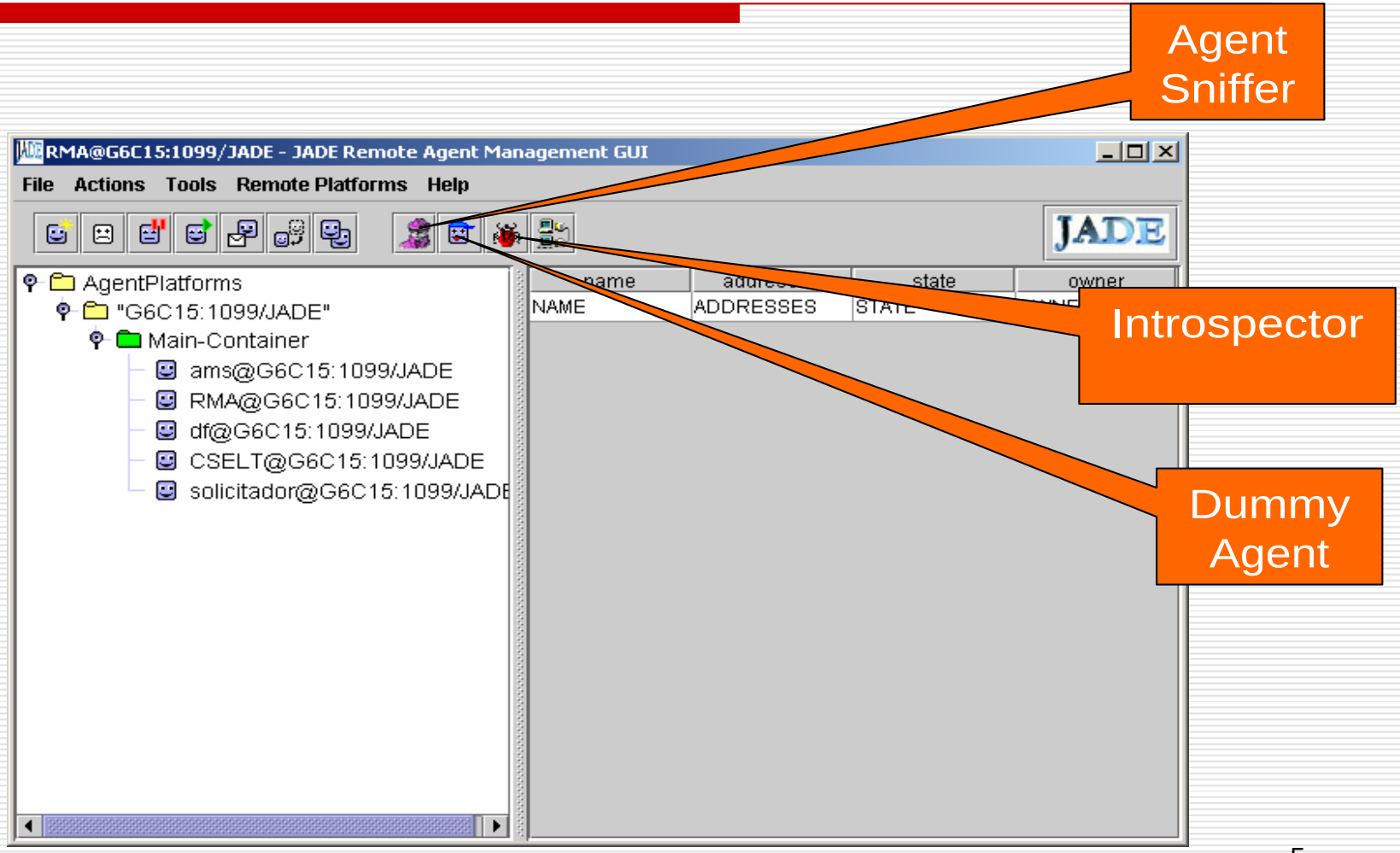
- Requisitos básicos para a execução do framework: versão 1.1x ou 1.2 de Java (JVM) com JDK instalado e 64 MB no mínimo.

Características

- Possui ferramentas de suporte;
  - Remote Monitoring(rms);
  - Agent Dummy;
  - Agent Sniffer;
  - Agent Introspector;

- Possui a habilidade de aceitar registros de outros *containers*
- Possui dois agentes especiais:
  - Agent Management System
    - Serviço de nomes, “autoridade da plataforma”
  - Directory Facilitator
    - Serviço de páginas amarelas

Telas e descrições



Funções gráficas

- Remote Monitoring Agent (RMA)
  - Funciona como uma console gráfica para gerenciamento e controle da plataforma;
  - É usada para monitorar e administrar o status de todos os componentes da plataforma distribuída, incluindo agentes e *containers*;
  - Serve para controlar o ciclo de vida dos agentes, conexões com plataformas remotas e a instalação de protocolos de transporte de mensagens

Funções gráficas

- Sniffer Agent
  - É usado para interceptar as mensagens ACL e exibir a conversação através de uma notação similar ao diagrama de seqüência da UML;
  - Útil para depuração de conversação entre agentes;
  - Permite que as conversações sejam salvas ou carregadas em arquivos.

Funções gráficas

- Introspector Agent
  - É usado para monitorar o ciclo de vida de um agente, suas mensagens ACL trocadas e seus comportamentos em execução.
  - Permite controlar a execução de um agente.

Funções gráficas

- Dummy Agent
  - É uma ferramenta utilizada para compor e enviar mensagens ACL para outros agentes, bem como para exibir as mensagens recebidas;
  - Permite que as mensagens sejam salvas ou carregadas em arquivos;
  - Uma alternativa para depuração das mensagens.

Funções gráficas

- Directory Facilitator (DF)
  - Representa o FIPA DF, o componente de páginas amarelas do sistema;
  - Permite registrar / de-registrar / modificar / buscar agentes e serviços;
  - Permite criar confederações de DF e realizar propagação de busca através de domínios e sub-domínios.

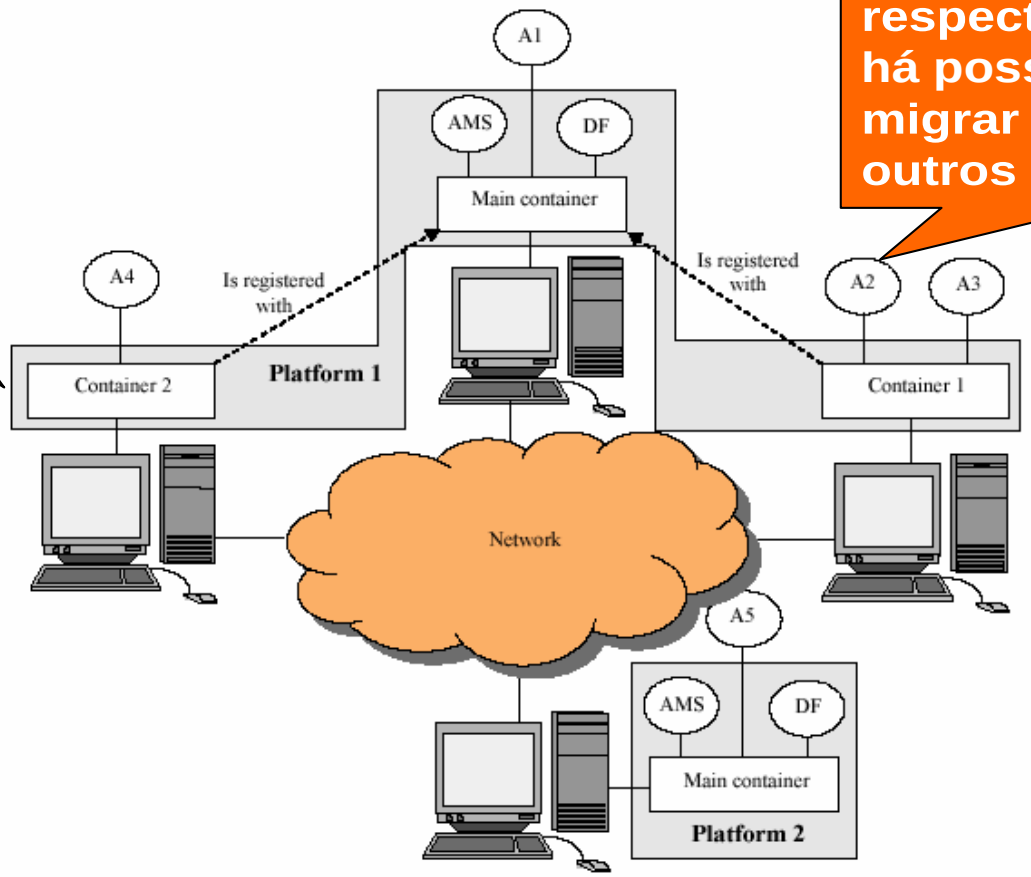
Main-container

- É baseado no conceito de *container*
  - Um *container* = instância do ambiente de execução JADE
  - Diferentes *containers* na mesma plataforma (1 JVM por *container*)
  - Diferentes agentes no mesmo *container*
  - Cada agente tem sua própria thread de execução
- Uma *Plataforma* é composta por um conjunto de *containers* ativos
- Cada plataforma possui, obrigatoriamente, um *Main Container*

Containers Distribuídos

Instância de jade rodando em um Host Distribuído

A1,A2,A3... Agentes rodando nos respectivos container, há possibilidade de migrar os agentes para outros containers



Agentes

- Forma de comunicação;
- Estados dos agentes;
- Heranças classe agent;
- Construção de um novo agente();
- AID
- Comportamento dos agentes;

Agentes

- Um agente em JADE é uma instância da classe *Agent*
- Heranças da classe *Agent:*
  - Características para realizar as interações básicas com a plataforma de agentes (registro, configuração, gerenciamento remoto, ...);
  - O conjunto básico de métodos que podem ser chamados para implementar o comportamento personalizado do agente (enviar/receber mensagens, usar protocolos de interação, ...)

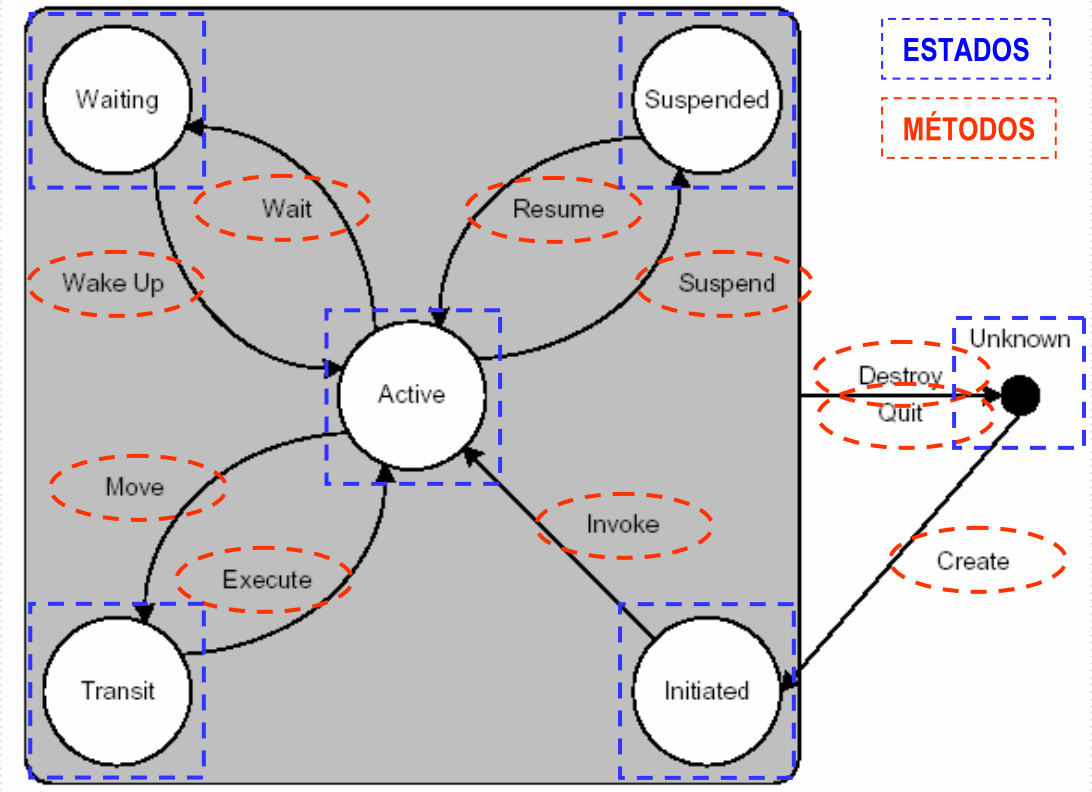
Agentes

- Um **AID** (*Agent Identifier*) é constituído por:
  - Um nome globalmente único:
    - <localname>@<hostname>:<port>/JADE
    - Ex: **ams@smeagol:1099/JADE**
  - Um conjunto de endereços de agentes
    - Endereços das plataformas em que o agente reside
    - Utilizados apenas quando agentes desejam se comunicar com agentes em outras plataformas
  - Agentes também possuem um "apelido"

Estado dos Agentes

A classe *Agent* representa os estados do agente através de constantes:

- AP_INITIATED
- AP_ACTIVE
- AP_SUSPENDED
- AP_WAITING
- AP_DELETED
- AP_TRANSIT
- AP_COPY



Exemplo de Agente

```
import jade.core.Agent;

public class AgenteSimples extends Agent {
  protected void setup() {
     System.out.println(“Oi Agente ” +
          getAID().getName() + “ está
  pronto!”);
  }
}
```

Comunicação dos Agentes

- Em JADE os agentes se comunicam através de passagem assíncrona de mensagens ACL
- A classe *ACLMessage* representa as mensagens de acordo com as especificações da FIPA
- Todos os parâmetros são pares ***palavra-chave: valor***
- Os valores dos parâmetros podem ser inseridos através dos métodos ***set()*** e podem ser lidos através de métodos ***get()***
- As performativas da FIPA são identificadas por constantes.

Comunicação dos Agentes

- Todo agente tem uma fila privada de mensagens ACL criada e preenchida pelo subsistema de comunicação de JADE
- O agente é informado sempre que uma mensagem é adicionada a sua lista de mensagens
- Cabe ao programador como se dará o tratamento dessas mensagens
- Se uma mensagem é enviada e o sub-sistema não consegue encontrar o destinatário, ele a envia para ser gerenciada pelo AMS

Criando/ Enviando mensagens

```
public void action() {
   ACLMessage msg = new
  ACLMessage(ACLMessage.INFORM);
   msg.addReceiver(new AID(“ze”, AID.ISLOCALNAME);
   msg.setLanguage(“portugol”);
   msg.setOntology(“Sistemas Distribuídos”);
   msg.setContent(“Comunicação entre processos”)
   // Chama o Método da classe Agent para enviar
  mensagens
   send(msg);
}
```

Mensagens em Acl

PERFORMATIVA
ENDEREÇAMENTO
COMUNICAÇÃO
MENSAGEM

```
QUERY-IF
  :sender solicitador@G6C15:1099/JADE
  :receiver CSELT@G6C15:1099/JADE
  :protocol fipa-query
  :conversation_id C2471722_1067382505429
  :reply_with
  :reply_by
  :language fipa-sl0
  :ontology curso
  :content  ((DISCIPLINAS
                (COMPUTACAO
                          :nome “Computação Distribuída"
                          :professor “Bosco”
                          :Horario “a tarde”
                (ENGENHARIA
                          :nome “Circuitos Digitais”
                             :professor “Carlos”
                          :Horario “a noite”))))
```

Instalação do Jade

- Download dos arquivos
  - http://jade.tilab.com/ ;
  - Registro para acesso aos downloads;
  - Arquivos
    - jadeAll.zip;
    - jadeBin.zip;
    - jadeDoc.zip
    - jadeSrc.zip
    - jadeExamples.zip

Note: que todos os binários devem ser executados usando jdk versão 1.4;

Rodando o Jade

- Como Main-Container
  - java jade.Boot [options] nome_agente: codigo
- Como Container
  - java jade.Boot –container [options]
    nome_agente: codigo compilado
- Sem classpath
  - java –jar lib\jade.jar –nomtp [options]
    Nome_agente: código_compilado

Options:
- **container**
- **gui**
- **mtp**
- **host**
- **port**
- **container-name**
- **nomobility**
- **version**
- **help**
- **conf**

Referências

- ***Java Agent DEvelopment Framework***
  - http://jade.cselt.it/
- **API de JADE**

  http://jade.cselt.it/doc/api/index.html
- **Documentação *on-line* de JADE**
  - http://jade.cselt.it/doc/index.html
- **Exercício proposto**
  - **Criar um agente o qual ele possa migrar em ambientes heterogêneos Ex: Main-Container rodando em Linux e um container em Windows. E fazer a comunicação entre os agentes utilizando as performativas do ACL.**