JADE: Java Agent Development Framework

Fábio V. Teixeira

***Abstract*—JADE is a software environment that provides a middleware for the development and execution of agent-based applications which can interoperate both in wired and wireless systems, in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. It supports these kinds of applications through a predefined programmable and extensible agent model in addition of a set of management and testing tools. The platform aims to simplify development while ensuring standard compliance through a comprehensive set of system services and agents, so JADE is one of the most used and promising agent development framework and it has been used to realize real systems in different application sectors.**

***Index Terms*—agent model, FIPA, interoperable intelligent multi-agent systems, middleware.**

I. Introdução

Há dez anos atrás, logo após a explosão da inteligência computacional, as tecnologias de agentes era consideradas uma das mais promissoras tecnologias de informação da época. No entanto, os pesquisadores perceberam que essa tecnologia não poderiam ser utilizadas em larga escala, dado que não existiam normas adequadas para apoiar a interoperabilidade entre elas, nem ambientes adequados para o desenvolvimento desse tipo de sistema [5], [8]. A fim de solucionar essa questão, recentemente, diferentes grupos de pesquisadores começaram a trabalhar no sentido de definir padrões para tecnologias de agentes e para a criação de ambientes de desenvolvimento de sistemas multi-agentes, o que originou dois dos resultados mais importantes desses dois campos: as especificações FIPA (*Foundation for Intelligent, Physical Agents*) e a plataforma JADE. As especificações definem um modelo de referência para uma plataforma de agentes e um conjunto de serviços que devem ser fornecidos ao se conceber sistemas multi-agentes interoperáveis, e o JADE é um ambiente de *software* (em conformidade com as especificações da FIPA) para construção de sistemas baseados nessa tecnologia, que emprega um conceito relativamente novo: o paradigma da programação orientada a agente [8]. Esse paradigma de *software* que traz conceitos da inteligência artificial para o domínio de sistemas distribuídos e, essencialmente modela uma aplicação como uma coleção de componentes chamados de agentes, cujas características são, entre outras, autonomia, pró-atividade e sociabilidade. Sendo autônomos eles podem de modo independente realizar tarefas complexas e, muitas vezes em longo prazo; sendo pró-ativos eles podem tomar a iniciativa para executar uma determinada tarefa, mesmo sem um estímulo explícito do utilizador; e sendo sociável eles podem interagir com outras entidades com a finalidade de alcançar o seu próprio objetivo ou um objetivo comum entre todos (cooperação e colaboração, respectivamente) [1].

Apesar da tecnologia de agentes já vir sendo há vários anos objeto de ampla discussão e investigação entre a comunidade científica, apenas recentemente começou a ser vista com algum grau de significância para aplicações comerciais. Sistemas multi-agentes estão sendo usados em uma variedade cada vez maior de aplicações, desde sistemas relativamente pequenos até sistemas de missão crítica para aplicações industriais. Exemplos de domínio industrial, onde esses sistemas tem sido proveitosamente empregados, inclui controle de processos, diagnóstico de sistema, manufatura, logística de transporte e gerenciamento de rede.

A plataforma JADE foi inicialmente desenvolvida pelo grupo de pesquisa e desenvolvimento da Telecom Itália em parceria com a Universidade de Parma, entretanto, há alguns anos passou a ser um projeto da comunidade e se tornou uma tecnologia open source (sob as normas da licença LPGL (*Lesser General Public License*)), dentre as mais comuns em utilização hoje.

II. Especificações FIPA

A FIPA é uma associação internacional de companhias e organizações que compartilham esforços a fim de produzir especificações para tecnologias de agentes genéricas. Ela promove um conjunto de tecnologias para diferentes áreas de aplicação de tal forma que os desenvolvedores possam integrá-las para criar sistemas complexos com um alto grau de interoperabilidade.

Os documentos da FIPA definem regras que permitem a uma sociedade de agente existir, operar e ser gerenciada. A partir desses princípios, a FIPA define um modelo de referência para uma plataforma de agente, de acordo com a Fig 1. Ele se baseia em três agentes especiais, responsáveis pela gerencia da plataforma e pela descrição da linguagem de conteúdo para gerenciamento de agentes e ontologias: o AMS (*Agent Management System*), o DF (*Directory Facilitator*) e o ACC (*Agent Communication Channel*) [1], [2], [4], [6], [8].

Trabalho entregue em junho de 2010, como requisito para obtenção da nota final da disciplina IA009 – Introdução a Teoria de Agentes ministrada pelo professor R. R. Gudwin.

F. V. Teixeira é membro do Laboratório de Controle e Automação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, Campinas-SP, Brasil (e-mail: fabiovt@feec.dca.unicamp.br).

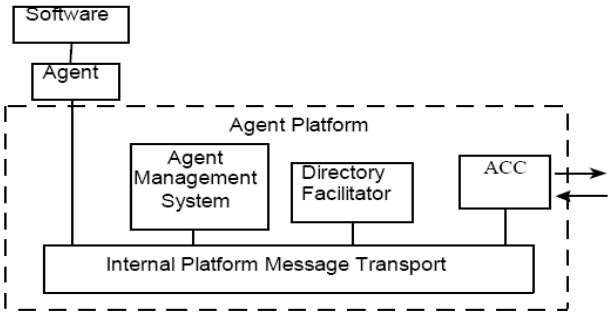

Fig. 1. Modelo de referência para plataformas de agentes definido pela FIPA

As especificações definem a FIPA-ACL para troca de mensagens, levando em consideração sua codificação e semântica, mas não os mecanismos de transporte de mensagens [4], [6], [8].

Outros aspectos abordados pela FIPA são questões relacionadas gerenciamento de ciclo de vida, mobilidade, segurança, ontologias, comunicação, entre outros [3]. No estado atual, a tecnologia JADE é capaz de atender por completo as especificações da FIPA, dado que a equipe do projeto esteve presente em todos os eventos realizados pela FIPA além de ter participado de vários processos de padronização que deram origem a diversos documentos de especificação [1].

III. A PLATAFORMA JADE E SUAS CARACTERÍSTICAS

O JADE se trata de um *framework* de *software*, independente de aplicação específica, capaz de prover funcionalidades de camada de *middleware*. Ele possui uma infra-estrutura bastante flexível para o desenvolvimento de aplicações onde o elemento de abstração é um agente de software. Para isso ele oferece suporte ao ciclo de vida e a lógica do núcleo de agentes em si, além de oferecer uma variedade de ferramentas gráficas que favorecem o desenvolvimento. A tecnologia é totalmente escrita em Java, o que traz benefícios a partir do enorme conjunto de recursos e bibliotecas oferecidas pela linguagem, que por sua vez oferece um vasto conjunto de abstrações de programação e possibilita a construção de sistemas baseados em agentes com competências relativamente mínimas em relação à teoria de agentes. Com isso, o programador pode economizar o tempo com o trabalho inicial necessário para construir uma infra-estrutura baseada em agentes, e dedicar seus esforços para o negócio da aplicação propriamente dito.

O JADE se baseia nos seguintes princípios [8]:

- Interoperabilidade: está em conformidade com as especificações FIPA, possibilitando que os agentes possam se comunicar com outros que não estejam executando em ambiente de execução JADE.
- Uniformidade e Portabilidade: provê aplicações com um conjunto homogêneo de APIs (*Application Programming Interfaces*) que são independentes da estrutura subjacente e da versão Java (o ambiente de execução JADE provê as mesmas APIs tanto para J2SE como para J2SE e J2ME e, em teoria, o desenvolvedor pode escolher o ambiente de execução Java em tempo de compilação).
- Facilidade de Uso: a complexidade do *middleware* é ocultada por trás de um simples e intuitivo conjunto de APIs.
- Filosofia *Pas-you-go*: programadores não são obrigados a usar todas as características providas pelo *middleware*. Aquelas que não serão utilizadas não requerem nenhum tipo de conhecimento a respeito por parte do programador, tampouco adiciona qualquer sobrecarga computacional.

As escolhas feitas para projetar o JADE foram inspiradas basicamente na inteligência, iniciativa, informação, nos recursos e no controle. O JADE é um sistema completamente distribuído onde habitam agentes. Esses agentes não podem prover *call-backs* ou referências do seu próprio objeto para outros agentes a fim de minimizar a possibilidade de outras entidades obterem controle de seus serviços. Além disso, cada agente deve ter sua própria *thread* de execução, provavelmente localizadas em diferentes máquinas remotas e capazes de se comunicar transparentemente uma com a outra. Elas devem ser capazes de gerenciar seu próprio ciclo de vida e decidir autonomamente quando executar uma tarefa.

A forma básica de comunicação entre agentes no JADE é a baseada em mensagens assíncronas. Um agente que deseja se comunicar deve apenas transmitir uma mensagem para um destino identificado (ou conjunto de destinos), ou seja, não existe nenhum tipo de dependência temporal entre o transmissor e o receptor; o fato do receptor não estar disponível, por exemplo, devido a uma conexão oscilante, não irá provocar nenhum tipo de problema funcional na comunicação. No *framework* não há a necessidade de obter a referência de objeto do agente receptor, mas sim apenas as identidades dos nomes que o sistema de transporte de mensagem é apto a converter para endereços de transporte adequados. É possível também que a identidade do receptor seja desconhecida para o remetente, que pode então utilizar um grupo de receptores através da definição de grupos de interesses (por exemplo, enviar uma mensagem para todos os agentes interessados em futebol). Além disso, essa forma de comunicação permite ao receptor a selecionar qual mensagem processar e qual descartar, assim como definir suas próprias prioridades de processamento. O método permite também o emissor a controlar sua *thread* de execução e então não ser bloqueado ate o receptor processar a mensagem. Finalmente, o mecanismo também provê uma vantagem interessante quando implementa comunicação *multicast* como uma operação atômica ao invés de varias chamadas de métodos consecutivas (uma operação de envio com uma lista de múltiplos receptores de mensagens ao invés de uma chamada de método para cada objeto remoto cujo se deseja realizar a comunicação). Apesar do uso desse tipo de comunicação, a segurança da plataforma é preservada, pois ela provê mecanismos próprios de autenticação e verifica os direitos e prioridades atribuídos a cada agente, ou seja, quando é necessário, a aplicação pode verificar a identidade do emissor da mensagem e prevenir ações relacionadas a ele que são limitadas ou não permitidas [1].

A estrutura das mensagens trocadas na comunicação entre agentes é baseada na linguagem ACL (*Agent Communication Language*) definida pela FIPA e contém campos tais como, variáveis que indicam o contexto ao qual a mensagem se refere e, o tempo limite que pode ser aguardado até a resposta ser recebida (*timeout*), visando suporte a interações complexas e conversas paralelas múltiplas [4], [6].

O sistema de agentes do JADE é baseado no paradigma *peer-to-peer*. Cada agente é identificado por um nome globalmente único, definido como AID (*Agent Identifier*) pela FIPA, que o JADE constrói concatenando um apelido definido pelo próprio usuário ao nome da plataforma. Ele pode ser associado ou desassociado a qualquer momento e permite descobrir dinamicamente outros agentes a partir dos serviços de páginas brancas e de páginas amarelas, providos pela AMS e pelo DF, respectivamente (*Naming Service*). Ou seja, um agente pode iniciar uma comunicação com qualquer outro agente a qualquer momento que deseje. Um AID é composto por um conjunto de campos que estão de acordo com a estrutura semântica definida pela FIPA. Os elementos mais básicos da AID são: o nome do agente e os endereços (os endereços dos agentes nada mais são do que endereços de transporte herdados da plataforma)

A tecnologia JADE suporta tanto mobilidade de código como mobilidade de estado de execução. Isto é, um agente pode parar de executar em um *host*, migrar para um *host* remoto diferente (sem a necessidade de ter o código do agente já instalado naquele *host*) e então reiniciar sua execução lá, a partir do mesmo ponto cujo qual foi interrompido. Esta funcionalidade permite a distribuição de carga computacional em tempo de execução, por exemplo, movendo agentes para máquinas menos sobrecarregadas sem qualquer impacto na aplicação.

JADE fornece suporte a ontologias e a linguagens de conteúdo. Por padrão, a verificação de ontologia e a codificação de conteúdo são executadas automaticamente pela plataforma, cabendo ao programador apenas selecionar a ontologia e/ou a linguagem de sua preferência. Entretanto, os programadores podem também optar pela criação de ontologias e linguagens de conteúdo mais adequadas à aplicação específica através de um rico suporte para conversão automática entre formatos string, incluindo XML e RDF (forma apropriada para transmitir informações), e objetos Java (forma apropriada para manipular informações). Para isto é fornecido ferramentas, que os permitem fazer isso graficamente.

Para aumentar a escalabilidade ou lidar com as restrições de ambientes com recursos limitados, JADE fornece a possibilidade de execução de múltiplas tarefas em paralelo dentro da mesma *thread* Java (cada agente executa em uma *thread* separada e esta por sua vez pode conter *threads* lógicas correspondentes a comportamentos paralelos). Várias tarefas elementares podem então ser combinadas para formar tarefas mais complexas, estruturadas como máquinas de estados finitos concorrentes.

O framework JADE pode ser executado em uma vasta classe de dispositivos, variando desde servidores até telefones celulares, sendo que nesse último o único requisito é suportar o Java MIDP (*Mobile Information Device Profile*) 1.0 (ou versões mais altas). A fim de tratar adequadamente as limitações de memória e de poder de processamento dos dispositivos móveis, assim como as características das redes sem fio, GPRS (*General Packet Radio Service*) em particular, em termos de largura de banda, latência, conectividade intermitente e variabilidade do endereçamento IP (*Internet Protocol*), e ao mesmo tempo ser eficiente quando executados em hosts de uma rede fixa, a plataforma pode ser configurada para adaptar-se a essas possíveis características do ambiente de implantação. Isso pode ser feito basicamente através da ativação de certos módulos, tornando possível a simulação de diferentes requisitos relacionados à conectividade, memória e poder de processamento [8].

Adicionalmente, o JADE possui uma série de características que o favorece em relação às outras tecnologias, tais como: suporte ao mecanismo *publish-subscribe*, onde agentes podem publicar seus serviços e solicitar à plataforma para serem notificados sobre todos seus eventos; recursos para depuração e monitoramento; interface *in-process*, que permite inicializar/controlar uma plataforma JADE e seus componentes distribuídos a partir de uma aplicação externa; integração com uma variedade de tecnologias baseadas na web, tais como applets, *web services*, entre outras; e *kernel* extensível, o que permite aos programadores adicionarem funcionalidade de acordo com suas necessidades e a aplicação específica.

IV. A ARQUITETURA JADE

O JADE possui vários elementos que o compõe. O *framework* é composto por *containers* de agentes e possivelmente podem estar distribuídos pela rede. Esses *containers* devem ser habitados por agentes e, correspondem ao processo Java que provê o ambiente de execução JADE juntamente com todos os seus serviços necessários para hospedar e executar esses agentes. O conjunto de todos os *containers* por sua vez, é chamado de plataforma e é esse todo que provê uma camada homogênea que oculta completamente dos agentes (a partir da aplicação) a complexidade e a diversidade das entidades subjacentes tais como *hardware*, sistema operacional, tipos de rede, JVM (Java *Virtual Machine*), etc.

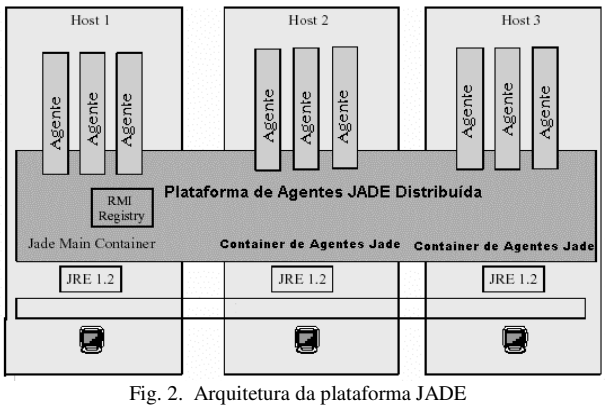

Fig. 2. Arquitetura da plataforma JADE

Dentre os *containers* da plataforma JADE, existe um especial chamado de Container Principal, que representa o ponto de partida para inicialização da plataforma, ou seja, é o primeiro container a ser inicializado e todos os outros devem juntar-se a ele através de um processo de registro, de acordo com a Fig. 2.

Como um ponto de partida, o Container Principal tem as seguintes responsabilidades especiais, ilustradas pela Fig. 3:

- Administrar o *Container Table* (CT), que é o registrador das referências de objeto e endereços de transporte de todos os outros nós *containers* que compõe a plataforma;
- Gerenciar a *Global Agent Descriptor Table* (GADT), que é o registrador de todos os agentes presentes na plataforma (incluindo seus estados correntes e suas localizações);
- Hospedar os três agentes especiais definidos no modelos de referencia da FIPA (AMS, DF e ACC).

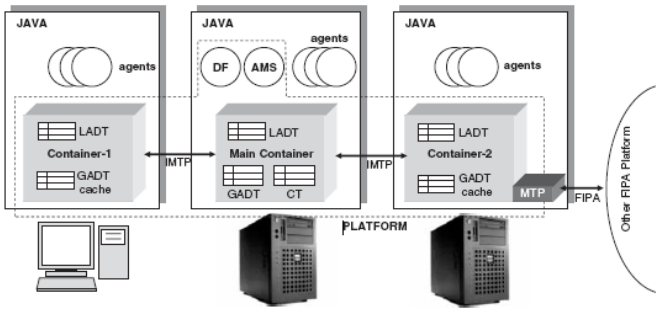

Fig. 3. Relacionamento entre os principais elementos arquiteturais do JADE

Quando a plataforma é inicializada, os três agentes especiais são automaticamente instanciados e inicializados pelo JADE. Seus papeis, definidos pelo padrão de Gerenciamento de Agente da FIPA, são descritos a seguir [2]:

- O AMS é o agente que supervisiona a plataforma inteira. Ele é o ponto de contato para todos os agentes que precisam interagir a fim de acessar as páginas brancas da plataforma e gerenciar seu ciclo de vida. Todo agente é solicitado a se registrar ao AMS para que possa obter um AID válido.
- O DF é o agente que implementa o serviço de páginas amarelas que é usado por qualquer agente que deseja registrar seus serviços ou buscar por outros disponíveis. O DF também aceita subscrições de agentes que desejam ser notificados quando um registro de serviço ou modificação for feita. DFs múltiplos podem ser inicializados concorrentemente a fim de distribuir o serviço de páginas amarelas através de vários domínios.
- O ACC é o agente capaz de prover o caminho para que haja o contato básico entre agentes dentro e fora da plataforma. Ele provê o método de comunicação default do sistema que oferece um serviço de roteamento de mensagens confiável, ordenado e preciso.

Uma questão comumente levantada se refere ao fato do Container Principal se tornar um gargalo para o sistema como um todo. Entretanto a própria arquitetura JADE trata essa questão ao prover um cache GADT, que cada container gerencia localmente. Com isso, as operações da plataforma, em geral, não envolvem o Container Principal, mas sim apenas o cache local e os dois *containers* hospedando os agentes envolvidos na operação (por exemplo, remetente e receptor da mensagem). Quando um *container* deve descobrir onde o destinatário de uma mensagem habita, ele primeiramente pesquisa em sua LADT (*Local Agent Descriptor Table*) e então, só no caso da busca falhar que o Container Principal é contatado para obtenção da referência remota adequada que, em seguida, é cacheada localmente para uso futuro. Devido ao dinamismo do sistema (agentes podem migrar, finalizar, ou novos agentes podem surgir), ocasionalmente o sistema pode utilizar-se de um valor cacheado legado, o que resultará em um endereço inválido. Neste caso, o container lança uma exceção e é forçado a fazer a atualização do seu cache levando em consideração o cache do container principal através da política LRU (*Least Recently Used*) para realizar as devidas substituições [1].

Embora o container principal não seja um ponto de gargalo, ele ainda é um ponto único de falha dentro da plataforma e para lidar com isso o JADE utiliza o Serviço de Replicação Principal que permite que o container principal seja replicado de forma a garantir que a plataforma permaneça operacional até mesmo no caso de falhas no próprio container principal. Com este serviço o administrador pode controlar o nível de tolerância a falha, o nível de escalabilidade e o nível de distribuição da plataforma.

V. Serviço de Transporte de Mensagem

O Serviço de Transporte de Mensagem é um dos três serviços mais importantes que uma plataforma de agentes precisa prover segundo a FIPA (os outros dois são providos pelo AMS e pelo DR) e é oferecido pelo ACC. Esse serviço é responsável pela gerência de todas as trocas de mensagens inter e intra-plataforma do sistema [2].

Para promover interoperabilidade entre diferentes plataformas, o JADE implementa todos os Protocolos de Transporte de Mensagens padrões (MTPs) definidos pela FIPA, onde cada um inclui a definição de um protocolo de transporte e uma codificação padrão para o encapsulamento da mensagem.

Por default, o JADE sempre inicia um MTP baseado no HTTP (*Hyper Text Transfer Protocol*) durante o processo de inicialização do Container Principal; nenhum MTP é ativado em *containers* normais. Isso então cria um *socket server* no *host* do Container Principal que fica escutando possíveis conexões de entrada HTTP. Sempre que uma conexão de entrada é estabelecida e uma mensagem válida recebida naquela conexão, o MTP rotea a mensagem para o seu destino final que, em geral, é um dos agentes localizados na plataforma distribuída. Internamente, a plataforma usa um protocolo de transporte proprietário chamado IMTP (*Internal Message Transport Protocol*); o JADE executa o roteamento de mensagens tanto para as mensagens de entrada como para as de saída usando uma tabela de roteamento *single-hop* que requer visibilidade IP direta entre *containers* [1]. O IMTP é

exclusivamente usado para troca de mensagens entre agentes habitando diferentes containers da mesma plataforma. Ele é consideravelmente diferente dos MTPs inter-plataformas, como o HTTP. Primeiramente, como ele é usado apenas para comunicação interna de plataformas, ele não precisa ser compatível com qualquer padrão FIPA; ele pode ser projetado para afirmar o desempenho da plataforma. O protocolo não é utilizado apenas para transportar mensagens, mas também para transportar comandos internos necessários para gerenciar a plataforma distribuída, tão bem como monitorar o estado de *containers* remotos. Por exemplo, ele pode ser usado tanto para transportar um comando ordenando o desligamento de um container, como para monitorar quando um container é finalizado ou se torna inalcançável.

JADE foi projetado para permitir a seleção do IMTP no momento de inicialização da plataforma. Por enquanto, duas implementações do IMTP estão disponíveis. Uma é baseada no Java RMI (*Remote Method Invocation*) e é a opção default. A segunda é baseada em um protocolo proprietário usando *sockets* TCP (*Transmission Control Protocol)* que resolve a questão da ausência de suporte para Java RMI. Ambas as implementações provêem varias opções de configurações para permitir a boa adequação do IMTP em relação às características de dispositivos e redes específicas [1].

VI. Princípios Básicos para o Desenvolvimento do Primeiro Agente JADE

O JADE pode ser obtido através do *download* a partir da página do projeto: http://jade.tilab.com. A distribuição principal é composta por cinco arquivos [3]:

- jadeBin.zip: contém apenas os arquivos Java pré-compilados (.jar) prontos para uso
- jadeDoc,zip: contém a documentação, incluindo os guias para administrador e para programadores.
- jadeExamples.zip: contém o código-fonte de vários exemplos
- jadeSrc.zip: contém todos os códigos-fonte de JADE
- jadeAll.zip: contém todos os arquivos citados acima

A instalação do *framework* é feita basicamente pela descompactação dos arquivos, e também, requer a configuração das variáveis de ambiente JADE_HOME e CLASSPATH, referenciando o diretório raiz do JADE e a biblioteca jade.jar, respectivamente, de acordo com a Fig. 4.

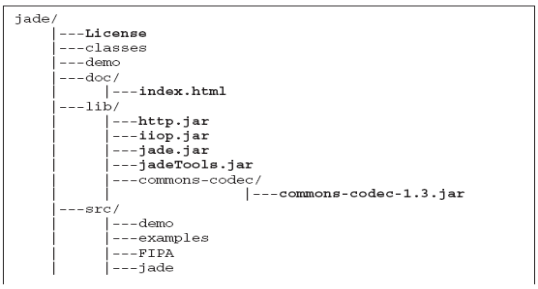

Fig. 4. Estrutura de diretórios JADE (apenas os arquivos mais relevantes)

Os fontes da plataforma JADE são organizados em uma hierarquia de pacotes e sub-pacotes Java, onde cada um, em princípio, contém um conjunto de classes e interfaces que implementam alguma funcionalidade especifica. Os pacotes principais são [1]:

- jade.core: implementa o *kernel* do JADE, o ambiente de execução distribuído que suporta a plataforma como um todo e suas ferramentas. Ele inclui a classe fundamental jade.core.Agent e todas as classes de execução básicas necessárias para implementar os *container*. Ele também inclui um conjunto de sub-pacotes cada um implementando um serviço especifico em nível de *kernel* (mobilidade, gerenciamento, comportamento, etc.
- jade.content: juntamente com seus sub-pacotes, contêm uma coleção de classes que provê suporte para criação e manipulação de expressões de conteúdo complexas de acordo com uma dada linguagem de conteúdo e ontologia. Isso inclui todos os mecanismos de codec necessários para a conversão automática entre o formato de representação interna JADE e o formato de transmissão de conteúdo de mensagem definido pela FIPA.
- jade.domain: contém a implementação dos agentes AMS e DF. Cada sub-pacote contém as classes representando as várias entidades de uma ontologia JADE pré-definida.
- jade.gui: contém alguns componentes Java de propósitos gerais e ícones que podem ser úteis para construir GUIs (*Graphical User Interfaces*) baseadas em Swing para agentes JADE.
- jade.imtp: contém a implementação do IMTP JADE. Em particular, o sub-pacote jade.imtp.rmi é o que implementa o IMTP default do JADE .
- jade.lang.acl: contém suporte para a FIPA-ACL incluindo a classe ACLMessage, o parser, o encoder e a classe helper para representação de *templates* das mensagens ACL.
- jade.mtp: contém o conjunto de interfaces Java que devem ser implementadas por um MTP JADE.
- Jade.wrapper: juntamente com as classes jade.coreProfile e jade.core.Runtime provê suporte para a interface in-process do JADE que permite aplicações Java externas a usarem o JADE como uma biblioteca.
- Jade.proto: contém a implementação de alguns protocolos de interação de propósito geral, incluindo os especificados pela FIPA.

A criação de sistemas agentes baseados em JADE apenas implica na criação de classes Java, sem ser necessário qualquer tipo de especialização em programação Java. Para a criação de um agente JADE basicamente é necessário estender a classe jade.core.Agent e implementar o método setup() (responsável por incluir as inicializações do agente) a partir da classe criada (várias instâncias do agente podem ser inicializadas em tempo de execução). Quando agentes são criados eles são atribuídos automaticamente a um GUID (*Global Unique Identifier*) e a um endereço de transporte que são usados para realizar o registro com o serviço de páginas brancas da plataforma.

No JADE, o AID que representa cada instância de agente é representado como uma instância da classe jade.core.AID. O método getAID() da classe Agent permite a recuperação do AID local. Um objeto AID inclui um GUID mais um número de endereços, assumindo a seguinte forma: <local-name>@<plataform-name>. Os endereços incluídos se tratam dos endereços das plataformas que o agente habita e são apenas usados quando ele necessita se comunicar com outro agente situado em uma plataforma diferente da qual se encontra. A classe AID provê métodos para recuperar o nome local (getLocalName()), o GUID (getName()) e os endereços (getAllAddresses()).

O código a seguir mostra um exemplo de um agente exibindo uma mensagem de apresentação [1]:

```
import jade.core.Agent;
public class HelloWorldAgent extends Agent {
   protected void setup() {
      // Printout a welcome message
      System.out.println("Hello World. I'm an agent!");
      System.out.println("My local-name is
   "+getAID().getLocalName());
      System.out.println("My GUID is "+getAID().getName());
      System.out.println("My addresses are:");
      Iterator it = getAID().getAllAddresses();
      while (it.hasNext()) {
         System.out.println("- "+it.next());
      }
   }
}
```

O nome local de um agente é atribuído no momento de inicialização das atividades pelo próprio criador do programa e deve ser único na plataforma. Se algum agente com o mesmo nome local já existir na plataforma, a criação do novo agente é prevenida.

Antes de executar o agente, como uma classe Java normal, um ambiente de execução JADE deve ter sido inicializado (a inicialização do ambiente de execução JADE pode ser feita também juntamente com a GUI de administração do *framework*, através do comando **`java jade.Boot –gui`**). Isso pode ser feito através do seguinte comando [1]:

java –classpath <JADE-classes>;. jade.Boot Fabio:HelloWorldAgent

O comando anterior inicializa o ambiente de execução JADE e diz a ele para lançar um agente de nome Fabio cuja classe é HelloWorldAgent (tanto as bibliotecas JADE quanto a classe HelloWorldAgent devem estar no CLASSPATH). Como não foi definido nenhum nome de plataforma, o JADE cria um por default usando o *host* local e a porta do Container Principal. Para atribuir um nome para a plataforma é necessário usar a opção –name, de acordo com o comando a seguir [1]:

java -classpath <JADE-classes> jade.Boot -name Ptfm Fabio:HelloWorldAgent

Após emitir a mensagem de apresentação e terminar todas suas tarefas, mesmo que não tenha mais nada para fazer, o agente permanecerá vivo. Para finalizá-lo o método doDelete() deve ser invocado, de forma similar ao setup. Outra possível opção é método takeDown(). Ele funciona de forma similar ao doDelete(), porém, antes de finalizar o agente ele executa várias operações de limpeza na plataforma como um todo.

VII. CONCLUSÃO

A tecnologia JADE foi um dos primeiros *frameworks* a considerar as especificações FIPA para o desenvolvimento de agentes de *software*. Atualmente o *framework* se encontra na versão 4.0 e os atuais parceiros do projeto, responsáveis por gerenciá-lo e supervisioná-lo, são os grupos: Telecom Itália, Motorola, Whitestein Technologies AG, Profactor GmbH, and France Telecom R&D [3].

JADE é capaz de juntar abstração e eficiência a fim de facilitar o trabalho do desenvolvedor, além de ser totalmente escrito em uma linguagem bem conhecida, o Java, que oferece uma variedade de funcionalidades e recursos favoráveis. Ele provê um modelo de agentes primitivos bastante geral, o que permite a implementação de arquiteturas de agentes robustas e sofisticadas [7], [8]. A tecnologia é *open source* e vem atraindo cada vez mais adeptos e membros da comunidade, que vêm trabalhando continuamente no sentido de mantê-la, visando a sua inclusão de forma mais ampla no âmbito das aplicações comerciais e industriais. Apesar de estar sendo cada vez mais aceita, ainda sofre bastante resistência em muitos setores, principalmente por estar relacionada à inteligência artificial, cujos benefícios ainda são mais sólidos no meio cientifico do que no meio industrial. Essa é uma das grandes razões pela qual não é feita a padronização da programação orientada a agentes na indústria e pela qual a popularização da tecnologia é dificultada.

REFERÊNCIAS

[1] *BELLIFEMINE, Fabio; CAIRE, Giovanni; GREENWOOD, Dominic.* ***Developing Multi-Agent Systems with JADE.*** *Londres: Wiley, 2007.*

[2] SILVA, Leonardo. **Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework.** Fortaleza: 2002.

[3] TELECOM ITALIA. **JADE.** Disponível em: <http://jade.tilab.com/>. Acesso em: 20 jun. 2010.

[4] BELLIFEMINE, Fabio; POGGI, Agostino; RIMASSA, Giovanni. Developing Multi-agent Systems with JADE. **Intelligent Agents VII**, Berlin, p.89-103, 2001.

[5] BELLIFEMINE, Fabio; POGGI, Agostino; RIMASSA, Giovanni. Developing multi-agent systems with a FIPA-compliant agent framework. **Software—practice And Experience,** Torino, p. 103-128. 2001.

[6] BELLIFEMINE, Fabio; POGGI, Agostino; RIMASSA, Giovanni. **JADE – A FIPA-compliant agent framework.** Parma: 2002.

[7] BELLIFEMINE, Fabio et al. JADE: A software framework for developing multi-agent applications. Lessons learned. **Information And Software Technology**, Karlskrona, n., p.10-21, 2007.

[8] BELLIFEMINE, Fabio et al. JADE - A JAVA AGENT DEVELOPMENT FRAMEWORK.