UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Licenciatura em Engenharia
de Informática e Computadores

(2001/2002)

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

TRABALHO FINAL DE CURSO

– ADAP, Plataforma de Agentes Distribuída –

RELATÓRIO FINAL

N.º da Proposta: 181

Professor Orientador:

Alberto M. Rodrigues da Silva _____(assinatura)_____

Co-Orientador:

_____(assinatura)_____

Professor Acompanhante:

_____(assinatura)_____

Alunos:

Hugo Miguel Álvaro Manguinhas _____(assinatura)_____

Francisco José Andrade Costeira _____(assinatura)_____

PLATAFORMA DE AGENTES DISTRIBUÍDA

AGRADECIMENTOS

Ao Prof. Alberto M. Rodrigues da Silva, por ter orientado este trabalho, contribuindo através do seu conhecimento em plataformas e metodologias de desenvolvimento, sem os quais teria sido impossível a construção de uma plataforma com os requisitos propostos.

Ao aluno José Alexandre Antunes Faria pelo seu apoio na construção da plataforma inicial, proposta para a cadeira de Introdução aos Agentes Autónomos, leccionada pela Prof. Ana Paiva. O José, através do seu espírito crítico, proporcionou uma visão diferente, permitindo a construção de uma plataforma mais robusta e à luz de conhecimentos arquitecturais mais avançados.

À professora Ana Paiva, pela transmissão dos conhecimentos base para a construção de uma plataforma de agentes, sem os quais, não teríamos desenvolvido este trabalho.

A todos os colegas do INESC (Instituto de Engenharia e Sistemas Computacionais) e do IST (Instituto Superior Técnico), pela ajuda diversa, mostrando-se sempre disponíveis para a discussão e troca de conhecimento.

Finalmente, não poderíamos deixar de agradecer às nossas famílias, nossos pais e irmãos, pelo seu apoio diário, partilhando connosco todos os momentos no período em que decorreu o trabalho.

Lisboa, 30 de Setembro de 2002

Hugo Miguel Álvaro Manguinhas

Francisco José Andrade Costeira

RESUMO

Este trabalho consiste na concepção e no desenvolvimento de uma plataforma distribuída para agentes (MAS – *Mobile Agent System*), tanto estacionários como móveis, responsável pelo suporte, execução e gestão dos mesmos. Entende-se como plataforma um ambiente de gestão de agentes suficientemente genérica que permita ao programador abstrair-se de diversos problemas existentes no desenvolvimento de soluções. Uma plataforma de agentes móveis suporta a execução e mobilidade de modo a permitir o uso de agentes móveis. Esta garante principalmente a execução homogénea de agentes em qualquer servidor de agentes, constituinte da plataforma, bem como, a migração transparente de agentes entre locais.

Entre outras características, uma plataforma de agentes pode garantir a persistência dos agentes e da informação por si acedida, fornecer facilidades de comunicação entre agentes e entre aplicações e agentes, garantir a segurança de modo a que os agentes não interfiram negativamente com a plataforma ou mesmos uns com os outros, fornecer facilidades de acesso a recursos externos à plataforma e gerir as identidades dos agentes e dos servidores de agentes a esta ligados.

PALAVRAS-CHAVE

Agentes; MAS (*Mobile Agent Systems*); Plataformas distribuídas para Agentes; Aplicações baseadas em Agentes (ABA - A*gent Based Applications*); Segurança de sistemas e de agentes; JAVA; Metodologia RUP (*Rational Unified Process*); UML (*Unified Modeling Language*); Padrões de desenho; Estilos Arquitecturais.

ÍNDICE

LISTA DE FIGURAS

LISTA DE SIGLAS

ABA – Agent Based Applications.

ACF – All-purpose Cipher Framework.

ADAP – Advanced Distributed Agent Platform.

AMF – All-purpose Marshalling Framework.

APL – Agent programming language.

ATF – Advanced Testing Framework.

AWS – Advanced Windowing System.

FIPA – Foundation for Intelligent Physical Agents.

HTTP – Hypertext transfer protocol.

IAA – Introdução aos Agentes Autónomos.

JADE – Java Agent Development Environment.

MAS – Mobile Agent Systems.

MTS – Monitor and Tracking System.

ORB – Object Request Broker.

RMI – Remote Method Invocation.

RTTI – Run-Time Type Identification.

RUP – Rational Unified Process.

TAS – Transport and Authentication System.

UML – Unified Modeling Language.

VDS – Virtual Database System.

WIAA – Workshop of Introduction to Autonomous Agents.

XML – Extended Markup Language.

Capítulo 1

INTRODUÇÃO

RESUMO

Neste capítulo serão apresentados os objectivos e motivações para o desenvolvimento deste trabalho. É ainda apresentado a integração do problema no contexto actual.

1.1 MOTIVAÇÃO

Necessidade de criar uma ferramenta para gestão de agentes (não apenas criação, mas também, manutenção) suficientemente genérica que permitisse que o programador se abstraísse de problemas típicos de sistemas operativos (criação de *threads*, gestão de *sockets*, ligações e troca de mensagens, etc.). Deverá ainda permitir toda a gestão em modo de execução, permitindo que a mesma plataforma sirva diversas e variadas aplicações baseadas em agentes, sem interromper a sua execução.

Pessoalmente, o que nos levou a desenvolver esta plataforma parte de um desejo nosso de aprofundar o conhecimento sobre plataformas de agentes e de aplicações baseadas em agentes.

1.2 CONTEXTUALIZAÇÃO DO PROBLEMA

Ao longo da evolução da informática existiu a necessidade da construção de um paradigma que melhor se integrasse às necessidades dos programadores, um paradigma de fácil compreensão, que permitisse grande flexibilidade, e acima de tudo, que permitisse um rápido desenvolvimento.

Um destes paradigmas, consiste no paradigma de aplicações baseadas em agentes (ABA - *agent based aplications*), o qual permite uma abstracção superior em termos de desenvolvimento de aplicações. Assim, o desenvolvimento de uma plataforma que permita a execução, manutenção e gestão destes agentes seria uma mais valia para o desenvolvimento de aplicações informáticas.

Na última década, a distribuição foi um dos temas que maior interesse despertou e que se traduziu numa evolução constante nos sistemas computacionais, assim, pretendemos que para além das características referidas, a plataforma também suporte esta distribuição.

Historicamente as políticas de segurança têm origem nos sistemas militares, onde a principal ameaça consiste na divulgação de informação não autorizada. No campo comercial se bem que o acesso à informação constitua uma ameaça significativa, a integridade da informação assume grande relevância. É evidente que a informação nas bases de dados, se ligada a informação com implicações económicas, tem uma enorme probabilidade de ser alvo de ataques. Nos sistemas informáticos (e, em particular, nas aplicações baseadas em agentes) está, portanto, em causa não só o confinamento da informação sensível a domínios de segurança como a protecção da sua integridade. Assim, a segurança é um dos assuntos de maior interesse, sendo também focado neste trabalho.

1.3 DESCRIÇÃO DE PLATAFORMA

Uma plataforma consiste num ambiente para execução e gestão de agentes, (não apenas criação, mas também, manutenção) suficientemente genérica que permita ao programador abstrair-se dos problemas típicos de sistemas operativos, existentes no desenvolvimento de agentes ou aplicações baseadas em agentes. O programador apenas se deve preocupar com as funcionalidades e tarefas que pretende que cada agente seja capaz de resolver.

Uma plataforma de agentes móveis consiste num sistema que suporta a execução e mobilidade de modo a permitir o uso de agentes móveis. Esta garante principalmente a execução homogénea de agentes em qualquer servidor de agentes, constituinte da plataforma, bem como, a migração transparente de agentes entre locais (ou seja, transfere a execução de um dado agente de um servidor de agentes para outro).

Uma plataforma de agentes pode também garantir a persistência dos agentes e da informação por si acedida, fornecer facilidades de comunicação entre agentes e entre aplicações e agentes, garantir a segurança de modo a que os agentes não interfiram negativamente com a plataforma ou mesmos uns com os outros, fornecer facilidades de acesso a recursos externos à plataforma e gerir as identidades dos agentes e dos servidores de agentes a esta ligados.

1.4 OBJECTIVO DO TRABALHO

Este trabalho consiste na concepção e no desenvolvimento de uma plataforma distribuída para agentes (MAS – *Mobile Agent System*), tanto estacionários como móveis, responsável pelo suporte e gestão dos mesmos. Esta plataforma permite ainda o desenvolvimento e manutenção de aplicações baseadas em agentes (ABA – *Agent based Aplications*).

De forma a melhor compreender todas as funcionalidades a ser suportadas pela plataforma, bem como funcionalidades requeridas por aplicações baseadas em agentes, foi necessário efectuar um intenso trabalho de pesquisa sobre algumas das plataformas existentes. Apresentamos algumas das plataformas analisadas no decorrer do nosso trabalho de pesquisa: *AgentSpace*, *Zeus*, *Jade*, *Aglets Workbench*, *Telescript*, *D'Agents*, *Odyssey* e *Jumping Beans*.

Após a análise efectuada a estas plataformas de agentes, e tendo em consideração que queremos um plataforma robusta e que permita um rápido desenvolvimento de aplicações baseadas em agentes, decidimos quais as principais características que deverão orientar o desenvolvimento da plataforma, designadamente: distribuição, persistência de código e dados de agentes, comunicação entre agentes, acesso a recursos externos, flexibilidade, mobilidade de agentes e segurança.

Um dos factores relevantes na avaliação da plataforma a ser desenvolvida será o seu desempenho na resolução de problemas reais. Assim existirá um elevado esforço da nossa parte no desenvolvimento de algumas aplicações e seus agentes de modo a demonstrar as potencialidades da mesma. Considera-se ainda de elevada importância a compreensão das aplicações orientadas a agentes existentes de forma a melhor as satisfazer.

- ***Filosofia adoptada***

  Toda a filosofia de agentes encontra-se confinada à plataforma, ou seja, cabe a esta gerir de forma transparente os seus agentes. As aplicações baseadas em agentes apenas usufruem das funcionalidades disponibilizadas por esses agentes. Esta filosofia permite que exista um conjunto diversificado de investigadores (programadores), no desenvolvimento de diversos agentes, que à partida podem não ter um objectivo comum, mas com o seu desenvolvimento e evolução poderão desenvolver uma sociedade extremamente complexa e inteligente.

  Clarificando a ideia colocamos a seguinte abstracção:

  Imaginemos um ecossistema, que é constituído por um diverso e variado conjunto de habitats, cada um destes albergando um conjunto de seres vivos. O ecossistema suporta toda a existência dos seres vivos fornecendo-lhes tudo o que estes necessitam para sobreviver (comida, bebida e abrigo), podendo estes depender de outros seres para a sua sobrevivência e comunicar com outros seres da mesma ou de outras espécies. O habitat corresponde ao local físico onde cada ser vivo vive.

  No âmbito da plataforma, um ecossistema corresponde à própria plataforma, composta por vários servidores (habitats), cada um destes capaz de suportar vários agentes (seres vivos). Os agentes podem comunicar entre si e podem depender de outros para o seu desempenho na resolução de um problema.

1.5 DESCRIÇÃO DO CONTEÚDO E ESTRUTURA DO RELATÓRIO

Este documento descreve todo o trabalho efectuado no âmbito do Trabalho Final de Curso dos alunos, Hugo Miguel Álvaro Manguinhas e Francisco José Andrade Costeira intitulado ADAP, Plataforma de Agentes Distribuída. Começa por introduzir ao leitor os conceitos essenciais para a compreensão do trabalho em questão, continua com uma análise das plataformas existentes, descrevendo ainda a captura dos requisitos, sua análise e desenho do sistema. Por fim, são descritos os resultados obtidos com o desenvolvimento deste trabalho e consequente conclusão do mesmo.

Capítulo 2

CONCEITOS

RESUMO

Neste capítulo serão apresentados alguns conceitos necessários para uma melhor compreensão do trabalho em questão.

2.1 INTRODUÇÃO

Os conceitos envolvidos neste trabalho englobam algumas áreas da informática, desta forma pretendemos ser breves na sua descrição e explicação. De forma a não tornar excessivamente extensa pretendemos apenas apresentar os conceitos essenciais, explicando também conceitos que não são do conhecimento comum de um engenheiro de informática.

2.2 AGENTES

2.2.1 Definição de Agente

Desde o início dos anos oitenta que o termo agente parece ter entrado na gíria do mundo dos computadores, sendo numerosas e divergentes as perspectivas dadas aos agentes, não existindo assim uma definição consensual para a designação de um "agente de software".

De acordo com definições apresentadas por diversos investigadores do ramo é natural que o consenso relativo a uma noção única seja difícil de atingir. No entanto, pode observar-se na maioria destas definições, que quando se fala de agente (de software) referimo-nos a uma das entidades (programas) activas que observam o mundo, actuando sobre ele. Estas entidades fazem parte de um sistema (dito multi-agente), onde outras entidades existem e se comunicam. Um exemplo de um desses sistemas poderá ser esta plataforma.

Outra definição consiste em encarar um agente como algo que raciocina sobre o ambiente, sobre os outros agentes e decide racionalmente quais são os objectivos a alcançar ou as acções a empreender nesse ambiente, isto é, possui critérios de avaliação e de selecção.

2.2.2 Noção de agente utilizada

Um agente é um actor fundamental num domínio. Ele combina uma ou mais capacidades de serviço num modelo de execução unificado e integrado que pode incluir acesso a software externo, utilizadores humanos e facilidades de comunicação. Os agentes são também processos computacionais que funcionam contínua e autonomamente, que se deslocam de um lugar para outro num ambiente onde outros processos ocorrem e outros agentes existem.

2.2.3 Importância dos agentes

Agentes são uma abstracção e como tal, servem para lidar com a complexidade dos problemas que surgem no desenvolvimento de aplicações e na resolução de problemas. Para além de ser fácil descrever os agentes através de comportamentos, existem ainda características que os agentes possuem, que facilitam o modo como se lida com os problemas (ex. comunicação).

2.2.4 Características dos agentes

- ***Acompanhamento***

  Capacidade de controlo sobre o ambiente envolvente de forma a garantir a autonomia da realização de tarefas.

- ***Adaptação***

  Capacidade do agente se adaptar e aprender com mudanças que ocorrem no meio onde se encontra inserido.

- ***Autonomia***

  Capacidade de operação sem a intervenção directa dos seres humanos e de controle sobre as suas acções e estados internos.

- ***Benevolência***

  Quando não existem objectivos conflituosos e podem executar o que lhes é pedido.

- ***Carácter***

  Possuidor de veracidade e emotividade.

- ***Colaboração***

  Capacidade de empenhamento, isto é, não obediência cega a ordens, e de modificação dos pedidos através da sua clarificação para o alcance de objectivos comuns.

- ***Comunicabilidade***

  Capacidade de interacção comum utilizador (ou outros agentes) para receber instruções (de delegação) e para informar sobre o estado da realização de uma tarefa.

- ***Continuidade***

  Capacidade de ter processos em execução continua (do ponto de vista temporal).

- ***Delegação***

  Execução de um conjunto de tarefas (actuação) em nome de um utilizador (ou outro agente) e sob sua aprovação (propriedade fundamental da agência).

- ***Emotividade***

  Capacidade de exibição de estados emocionais.

- ***Personalidade***

  Capacidade de exibir um comportamento que demonstra estar de acordo com uma dada personalidade.

- ***Intencionalidade***

  Capacidade de ter um comportamento que pode ser guiado por objectivos (realizar uma acção), e explicado através das noções crenças, desejos e intenções.

- ***Mobilidade***

  Capacidade de deslocação (transporte) de sitio (frequentemente uma máquina) para outro durante a sua execução, e tendo em atenção aspectos como a privacidade e a segurança.

- ***Pró-Actividade***

  Capacidade de exibição de um comportamento dirigido por objectivos e de tomada de iniciativas.

- ***Reactividade***

  Capacidade de reacção ao ambiente em que o agente está inserido, bem como, às suas mudanças. Existe um conjunto de estímulos aos quais o agente reage.

- ***Racionalidade***

  Capacidade de intervenção de acordo com critérios de avaliação (utilidade) das suas acções, e de justificação das decisões.

- ***Aprendizagem***

  Capacidade de aprender à medida da sua utilização, reconhecendo padrões de comportamentos, padrões de preferências etc., actualizando a sua base de conhecimento.

- ***Sociabilidade***

  Possibilidade de interacção com outros agentes através de uma linguagem de comunicação.

- ***Persistência***

  Capacidade de existir para além do tempo de execução.

- ***Veracidade***

  Propriedade de não comunicação de informações falsas.

2.2.5 Classificação de Agentes

A classificação adoptada pela grande maioria dos investigadores encara a existência de dois tipos distintos de agentes, agentes reactivos e deliberativos.

- ***Agentes Reactivos***

  São geralmente concebidos como soluções para problemas, e baseiam-se em modelos de uma organização biológica ou etológica, como a sociedade de formigas. Nestas sociedades, podemos observar a emergência de alguns comportamentos inteligentes, tornados evidentes ao nível da sociedade, mas jamais ao nível do agente. Um agente reactivo funciona segundo um modelo estímulo/resposta, sendo a sua construção realizada em geral num nível sub simbólico. A representação do ambiente e dos outros agentes não é explícita, isto é o agente não é capaz de se lembrar do que lhe aconteceu no passado (falta de memória), nem de planificar as suas acções no futuro. Cada agente toma conhecimento das acções e dos comportamentos dos outros percepcionando as modificações no ambiente. Não existe comunicação directa entre os agentes, e em geral as sociedades destes agentes são constituídas por um grande número (milhares) de elementos simples.

- ***Agentes Deliberativos***

  Têm uma representação explícita do ambiente e possuem capacidades de decisão baseada nos seus objectivos. Assim, são por vezes designados de deliberativos, reflexivos, racionais ou baseados em planos. Produzem frequentemente a solução correcta, são inflexíveis ao ambiente em mudança (problema errado, solução certa), e inspiram-se em modelos de organizações sociais humanas, como os grupos, as hierarquias e os mercados. Estes agentes têm uma representação explícita do ambiente e dos outros agentes, e ao contrário dos agentes reactivos têm memória, isto e podem raciocinar sobre o passado e planear as suas acções no futuro, em suma resolver problemas. A sua construção realiza-se no nível simbólico, recorrendo a noções cognitivas, como os estados ou atitudes mentais, tais como crenças, os desejos e as intenções. No que respeita às suas interacções, comunicam directamente pelo envio e passagem de mensagens. As suas actividades de percepção e de comunicação são distintas, em oposição ao que acontece com os agentes reactivos. Normalmente, o número de agentes numa sociedade deste tipo não é elevado, uma ou duas dezenas.

2.2.6 Agentes Móveis

Agentes capazes de interromper o seu estado de execução numa determinada máquina e por sua iniciativa ou de outrem, continuam a sua tarefa noutra máquina distinta.

- ***Vantagens***

  A mobilidade permite a redução de carga na rede, introduz uma diminuição da latência (pois é permitido ao agente mover-se para o local onde ele poderá realizar a sua tarefa mais eficientemente), permite uma execução assíncrona e autónoma, permite uma adaptação dinâmica ao ambiente de execução e fornece uma maior robustez e heterogeneidade.

2.2.7 Sociedade de Agentes

O conceito de organização não deve ser considerado como construído pelas relações estruturais entre os elementos do grupo de agentes (estrutura), nem por um conjunto de limitações às suas actividades (regras de comportamento), mas sim ser algo que está embebido nas crenças, intenções e compromissos dos agentes individuais. Assim, a noção de organização está associada à de um grupo de agentes com compromissos mútuos, globais, crenças mútuas e intenções partilhas tal que agem conjuntamente para atingir um dado objectivo, podendo estar organizados de acordo com uma dada estrutura.

A sociedade de agentes é algo mais geral e pode inclusive ser visto como um conjunto de agentes ou de um ou mais grupos ou organizações de agentes que estão sujeitos às mesmas regras sociais podendo mesmo estes grupos e organizações interagir entre si.

A criação de sociedades de agentes em detrimento da criação de um simples agente surge por diversas razões, de entre as quais:

- Da necessidade de adoptar uma solução distribuída para a resolução de problemas complexos;
- Da necessidade de criar modelos de sociedades para explicar determinados factores de emergência de comportamento social; e
- Da necessidade de simular sociedades reais para fins educacionais ou mesmo de entretenimento.

2.2.8 Comunicação entre Agentes

A comunicação entre agentes é um dos aspectos mais importantes da construção de sociedades de agentes. Em geral, a comunicação entre duas entidades pode ser vista como uma troca intencional de informação originada pela produção de símbolos e a sua percepção derivada de um sistema convencional de sinais. Esta comunicação é o que suporta a resolução cooperativa de problemas, e sem ela não é possível sincronizar as acções dos agentes e resolver os conflitos de recursos e de objectivos.

Existem dois tipos principais de comunicação, a comunicação por envio de mensagens e a comunicação por partilha de informações. Contrariamente à comunicação por partilha de

informação, que foi idealizada em redor dos sistemas periciais (fontes de conhecimento) cooperantes, onde os componentes não estão em ligação directa, mas comunicam através de uma estrutura de dados partilhada (o quadro preto), na comunicação por envio de mensagens, os agentes estão em ligação directa e enviam as suas mensagens directamente e explicitamente ao destinatário. Neste modo atinge-se uma distribuição total, quer do conhecimento, dos resultados ou dos métodos utilizados para a resolução do problema.

2.3 SEGURANÇA

O estabelecimento de uma política de segurança em qualquer situação real advém da necessidade de proteger um bem de um determinado conjunto de ameaças, como o acesso ou modificação de alguma informação. A segurança de um sistema de informático baseia-se na definição global de uma estratégia ou política de segurança que se apoia em diversos mecanismos de protecção à informação e que deverá ter em conta o valor potencial dessa informação, as ameaças esperadas, as formas de ataque e os custos associados à implementação.

2.3.1 Segurança dos agentes

A segurança a nível do agente prende-se com, aspectos de confidencialidade da informação guardada pelo agentes (por exemplo, um agente pode conter informação pessoal de um dado utilizador) e integridade do próprio agente (proteger de adulterações o código e os dados do agente).

2.3.2 Segurança da plataforma

A segurança a nível da plataforma corresponde à construção de uma política de segurança que garanta que todos os intervenientes (servidores, clientes, etc.) da mesma se encontrem devidamente autenticados perante esta. Da interacção entre agentes e plataforma poderão surgir aspectos de segurança que deverão ser contemplados.

2.3.3 Criptografia simétrica (chave secreta)

Este mecanismo é utilizado sempre que os dois interlocutores da comunicação conhecem as chaves a ser utilizadas, permitindo uma cifra e decifra rápida e eficiente.

2.3.4 Assinatura digital

Este mecanismo é importante, na medida em, que não é necessário a cifra da informação por completo, apenas um resumo do texto é cifrado e depois verificada a sua integridade, qualquer ataque à informação será infrutífero. Este método é utilizado sempre que é necessário uma autenticação do emissor e a informação transmitida não é confidencial.

Capítulo 3

CONTEXTO DO TRABALHO

RESUMO

Este capítulo começa por familiarizar o leitor com algumas das plataformas desenvolvidas antes da elaboração deste trabalho. Em seguida irá apresentar as equivalências e divergências em relação a estas plataformas, bem como, algumas inovações apresentadas.

3.1 INTRODUÇÃO

Previamente à elaboração deste trabalho existiu uma enorme preocupação em criar plataformas com as mesmas características da plataforma que desenvolvemos. Algumas destas são por exemplo a *Aglets Workbench*, *D'Agents*, *Grasshopper*, *Odyssey*, *Tacoma*, *Concordia*, *Jumping Beans*, *AgentSpace*, *Mole*, *MOA* e *PageSpace*.

A plataforma *Telescript* foi uma referência para a construção destas plataformas sendo mesmo considerada um *standard* devido ao seu avanço e inovação no âmbito de agentes.

3.2 ARQUITECTURAS EXISTENTES

Apresentamos em seguida uma pequena descrição de plataformas desenvolvidas no passado que se mostraram relevantes na concepção deste trabalho, quer pelo seu carácter único, quer pela inovação acrescentada no âmbito das plataformas distribuídas.

3.2.1 Telescript

Plataforma desenvolvida pela *General Magic*, foi uma das primeiras plataformas comerciais distribuídas com suporte para agentes móveis. Esta plataforma apresenta um grande número de funcionalidades implementadas e disponíveis para os agentes como migração, encontros, sistema de permissões, autoridades, ligações, noção de locais, bem como outros. Apresentou um grande desenvolvimento a nível de segurança, mobilidade, persistência e comunicação.

No entanto uma das suas grandes falhas corresponde à sua incapacidade de carregamento de agentes em tempo de execução.

3.2.2 Zeus, Agent Building Toolkit

Plataforma desenvolvida pela *British Telecom* para a construção de aplicações baseadas em agentes colaborativos (agentes de software autónomos que tipicamente interagem, partilham informações e cooperam com um grupo de outros agentes com o intuito de resolverem algum tipo de problema, formando assim uma sociedade de agentes).

3.2.3 Aglets Workbench

Plataforma desenvolvida pela IBM^{TM}, foi uma das primeiras plataformas desenvolvidas com tecnologia JAVA^{TM}. Apresenta todas as funcionalidades permitidas pela linguagem JAVA^{TM}, como carregamento de classes em tempo de execução, identificação de tipos, serialização e invocação remota.

Esta plataforma utiliza, como muitas outras plataformas para agentes móveis, funções de ***callback*** para continuar a execução após a sua deslocação entre máquinas (servidores). Os principais pontos fracos desta plataforma correspondem à segurança, quer interna (interior às aplicações, acesso a recursos e variáveis), quer externa (comunicação entre servidores).

3.2.4 JADE

Ferramenta desenvolvida pela CSELT, que consiste numa moldura para desenvolvimento e exploração de sistemas multi-agentes. Esta ferramenta segue os padrões da FIPA 97/2000 (*Foundation for Intelligent Physical Agents*). Como tal, apresenta diversos mecanismos para suporte de agentes; comunicação por mensagens, sistemas de deliberação (JESS), assim como outros.

O JADE é uma plataforma avançada e aberta para desenvolvimento e exploração, explicando desta forma a sua forte integração no âmbito de sistemas multi-agente.

3.2.5 ffMain

Projecto desenvolvido na universidade de Goethe na Alemanha, a sua maior importância consiste no seu paradigma de agentes móveis para ambientes abertos. Permite o suporte da execução e interpretação do código de agentes, estas características são suportadas pela linguagem Tcl, visto esta ser interpretada. O seu desenvolvimento foi motivado pela sua utilização para ambientes Web, utilizando exclusivamente tecnologias baseadas na mesma.

3.2.6 D'Agents

Infraestrutura para agentes móveis, desenvolvida pela Universidade de Dartmouth, nos EUA. Apresenta uma independência entre a unidade de execução dos agentes e a linguagem em que estes foram desenvolvidos, como tal, permite a utilização de diversas linguagens para a programação de agentes, como por exemplo, Java, Tcl, Scheme, etc.

3.2.7 Concordia

Desenvolvida em *JAVA™* pela *Mitsubishi Electric™* ITA, esta apresenta grande parte das funcionalidades já referidas para ambientes desenvolvidos em *JAVA™*. É composto por um grande número de *Managers* para segurança, persistência, administração, directório (localização de servidores) e eventos. Esta plataforma contém ainda uma aplicação de administração que permite a gestão dos agentes e sua segurança, bem como, a gestão dos servidores. As suas vantagens são a existência de *callback* variado, encontros e desenvolvimento de agentes e comunicação cooperativa.

3.2.8 AgentSpace

Plataforma desenvolvida pelo Prof. Alberto Silva como apoio à sua tese de Doutoramento no IST. Esta plataforma apresenta conceitos interessantes relativos ao desenvolvimento de agentes, conseguindo incorporar vantagens encontradas em outras plataformas, como, a possibilidade de manter os canais de comunicação abertos durante a navegação, existência de diferentes funções de ***callback*** e a possibilidade de envio de mensagens de objectos com tipos arbitrários.

É necessário realçar que toda esta plataforma se encontra desenvolvida numa camada acima do *Voyager Orb* (Sistema de distribuição e de acesso a objectos através de uma rede de servidores).

3.3 AVALIAÇÃO DAS ARQUITECTURAS EXISTENTES

3.3.1 Objectivo

O desenvolvimento desta plataforma parte da análise de outras plataformas existentes no mercado, bem como, de outras desenvolvidas mas sem sucesso comercial. Apresentamos nesta secção a análise cuidada efectuada da comparação das plataformas (*AgentSpace*, *Zeus*, *Jade*, *Aglets Workbench*, *Telescript*, *D'Agents*, *Odyssey* e *Jumping Beans*) com a plataforma desenvolvida.

3.3.2 Avaliação

1. Ao contrário do que acontece no *Aglets* e no *AgentSpace* não é chamada uma função específica de ***callback*** após a operação de migração. A execução continua na função em que estava (mantendo o estado da pilha de funções como se encontrava antes da migração) e na instrução imediatamente a seguir (se for usada a linguagem de programação de agentes, ou, na falta desta, na instrução indicada directamente pelo programador). Esta **pilha de instruções** (conceito explicado adiante) permite uma maior flexibilidade na construção das operações efectuadas por cada agente, permitindo ainda chamadas encadeadas de funções ou mesmo chamada recursiva de funções. A existência desta pilha empurra-nos um passo à frente na construção de **agentes *statefull*** (agentes que conseguem manter o seu estado de execução após navegação), embora o JAVA^{TM} não permita internamente manter o estado de execução. Arquitecturas baseadas em funções de ***callback*** dificultam a tarefa do programador e não apresentam uma forma intuitiva de programar.
2. Não existe uma associação directa entre um agente e a *thread* que o executa, ao contrário do Jade e outras soluções baseadas em *Threads*.
3. Apesar da maior parte das arquitecturas apresentarem uma **sociedade de agentes**, pensamos que a existência de uma **sociedade de agentes hierarquizada** seria uma mais valia para o desenvolvimento de aplicações baseadas em agentes. A existência de uma sociedade hierarquizada permite ainda a comunicação entre níveis da hierarquia diferentes e entre um agente e todos os agentes de um nível hierárquico (***broadcast hierárquico***).
4. A comunicação de mensagens é feita de **modo assíncrono**, como acontece também no *ffMain* e no *D'Agents* e pode ser efectuada localmente ou remotamente como na maioria das plataformas estudadas. Pode também ser pedido à plataforma que entregue uma mensagem numa determinada data e hora correspondente ao **modo futuro** suportado pelo *Aglets* e *AgentSpace*.
5. Também permite, como o *AgentSpace*, mas contrariamente ao *Aglets*, o envio de mensagens compostas de objectos de um tipo arbitrário, incluindo tipos (classes) que não existam ainda na máquina receptora da mensagem.

6. Nesta plataforma, ao contrário da maioria das outras plataformas, não existem outros canais de comunicação para o exterior do agente que não o envio de mensagens ou o acesso às funções base da plataforma, não existindo assim o problema de manter ou não os **canais de comunicação abertos** numa operação de **navegação**. Isto permite simplificar não só o funcionamento dos agentes, como também o funcionamento dos servidores de agentes, sem grande perda de flexibilidade (a comunicação passa a ser toda feita através de mensagens).

7. Ao contrário do Zeus, esta permite a **navegação dos agentes** entre servidores da plataforma (navegação tanto do código do agente, como do conteúdo).

8. É usada a mesma noção de **local de execução** que na *D'Agents*, na qual cada local de execução corresponde à identificação de um servidor de agentes, em vez de cada local de execução ser visto como uma local estruturado para o encontro e execução de agentes.

9. Assim como no *Jumping Beans*, esta plataforma possibilita uma gestão fácil de agentes através das possibilidades dadas aos utilizadores (administradores da plataforma).

10. Ao contrário do *Telescript*, mas em comum com outras plataformas baseadas em *JAVA™*, a interacção directa entre agentes não implica que tenham de se conhecer um ao outro durante a fase de desenvolvimento.

11. A **gestão da identidade** das metaclasses dos agentes pelo nome é hierárquica como nos *Aglets*, no *ffMain* e no *AgentSpace*.

12. A persistência é externa como na maioria das plataformas estudadas.

13. Assim como no *Telescript* e no *D'Agents* vão ser garantidos os aspectos relativos à **autenticidade** e à **privacidade dos dados**.

14. Permite a construção de agentes a qualquer momento, mesmo enquanto outros agentes se encontram a correr, esta característica permitirá a construção de aplicações baseadas em agentes a qualquer momento e com uma enorme flexibilidade e robustez.

15. Outro dos aspectos, corresponde à flexibilidade da plataforma, esta permite o desenvolvimento de qualquer tipo de agentes (agentes emocionais, educativos, inteligentes, interfaces, etc.) sem qualquer compromisso.

16. Não é possível o controlo de alguns recursos usados por cada agente, por limitações do *JAVA™* (memória ocupada), à semelhança das restantes plataformas baseadas nesta linguagem.

17. A plataforma pode ser incorporada em qualquer outra aplicação baseada em *JAVA™*, incluindo servidores, aplicações cliente, aplicações de Internet (applets), etc., assim, para além de estas terem disponíveis, de imediato, grande parte das funcionalidades da plataforma, possibilita que os agentes naveguem para junto destas de forma a facilitar a comunicação, eficiência e desempenho.

18. Finalmente, integra grande parte dos pontos fortes encontrados nas diversas arquitecturas analisadas que pensamos terem sido importantes na construção desta plataforma.

3.4 INOVAÇÕES DESENVOLVIDAS

O agente não tem controlo directo sobre a sua execução, é da responsabilidade da plataforma colocá-lo em execução sempre que existir uma mensagem nova para este, e retirá-lo de execução sempre que não existam mensagens a processar. Pensamos que este aspecto é de elevada importância porque reduz consideravelmente a ocupação de recursos, e facilita a gestão dos mesmos por parte do agente. É importante realçar que este controlo se refere à execução do agente e não às suas acções.

As mensagens são também elas inovadoras, no sentido que, correspondem exactamente às funções fornecidas por cada agente, adicionando uma maior flexibilidade e simplicidade. Este aspecto, não é uma limitação, na medida em que, permite desenvolver sistemas de mensagens e mensagens mais complexas.

Capítulo 4

METODOLOGIA

RESUMO

Neste capítulo é apresentada a metodologia adoptada para o desenvolvimento deste trabalho. Dando ênfase ao trabalho realizado para a cadeira de IAA (Introdução aos Agentes Autónomos) que serviu de protótipo para o desenvolvimento deste trabalho.

4.1 INTRODUÇÃO

Este trabalho foi desenvolvido segundo duas iterações, uma primeira iteração, na qual o conhecimento sobre o problema era reduzido, como tal foi adoptada uma filosofia de desenvolvimento ***bottom-up***, na segunda iteração, após um conhecimento e estudo mais alargado e conhecedor do problema, foi possível uma construção assentando numa filosofia ***top-down***.

Para melhor compreender estas duas etapas e as metodologias adoptadas em cada uma destas, iremos analisá-las com maior pormenor neste capítulo.

4.2 CONTEXTUALIZAÇÃO

4.2.1 Primeira Iteração

No início do primeiro semestre do ano lectivo de 2001/02 foi escolhida, como cadeira de opção, a cadeira de Introdução aos Agentes Autónomos, leccionada pela professora Ana Paiva. No âmbito desta cadeira foi proposto a construção de uma plataforma de agentes distribuída para desenvolvimento e manutenção de agentes reactivos e/ou deliberativos (ver apêndice E).

Após o desenvolvimento do trabalho da cadeira de agentes verificámos que a plataforma apresentava determinados problemas, uns devido à abordagem tomada no processo de desenvolvimento, outros, relacionados com a especificação e os padrões de desenho.

- ***Abordagem tomada no processo de desenvolvimento***

  No trabalho da cadeira de agentes autónomos foi construída uma especificação, na qual foram delineados os requisitos principais, um conjunto de arquitecturas de alto nível, mas não foram apresentadas arquitecturas para os intervenientes da plataforma (servidores), nem foram desenvolvidos desenhos de baixo nível.

  - Uma arquitectura bem elaborada de raiz para os intervenientes teria oferecido uma grande robustez ao sistema e teria prevenido a existência de falhas.
  - O desenvolvimento de desenhos de baixo nível teria permitido *a priori* o reconhecimento e detecção de problemas que se mostraram difíceis de resolver.

  Na etapa de desenho do sistema foi abordado uma filosofia orientada ao agente, na qual o agente seria o centro da plataforma. O agente teria acesso a um conjunto de serviços disponibilizados pela plataforma, todos estes serviços seriam isolados e independentes. Esta filosofia encontra-se perfeitamente apropriada para o desenvolvimento de uma plataforma de agentes, no entanto, o desenho dos serviços, no qual estes seriam considerados isoladamente e independentemente mostrou-se ineficaz para o trabalho final de curso, nomeadamente devido aos requisitos de segurança. Como é fácil de imaginar

seria complicado impedir acessos a informação confidencial de outros agentes seguindo a anterior abordagem, este deve-se principalmente aos múltiplos pontos acesso do agente.

Na realidade não existiu uma filosofia base para o processo de desenvolvimento, de certa forma poderá se dizer que a metodologia adoptada tenha sido uma metodologia próxima do ***Extreme Programming***, no entanto, tal facto não foi premeditado. Assim, com esta metodologia, algumas etapas essenciais no processo de desenvolvimento de sistemas reais foram esquecidas e/ou subestimadas. A etapa de testes não existiu na sua totalidade, foram efectuados testes genéricos, mas não nos padrões necessários para um bom desenvolvimento. Assim as etapas de teste de unidade e integração foram menosprezadas e inexistentes.

Desta má abordagem, surgiram inúmeros problemas quando o sistema foi testado com problemas reais, como é de esperar foi extremamente difícil a resolução destes problemas devido ao efeito de cascata de erros.

4.2.2 Segunda iteração

Após a análise da primeira iteração tornou-se obvio a necessidade da construção de uma plataforma de agentes de início baseando em conceitos mais sólidos adquiridos na construção da anterior plataforma. Esta nova plataforma seria potencialmente mais robusta, eficiente, eficaz e, principalmente, modular, o que seria impossível com a anterior abordagem. Como é natural, conceitos de arquitecturas e padrões de desenho acrescentam uma maior robustez ao trabalho, facilitando as etapas de teste e a inserção de novos requisitos.

Para tal e após o estudo de algumas das metodologias existentes que apresentassem as características anteriormente descritas e que permitissem o desenvolvimento de um trabalho com este grau de complexidade, considerámos que o RUP (*Rational Unified Process*) oferecia as vantagens desejadas. O RUP pode ser considerado como um processo de desenvolvimento, mas é mais do que um simples processo, podendo mesmo ser considerado uma metodologia. Caso deseje aprofundar os seus conhecimentos sobre este processo retemo-lo para o apêndice B, no qual, é descrito muito brevemente este processo.

Capítulo 5

CAPTURA DE REQUISITOS

REQUISITOS DO SISTEMA

RESUMO

Neste capítulo são apresentados os requisitos recolhidos do sistema a ser desenvolvido, bem como, dos seus constituintes. Serão ainda representados os requisitos capturados sobre a forma de casos de uso do sistema.

5.1 INTRODUÇÃO

Os requisitos de um sistema consistem em funcionalidades de um sistema ou uma descrição de algo que o sistema é capaz de fazer de forma a satisfazer o seu propósito.

5.2 REQUISITOS DO SISTEMA

Para melhor compreensão de todos os aspectos importantes nos quais a plataforma se baseia, pensámos em caracterizá-la segundo diversos padrões e funcionalidades a suportar para a execução de agentes.

- ***Tipos de Agentes***

  Esta plataforma pretende apresentar um sistema para desenvolvimento de qualquer tipo de agentes. Como tal, serão divididos os agentes em dois tipos distintos, tendo em conta a sua mobilidade, **agentes estacionários** (*stationary agents*) e **agentes móveis** (*mobile agents*). O motivo desta divisão prende-se com simplificação e facilidade de desenvolvimento dos agentes proveniente desta divisão, para além de, permitir reduzir o grau de complexidade dos agentes estacionários, mais simples em relação aos restantes.

  Existem ainda **agentes de sistema**, internos aos servidores de agentes, e apenas conhecidos por estes, servindo para executar tarefas específicas (como por exemplo: receber pedidos de entrega de mensagens e reencaminhá-los para o agente respectivo) para o servidor de agentes a que pertencem.

- ***Execução do Agente***

  A execução do agente deverá ocorrer segundo distintas linhas de execução (*Thread*), no entanto, embora a execução do agente seja continua, quando este terminar as suas tarefas, é retirado de execução, sendo reposto assim que existir alguma(s) tarefa a desempenhar.

- ***Gestão de identidades***

  Cada entidade, **agente** ou ***agentclass*** (classe de um agente), pode ser identificada de duas formas, por **identificador** ou, no caso das classes de agentes, através de um **nome** (designação), **único em cada nível hierárquico**. O Agente quando identificado por identificador, este será único, no entanto, este pode ser referenciado através da classe, cabendo a esta referenciar o agente que esta achar o mais adequado.

- ***Desenvolvimento contínuo***

  Esta plataforma permite o desenvolvimento de agentes **fora do tempo de desenvolvimento**, ou seja, a qualquer momento poderá ser criada uma classe de agentes e seus agentes, sem que para isso seja necessário reiniciar a plataforma.

- ***Relação entre agentes***

Um agente encontrar-se-á incorporado numa **sociedade**, a qual deverá ser **hierárquica** e permitir a **socialização** (comunicação) entre agentes. Cada agente tanto detém as suas responsabilidades, como também responsabilidades herdadas dos níveis hierárquicos superiores. Clarificando a ideia, todos os agentes terão uma classe bem definida, da qual poderão existir inúmeros agentes, esta por sua vez poderá pertencer a um outro nível hierárquico de classes e assim sucessivamente.

- ***Navegação***

Esta plataforma, como foi anteriormente dito, permitirá o desenvolvimento de agentes móveis, ou seja, agentes que durante o seu período de execução, se podem mover do local onde se encontram para outro local distinto, conhecido da plataforma, que suporte a execução de agentes. No entanto, o estado de execução deixado pelo agente no momento de transporte pode ser considerado segundo duas perspectivas, ***statefull*** (mantém o estado completo de variáveis locais e globais do agente, bem como, o seu estado de execução), ***stateless*** (mantém o estado apenas das variáveis globais). Nesta plataforma só se encontra implementado o segundo caso devido às limitações do *JAVA™* no acesso ao *stack* de execução, no entanto foi dado um passo em frente na construção do primeiro caso, pormenor que será posteriormente alvo de maior foco.

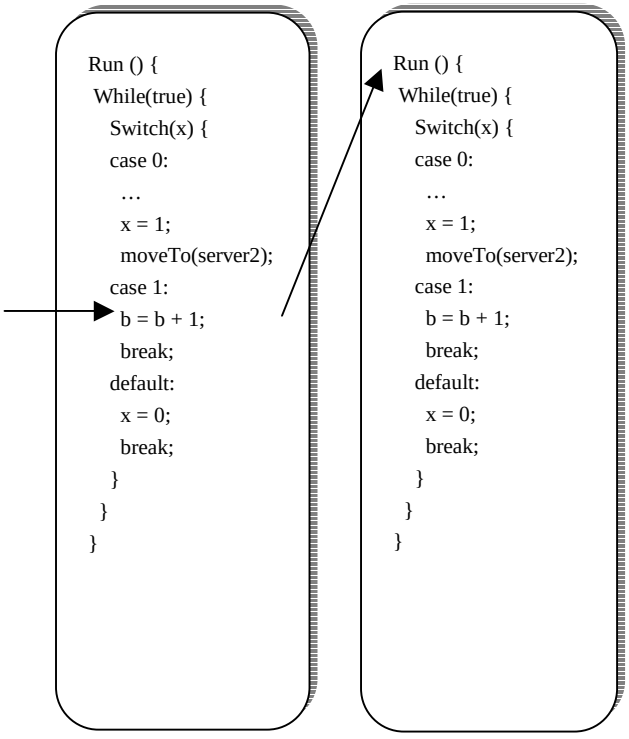

FIGURA 5-1. NAVEGAÇÃO *STATELESS*.

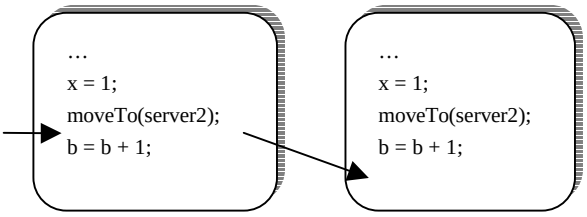

FIGURA 5-2. NAVEGAÇÃO *STATEFULL*.

Outro ponto importante a referir consiste na migração do agente, esta pode ser **completa** ou **incremental**. Optámos por desenvolver uma solução que implica uma migração incremental da informação do agente, a informação relativa a código e imagem é enviada de imediato, no entanto, referências para informação externa é transmitida a pedido do agente e garantida conjuntamente pela plataforma e sistema de base de dados.

Outro ponto importante a referir consiste na responsabilidade da navegação (*será que devo mudar ou estou bem aqui?*), ou seja, a quem compete a decisão sobre a transferência de fluxo de execução. Aqui apresentamos duas hipóteses, **por decisão do próprio agente** ou **por decisão da plataforma** sendo que esta última não foi possível desenvolver.

Existe ainda outro ponto importante a relatar, que consiste na determinação do alvo de transferência (local destino) do agente. Este local é identificado através de um nome conhecido da plataforma. Será da responsabilidade desta a sua resolução de forma a permitir a comunicação com o servidor alvo.

- ***Comunicação***

A interacção entre agentes será baseada em métodos assíncronos de comunicação, via mensagens. Esta interacção será sempre indirecta, e pode ser efectuada sempre que um agente o achar necessário. O agente **destinatário** tanto pode estar presente **localmente** como em execução num **local distinto.** A mensagem é transmitida transparentemente para o agente **emissor** como se a mensagem fosse local.

Existirá ainda outro método para o envio de mensagens, o qual permitirá que o agente defina uma mensagem que deverá ser **enviada** numa **determinada data e hora**, podendo ainda, definir uma **calendarização** para o envio de uma mensagem.

Em geral quando um agente interagir com outros agentes este segue uma comunicação ponto a ponto. Contudo, é permitido também o envio de uma mensagem para todos os agentes de uma classe (***broadcast***), bem como, para um nível hierárquico bem definido. Neste último, a mensagem é enviada para todos os agentes de todas as classes de todos os níveis abaixo do mesmo (***broadcast* hierárquico**).

- ***Comunicação durante navegação***

Como foi anteriormente dito, o agente pode navegar de um local de execução para outro, no entanto, o que acontece quando alguém tenta comunicar com o mesmo e este não se encontra a operar? Como a comunicação é assíncrona este problema aparentemente não existe, mas como é que a plataforma sabe o local onde se encontra a executar um determinado agente? Este problema resolve-se por uma solução semelhante ao de algumas plataformas (proxy), em que as mensagens são encaminhadas pelos servidores de agentes por onde passou o agente até chegarem ao mesmo.

- ***Persistência***

Embora seja correcto designar o nível de persistência como **externo**, visto que, toda a informação persistente é garantida por um sistema complexo de base de dados, não seria de todo incorrecto intitular esta como interna, na medida em que, toda a persistência será garantida de forma transparente para o agente, **a pedido do mesmo**, quer para a informação por este manipulada, quer para a sua própria informação.

É no entanto importante realçar o facto de o próprio sistema de base de dados poder suportar um modelo para a execução interna de agentes, esta capacidade, permite aos agentes ter uma maior facilidade na obtenção da informação desejada, num curto espaço de tempo, sem sobrecarregar o sistema.

A Plataforma suportará toda a persistência de um agente, incluindo **Data** (informação manipulada pelo agente, informação recolhida ou inferida ao longo do sua vida, bem como todo o conjunto de variáveis locais, partilhadas e globais que este incorpora), **Código** (código do agente, incluindo referências e código proveniente da herança) e **Imagem** (estado de execução), este último aspecto apenas é suportado parcialmente, através da introdução de uma pilha de instruções nos agentes que lhes permite guardarem a sua posição actual de execução. A persistência de um agente é garantida apenas a pedido do agente e não automaticamente pela plataforma, à excepção dos seguintes casos, nos quais a plataforma será encarregue de garantir a persistência, sendo estes; sempre que um agente é transferido para outro servidor e quando uma agente é retirado de execução.

- ***Acesso a recursos externos***

Como foi dito anteriormente, a plataforma permite a persistência de um agente como um todo (dados, código e imagem[1]), através de um acesso a um sistema de base de dados. No entanto, a plataforma permitirá ainda aceder a *HTTP servers*, aplicações de gestão e administração, dispositivos de *input* e *output*, assim como muitos outros.

- ***Segurança***

Foi desenvolvida uma política de segurança robusta para a execução dos agentes, e para a comunicação entre servidores de agentes (a comunicação entre servidores é autenticada utilizando o algoritmo de autenticação de chave pública, ver referência [MG99]). O código é sempre transferido de forma segura. Também este aspecto é garantido pelos servidores intervenientes na migração.

- ***Serviços***

A plataforma disponibiliza um conjunto de serviços aos seus intervenientes e aos agentes. Estes serviços podem ser de qualquer tipo, no entanto, apenas foi desenvolvido o serviço de notificação da alteração de execução de um agente na plataforma.

- ***Linguagem de programação de agentes***

Não foi desenvolvida, à partida, nenhuma linguagem para programação de agentes (***Agent programing Language* APL**).

[1] A persistência da imagem encontra-se dependente do desenvolvimento de agentes statefull.

5.3 GLOSSÁRIO DE TERMOS

TERMO	DESCRIÇÃO
Agente	Agente é um **actor** fundamental num domínio. Ele combina uma ou mais capacidades de serviço num modelo de execução unificado e integrado que pode incluir acesso a software externo, utilizadores humanos e facilidades de comunicação. Os agentes são também processos computacionais que funcionam **continuamente** e **autonomamente**, que se deslocam de um lugar para outro num ambiente onde outros processos ocorrem e outros agentes existem.
Agentclass	Classe de um agente. Entidade que representa as acções a executar por uma determinada classe de agentes.
Agentes estacionários	Agentes incapazes de se mover entre servidores distintos.
Agentes móveis	Agentes com a capacidade de movimento entre servidores.
Agentes *statefull*	Agente com a capacidade de manter o estado completo das suas variáveis locais e globais, bem como, o seu estado de execução após a transferência para outro servidor.
Agentes *stateless*	Agente incapaz de manter o estado completo de execução após a transferência para outro servidor. Este agente apenas mantém o estado das variáveis globais.
Hierarquia	Forma estruturada para representação das dependências entre agentes e classes de agentes. Numa hierarquia existem nós (níveis hierárquicos) dos quais dependem (esta dependência pode ser funcional, estrutural ou relativo à informação manipulada) outros nós ou folhas (classes de agentes).
Nível hierárquico	Posição ou patamar numa determinada hierarquia.
Sociedade	Conjunto de agentes que interagem de forma a desempenhar as suas funções. Numa sociedade todos os agentes são geridos por um conjunto de regras e obrigações que deverão obedecer.
Comunicar	Forma através da qual um determinado indivíduo (plataforma, agente, utilizador) envia uma mensagem a um agente de forma no intuito de este desempenhar uma determinada função.
Comunicação assíncrona	Este tipo de comunicação pressupõe que o emissor não aguarde pela entrega da mensagem ao destinatário (receptor da mensagem).

Comunicação síncrona	Este tipo de comunicação pressupõe que o emissor aguarde pela entrega da mensagem ao destinatário.
Mensagem	Conjunto de informações enviadas a um determinado agente, esta contêm o emissor o receptor, a função a ser exercida e seus argumentos.
Emissor	Individuo que envia uma determinada mensagem, este pode representar a plataforma, um agente ou um utilizador.
Receptor	Individuo, neste caso um agente, classe de agentes ou nível hierárquico, que recebe uma determinada mensagem.
Broadcast	Processo através do qual um emissor envia um mensagem para um conjunto definido de destinatários. O termo usualmente utilizado para referir o envio de mensagens para todos os agentes de uma classe.
***Broadcast* hierárquico**	Processo de ***broadcast*** através do qual é enviada uma mensagem para todos os agentes de todas as classes de um determinado **nível hierárquico**.
Calendarização	Processo através do qual um emissor poderá definir um calendário (definir dias e horas) para o envio de uma determinada mensagem que se manterá inalterável.
Estado de execução	Conjunto de informação sobre a pilha de execução de uma função, ou seja, o conjunto de instruções executadas até ao momento.
Local	No âmbito desta plataforma um local representa um determinado **servidor** conhecido da plataforma, o qual poderá executar agentes.
Migração	Processo através do qual um agente termina a execução no corrente servidor e recomeça-a no servidor alvo.
Navegação	O mesmo que migração.
Migração completa	Processo através do qual todo o código relativo a um determinado agente é transferido para o servidor alvo.
Migração incremental	Processo através do qual apenas o código do agente é transferido para o servidor alvo, o restante código (heranças, tipos adicionais,...) é transferido à medida que é necessário.
Persistência	Processo através do qual um agente (ou conjunto de objectos) é armazenado segundo um determinado formato sobre um suporte físico.
Recursos externos	Toda e qualquer funcionalidade que um sistema pode disponibilizar a um agente que a plataforma por si só não apresente.
Serviços	Funcionalidade prestada aos intervenientes da plataforma e aos agentes.
Chave pública	Chave de conhecimento público (Todos os intervenientes e potenciais atacantes).

Chave secreta / Chave privada	Chave do conhecimento apenas dos intervenientes da plataforma (Idealmente fora do acesso de potenciais atacantes).

5.4 DEFINIR REQUISITOS SUPLEMENTARES

FÍSICOS

Independência de máquina

O sistema deverá ser independente da máquina onde executa e do local em que se encontra.

NÃO FUNCIONAIS

Modularidade

Deverá poder permitir com facilidade a sua extensão através do acréscimo de novas funcionalidades e/ou alteração das existentes, adição e substituição de componentes.

Fiabilidade e Robustez

Deverá garantir a sua execução durante um elevado período de tempo, sem falhas, faltas ou quebras.

Eficiência

Deverá satisfazer as suas funcionalidades de forma rápida e/ou ocupando o mínimo de recursos necessários à sua correcta execução.

Segurança

Deverá sempre opor-se a potenciais intrusos (indivíduos ou código malicioso), impedindo a sua interferência na correcta execução das suas funcionalidades.

Desenvolvimento em tempo de execução

A qualquer momento a plataforma deverá permitir que seja criado ou alterado o corpo do agente sem que seja reiniciada a plataforma e sem favorecer a existência de falhas.

5.5 REQUISITOS FUNCIONAIS

A forma mais conveniente encontrada para determinar os requisitos funcionais para o sistema consiste em identificar os seus casos de uso (use cases).

5.5.1 Actores

Esta parte descreve os actores (utilizadores) existentes neste sistema. Um actor representa uma pessoa ou entidade que desempenha uma determinada função no sistema, cada actor terá à sua disposição um leque variado de operações que poderá desempenhar. Cada utilizador terá apenas um perfil no sistema (apenas desempenha o papel de um actor). Neste trabalho, em particular, existe duas perspectivas a analisar, a nível de utilizadores do sistema. Assim, existem os utilizadores (pessoas) do sistema (plataforma) propriamente ditos, os agentes, também estes utilizadores do sistema visto dependerem deste a nível funcional.

- ***Utilizadores***

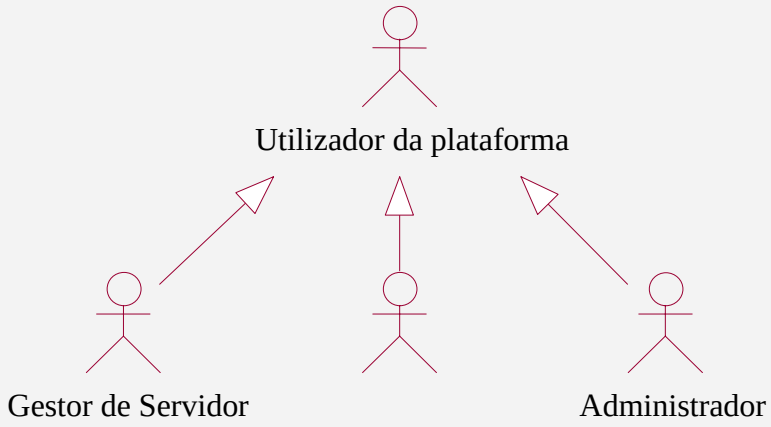

FIGURA 5-3. ACTORES DA PLATAFORMA (UTILIZADORES).

UTILIZADOR DA PLATAFORMA

Individuo que usufrui da plataforma para requerer funcionalidades a agentes através do envio de mensagens.

GESTOR DE SERVIDORES

Indivíduo responsável pela gestão de um servidor da plataforma.

GESTOR DE AGENTES

Indivíduo responsável pela gestão dos agentes na plataforma, podendo criar, alterar e remover classes de agentes, agentes e hierarquias.

ADMINISTRADOR

Individuo responsável por gerir todos os servidores da plataforma e seus agentes, bem como, monitorar o seu desempenho.

➢ ***Agentes***

À semelhança do tópico anterior, dedicamos esta parte à descrição dos actores (agentes) existentes neste sistema. Um actor representa um agente com uma determinada identidade, este terá à sua disposição um leque variado de operações que poderá exercer.

Existem três tipos de agentes distintos (representados no modelo seguinte), cada um com leque de operações bem definidas que poderá desempenhar.

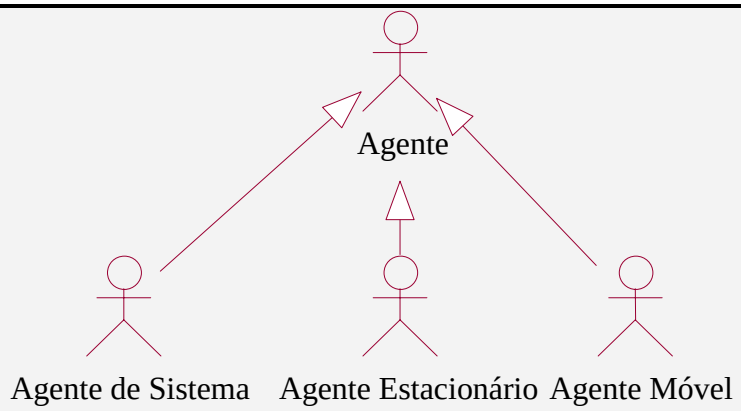

FIGURA 5-4. ACTORES DA PLATAFORMA (AGENTES).

AGENTE

Representa o agente, dispondo das funcionalidades e responsabilidades usuais comuns a todo o tipo de agentes.

AGENTE ESTACIONÁRIO

Corresponde a um Agente que opera num qualquer servidor mas sem a capacidade de se mover.

AGENTE MÓVEL

Agente que se pode mover entre vários servidores de agentes à sua disposição. Este pode mover-se a qualquer momento de modo a realizar a sua tarefa da forma que achar mais eficiente.

AGENTE DE SISTEMA

Agente que responde a mensagens do servidor onde se encontra e realiza a tarefa necessária para satisfazer o pedido.

5.5.2 Casos de uso

➢ ***Agente***

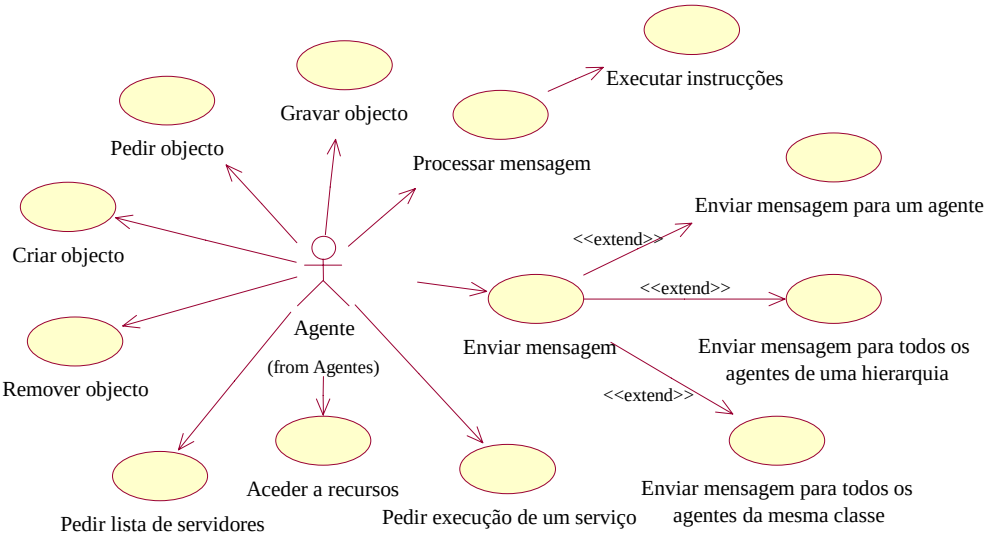

FIGURA 5-5. CASOS USO DO AGENTE.

- Processar mensagem

Objectivo

Processar uma mensagem que lhe foi entregue pela plataforma.

Caminho de sucesso

1. Retira a mensagem da sua caixa de mensagens.
2. Caso o agente seja estacionário, este apenas invoca a mensagem pretendida.
3. Caso seja uma agente móvel, subdivide a mensagem em instruções. *Extends Executar Instruções*.

- Executar instruções

Objectivo

Executar uma instrução de uma mensagem.

Caminho de sucesso

1. O agente retira uma instrução da pilha de instruções.
2. Executa a instrução e actualiza sempre que necessário a pilha de instruções.

- Enviar mensagem

Objectivo

Enviar uma mensagem para um agente destinatário, para todos os agentes de uma determinada classe ou mesmo para todos os agentes de todas as classes de uma determinada hierarquia na sociedade de agentes.

Caminho de sucesso

1. Requer à plataforma o envio da mensagem.
2. A plataforma determina se o agente destinatário se encontra em execução localmente.
   a. Caso afirmativo entrega a mensagem de imediato ao agente.
   b. Caso contrário:
      1. Verifica se detêm o registo do local de execução do agente.
         i Se a situação se verificar o agente reencaminha a mensagem para esse servidor. O servidor recebe a mensagem e entrega a um agente sistema responsável pela sua entrega. Este recebe a mensagem e reinicia no primeiro passo.
         ii Caso contrario questiona, ao servidor responsável pela gestão dos agentes, o servidor que detêm o agente. Se o servidor supremo indicar um servidor, este continua no passo (2.b.1.i), caso contrario coloca em execução localmente.

Caminhos alternativos

1. No 2º passo, a plataforma ao determinar o destinatário poderá verificar que se trata de uma classe de agentes ou de uma hierarquia, nestes casos a função irá ser executada recursivamente para todos os agentes pertencentes à classe ou hierarquia em questão. Estas extensões ao passo correspondem ao *Enviar mensagem para um agente*, *Enviar mensagem para todos os agentes de uma hierarquia* e *Enviar mensagem para todos os agentes da mesma classe.*

- Aceder a um recurso

Objectivo

Pedir à plataforma que lhe forneça um determinado recurso pretendido. (e.g. ficheiro ou directório disponível, ligação aberta, etc.).

Caminho de sucesso

1. O agente acede ao servidor de agentes em que se encontra de momento.
2. Questiona ao servidor os recursos que este disponibiliza.

- Pedir execução de um serviço

Objectivo

Pedir à plataforma que esta notifique o agente da ocorrência de um determinado serviço.

Caminho de sucesso

1. O agente acede ao servidor de agentes em que se encontra de momento.
2. Regista perante o servidor a sua pretensão de receber notificações de um determinado tipo.
3. O servidor de agentes regista no seu registo de serviços de um novo receptor de serviços.

- Pedir lista de servidores

Objectivo

Pedir à plataforma que lhe forneça a lista de servidores existentes nesta.

Caminho de sucesso

1. O agente acede ao servidor de agentes em que se encontra de momento.
2. Requer o fornecimento da lista de servidores.
3. O servidor de agentes determina junto do sistema virtual de transporte, quais os servidores existentes na plataforma.

- Criar, Obter, Gravar e Remover Objecto

Objectivo

Pedir à plataforma para criar, obter, gravar ou remover um objecto de uma determinada classe.

Caminho de sucesso

4. O agente acede ao servidor de agentes em que se encontra de momento.
5. Requer a operação sobre um objecto de uma determinada classe (criar, obter, gravar ou remover).
6. O servidor de agentes requer ao sistema virtual de objectos a operação.

➢ ***Agente Móvel***

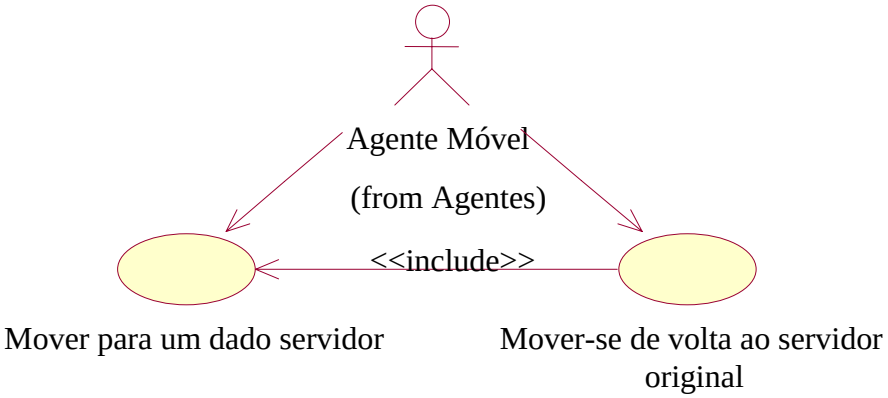

FIGURA 5-6. CASOS USO DO AGENTE MÓVEL.

- Mover para um dado servidor

Objectivo

Pedir à plataforma para transportar o agente entre servidores, colocando-o em execução no servidor destino.

Caminho de sucesso

1. O agente acede ao servidor de agentes em que se encontra de momento.
2. Requer o seu transporte junto do servidor.
3. O servidor retira o agente de execução, grava-o, e envia-o para o servidor destino.
4. O servidor destino recebe o agente, comunica ao supremo da chegada do agente e coloca-o em execução localmente.

- Mover para o servidor original

Objectivo

Pedir à plataforma para transportar o agente para o servidor de sua origem.

Caminho de sucesso

1. O agente determina o servidor de origem.
2. *Include Mover para um dado servidor*.

➢ ***Agente Sistema***

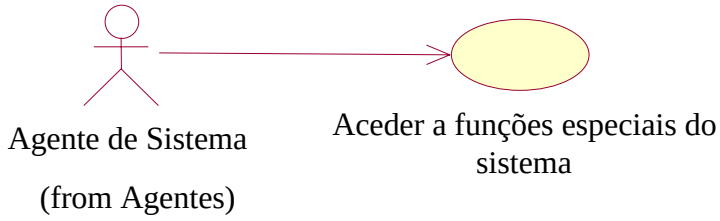

FIGURA 5-7. CASOS USO DO AGENTE SISTEMA.

- Aceder às funções especiais do sistema

Objectivo
Requer funcionalidades privadas do sistema não disponíveis a outros agentes.

➢ ***Utilizador da plataforma***

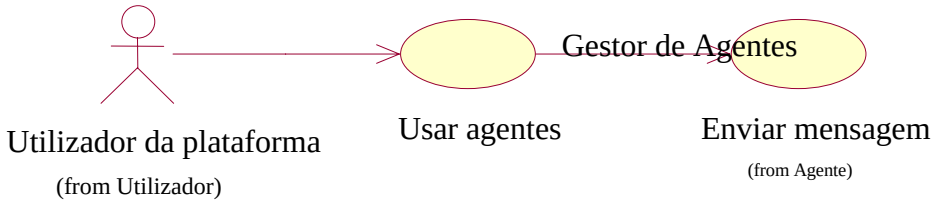

FIGURA 5-8. CASOS USO DO UTILIZADOR DA PLATAFORMA.

- Usar agentes

Objectivo
Pedir ao sistema para enviar uma mensagem para um agente para que este execute uma operação pretendida.

➢ ***Gestor de Agentes***

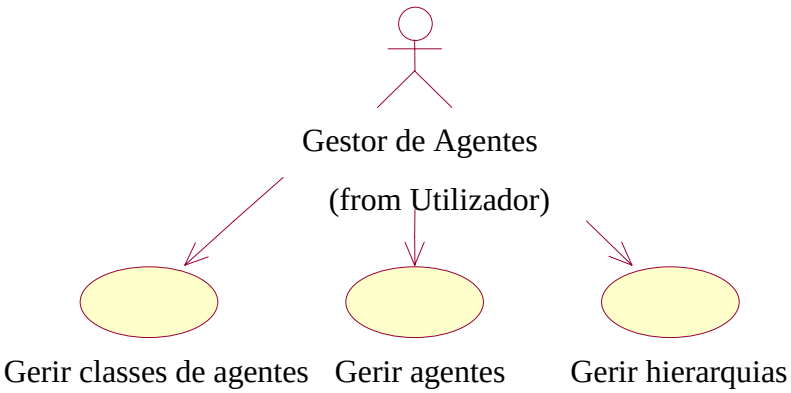

FIGURA 5-9. CASOS USO DO GESTOR DE AGENTES.

- Gerir hierarquias de agentes (criar, obter, alterar e remover hierarquias)

Objectivo

Pedir ao sistema para criar, obter, alterar ou remover uma hierarquia de agentes.

Caminho de sucesso

1. O utilizador visualiza a hierarquia de agentes, escolhe a que pretende para executar a operação.
2. Requer a operação pretendida (criação, obtenção, alteração ou remoção).
3. O servidor de agentes requer ao sistema virtual de objectos a operação.

- Gerir classes de agentes (criar, obter, alterar e remover classes de agentes)

Objectivo

Pedir ao sistema para criar, obter, alterar ou remover uma classe de agentes da hierarquia de agentes.

Caminho de sucesso

1. O utilizador visualiza a hierarquia de agentes.
2. Escolhe a classe de agentes que pretende.
3. Requer a operação pretendida (criação, obtenção, alteração ou remoção).
4. O servidor de agentes requer ao sistema virtual de objectos a operação.

- Gerir agentes (criar e remover agentes)

Objectivo

Pedir ao sistema para criar ou remover um agente de uma determinada classe de agentes da hierarquia.

Caminho de sucesso

1. O utilizador visualiza a hierarquia de agentes, escolhe a classe de agentes.
2. Requer a operação pretendida (criação ou remoção de um agente).
3. O servidor de agentes requer ao sistema virtual de objectos a operação.

➢ ***Gestor de Servidor***

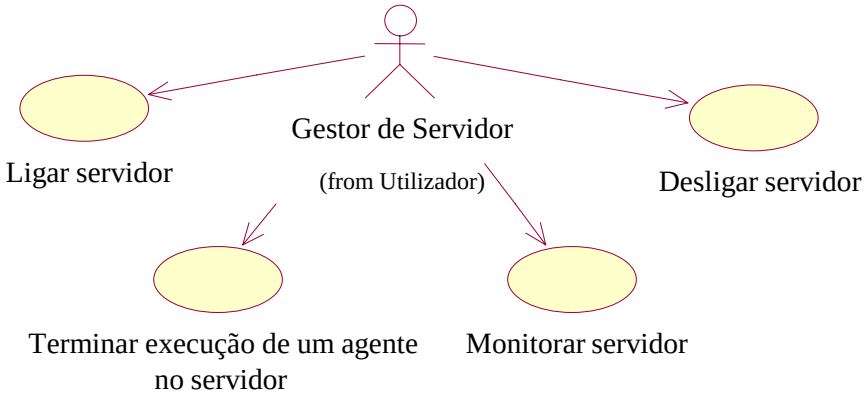

FIGURA 5-10. CASOS USO DO GESTOR DE SERVIDOR.

- Ligar servidor

Objectivo

Ligar um servidor de agentes.

- Desligar servidor

Objectivo

desligar um servidor de agentes.

- Monitorar servidor

Objectivo

Visualizar a informação do servidor relacionada com as ligações estabelecidas com outros servidores e os agentes que se encontram em execução localmente.

- Terminar a execução de um agente no servidor

Objectivo

Terminar a execução de um agente que se encontra em execução no servidor.

Caminho de sucesso

1. O utilizador indica o agente em execução que pretende terminar.
2. O sistema retira de execução esse agente.

➢ ***Administrador***

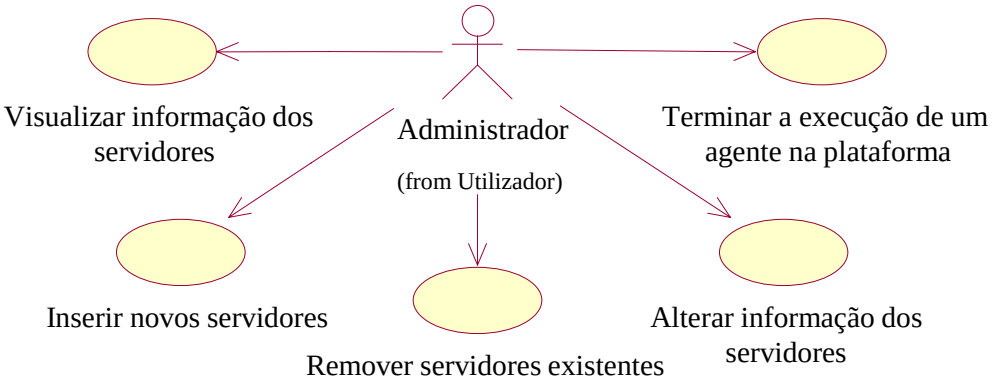

FIGURA 5-11. CASOS USO DO ADMINISTRADOR.

- Visualizar informação dos servidores

Objectivo

Visualizar a informação de todos os servidores registados na plataforma de agentes.

- Inserir novo servidor

Objectivo

Inserir um novo servidor na plataforma de agentes.

Caminho de sucesso

1. O utilizador indica o nome do servidor que pretende inserir, seguido do endereço IP (*Internet Protocol*) e porto.
2. O sistema regista no sistema o novo servidor e a sua chave pública.
3. O utilizador insere uma disquete na *drive*, na qual será inserida os ficheiros necessários para a instalação do servidor contendo informação sobre a chave privada deste e chave pública do servidor responsável pela autenticação.

- Remover servidor existente

Objectivo

Remover um servidor da plataforma de agentes.

Caminho de sucesso

1. O utilizador indica o nome do servidor que pretende remover.
2. O sistema remove o servidor pretendido.

- Alterar informação dos servidores

Objectivo

Alterar a informação de um servidor da plataforma.

Caminho de sucesso

1. O utilizador indica o servidor que pretende alterar.
2. Em seguida altera a informação do servidor.
3. O sistema actualiza a informação do servidor com a nova informação inserida.

- Terminar a execução de um agente na plataforma

Objectivo

Terminar a execução de um agente na plataforma.

Caminho de sucesso

1. O utilizador indica o agente em execução que pretende terminar.
2. O sistema retira de execução esse agente.

Capítulo 6

ANÁLISE

RESUMO

Neste capítulo é apresentada a análise dos requisitos do sistema.

6.1 ANÁLISE ARQUITECTURAL

6.1.1 Pacotes de análise do Sistema

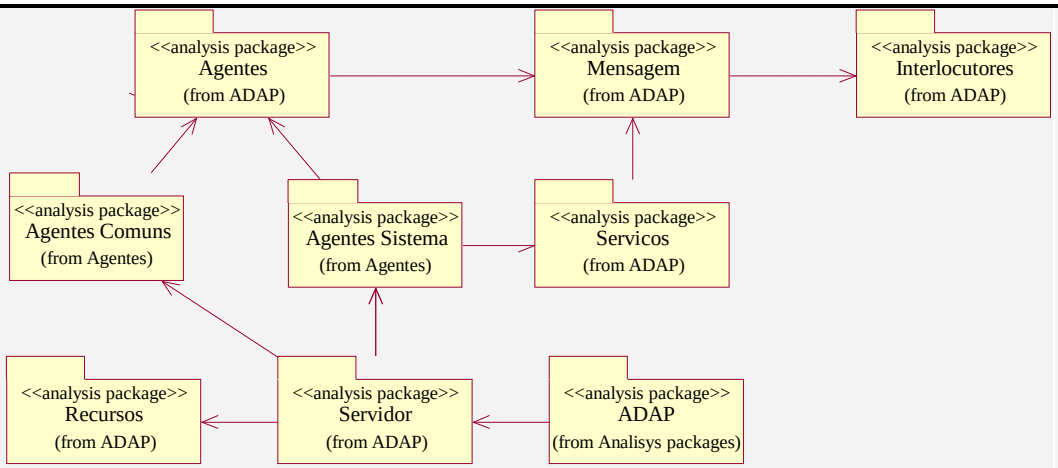

FIGURA 6-1. PACOTES DE ANÁLISE.

AGENTES

Este pacote representa a parte do sistema responsável pela gestão dos agentes (casos de utilização e informação dos agentes), apresentando também os diversos tipos de agentes e alguns agentes de sistema.

INTERLOCUTORES

Este pacote representa todos os módulos que representam os interlocutores das mensagens (emissor e receptor).

MENSAGEM

Este pacote representa as entidades relativas às mensagens e aos seus tipos.

RECURSOS

Pacote representativo dos recursos e seus tipos.

SERVIÇOS

Pacote que contem os serviços da plataforma.

SERVIDOR

Pacote que representa todas as acções da plataforma e suas entidades.

6.1.2 Classes de análise

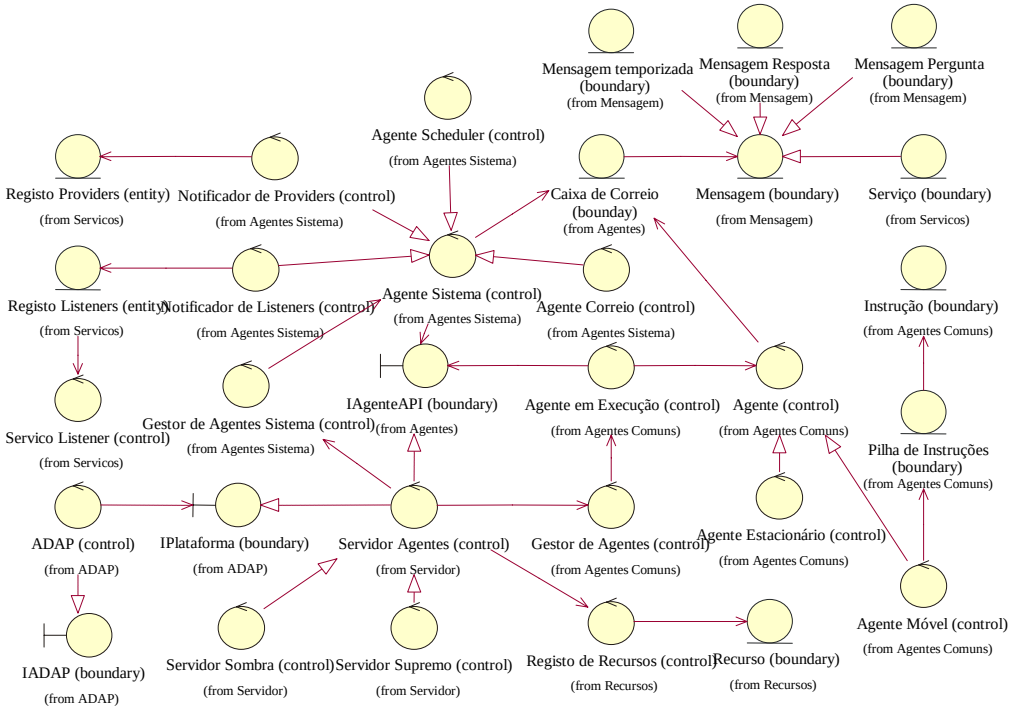

FIGURA 6-2. VISÃO GLOBAL DAS CLASSES DE ANÁLISE.

De forma a facilitar a leitura deste relatório e não torná-lo excessivamente descritivo, as classes de análise presentes no diagrama acima, encontram-se descritas com maior detalhe no apêndice A. Optámos desta forma apenas por evidenciar a visão global das classes sem ir ao pormenor.

Capítulo 7

ARQUITECTURA DE COMPONENTES

RESUMO

Neste capítulo serão apresentados todos os *deliverables* relacionados com a arquitectura de componentes da plataforma.

7.1 INTRODUÇÃO

Um arquitecto usualmente trabalha desde o topo até à base, produzindo desenhos que evidenciam cada vez mais detalhe sobre um sistema. Uma arquitectura associa as capacidades do sistema identificadas na especificação dos requisitos com os componentes do sistema que os irão implementar. Um estilo arquitectural caracteriza um determinado sistema. Existem alguns estilos arquitecturais, entre estes incluem-se, *pipes* e filtros, objectos, invocação implícita, camadas, repositórios, interpretadores, e controlo de processos.

7.2 COMPONENTES DE BASE

7.2.1 Visão geral dos componentes do sistema

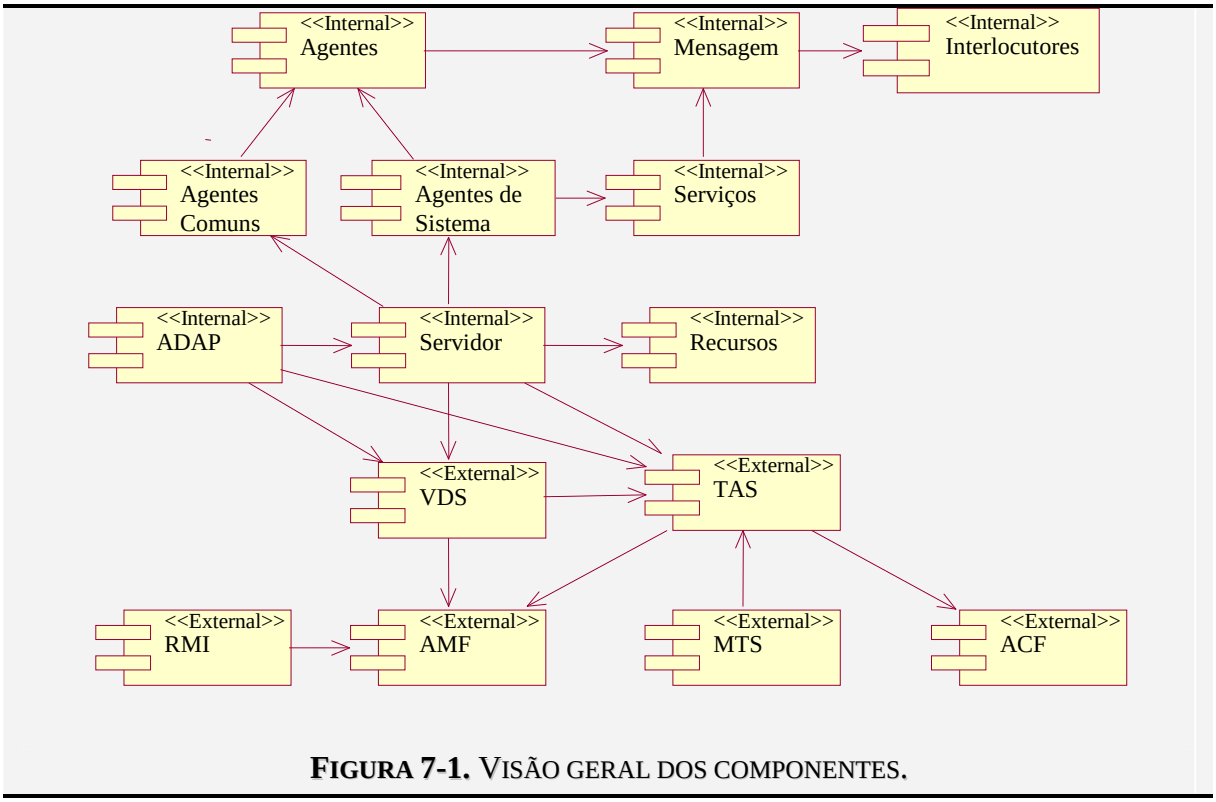

FIGURA 7-1. VISÃO GERAL DOS COMPONENTES.

7.2.2 Componentes Específicos da Plataforma

Componentes específicos da plataforma que modelam os requisitos são estereotipados como <<Internal>>. Caso deseje uma descrição mais detalhada destes componentes sugere-se a leitura do apêndice A.

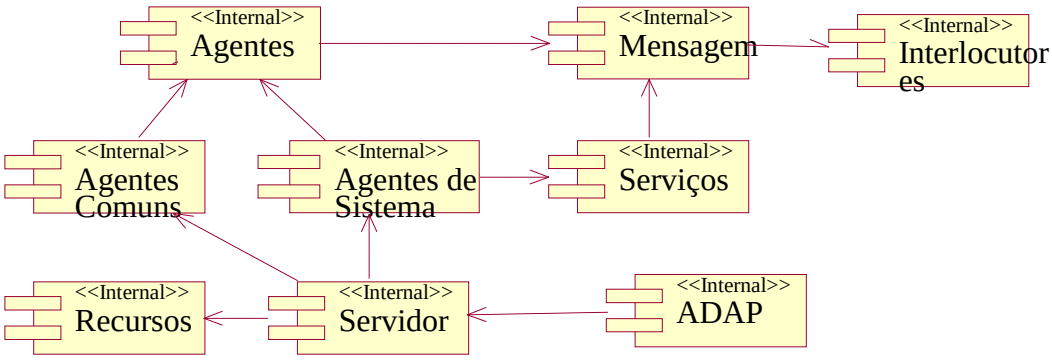

FIGURA 7-2. COMPONENTES INTERNOS.

7.2.3 Componentes de Suporte Geral

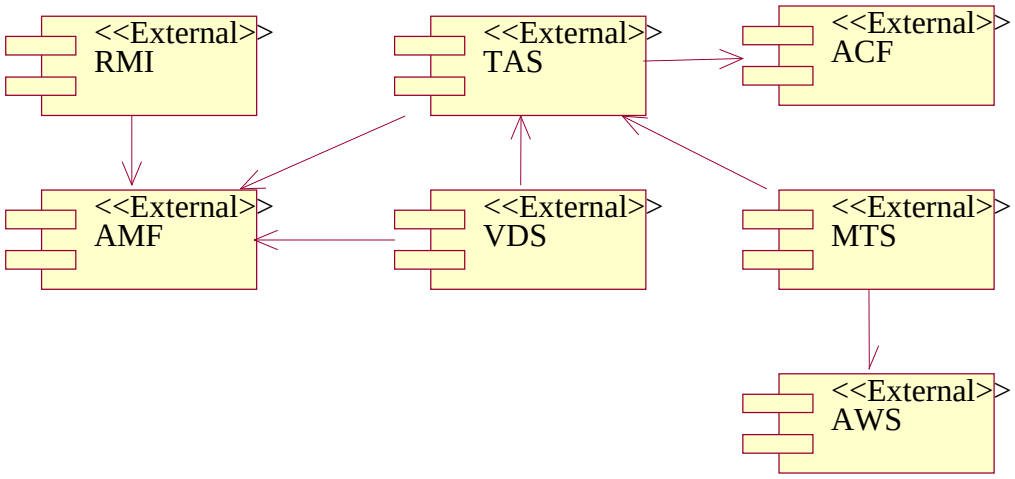

FIGURA 7-3. COMPONENTES EXTERNOS.

Durante a análise dos requisitos do sistema foi identificado um conjunto de funcionalidades que foram confinadas a um conjunto distinto de componentes, sendo estes estereotipados como <<External>>. Passamos a descrever os principais componentes identificados (caso deseje uma descrição mais detalhada destes componentes sugere-se a leitura do apêndice B).

- ***AMF (All-purpose Marshalling Framework)***

  Componente responsável pela serialização de objectos. A sua construção foi motivada pela necessidade de poder serializar qualquer agente, a qualquer momento, mesmo agentes recentemente importados para a plataforma. Este sistema permite a parametrização do responsável pela resolução e obtenção da classe. Outra motivação consiste na necessidade de poder efectuar invocações remotas sem que a assinatura das funções seja conhecida em *compile-time*, permitindo a inserção de novas funcionalidades a qualquer momento. Desta

forma é desnecessário a construção de *stubs*, essenciais para o sistema de RMI existente no JAVA. Este novo sistema permite uma maior flexibilidade e eficácia no desempenho da plataforma.

Este sistema é utilizado pelos nossos sistemas de RMI e TAS para o transporte de informação entre servidores e pelo VDS para o armazenamento da informação dos agentes e objectos por estes manipulados.

- ***ACF (All-purpose Cipher Framework)***

Framework que encapsula todas as funcionalidades de cifra de informação e chaves.

- ***RMI (Remote Method Invocation)***

Sistema de invocação remota que utiliza o AMF como pacote de serialização adquirindo as melhorias que esta lhe proporciona, como por exemplo, permite o registo de atendedores de chamadas durante a execução, permitindo mesmo a escolha de quais as funções que se pretende disponibilizar a cada momento. O *provider*, neste caso intitulado de atendedor pode, caso seja do seu interesse, disponibilizar uma funcionalidade durante uns instantes, retirando-a após um período tempo.

- ***TAS (Transporte and Authentication System)***

Este sistema permite virtualizar um conjunto de servidores, permitindo de uma forma transparente, a comunicação entre estes. Na realidade este sistema pouco difere do sistema anterior (RMI), no entanto, existe um aspecto importante que os distingue, o qual prende-se com a resolução do servidor: no RMI a invocação necessita de indicar o endereço IP e porto que se pretende aceder enquanto que no TAS apenas necessita de um nome que o identifique o servidor. Este nome é conhecido no momento do registo no sistema, nesse momento, é memorizado o seu endereço de Internet e porto, no qual este se encontra à escuta, para além da chave pública para o estabelecimento de um canal seguro. Este sistema permite ainda a autenticação dos intervenientes existindo para tal, um servidor de autenticação com esta responsabilidade.

- ***VDS (Virtual Database System)***

Este sistema pouco difere de um usual ORB (*Object Request Broker*), o qual é responsável pela gestão dos objectos e sua coerência entre diversos servidores possuidores de cópias dos objectos. A necessidade da sua construção prende-se com o carácter anteriormente referido relacionado com a inserção de novos objectos (pertencentes a novas classes) com o sistema em execução. Este sistema tanto suporta a gestão dos agentes como dos objectos por estes manipulados.

- ***MTS (Monitor and Tracking System)***

Este sistema encontra-se construído sobre o módulo do TAS de rastreio de informação. Qualquer componente pode enviar a qualquer momento informações de rastreio para um

servidor específico, ao qual compete guardar e apresentar essa informação. Este aspecto foi logo contemplado no protocolo do TAS para que logo nas camadas inferiores esta potencialidade seja usufruída.

Para que a visualização da informação e sua gestão seja mais fácil e acessível foi construído um sistema intitulado MTS com estas competências. Este sistema recebe todas as informações de rastreio que lhe são transmitidas e apresenta-as para posterior resolução de possível falhas, faltas ou quebras, que não tenham sido previstas na sua construção.

- ***AWS (Advanced Windowing System)***

  Este sistema foi construído no intuito de apresentar um sistema de gestão de janelas completamente genérico que permitisse o desenvolvimento de aplicações gráficas com extrema facilidade. Desta forma, todas as aplicações cliente complexas utilizam este sistema como interface. Embora este sistema seja autónomo, permite a sua extensão para inserção de novas funcionalidades apresentando ainda um conjunto de ferramentas para construção de interfaces (*toolkit* de componentes de interface).

7.3 COMPONENTES DE SERVIDORES

7.3.1 Arquitectura de componentes de servidores

- ***Visão geral dos componentes de servidores***

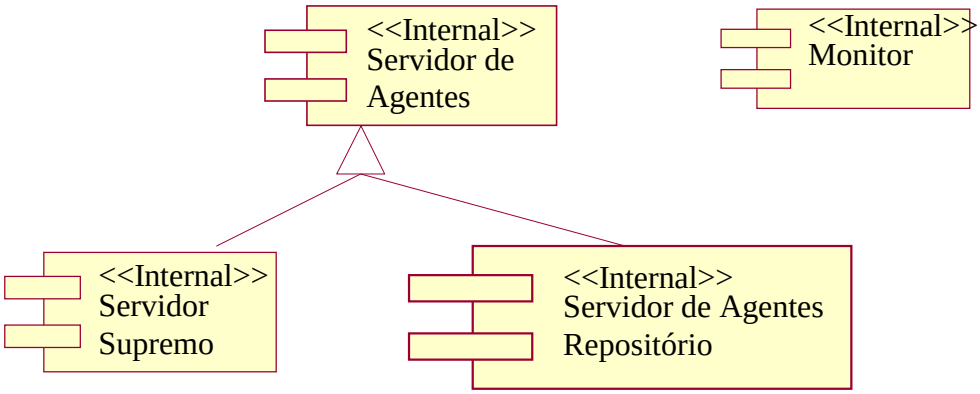

FIGURA 7-4. VISÃO GERAL DOS COMPONENTES DE SERVIDORES.

- ***Dependências dos componentes de servidores***

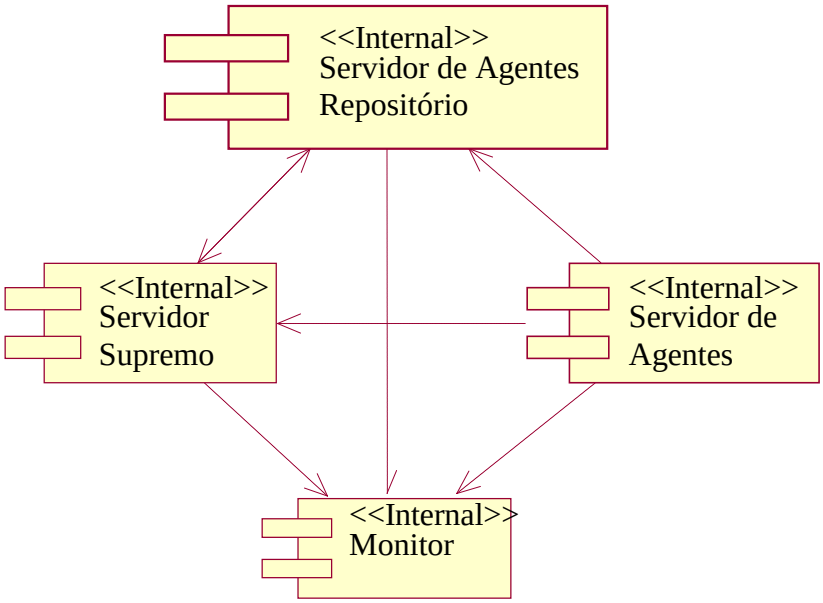

FIGURA 7-5. DEPENDÊNCIAS DOS COMPONENTES DE SERVIDORES.

➢ ***Supremo***

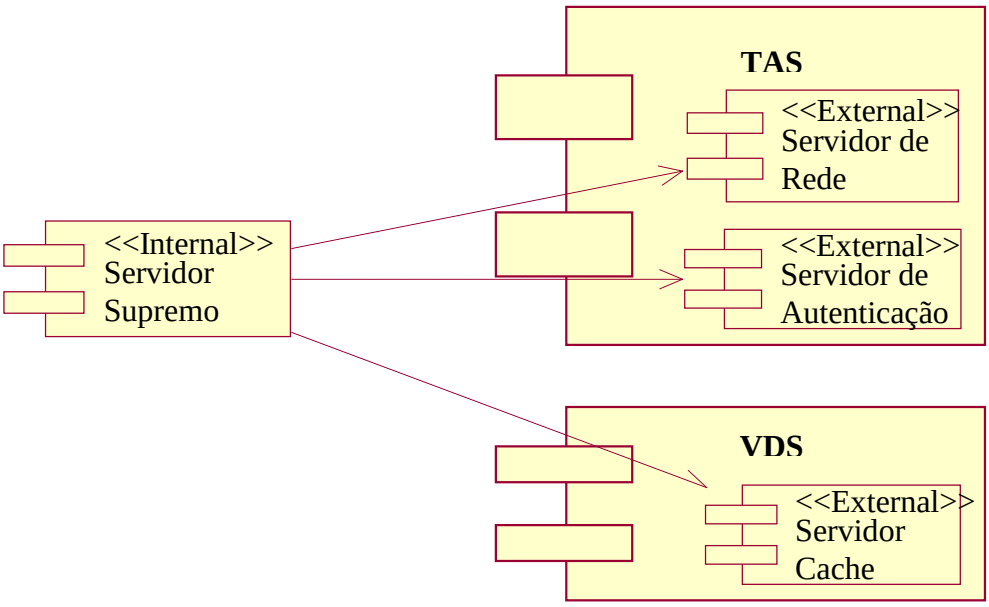

FIGURA 7-6. ARQUITECTURA DE COMPONENTES DO SERVIDOR SUPREMO.

- Responsabilidades

Este servidor distingue-se dos restantes por desempenhar funções específicas da sincronização da plataforma. As suas funções principais são as seguintes:

- Ponto central de identificação dos servidores (através de autenticação).
- Controla a criação de agentes de forma a garantir a unicidade de execução de um agente (um determinado agente apenas pode executar num servidor de agentes a cada instante).
- Verificar a disponibilidade de cada servidor.
- Notificar aplicações de gestão de acontecimentos ocorridos na plataforma.

➢ ***Servidor de Agentes***

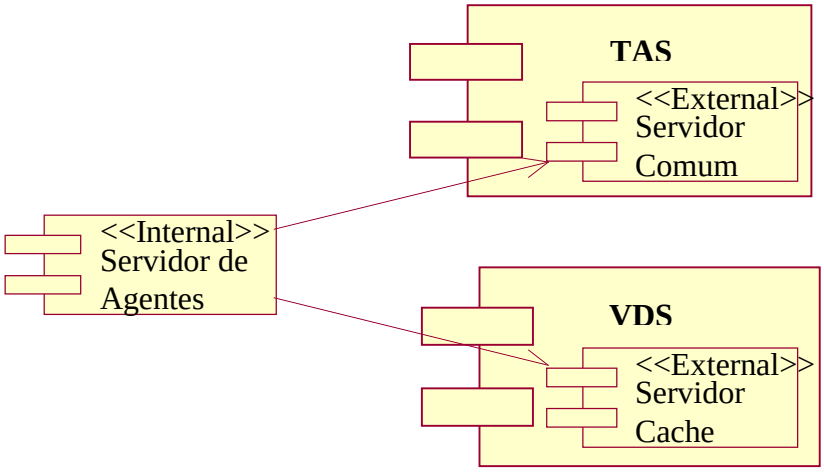

FIGURA 7-7. ARQUITECTURA DE COMPONENTES DO SERVIDOR DE AGENTES.

- Responsabilidades

Funcionalidades comuns a todos os servidores de agentes. Cada servidor é responsável por:

- Garantir a execução de agentes.
- Garantir a entrega de mensagens aos agentes.
- Permitir a deslocação de agentes entre dois servidores de agentes autorizados.
- Assegurar a transmissão segura do código do agente.
- Assegurar a transmissão de mensagens de forma segura, caso o seja pretendido pelo agente.

➢ ***Servidor Repositório***

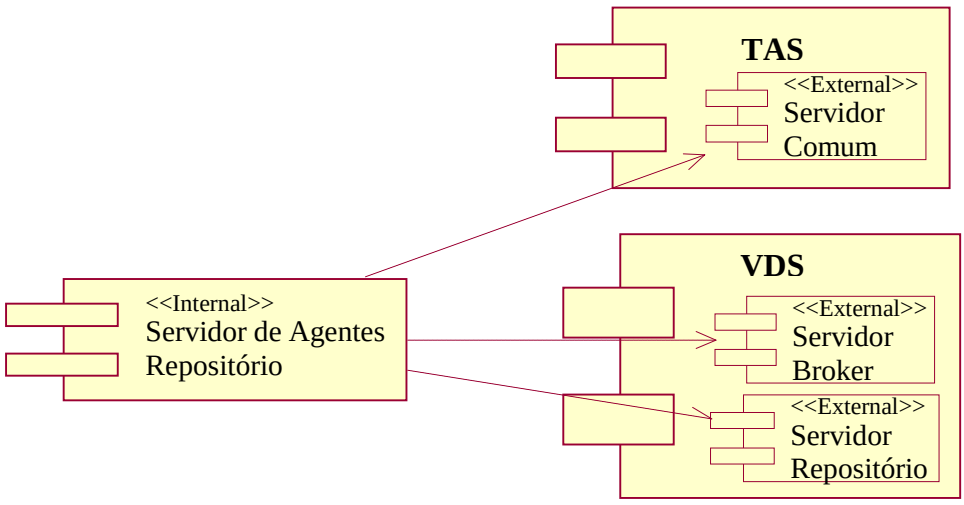

FIGURA 7-8. ARQUITECTURA DE COMPONENTES DO SERVIDOR DE AGENTES REPOSITÓRIO.

- Responsabilidades

Este servidor detém as mesmas funcionalidades e obrigações de um servidor de agentes, no entanto, indica aos restantes servidores que possui integrado o módulo repositório da VDS (ver apêndice B).

➢ ***Monitor***

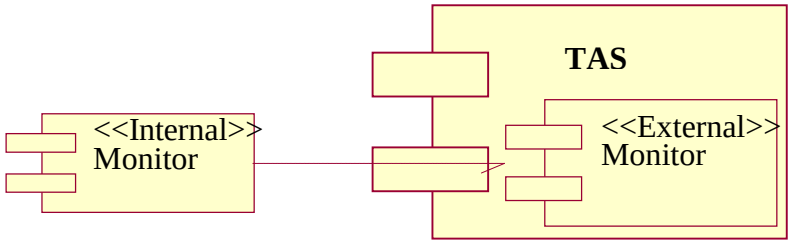

FIGURA 7-9. ARQUITECTURA DE COMPONENTES DA APLICAÇÃO MONITOR.

- Responsabilidades

Este cliente não se encontra integrado na plataforma, este pertence ao sistema TAS (ver apêndice B) que representa o sistema de rede de servidores. O monitor surge como um ponto central para o rastreio da plataforma, para o qual, todos os servidores podem enviar informações de diversos tipos (excepções, notificações e erros).

7.3.2 Arquitectura de servidores

➢ ***Arquitectura mínima***

Para que a plataforma desempenhe as suas funcionalidades esta deverá, no mínimo, deter o **servidor supremo** e um **servidor de agentes repositório**. É importante realçar que qualquer servidor pode executar simultaneamente na mesma máquina, o que significa que, no mínimo será necessário uma máquina para executar a plataforma. No entanto, para que todas as vantagens da distribuição sejam usufruídas é aconselhável a utilização do máximo de servidores possível.

Para que a plataforma exerça as suas funcionalidades da melhor forma, é aconselhável que o servidor supremo execute separadamente dos restantes. A razão para tal deve-se à excessiva carga operacional que este comporta motivada pela gestão de todos os agentes da plataforma.

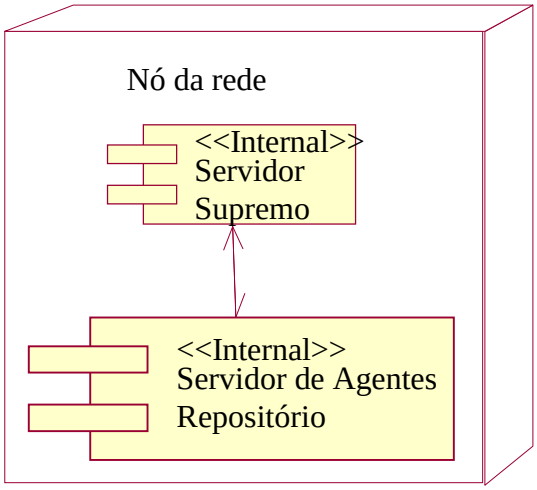

FIGURA 7-10. ARQUITECTURA MÍNIMA DE SERVIDORES.

- ***Arquitectura usual***

Como é de esperar a arquitectura usual desta plataforma compreende um servidor supremo, um servidor de agentes repositório, uma aplicação monitor e vários servidores de agentes, cada um destes módulos operando num nó da rede distinto, potenciando uma melhor distribuição e balanceamento de carga, escalabilidade e disponibilidade global do sistema.

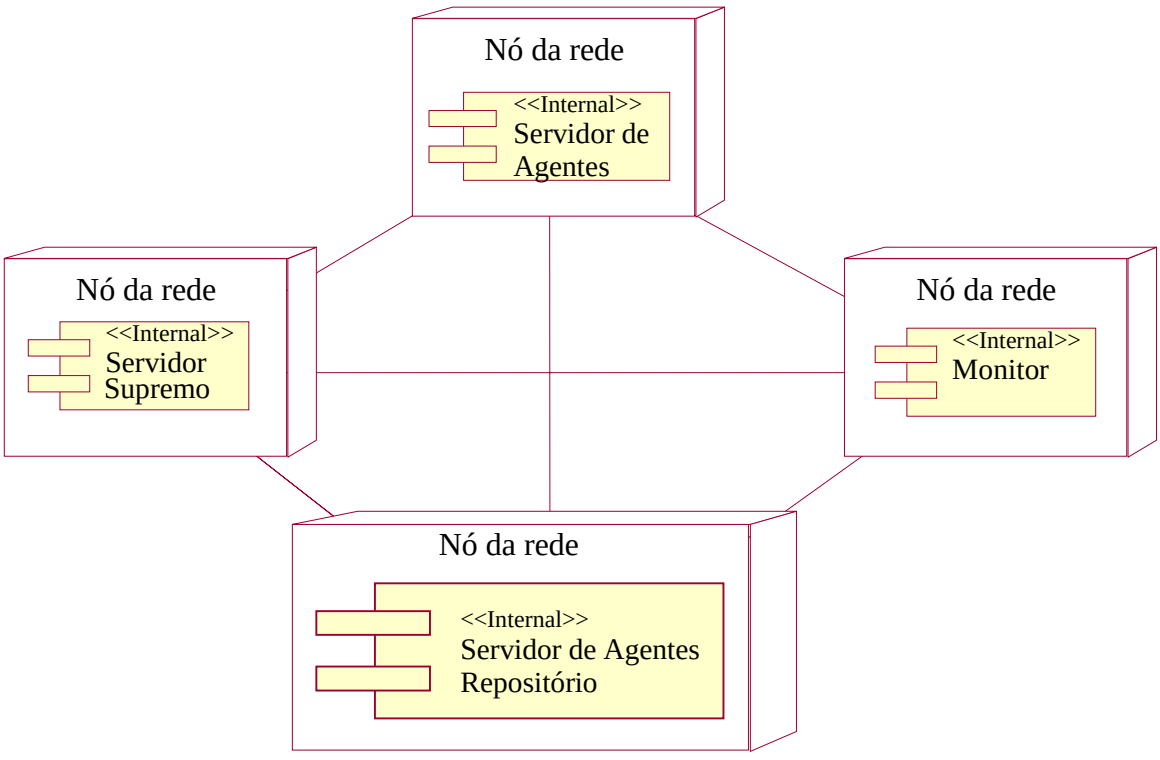

FIGURA 7-11. ARQUITECTURA DE USUAL DE SERVIDORES.

Capítulo 8

IMPLEMENTAÇÃO

RESUMO

Neste capítulo serão apresentadas as tecnologias utilizadas para o desenvolvimento desta plataforma, bem como, a nossa motivação para o seu uso.

8.1 ANÁLISE DAS TECNOLOGIAS UTILIZADAS

8.1.1 Linguagem de programação

Existem inúmeras linguagens de programação no mercado que possuem as capacidades para o desenvolvimento de um sistema desta envergadura, no entanto, poucas são as linguagens que permitam satisfazer os requisitos determinados para este, como a criação de agentes em tempo de execução, este pormenor é de elevada importância, na medida em que, oferece uma maior flexibilidade ao desenvolvimento de aplicações baseadas em agentes. Para satisfazer alguns destes requisitos será necessário uma linguagem que permita a sua interpretação em modo de execução.

Existem algumas linguagens que preenchem estes requisitos como o *JAVA™*, desenvolvido pela *Sun Microsystems*, o *Ansi Common Lisp* e o *DotNet* desenvolvida pela *Microsoft*. Destas linguagens escolhemos a linguagem de programação *JAVA™*, visto se encontrar na vanguarda das linguagens de programação, sendo a mais requerida pelos demais programadores. Este aspecto permite uma maior integração dos esforços no desenvolvimento de ABA (*Agent Based Aplications*, aplicações baseadas em agentes), existindo desta forma um ponto comum importante com o desenvolvimento das restantes aplicações.

O grande êxito desta linguagem deve-se ao facto de incorporar as melhores características das linguagens modernas (gestão de lixo - *garbage collection* e ligação dinâmica do *Lisp*, interfaces do *Objective-C*, pacotes do *Modula*, concorrência do *Mesa* e as excepções do *Modula-3*). Esta linguagem permite ainda, para além de outras características, a identificação de tipos em modo de execução através de um mecanismo denominado RTTI (*Run-Time Type Identification*) e o mecanismo *Reflection* que consiste na introspecção de objectos e/ou classes de objectos.

8.1.2 Ambientes de desenvolvimento

Durante a nossa formação trabalhamos com diversos ambientes de desenvolvimento para desenvolvimento de sistemas, no entanto, apenas alguns destes ambientes destinavam-se ao desenvolvimento sobre a linguagem Java, como por exemplo *VisualAge™*, *JBuilder™* e *Forte™*.

Pensamos que o ambiente que mais se adequa às nossas necessidades, corresponde à aplicação desenvolvida pela *Sun Microsystems*, denominada *Forte™*, para além de que, toda a implementação assentou na linguagem JAVA™, desenvolvida pela mesma empresa. Esta motivação para o seu uso assenta apenas na maior facilidade de implementação que esta ferramenta acresce em oposição às restantes.

Capítulo 9

ETAPA DE TESTES

9.1 INTRODUÇÃO

A etapa de testes apresenta-se como uma das etapas mais importantes do processo de desenvolvimento. É nesta etapa que todos os requisitos são verificados de forma a caminhar para um produto final robusto e estável.

9.2 FERRAMENTAS PARA TESTE AUTOMÁTICO

9.2.1 Ferramentas de análise de Código

Existem duas categorias de ferramentas de análise de código. Análise estática que é desempenhada quando o programa não se encontra em execução e análise dinâmica que, ao contrário da anterior, é efectuado quando o programa se encontra em execução.

Neste trabalho apenas foi dada ênfase apenas ao segundo tipo, tendo sido mesmo desenvolvido um sistema para satisfazer os requisitos necessários para a satisfação das suas tarefas. Este sistema intitula-se de ***Monitor and Tracking System*** (MTS) e será descrito neste documento (ver apêndice B).

- ***Análise Dinâmica***

  Em muitas ocasiões, sistemas são difíceis de testar devido à execução paralela (concorrência) de diversas operações. Esta situação é especialmente verdadeira para sistemas em tempo real. Nestes casos, é difícil antecipar condições e gerar casos de teste representativos. Ferramentas automáticas possibilitam à equipa de teste capturar o estado dos eventos durante a execução de um programa por preservação de *snapshots* de condições. Estas ferramentas são por vezes chamadas monitores de programas porque estas observam e reportam o comportamento do programa.

9.2.2 Ferramentas de execução de testes

As ferramentas anteriormente mencionadas focam no código. Outras ferramentas podem ser utilizadas para automatizar o planeamento e execução dos próprios testes. Dado o tamanho e complexidade da maioria dos sistemas de hoje, ferramentas de automatização de execução de testes são essenciais para manipular um grande número de casos de teste que necessitam de ser executados para testar completamente o sistema. Um tipo de ferramentas consiste na captura e replicação.

- ***Captura e replicação***

  Quando os testes são planeados, a equipa de testes deverá especificar um caso de teste, qual a informação que deverá receber (*input*) e qual o resultado que é esperado pelas acções que se encontram a ser testadas. Ferramentas de **captura e replicação** ou de **captura e *playback***, capturam as teclas pressionadas, informações recebidas e respostas ao longo da execução do teste, para além de, comparar o que é esperado com o que realmente foi retornado. Discrepâncias são reportadas à equipa de teste e as informações capturadas

ajudam a equipa a rastrear as discrepâncias até à raiz do problema. Este tipo de ferramenta é especialmente útil, após uma falta ter sido encontrada e corrigida, para verificar se a correcção foi bem sucedida e não introduziu outras faltas no código.

Reconhecendo a importância desta ferramenta foi desenvolvido um sistema avançado de testes que satisfaz os requisitos acima mencionados. Este sistema intitula-se de ***Advanced Testing System*** e será alvo de uma descrição alargada neste documento (ver apêndice B).

Capítulo 10

RESULTADOS

RESUMO

Neste capítulo serão apresentados os resultados obtidos da construção desta plataforma. Nomeadamente o seu desempenho na resolução de problemas.

A plataforma presentemente desenvolvida encontra-se extremamente estável em relação à primeira, conseguindo manter padrões de resposta aceitáveis e satisfazer a concorrência por completo sem a ocorrência de falhas. A modularidade é um dos factores que surpreendeu, visto a plataforma por si só, e através dos seus componentes permitir o acréscimo, com extrema facilidade, de novas funcionalidades.

Todos os requisitos suplementares foram satisfeitos, chegando mesmo a ultrapassar os níveis esperados, estes requisitos compreendem aos factores de modularidade (já referido), robustez, eficiência e segurança.

É importante realçar que grande parte dos resultados positivos advém da metodologia de desenvolvimento usada e dos sistemas anexos de teste, que permitiram o teste exaustivo dos módulos e componentes antes da sua integração no sistema.

Resumindo, pode-se considerar que a segunda iteração foi um completo sucesso em relação à primeira apesar de poucas funcionalidades terem sido acrescidas com a nova iteração.

Apesar de na segunda iteração não terem sido desenvolvidos tantos exemplos como para a primeira, estamos convictos que a nova responderá a estes com um melhor grau de desempenho.

É importante realçar que um desenvolvimento de raiz apresenta um maior grau de desempenho na resolução de problemas, contudo um desenvolvimento sobre a plataforma de agentes apresenta inúmeras vantagens: rápido desenvolvimento, maior grau de abstracção sobre o problema, garantia que as funções são executadas sem ocorrência de falhas ou erros, construção sobre uma arquitectura predefinida robusta e estável.

Estes aspectos verificaram-se na construção dos exemplos que serviram de demonstração da plataforma. Em poucos dias foi possível a construção destes exemplos, com um elevado grau de completude e satisfação do problema.

Capítulo 11

TRABALHO FUTURO

RESUMO

Neste capítulo serão apresentadas algumas ideias para trabalho futuro sobre a plataforma ADAP.

11.1 INTRODUÇÃO

É difícil chegar a um acordo sobre quando um sistema se encontra completo, esta plataforma não foge à excepção, como tal, existem algumas funcionalidades que podiam ser acrescidas que apresentassem uma mais valia para a plataforma em si e para os seus utilizadores. Optámos por diferenciar em dois grupos distintos: as funcionalidades relacionadas com a plataforma em si e as funcionalidades relacionadas com os agentes.

11.2 PLATAFORMA

11.2.1 Distribuição dos agentes

Distribuição da execução de agentes através da plataforma de forma a dispersar a carga de utilização. Podendo cada servidor definir um número máximo e médio aceitável de agentes que deseja executar em simultâneo.

11.2.2 Segurança

A nível de segurança, esta poderia ser completada com integridade do código e controlo de acesso (ACL) dos agentes.

11.2.3 Bancada de desenvolvimento

Desenvolvimento de uma **bancada de desenvolvimento** (*Workbench*) para criação e gestão de agentes segundo grupos de trabalho ou mesmo organizações empresariais. Podendo mesmo definir perfis de utilizadores, cada um com restrições de acesso.

11.3 AGENTES

11.3.1 Mensagens de alto nível

Sobre as mensagens existentes poderiam ser construídas mensagens mais complexas, as quais poderiam obedecer a determinadas ontologias ou mesmo linguagens de comunicação de agentes, permitindo desta forma a recepção de mensagens provenientes do exterior. Um exemplo seriam mensagens formatas em XML (*Extended markup language*) que obedecessem aos *standarts* da FIPA (*Foundation for Intelligent Physical Agents*).

11.3.2 Agentes broker

Adicionalmente poderia ser construído um agente *broker* de mensagens, cuja responsabilidade seria encaminhar mensagens entre agentes de acordo com determinados requisitos (e.g. tema de conversação, assunto pretendido, contexto do dialogo de conversação entre agentes, etc.).

11.3.3 Linguagem de programação de agentes

Desenvolver uma linguagem para programação de agentes (***Agent programing Language*** **APL**) que permita uma maior abstracção do programador dos pormenores intrínsecos à plataforma.

Capítulo 12

CONCLUSÕES

RESUMO

Neste capítulo serão apresentadas as principais conclusões deste trabalho.

O desenvolvimento desta plataforma de agentes mostrou-se ser bastante complexo, mas ao mesmo tempo motivador pelo conjunto de problemas de sincronização e eficiência abordados no seu desenvolvimento.

No desenvolvimento desta plataforma foi necessária a utilização de grande parte dos conhecimentos adquiridos ao longo do nosso percurso como futuros engenheiros, elevando as nossas capacidades a níveis superiores aos conseguidos em qualquer outra disciplina.

Pensamos que grande parte dos objectivos foram atingidos no desenvolvimento desta plataforma, quer a nível pessoal (alargar os conhecimentos sobre plataformas e toda a sua arquitectura), quer a nível do próprio sistema (desenvolvimento de uma plataforma suficientemente genérica de suporte de agentes móveis). No entanto, pensamos que existe ainda um grande trabalho de desenvolvimento sobre esta plataforma, de forma a torná-la realista a nível do mercado.

A utilização da plataforma na resolução de problemas cujo objectivo implica um rápido desenvolvimento, robustez, estabilidade e, principalmente, um elevado grau de abstracção, apresenta-se como uma solução elevadamente satisfatória, ou mesmo, ideal.

Pretendemos que esta plataforma não tenha apenas interesse a nível académico, mas venha a servir de base para o desenvolvimento de aplicações reais baseadas em agentes, ou mesmo, que seja uma referência para futuras implementações de plataformas.

Após todo o desenvolvimento desta plataforma e de toda a investigação a nível de plataformas e aplicações baseadas em agentes, esperamos que o desenvolvimento de aplicações baseadas nestes princípios e desenvolvidas sobre plataformas se tornem uma prática comum no futuro. Gostaríamos que todo este desenvolvimento não se desperdice e não fique esquecido no tempo.

REFERÊNCIAS

- ***Agentes de Software***

[Wei] Weiss G. Multiagents Systems – A modern approach to distributed artificial intelligence. USA: MIT Press.

[SRDS01] Alberto Rodrigues da Silva, Artur Romão, Dwight Deugo and Miguel Mira da Silva. Towards a Reference Model for Surveying Mobile Agent Systems. Autonomous Agents and Multi-Agent Systems Journal. Kluwer Academic Publishers, September 2001.

[Sil98] Alberto Silva. ESPAÇO DE AGENTES: Suporte, Desenvolvimento e Gestão de Aplicações Baseadas em Agentes, Dinâmicas e Distribuídas. Tese de Doutoramento, Universidade Técnica de Lisboa, Instituto Superior Técnico, Junho 1998.

[Sil99] Alberto Silva. *Agentes de Software na Internet*. Centro Atlântico, 1999.

[Bra97] Bradshaw J. *Software Agents*, MIT Press (1997).

[CP] Coelho H. e Paiva A. *A mente e o mundo lá fora.* (Draft)

- ***Distribuição***

[MG99] José Alves Marques e Paulo Guedes. *Tecnologia de Sistemas Distribuídos*, FCA Editora de Informática, 2ª edição, 1999.

- ***Engenharia de Software***

[Mcc98] Steve McConnell. *Software Project Survival Guide*, Microsoft Press, 1ª edição, 1998.

[Pfl01] Shari Lawrence Pfleeger. *Software Engineering, Theory and Practice*, Prentice Hall, 2ª edição, 2001.

- ***Análise de Requisitos***

[Wie99] Karl E. Wiegers. *Software Requirements*, Microsoft Press, 1ª edição, 1999.

- ***Desenho e Arquitextura***

[CD01] John Chessman, John Daniels. UML Components: A Simple Process for Specifying Component-Based Software, Addison-Wesley, 1ª edição, 2001.

[HNS00] Christine Hofmeister, Robert Nord, Dilip Soni. Applied Software Architecture, Addison-Wesley, 1ª edição, 2000.

[GHJV00] Erich Gamma, Richard Helm, Ralph Jonhson, John Vlissides. Design Patterns: Elements of reusable Object-Oriented Software, Addison-Wesley, 1ª edição, 2000.

- ***Testes***

[Bla99] Rex Black. Managing the Testing Process, Microsoft Press, 1ª edição, 1999.

- ***Metodologias***

[SV01] Alberto Silva, Carlos Videira. *UML, Metodologias e ferramentas CASE*, Edições Centro Atlântico, 1ª edição, 2001.

- ***Tecnologia Java***

[Eck00] Bruce Eckel. *Thinking in Java*, Second Edition, Prentice Hall, June 2000.

[CWH00] Mary Campione, Kathy Walrath, Alison Huml. *The Java Tutorial*, Third Edition, Addison Wesley, January 2000.

[AGH00] Ken Arnold, James Gosling, David Holmes. The Java Programming Language, Third Edition, Addison Wesley, June 2000.