Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

Self-Organization and Complexity Theory to Support Evolvable Production Systems

Bruno Domingos Ferreira

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador: *José António Barata de Oliveira*

Monte de Caparica
Fevereiro de 2009

Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

Self-Organization and Complexity Theory to Support Evolvable Production Systems

Bruno Domingos Ferreira

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador: *José António Barata de Oliveira*

Monte de Caparica
Fevereiro de 2009

Agradecimentos

Começo por agradecer à minha família e em particular à minha mãe, e irmãos por todo o apoio que me deram em todos os sentidos, durante o processo que permitiu a conclusão desta tese. Ao meu pai, que apesar de já não se encontrar entre nós, sempre me apoiou e deu conselhos. Um agradecimento muito especial também à Nicole, a minha namorada por toda a paciência que teve e pelo apoio que sempre mostrou.

Um muito obrigado ao meu orientador, o Professor Dr. José Barata por toda a confiança que depositou em mim e pela disponibilidade para esclarecimentos que sempre mostrou do decorrer desta tese.

Agradeço também à doutoranda Regina Frei, cujas discussões se transformaram em ideias, e ideias em soluções.

Ao meu colega e amigo António Amado, que de uma forma bastante significativa me apoiou ao longo de toda a tese, desde a troca de ideias, até às noites de debug, um muito obrigado.

Aos meus colegas Nelson Silvério, Michel Rodrigues, Pedro Mendes, Luís Ribeiro e a todos os colegas do Departamento de Engenharia Electrotécnica e de Computadores que de alguma forma contribuíram para o desenvolvimento desta tese, ou simplesmente pela companhia na máquina do café.

À Vanessa Inácio um obrigado de última hora, não sendo no entanto o menos importante.

A todos os colegas da residência universitária Fraústo da Silva, obrigado pela companhia ao longo destes cinco longos anos.

Faculdade de Ciências e Tecnologia, Fevereiro de 2009

_____

Bruno Domingos Ferreira

Resumo

De forma a tornar os sistemas de produção evolutivos uma realidade, devem ser elaboradas formas de implementar capacidades de auto-organização.

Os sistemas de produção evolutivos são compostos por módulos do sistema agentificados, os quais necessitam de interagir e colaborar entre eles. A aproximação de *"Plug&Play"* significa que a reprogramação desses módulos face às possíveis alterações deve ser evitada. Em vez disso os agentes organizar-se-ão entre eles de forma a conseguirem responder a todas as acções de montagem solicitadas. Para tal os agentes necessitam de conhecer informação sobre o módulo que representam: funcionalidades, geometrias, dimensões, interfaces, limitações, entre outras. Também as partes que irão compor o produto final, as instruções de montagem, e os processos, devem ser definidos de uma forma abstracta e genérica para que o sistema tenha o máximo de liberdade para se auto-reconfigurar em caso de necessidade.

Esta tese visa mostrar como é que se pode definir as características sobre os módulos de uma forma genérica, e que possam facilmente ser interpretadas pelo computador ou pelo utilizador. Por fim serão mostradas e detalhadas as interacções que mostram como é que a auto-organização surge.

Palavras-Chave:

Auto-organização, Sistemas Distribuídos, Reconfigurabilidade, Coligações Dinâmicas.

Abstract

To turn Evolvable Production Systems (EPS) a reality, ways of concretely implementing Self-Organization were elaborated.

The EPS are composed by agentified system modules, which need to interact and collaborate between them. A *"Plug&Play"* approach means that reprogramming of modules when changes occur should be avoided. Instead, the agents will organize themselves in order to fulfill the assembly tasks. To perform this, agents need to know information about the module they represent: functionalities, geometries, dimensions, interfaces, limitations, among others. Also the parts of the product, the assembly instructions and the processes must be defined in an abstract, generic way in order to give the system as much freedom as possible for reconfiguration in case of need.

This thesis will show how one can define the characteristics about the modules in a generic way, and that can easily be interpreted by the computer and by the user. In the end will be showed and detailed these interactions and show how Self-Organization takes place.

Keywords:

Self-Organization, Distributed Systems, Reconfigurability, Dynamic Coalitions.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas

ACC	Agent Communication Channel
ACL	Agent Communication Language
AMI	Agent Manufacturing Interface
AMS	Agent Management System
BDI	Belief Desire Intention
CFP	Call For Proposals
CoBASA	Coalition Based Approach for Shop Floor Agility
DCA	Dynamic Coalition Agent
DF	Directory Facilitator
DOM	Document Object Model
DTD	Document Type Definition
EAS	Evolvable Assembly Systems
EPS	Evolvable Production Systems
FA	Feeder Agent
FIPA	Foundation for Intelligent Physical Agents
FMS	Flexible Manufacturing Systems
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
IP	Internet Protocol
JADE	Java Agent Development Framework
JDBC	Java Database Connectivity

JRE	Java Run-time Environment
JVM	Java Virtual Machine
MRA	Manufacturing Resource Agent
NS	Naming Service
OA	Order Agent
OntA	Ontology Agent
PA	Product Agent
PSA	Palette Storage Agent
RMI	Remote Method Invocation
RMS	Reconfigurable Manufacturing Systems
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
TA	Transport Agent
UML	Unified Modeling Language
XML	EXtensible Markup Language
XSD	EXtensible Schema Definition
WhA	Warehouse Agent
WPCA	Work Piece Carrier Agent

Capítulo 1. **Introdução**

1.1 Motivação

Ao longo dos anos tem-se vindo a observar uma contínua evolução no mundo da indústria de manufactura, desde a forma como os produtos são desenhados, até à forma em que são produzidos e distribuídos, nas mais diversas áreas (calçado, alimentar, farmacêuticas, automóvel, electrónica, entre muitas outras).

Até ao início do século XX os produtos eram desenvolvidos à medida e ao gosto do cliente, tendo-se assim um elevado nível de personalização e de satisfação do cliente. No entanto os produtos eram caros e como tal, apenas abrangiam uma pequena percentagem da população.

A produção em massa teve início no princípio do século XX, tendo atingido o auge nas décadas de 40 e 50, produzindo-se grandes quantidades de produtos idênticos, o mais rápido e o mais barato possível, no entanto esta metodologia de produção era pouco flexível, deixando pouco espaço para a inovação e para rápidas mudanças no mercado. Nesta época, apesar de nunca ter sido provado que foi Henry Ford quem a proferiu, ficou conhecida a célebre frase sobre o modelo T da Ford "*The customer can have any color as long as it is black"* ("o cliente pode ter qualquer cor desde que seja preta"), evidenciado a consequência deste avanço tecnológico, que se reflectiu na limitação das gamas de produtos e consequentemente no consumidor que deixou de ter opções de escolha, ficando a satisfação do cliente para segundo plano.

A partir da década de 80, com o desenvolvimento socioeconómico e o aumento do nível de vida, houve um aumento do consumismo de produtos com ciclo de vida curto, descartáveis e mais personalizados. Estas novas circunstâncias levaram a uma grande necessidade de inovação na indústria para esta conseguir gerar produtos exclusivos a baixo preço e de qualidade que satisfizessem os novos padrões de exigência dos consumidores.

Esta nova filosofia de produção que ficou conhecida como "*mass customization"* isto é, produção em massa personalizada, requer unidades fabris mais pequenas e consequentemente mais flexíveis.

No entanto, só o facto das unidades fabris se tornarem mais flexíveis não chega, é necessário que se consigam auto-organizar de modo a que suportem elevados níveis de reconfigurabilidade.

Na Figura 1-1 é apresentada a evolução dos paradigmas de manufactura ao longo dos anos.

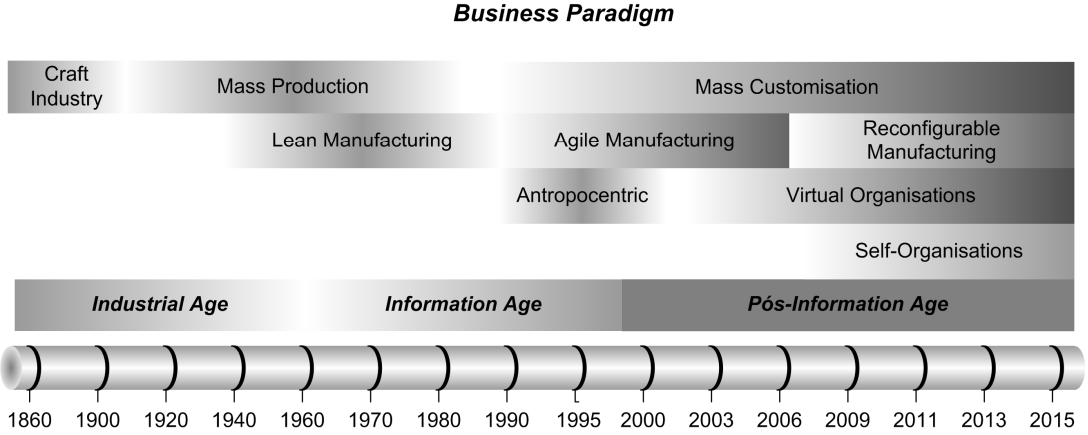

Figura 1-1- Paradigmas da Manufactura. [adaptada de (Barata 2003)]

As soluções para os novos sistemas de controlo de unidades fabris devem ir então ao encontro dos seguintes objectivos, de forma a preencher os requisitos do mercado:

- Os módulos têm de ter uma interface amigável e visualmente apelativa de modo a permitir uma interacção quase intuitiva;

- Tem de ser altamente flexível, de modo a ser personalizado facilmente para diferentes tipos de estrutura de produção, o software para controlo tem de se adaptar à organização do ambiente de manufactura. Com o objectivo de uma optimização contínua, este ambiente é reestruturado e reorganizado frequentemente. Deve ser possível a actualização das ferramentas de modo a suportar tais mudanças;
- Tem de ser possível a integração do sistema de controlo da unidade fabril com os sistemas de agendamento, de produção de planos e sistemas de controlo já existentes. Soluções isoladas não são mais aceitáveis na indústria de manufactura moderna;
- Tem de permitir o uso de interfaces de outros sistemas – A integração de informação e sistemas já existentes garantido a interoperabilidade torna-se cada vez mais importante;
- Tem de facilitar a cooperação e a comunicação entre diferentes células de manufactura dentro de uma mesma unidade de manufactura. A redução de níveis na estrutura organizacional cria a necessidade de uma comunicação e cooperação horizontal. A hierarquia de comando vertical é substituída por uma interacção directa no nível da unidade fabril;
- Tem de permitir a cooperação entre fornecedores, produtores e consumidores, permitindo um maior grau de integração ao longo da cadeia de fornecimento externa;
- Possuir sistemas de controlo, manutenção e correcção de falhas/ erros;
- Tem de existir um conjunto de regras bem definidas de modo a que o sistema não siga acções incoerentes ou que estas não ocorram de acordo com o desejado.

1.2 Descrição do trabalho efectuado

Tendo em conta os problemas descritos anteriormente, este trabalho visa colmatar algumas lacunas existentes nos sistemas de produção actuais implementado para tal alguns dos objectivos definidos anterioremente.

Deste modo, procurou-se implementar um sistema baseado em multi-agentes com uma interface amigável para o utilizador, que fosse independente das características do sistema ou dos próprios módulos que o constituem, utilizando para tal, uma forma genérica de descrever cada um dos módulos e em que a sua alteração não implica a reprogramação de nenhum destes.

O conceito de auto-organização foi conseguido através da formação dinâmica de coligações, as quais são iniciadas por um módulo que perceba que não consegue realizar uma acção que lhe foi requisitada sozinho. Para o funcionamento correcto das coligações foi implementado um algoritmo que define a ordem correcta das acções a executar.

Assim de uma forma sucinta este trabalho propõe-se a preencher requisitos apresentados anteriormente pelo que na construção desta arquitectura ter-se-á em conta alguns pontos importantes, tais como:

- A alteração do produto a fabricar, bem como alterações nas condições de produção não implicaram a alteração de qualquer código, bem como a adição, remoção ou alteração de módulos;
- Cada módulo da célula (tapete, robô, máquina) deverá ser controlado por um agente. Sobre cada agente haverá comunicação, que será feita através dos protocolos *"FIPA REQUEST"(FIPA 2002a)* e *"FIPA CONTRACT NET" (FIPA 2002c)*, suportados pelo ambiente de desenvolvimento JADE (*"Java Agent Development Framework"*) (Bellifemine, Poggi et al. 1999);
- Todos os módulos presentes na célula "*MOFA France*" terão um ficheiro de configuração inicial de forma a serem caracterizadas as suas funcionalidades com os respectivos requisitos, propriedades genéricas e limitações.

- A formação e gestão das coligações serão efectuadas de forma automática e dinâmica.

O sistema, a partir do plano de montagem do produto deverá ser capaz de realizar automaticamente as operações que conduzam à montagem total do produto.

1.3 Organização da tese

De acordo com o âmbito da tese, atrás definido, esta dissertação foi então organizada em seis capítulos:

O primeiro capítulo contextualiza o problema existente no mundo da indústria de manufactura, e são dados a conhecer alguns aspectos que devem ser tidos em atenção quando se propõe uma solução para o problema.

O segundo capítulo pretende mostrar a forma como outros autores abordaram o mesmo problema e as soluções que encontraram, bem como as suas vantagens e desvantagens.

De seguida é feita uma descrição das tecnologias que deram suporte à implementação desta tese.

No quarto capítulo é apresentada de forma breve e detalhada a arquitectura proposta para a solução do problema apresentado anteriormente.

O quinto capítulo refere-se à implementação, expondo-se um caso de estudo.

Por fim, são apresentadas as conclusões e perspectivas de trabalho futuro.

Capítulo 2. **Estado da arte**

Os problemas no mundo da manufactura têm sido amplamente abordados ao longo dos anos bem como a aplicabilidade dos agentes na manufactura (Hall, Staron et al. 2005; Monostori, Váncza et al. 2006; Barata, Cândido et al. 2007), de onde surgiram novos conceitos, tais como FMS (*"Flexible Manufacturing Systems"*) (Hoda 2006), RMS (*"Reconfigurable Manufacturing Systems"*) (Koren, Heisel et al. 1999), HMS (Valckenaers, Brussel et al. 2005), EPS (*"Evolvable Production Systems"*) (Barata, Onori et al. 2007), EAS (*"Evolvable Assembly Systems"*) (Onori, Barata et al. 2006), entre outros. Ao longo do próximo capítulo irá ser feita uma breve descrição destes conceitos de modo a verificar as diferenças entre eles.

Aplicações concretas no mundo da manufactura podem ser encontradas em (Mařík and Lazansky 2004; Mařík, Vrba et al. 2005).

2.1 Sistemas de Produção

2.1.1 Evolutivos (EPS/ EAS)

Os EPS / EAS são constituídos por um módulos orientados para o processamento representados por agentes, possuindo não só a habilidade de se adaptarem a novas condições de funcionamento, como também acompanhar a

evolução desses mesmos módulos. Os aspectos de manutenção são também suportados bem como a integração de componentes *"Plug&Play"*, estes tipos de sistemas reforçam ainda a ideia de que o controlo descentralizado é o mais adequado para tais requisitos.

Os EPS / EAS pretendem assim oferecer uma solução prática através de mecanismos que permitem uma rápida reconfiguração da plataforma fabril, aos níveis mecânicos e de controlo. Esta reconfiguração é conseguida através de alguns conceitos, os quais se encontram descritos em (Onori, Barata et al. 2006) e são enumerados de seguida:

- **Módulo –** É uma unidade capaz de realizar uma operação e integrar uma interface específica. Níveis de granularidade devem ser definidos (o nível mínimo, e o maior grau de emergencia);
- **Granularidade –** O menor nível de granularidade de um módulo dentro de uma arquitectura de referência será uma gripa ou um suporte; o maior comportamento de emergencia, será se uma gripa consegue comunicar com um robô. Novas características operacionais deveram emergir (virar produto, parte em movimento, ajustamento de posição, ...), no entanto isto implica que um conjunto de definições e formas de gerir informação surjam, os níveis mínimos devem ainda ser clarificados;
- **Conectividade** – A habilidade para voltar a encontrar e integrar módulos do sistema com a plataforma de trabalho de uma dada arquitectura. O novo *"layout"* não deverá excluir a performance simplesmente se irão juntar e formar uma nova disposição;
- **Configurabilidade/ Interoperabilidade** – A habilidade para encontrar novos módulos do sistema que tenham disponíveis novas, (no entanto pré-definidas) operações (conectividade mais as características que asseguram a performance eficiente do novo *"layout"*);
- **Evolução** – Um sistema completamente reconfigurável que exibe comportamentos emergentes os quais introduzem níveis de funcionalidade novos ou redefinidos. Para tal requer-se uma arquitectura de referência

rigorosamente bem definida de modo a permitir a correcta aplicação das características mais relevantes.

As diferenças entre os EPS e os RMS são descritos em (Barata and Onori 2006) bem como os objectivos a que os EPS se propõem, os quais são enumerados de seguida:

- Módulos orientados aos processos;
- Tarefas específicas (pequenas) por elemento (módulo);
- Sistema de controlo distribuído e descentralizado;
- Interoperabilidade;
- Contínua evolução do sistema através da troca, adição e remoção de módulos;
- Unidades de sistema auto-configuráveis;
- Habilidade para explorar as tecnologias de agentes de modo a captar comportamentos emergentes;
- Soluções avançadas de interfaces electromecânicas;
- Módulos com alguma "inteligência" embutida.

Uma descrição da evolução dos EPS pode ser encontrada em (Barata, Santana et al. 2006). Sendo alguns esclarecimentos, a arquitectura de referência e desenvolvimentos realizados encontrados em (Maraldo, Onori et al. 2006).

Os conceitos apresentados anteriormente têm sido aplicados a soluções inovadoras e com boas perspectivas, tais como o CoBASA (*"Coalition Based Approach for Shop Floor Agility"*) (Barata 2003; Barata and Camarinha-Matos 2003).

2.1.2 Reconfiguráveis, Flexíveis e Holónicos (RMS, FMS e HMS)

Numa primeira aproximação estes três paradigmas de produção podem parecer bastante semelhantes, no entanto apresentam características diferentes que se reflectem no tipo de produção.

Os sistemas flexíveis podem ser vistos como tendo a capacidade de alterar ou assumir diferentes posições ou estados em resposta a alteração dos requisitos de produção com uma pequena penalização no tempo, esforço, custo ou performance. Sendo assim um tipo de produção bastante orientada para a personalização e pouco para a produção em massa.

Os sistemas reconfiguráveis incorporam princípios de modularidade, integrabilidade, flexibilidade, escalonamento, convertibilidade e diagnóstico. Deste modo o principal objectivo dos RMS não é tanto a flexibilidade do sistema, mas sim estar permanentemente actualizado permitindo uma rápida reconfiguração. Em relação à flexibilidade pode-se afirmar então que os RMS são um meio-termo entre as linhas de produção dedicadas e os FMS.

As diferenças entre FMS e RMS são exaustivamente detalhadas em (Koren, Heisel et al. 1999; Hoda 2006; Mehrabi, Ulsoy et al. April 2002).

Os sistemas de manufactura holónicos (HMS) (Valckenaers, Brussel et al. 2005) concentram-se principalmente nos aspectos morfológicos do sistema.

Um *"holon"*, é caracterizado como sendo "a parte como um todo e o todo como uma parte", uma definição mais detalhada de o que é um *"holon"* ou uma holarquia, bem como uma arquitectura de referência para os HMS pode ser encontrada em (Brussel, Wyns et al. 1998)

As principais características de um HMS são descritas em (Gruver, Kotak et al. 2003), e são enumeradas de seguida:

- Composta por elementos autónomos, cooperativos, reutilizáveis e auto-configuráveis, isto é, os *"holons"*;
- Estrutura recursiva dos elementos (a dualidade natural dos *"holons"*);

- Reconhecimento do papel importante das pessoas para o sucesso global;
- Inexistência de controlo central na planta fabril;
- Integração de trabalho humano nas células de fabrico.

As diferenças entre os HMS e os EPS são realçadas em (Barata, Onori et al. 2007).

2.2 Auto-organização (Self-Organization)

Uma descrição do conceito de auto-organização aplicado a sistemas multi-agente pode ser encontrado em (Serugendo, Gleizes et al. 2005a; Serugendo, Gleizes et al. 2005b).

Auto-organização é o mecanismo ou processo que permite um determinado sistema alterar a sua configuração sem intervenção ou sem o controlo explícito durante o funcionamento do sistema, não existindo controlo centralizado ou hierárquico. É essencialmente uma (re-)organização espontânea e dinâmica, da estrutura ou composição do sistema que pode surgir das interacções locais.

As razões para a implementação de conceitos de auto-organização nos EPS são descritas em (Frei, Barata et al.):

- Minimizar e facilitar a interface com o utilizador, isto é, tornar transparente a complexidade e o aumento de autonomia do sistema.
- Desenhar programar e configurar um sistema composto por inúmeras entidades com múltiplas interacções laterais é uma tarefa complexa e demorada, assim quanto mais autónomo o sistema for, mais fácil se torna para o utilizador.
- Os sistemas de produção tendem a ter muitas componentes e de diversas naturezas, interagindo de diferentes formas. Os agentes têm assim de ter a capacidade de reorganizarem as suas colaborações, em formas e

composições diferentes de acordo com as necessidades, sem no entanto passarem por um ponto de coordenação central.

- Na presença de perturbações ou modificações o sistema é capaz de modificar a sua organização enquanto contínua a manter as suas funcionalidades. Isto significa na prática que o sistema de controlo deverá ser capaz de lidar com os problemas e se necessário encontrar formas alternativas de produção.

O maior desafio nas aplicações de manufactura é deixar que o sistema se auto-organize e ao mesmo tempo prever ou determinar o seu comportamento, deste modo os sistemas podem necessitar de um líder (BA - *"Broker Agent"* ou eventualmente um humano), para efectuar decisões

2.3 Coligações na Manufactura

O conceito de coligações ou a formação de coligações no mundo da manufactura não é particularmente novo, outros autores abordaram o mesmo problema, no entanto com soluções diferentes e não dinâmicas.

Um exemplo, é o CoBASA o qual se pode encontrar descrito em (Barata 2003), o qual será brevemente apresentado de seguida.

O CoBASA é constituido por uma sociedade de agentes, que representam os recursos fisicos e pertencentem à mesma estrutura fisica (unidade fabril) capazes de oferecer funcionalidades a essa mesma estrutura. Quando uma oportunidade de re-engenharia surge os agentes capazes de executar as novas acções, e que sejam compativeis são então escolhidos para participar em coligações, sendo essas coligações reguladas através de contratos bilaterais.

Os principais comportamentos do CoBASA são seis, e os quais são enumerados de seguida:

- Registo dos agentes num *"Cluster"*;
- Criação de novas coligações / consórcios;
- Modificação das coligações / consórcios;

- Funcionamento das coligações;
- Dissolução das coligações;
- Execução das acções requeridas.

No CoBASA as coligações eram criadas manualmente por um utilizador designado por BA. Esta abordagem tem como vantagem o facto de não serem criadas coligações que utilizem mais recursos que o necessário, por serem criadas para um objectivo bem definido irão ter um maior desempenho na execução das acções. No entanto, esta abordagem também tem as suas consequências, tais como a formação que é um processo lento e moroso comparado com a criada dinamicamente através de software, podem ocorrer erros de incompatibilidades por existirem demasiadas variáveis a serem tidas em conta, existe ainda o problema de não se conseguir fazer uma reutilização eficaz dos módulos.

O *"ProPlanT"* descrito em (Pěchouček, Mařík et al. 2000), também aborda o mesmo problema com soluções diferentes.

Uma primeira solução (*"Autonomous Coalition Formation in a Structured Community"*) era uma formação de coligações de forma semi-automática numa comunidade estruturada.

Esta abordagem para a formação de coligações é considerada pelos autores como sendo semi-automática no sentido em que tem de existir uma estrutura predefinida das possíveis combinações entre os agentes.

A segunda solução encontrada designada, por *"Autonomous Coalition Formation in a Community of Peer Agents"*, referia-se a uma formação completamente dinâmica, no entanto é descrita com não sendo trivial e com um número de combinações enorme, desta forma estes optaram pela primeira solução tendo surgido o *"TRI.Base Acquaintance Model"*.

Este modelo incluía uma estrutura de colaboração administrada por três bases distintas. O *"Cooperator base"* que armazena conhecimentos permanentes, tais como capacidades, endereços, linguagem de comunicação etc., sobre os membros que colaboram na comunidade. O *"State base"* que armazena informação de uma forma não permanente, essa informação incluía as capacidades actuais e os

objectivos dos agentes. Por fim, o *"Task base"* que continha os planos de como decompor e atribuir tarefas.

Tendo em conta as desvantagens das soluções apresentadas anteriormente, são necessários métodos para a formação dinâmica de coligações, tal como descritos em (Frei, Ferreira et al. 2008), é ainda necessário um conjunto de regras bem definidas que sirvam como guias durante o processo de produção, bem como o registo de dados relativos à performance e registo de actividades de cada módulo (Ferreira, Frei et al. 2009).

Existem ainda outros trabalhos relacionados com a formação de coligações em agentes, no entanto apenas de carácter teórico tal como o descrito em (Sandholm and Lesser 1997)

2.4 Teoria da Complexidade

Complexidade é um fenómeno presente em muitos sistemas e sociedades: natureza, células, organismos, ecossistemas, cidades, países, software em larga escala, negócios, governos, internet e muitos outros.

Os sistemas complexos consistem num grande conjunto de entidades heterogéneas que comunicam entre si. Destas interacções resultam comportamentos com muitos graus de liberdade e de elevada não linearidade tendo normalmente tendência a evoluir e adaptarem-se, exibindo assim comportamentos de aprendizagem.

A teoria do caos (tal como a auto-organização) são partes muito importantes da teoria da complexidade, no entanto existem algumas diferenças. O estudo da teoria da complexidade procura causas simples que levam aos comportamentos complexos, enquanto a teoria do caos estuda casos onde os resultados futuros são arbitrariamente sensíveis a pequenas mudanças nas presentes condições. Note-se que não se deve confundir o caos com a aletoriedade, uma vez que os

acontecimentos aleatórios não são possiveis de reproduzir novamente e são imprevissíveis.

Um estudo mais exaustivo e detalhado sobre as teorias da complexidade e do caos, bem com as definições das características dos sistemas complexos que seram apresentadas de seguida pode ser encontrada em (Érdi 2007).

Algumas características dos sistemas complexos são:

- Causalidade circular, retroacções, paradoxos lógicos;
- Pequenas alterações nas causas implicam efeitos dramáticos;
- Emergencia e imprevisibilidade.

Os EPS consistem em módulos de equipamento que estão conectados uns com os outros e têm múltiplas interacções laterais. Conjuntamente, os módulos formam um sistema com o comportamento global desejado.

A teoria da complexidade pode assim ser aplicada aos sistemas de manufactura uma vez que frequentemente exibem sensibilidade a condições específicas e a pequenos distúrbios. Certos factores levam a falhas de sistema, enquanto outros não têm efeito significativo. É assim difícil prever as circunstâncias críticas e lidar com estas.

Uma abordagem de reengenharia promissora baseada na teoria da complexidade é descrita em *"Foundations of Complex System Theories*: *the Synthetic Microanalysis"* (Auyang 1998), que propõem uma viagem interactiva do todo para as suas partes e destas para o todo.

Capítulo 3. **Tecnologias de Suporte**

O trabalho apresentado foi desenvolvido sobre a plataforma JADE (Bellifemine, Poggi et al. 1999) permitindo a comunicação entre os agentes. Para a inicialização dos agentes foram utilizados ficheiros XML (*"EXtensible Markup Language"*) (Bray, Paoli et al. 2008) com a respectiva descrição e validação efectuada através de ficheiros XSD (*"EXtensible Schema Definition"*) (W3C 2004).

3.1 Agentes

3.1.1 Características gerais dos Agentes

Os agentes inteligentes constituem um novo paradigma para o desenvolvimento de sistemas informáticos que tem sido aplicado com êxito, nos últimos anos, na solução de problemas tão diversos como:

- Pesquisadores na internet;
- Gestão dinâmica de redes de distribuição de electricidade;
- Gestão e selecção de recursos em empresas geograficamente distribuídas;
- Mercados de electricidade/ aquecimento;
- Gestão de "portfolio" de acções;

- Gestão inteligente de redes de computador;
- Gestão dinâmica de tráfego;
- Encaminhamento dinâmico de veículos de transporte;
- Integração de informação de pacientes em ambiente hospitalar;
- Sistemas multi-robôs;
- Virtualização de empresas.

De uma forma geral, um agente tem a capacidade de executar determinadas tarefas, tais como captar informação de uma rede, regular agendas, fazer negociações, sempre de uma forma autónoma. A sua popularidade resulta precisamente desta capacidade de relativa autonomia. Porém, devido à enorme complexidade e conhecimento que a criação e o desenvolvimento de um agente exigem, são, por vezes, cometidos erros que conduzem a resultados insatisfatórios, cujas consequências são imprevisíveis.

Nos últimos anos tem-se verificado uma grande expansão na área de investigação de agentes inteligentes/autónomos.

No entanto, esta evolução tem sido acompanhada por uma sucessiva controvérsia, que se prende com os seguintes factos:

- Inexistência de um paradigma de programação bem definido para sistemas distribuídos;
- O termo agente tornou-se vulgarmente utilizado para descrever software em geral, devido às definições vagas e contraditórias de que é objecto;
- O paradigma dos agentes tenta resolver o problema da existência do “mundo fechado” na orientação a objectos.
- O interesse inicialmente meramente científico foi explorado pela comunicação social e remetido para a opinião pública sem que o seu significado tenha sido correctamente definido e explicado.

A designação de agente tem sido continuamente alterada e discutida ao longo dos anos. Uma definição de agente mais precisa define-o como sendo um sistema computacional baseado em software capaz de representar uma peça de hardware (características, limitações, interações, etc...) e que goza das seguintes propriedades:

- **Autonomia -** Os agentes operam sem a intervenção directa de humanos ou outros agentes e possuem algum tipo de controlo sobre as suas acções e estado interno;
- **Reactividade -** Os agentes têm a percepção do seu ambiente e respondem rapidamente às alterações que nele ocorrem;
- **Pró-Actividade -** Os agentes não se limitam a agir em resposta ao seu ambiente, são igualmente capazes de tomar iniciativas e exibir comportamento direccionado por objectivos;
- **Habilidade Social -** Os agentes são capazes de interagir com outros agentes (e possivelmente com humanos) através de uma dada linguagem de comunicação entre agentes.

Um agente não é autónomo na sua criação e no seu arranque inicial, isto é, para ser originado e colocado em funcionamento necessita da intervenção de um humano ou de outro agente. Além disso, a actividade do agente é condicionada por um tempo de vida limitado e um final de operação.

A versatilidade de um agente é um aspecto igualmente importante, tendo em conta que, apesar de este possuir a capacidade de realizar acções sem intervenção directa do humano, também deve interagir com este sempre que necessário (aceitar ordens ou processar instruções vindas de humanos).

O grande interesse passa por definir agentes que saibam conjugar o comportamento reactivo com o comportamento pró-activo.

O primeiro tipo de comportamento aplica-se mais a ambientes dinâmicos, onde podem ocorrer, a cada instante, alterações significativas e às quais o agente se

deve adaptar de forma imediata. Por outro lado, o comportamento pró-activo é adequado para situações onde o ambiente seja estático, ou seja, o agente possui à partida determinados objectivos, que irá cumprir caso o ambiente não seja alterado durante a sua execução. Aliando estas duas características, tem-se um agente capaz de reagir às mudanças de ambiente e raciocinar a cada momento acerca da validade dos seus objectivos iniciais, anulando as suas acções caso as condições ideais não se verifiquem ou adequando o seu comportamento da melhor forma possível, face às alterações.

Uma propriedade igualmente importante e que é essencial num agente é a sua habilidade social. As características anteriores dizem respeito ao objecto agente em si e não focam a comunicação do agente com outras entidades (podem ser agentes ou humanos). De facto, o agente é frequentemente confrontado com situações de cooperação, competição ou negociação, onde são exigidas trocas de mensagens (de alto nível).

Tal como nos seres humanos, estes processos de interacção social devem coexistir no processo de funcionamento de um agente e todas as suas acções, sejam de que tipo forem, devem ser executadas, tendo em conta não só os objectivos pessoais do agente mas também o lado oposto da questão, isto é, a entidade com a qual se pretende interagir e quais os seus propósitos.

O balanceamento da habilidade social com as capacidades pró-activa e reactiva é de extrema importância, principalmente no que diz respeito a processos de cooperação entre um conjunto de agentes que partilham o(s) seu(s) objectivo(s).

Estas propriedades dizem respeito à vertente operacional e funcional do agente. Existem, no entanto, outras características que associam o agente a uma entidade cognitiva e consciente, capaz de exibir sentimentos, percepções e até emoções, assemelhando-os da melhor forma possível com os humanos.

Assim, destacam-se nos agentes as seguintes características:

- **Mobilidade -** Capacidade de um agente se movimentar de um local para outro. Usualmente esta capacidade é mencionada no contexto de agentes de

software e, como tal, a movimentação verifica-se no interior de uma rede de computadores;

- **Verdade -** Um agente deve sempre ser verdadeiro e não comunicar informação falsa propositadamente;
- **Benevolência -** Os agentes não devem assumir um comportamento contra-produtivo e devem sempre tentar fazer aquilo que lhes é solicitado;
- **Conhecimento e Crença -** Possuir conhecimento consiste em, para além de possuir uma colecção de informação dinâmica, possuir também capacidade de raciocínio sobre essa informação. É necessário definir qual a melhor estratégia de raciocínio a aplicar numa determinada situação e o porquê (meta-conhecimento). Uma crença representa a noção actual que o agente possui sobre determinado facto. As crenças são geralmente dinâmicas, isto é, podem alterar o seu valor de verdade com o tempo.
- **Intenções e Obrigações -** Intenções são objectivos de longo prazo do agente. Resultam em padrões de comportamento que levam à execução de um determinado conjunto de acções individuais. As obrigações estão relacionadas com compromissos que o agente assumiu anteriormente. A partir do momento em que este expressa a sua disponibilidade para executar determinada tarefa, passa a ser responsável por realizar as acções necessárias para essa execução.
- **Racionalidade -** Um agente agirá de forma a atingir os seus objectivos e não de forma a impedir que esses mesmos objectivos sejam atingidos. A cada instante, em função do seu conhecimento e de acordo com as suas capacidades, tentará executar a melhor acção para cumprir esses mesmos propósitos;
- **Inteligência -** O estado de um agente é formalizado por conhecimento (i.e. crenças, objectivos, planos e assunções) e este interage com outros agentes utilizando uma linguagem simbólica. Possui capacidade de raciocínio abstracto, suporta a resolução de novos problemas e adaptação a novas situações;

- **Continuidade Temporal -** O agente é um processo que é executado continuamente ao longo do tempo. Usualmente são utilizados também os termos persistente ou "com uma vida longa" para designar a continuidade temporal dos agentes.
- **Carácter -** O agente possui uma personalidade credível e eventualmente possui também um " estado emocional";
- **Aprendizagem -** Faz com que o agente adquira novo conhecimento e altere o seu comportamento baseado na sua experiência prévia.

Muitas destas propriedades não são absolutamente necessárias num agente e, algumas delas são até evitáveis. A aprendizagem, por exemplo, é o aperfeiçoamento baseado em experiências. No caso em que seja fatal uma experiência falhada esta propriedade não pode coexistir no agente. Um exemplo disso é um agente concebido para efectuar a aterragem de um avião. Nessa situação, pretende-se que o agente seja capaz de realizar a sua tarefa de forma autónoma e inteligente, mas pode ser argumentado que não será desejável que ele possua aprendizagem, pois interessa única e exclusivamente que a acção seja realizada com êxito, sem tentativas falhadas.

3.1.2 Classificação dos Agentes

De modo a caracterizar melhor a área científica dos agentes, para além de analisar os domínios que inspiraram a área, é útil dividir os agentes em classes, analisando os diferentes tipos.

Esta implementação está relacionada com o facto das características de um agente serem idealmente dependentes do tipo de aplicação que se pretende. Basicamente, os agentes estão divididos em sete tipologias:

- **Mobilidade -** Agentes estáticos ou móveis. Os agentes móveis podem estar residentes na sua máquina de origem ou temporariamente numa outra máquina;

- **Modelo de Raciocínio -** Presença ou não de um modelo de raciocínio simbólico, ou seja, um agente pode ser deliberativo ou puramente reactivo;
- **Função do Agente -** A função principal assumida pelo agente;
- **Autonomia -** Grau de autonomia do agente;
- **Cooperação -** Realização ou não de acções cooperativas com outros agentes;
- **Aprendizagem -** Inclusão ou não de capacidades de aprendizagem no agente;
- **Características Híbridas -** Estas combinam duas ou mais filosofias diferentes num mesmo agente.

Existem quatro tipos de agentes principais: colaborativos, colaborativos com capacidade de aprendizagem, agentes de interface e agentes verdadeiramente inteligentes. Esta separação teve em conta as características de autonomia, cooperação e aprendizagem do agente.

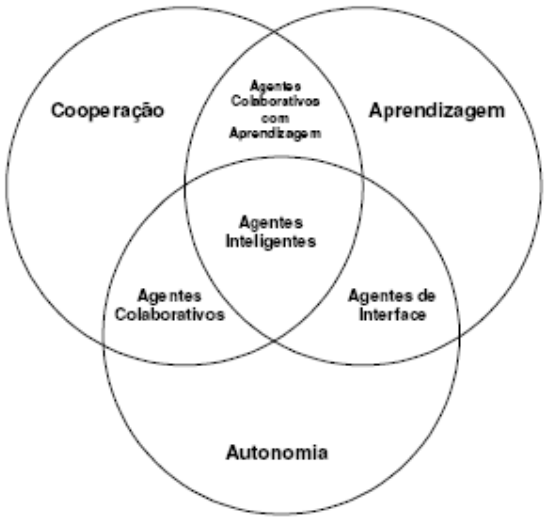

Figura 3-1: Categorias de um agente. [adaptada de (Nwana 1996)]

Analisando a Figura 3-1, verifica-se que podem existir diversas categorias para um agente. Os agentes que aliam autonomia com cooperação são agentes colaborativos; os que aliam cooperação com aprendizagem dizem-se agentes colaborativos com aprendizagem. Por outro lado, se um agente for autónomo e

possuir aprendizagem, é um agente de interface. Finalmente, o agente mais "completo" é aquele que alia as propriedades da cooperação, aprendizagem e autonomia, ou seja, é um agente verdadeiramente inteligente.

Outras considerações foram feitas acerca de agentes, onde estes eram vistos como um processo em contínua execução e, como tal, deveriam estar divididos em 3 grupos fundamentais: agentes biológicos; agentes robóticos e agentes computacionais, como se pode observar na Figura 3-2.

Figura 3-2: Categorias e subcategorias de um agente. [retirada de (Franklin and Graesser 1996)]

3.1.3 Arquitecturas de Agentes

A arquitectura de agentes, como o próprio nome indica, refere-se à arquitectura, não só do próprio agente, mas também do sistema multi-agente, isto é, o modo como estão organizados os agentes dentro de um sistema e a forma como estão estruturados os seus relacionamentos e interacções. De um modo semelhante ao que acontece com as diversas arquitecturas de software, as arquitecturas dos agentes possuem determinadas características que permitem a avaliação da sua qualidade e eficácia. No que respeita aos agentes robóticos, a sua arquitectura alia a arquitectura de software com a arquitectura de hardware (módulos físicos e a sua interligação).

A questão que se levanta quando se pretende discutir sobre a qualidade de uma arquitectura de agentes é bastante relativa, tendo em conta que esta depende

inteiramente do tipo de aplicação que se pretende desenvolver para os agentes, variando de domínio para domínio. No entanto, as seguintes características são válidas e aconselháveis para a implementação de uma arquitectura de agentes:

- **Simplicidade -** Idealizar a arquitectura e seus módulos para que sejam fáceis de entender, implementar e manter.
- **Funcionalidade -** Seleccionar uma arquitectura e ferramentas de desenvolvimento que focalizem os aspectos específicos do problema a ser abordado.
- **Expansibilidade -** A arquitectura deve poder ser facilmente ampliada, uma vez que nem todas as necessidades futuras podem ser previstas inicialmente.
- **Isolamento/ Portabilidade -** Uma arquitectura, para poder ser expandida, deve possuir uma implementação portável, evitando-se soluções não padronizadas.

Seguidamente, apresentam-se alguns agentes caracterizados por uma arquitectura simples e nos quais está presente a noção de raciocínio:

- **Agentes Reflexos Simples -** Utilizam um conjunto de regras condição-acção pré-estabelecidas. Perante a percepção de uma dada situação, seleccionam e executam uma dada acção pré-especificada. Os agentes deste tipo podem ser considerados os mais simples pois o seu comportamento é puramente reactivo;
- **Agentes com Representação do Estado do Mundo -** Estes agentes possuem uma representação do estado do mundo que é actualizada dinamicamente com a percepção do agente. Desta forma, a reactividade destes agentes é condicionada pelas experiências anteriores, que se encontram reflectidas nesse estado do mundo. Este modo de funcionamento implica que a mesma percepção ocorrida em momentos distintos e, consequentemente, estados diferentes do mundo, tenha como resultado acções diferentes;

- **Agentes Baseados em Objectivos** - Os agentes baseados em objectivos, para além da descrição do estado corrente do mundo, utilizam a informação sobre os objectivos que pretendem atingir. Neste caso, a decisão pode implicar a pesquisa e planeamento prévio, antes da selecção da acção a executar a cada instante. Este tipo de agentes é mais flexível pois diferentes comportamentos podem ser obtidos para o mesmo estado do mundo, dependendo dos objectivos do agente;
- **Agentes Baseados em Utilidade -** A utilidade correspondem a uma medida de satisfação do agente relativa ao cumprimento dos seus objectivos. A utilidade será mais elevada se o estado do mundo actual do agente estiver próximo de atingir os seus objectivos. As utilidades podem ser usadas para decidir entre objectivos em conflito ou ainda, quando existe incerteza nas acções, para medir a proximidade no alcance dos objectivos. Os agentes baseados em utilidade conseguem ser mais racionais pois possuem capacidades para avaliar a utilidade da execução de uma determinada acção.

Além destas arquitecturas, existem outras mais complexas e avançadas, baseadas sobretudo em agentes com capacidades de aprendizagem e de planeamento, e em agentes de decisão.

Ainda no que diz respeito às arquitecturas, existem ainda algumas, baseadas em sistemas computacionais, que os dividem em três subcategorias:

- **Arquitectura Deliberativa -** Segue a abordagem clássica da inteligência artificial, onde os agentes actuam com pouca autonomia e possuem modelos simbólicos explícitos dos seus ambientes;
- **Arquitectura Reactiva -** Procura não utilizar algum tipo de modelo ou raciocínio simbólico complexo e tomar decisões “em tempo real” ;
- **Arquitectura Híbrida -** Combina as características das duas abordagens anteriores, nomeadamente as capacidades deliberativas e reactiva.

Porém, numa tentativa de proporcionar às estruturas implementadas o conceito de hierarquia, surgiu a arquitectura por camadas, onde os vários subsistemas de agentes interagem por níveis.

A arquitectura BDI (*"Belief Desire Intention"*) é uma arquitectura deliberativa em que o estado interno de processamento de um agente é descrito através de um conjunto de estados "mentais", usualmente crenças, desejos e intenções.

- **Arquitectura Deliberativa**

Neste tipo de arquitectura, os agentes actuam com pouca autonomia e são interpretados como sistemas baseados em conhecimento, onde as suas acções/decisões são executadas tendo por base um raciocínio lógico. O agente possui uma representação interna do mundo e um estado mental explícito que pode ser modificado por alguma forma de raciocínio simbólico.

A Figura 3-3 representa a forma como se traduz o mundo real através de uma descrição simbólica, utilizando a percepção para manter essa estrutura actualizada e o raciocínio (sobre essa mesma informação simbólica) para que seja possível executar as acções a cada instante.

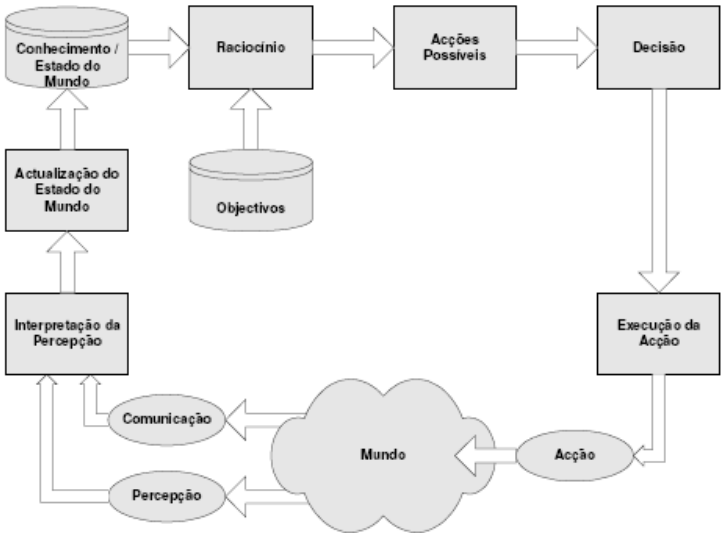

Figura 3-3: Esquema genérico de uma arquitectura deliberativa.

- **Arquitectura Reactiva**

Este tipo de arquitectura Figura 3-4 suporta a teoria de que o agente adquire inteligência através das interacções com o ambiente que o rodeia, não necessitando portanto de um modelo pré-estabelecido. As decisões são tomadas em tempo-real, com base num conjunto de informação muito limitado e regras simples de situação/acção que permitem seleccionar um certo tipo de comportamento.

Este tipo de arquitectura apresenta algumas vantagens em relação às restantes, destacando-se entre elas a simplicidade, a economia ou a boa robustez contra falhas. Em contrapartida, apresenta algumas desvantagens que tornam, por vezes, o seu uso inadequado, tais como o facto de os agentes decidirem com base na sua percepção actual, possuírem uma hierarquia pré-definida e serem incapazes de realizar acções que impliquem a execução de planos a longo prazo.

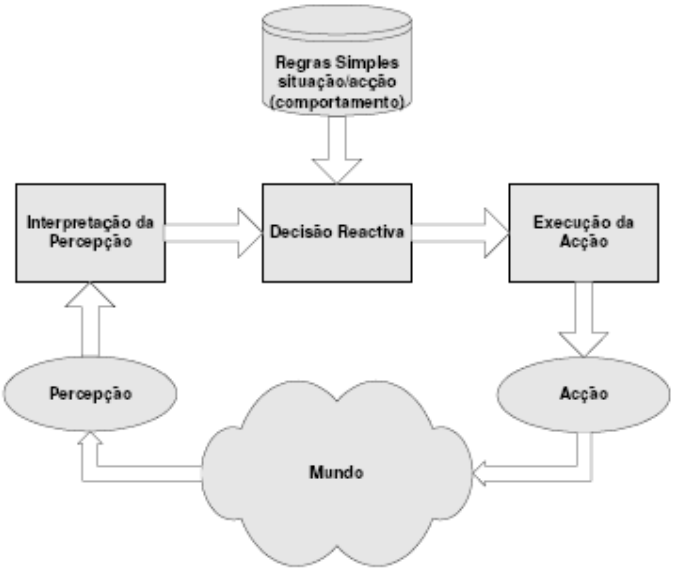

Figura 3-4: Esquema genérico de uma arquitectura reactiva.

- **Arquitectura Híbrida**

Esta arquitectura, ilustrada na Figura 3-5 surge como alternativa às limitações existentes nas arquitecturas reactiva e deliberativa. Efectivamente, os agentes puramente reactivos não são capazes de implementar um comportamento orientado a objectivos. Por outro lado, os agentes deliberativos tornam-se muitas vezes incapazes de responder rapidamente aos estímulos do exterior, isto é, têm um tempo de reacção lento. Um agente híbrido combina as duas componentes e caracteriza-se por uma arquitectura composta por níveis ou camadas, dispostas hierarquicamente e onde normalmente a camada reactiva tem prioridade sobre a camada deliberativa, de modo a permitir uma resposta rápida aos eventos mais importantes registados no ambiente.

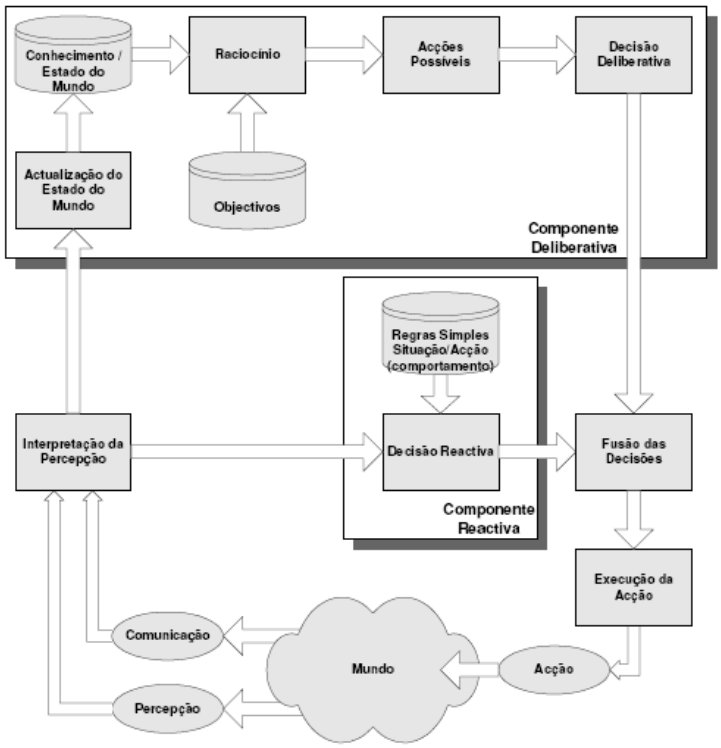

Figura 3-5: Esquema genérico de uma arquitectura híbrida.

Após a interpretação da percepção e das comunicações provenientes do ambiente, o agente possui duas componentes: uma componente reactiva e uma componente deliberativa. O funcionamento da componente reactiva é extremamente simples e baseia-se num conjunto de regras situação/acção que associam directamente certas decisões a determinados estímulos. A componente deliberativa implica a construção de um estado simbólico do mundo e a utilização de raciocínio simbólico de forma a decidir a cada instante as acções a executar, de forma a atingir os objectivos. Uma parte complexa e sujeita a um elevado número de trabalhos de investigação é a forma de efectuar a fusão das decisões deliberativa e reactiva como forma de seleccionar a acção final a executar. Note-se que existe uma constante interacção entre as duas componentes, sendo a componente deliberativa capaz de alterar as regras de situação/acção da componente reactiva e esta, em situações de emergência, tem a capacidade de se sobrepor à componente deliberativa.

- **Arquitectura por camadas**

Este tipo de arquitectura é usualmente recorrente, nomeadamente pelos sistemas híbridos. Existem dois tipos de camadas:

- Camadas horizontais;
- Camadas verticais;

No primeiro caso, cada camada actua como um agente (camadas de software ligadas directamente aos sensores de *"input"* e às acções de *"output"*).

Nas camadas verticais, os sensores de input e as acções de output têm, pelo menos, uma camada a separá-los.

Deste modo, a camada horizontal é vantajosa relativamente à vertical por ser conceptualmente mais simples. Assim, o número de camadas e o número de comportamentos que o agente suporta possuem uma relação de n para n. No entanto, na arquitectura horizontal existe obrigatoriamente uma camada extra que funciona como mediador e que tenta resolver o problema proveniente do facto de

existir competição entre camadas. A introdução desta camada extra, apesar de resolver um problema importante, pode provocar um estrangulamento no processo de decisão do agente.

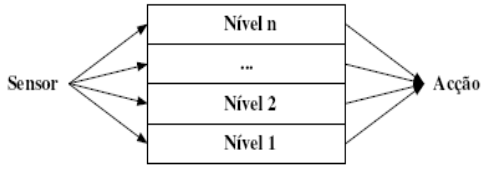

Figura 3-6: Arquitectura de camadas horizontais.

Na arquitectura vertical este problema não ocorre, pois o fluxo de controlo passa sequencialmente por cada camada até à última, altura em que se executa a acção. No entanto, este método é pouco flexível pois, para o agente tomar uma decisão, é necessário que o fluxo de controlo atravesse todas as camadas. No caso de existirem falhas numa camada a acção actual do agente poderá ser anulada.

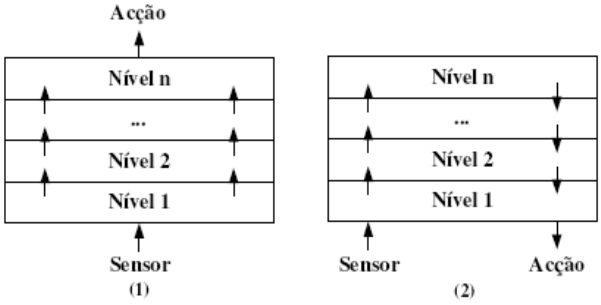

Figura 3-7: Arquitectura de camadas verticais.

Analisando a Figura 3-6, verifica-se uma única passagem de controlo. Contrariamente, na Figura 3-7 observam-se duas passagens de controlo por cada camada. Nesta última arquitectura, o fluxo de informação sobe as várias camadas até atingir o nível superior e, seguidamente, desce atravessando novamente todas as camadas, dando origem à acção a executar.

3.2 Plataforma Jade

3.2.1 Caracterização do JADE

O JADE consiste em um ambiente para desenvolvimento de aplicações baseadas em agentes de acordo com as especificações da FIPA (*"Foundation for Intelligent Physical Agents"*) para interoperabilidade entre sistemas multi-agentes totalmente implementado em Java.

O principal objectivo do JADE é simplificar e facilitar o desenvolvimento de sistemas multiagentes garantindo um padrão de interoperabilidade entre os mesmos através de um abrangente conjunto de serviços, os quais tanto facilitam como possibilitam a comunicação entre agentes, de acordo com as especificações da FIPA: serviço de nomes ("*naming service"*), páginas amarelas ("*yellow-page service")*, transporte de mensagens, serviços de codificação e descodificação de mensagens e uma biblioteca de protocolos de interacção pronta para ser usada. Toda a comunicação entre agentes é feita por troca de mensagens. Além disso, lida com todos os aspectos que não fazem parte do agente em si e que são independentes das aplicações tais como transporte de mensagens, codificação e interpretação de mensagens e ciclo de vida dos agentes. O Jade pode ser considerado como um "*middle-ware*" de agentes que implementa um "*framework"* de desenvolvimento e uma plataforma de agentes. O *Jade* foi desenvolvido na linguagem *Java* devido a características particulares da mesma, particularmente pela programação orientada a objectos em ambientes distribuídos heterogéneos.

Na Figura 3-8 ilustra-se a arquitectura de referência do JADE.

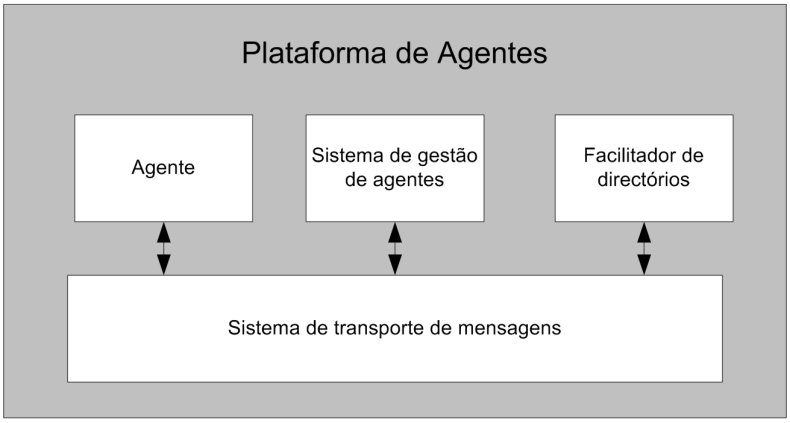

Figura 3-8: Arquitectura de referência da plataforma de agentes FIPA. [adaptada de (Bellifemine, Caire et al. 18-June-2007)]

3.2.2 Características do JADE

De seguida descrevem-se as principais características do JADE:

- Plataforma distribuída de agentes – JADE pode ser dividida em vários "*hosts*" ou máquinas (desde que eles possam ser ligados via RMI – *"Remote Method Invocation"*). Apenas uma aplicação Java e uma JVM (*"Java Virtual Machine"*) é executada em cada computador. Os agentes são implementados como "*threads"* Java e inseridos dentro de repositórios de agentes chamados "*Agent Containers"* que fornecem suporte para a execução do agente.
- GUI (*"Graphical User Interface"*) – Interface visual responsável pela gestão de vários agentes e *"containers"* de agentes.
- Ferramentas de "*Debugging"* – Ferramentas que ajudam o desenvolvimento de aplicações multiagentes baseados em JADE.
- Suporte a execução de múltiplas, paralelas e concorrentes actividades de agentes – através dos modelos de comportamentos ("*Behaviours"*).
- Ambiente de agentes complacente a FIPA – No qual se incluem o sistema gestor de agentes (AMS – *"Agent Management System"*), o DF (*"Directory Facilitator"*) e o canal de comunicação entre agentes (ACC – *"Agent Communication Channel"*) Todos esses três componentes são automaticamente carregados quando o ambiente é iniciado.

- Transporte de mensagens – Transporte de mensagens no formato FIPA-ACL (*"Agent Communication Language")* (FIPA 2002d) dentro da mesma plataforma de agentes.
- Biblioteca de protocolos FIPA – Para interacção entre agentes JADE, na Figura 3-9 são apresentados dois dos protocolos disponibilizados.
- Automação de registos – Registo e cancelamento automático de agentes com o AMS fazendo com que o desenvolvedor se abstraia disso.
- Serviços de nomes (NS – *"Naming Service"*) em conformidade aos padrões FIPA: Durante a inicialização dos agentes, estes obtêm seus GUID (*"Globally Unique Identifier"*) da plataforma que são identificadores únicos em todo o ambiente.
- Integração – Mecanismo que permite aplicações externas carregarem agentes autónomos JADE.

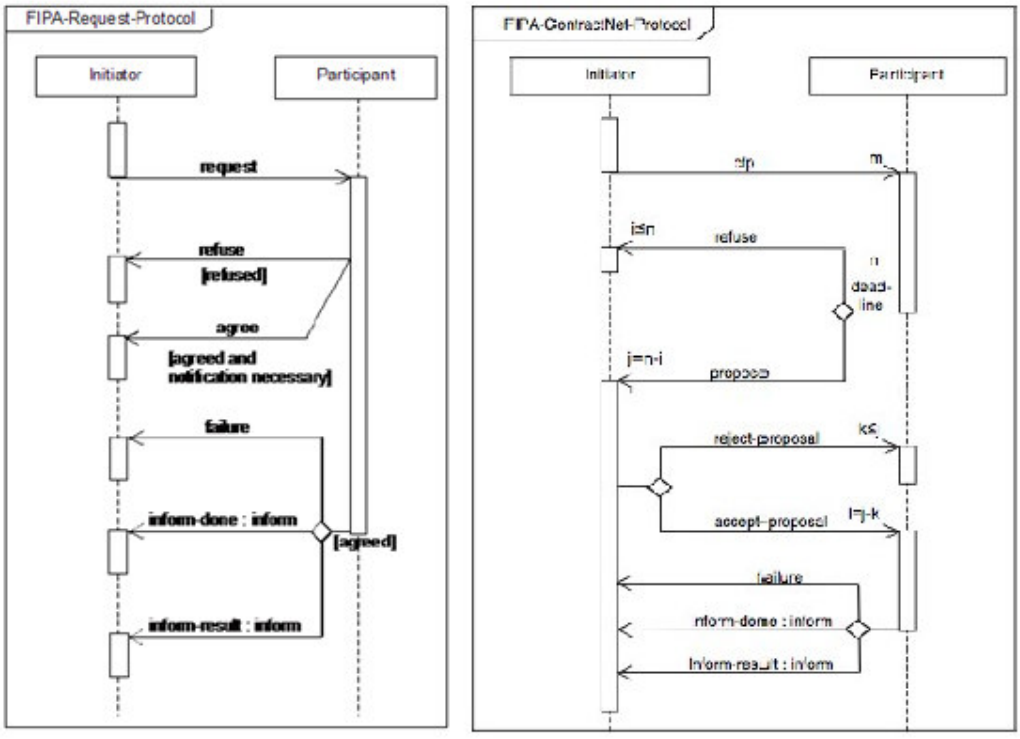

Figura 3-9: Protocolos FIPA Request e Contract Net. (FIPA 2002a; FIPA 2002b; FIPA 2002c)

3.2.3 Arquitectura interna do JADE

A arquitectura da plataforma JADE baseia-se na coexistência de várias máquinas virtual Java (JVM) podendo ser distribuída por diversas máquinas independente de sistema operacional que cada uma utiliza. Na Figura 3-10 existe uma visão distribuída da plataforma de agentes JADE dividida em três computadores. Em cada computador existe uma JVM executando um JRE (*"Java Run-time Environment"*) *e* possuindo cada uma delas um *"container"* de agentes que fornece um ambiente completo para execução destes, além de permitir que vários elementos possam correr concorrentemente no mesmo processador (computador*)*. Ou seja, durante a execução deve existir uma JVM por processador, sendo possível coexistir vários agentes por JVM. A comunicação entre JVMs é realizada através da invocação remota de métodos (*RMI)* do Java.

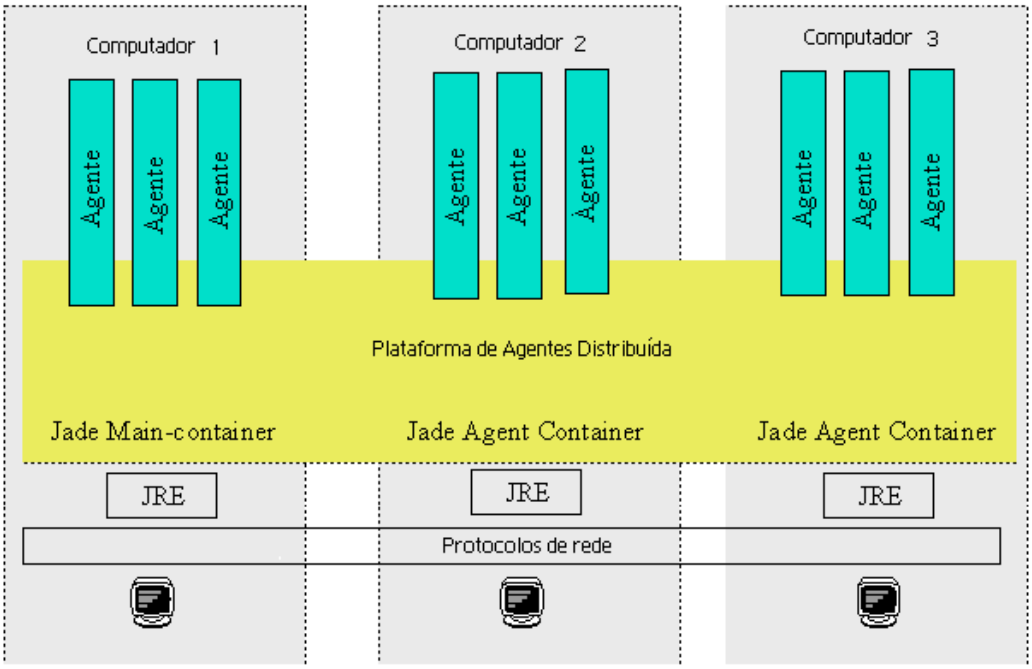

Figura 3-10: Plataforma de agentes JADE distribuída por vários computadores. [adaptada de (Bellifemine, Caire et al. 18-June-2007)]

O *"main-container",* localizado no computador 1 da figura ilustrada anteriormente, designa-se por *"container"* onde se encontra o AMS, o DF e registo RMI, na Figura 3-11 pode-se observar uma ilustração real do *"main-container"*. O registo RMI é um servidor de nomes que o Java usa para registar e recuperar referências a objectos através dos seus nomes, ou seja, é o meio que JADE usa em Java para manter as referências aos outros *"containers"* de agentes que se ligam à plataforma. Os outros *"containers"* de agentes ligam-se ao *"main-container"* fazendo com que quem devolve fique abstraído da separação física dos computadores, caso exista, ou das diferenças de plataformas que cada computador possa ter. Essa abstracção é ilustrada na figura anterior na zona central na qual a plataforma JADE integra todos os três computadores actuando como um elo de ligação e fornecendo um completo ambiente de execução para qualquer conjunto de agentes JADE.

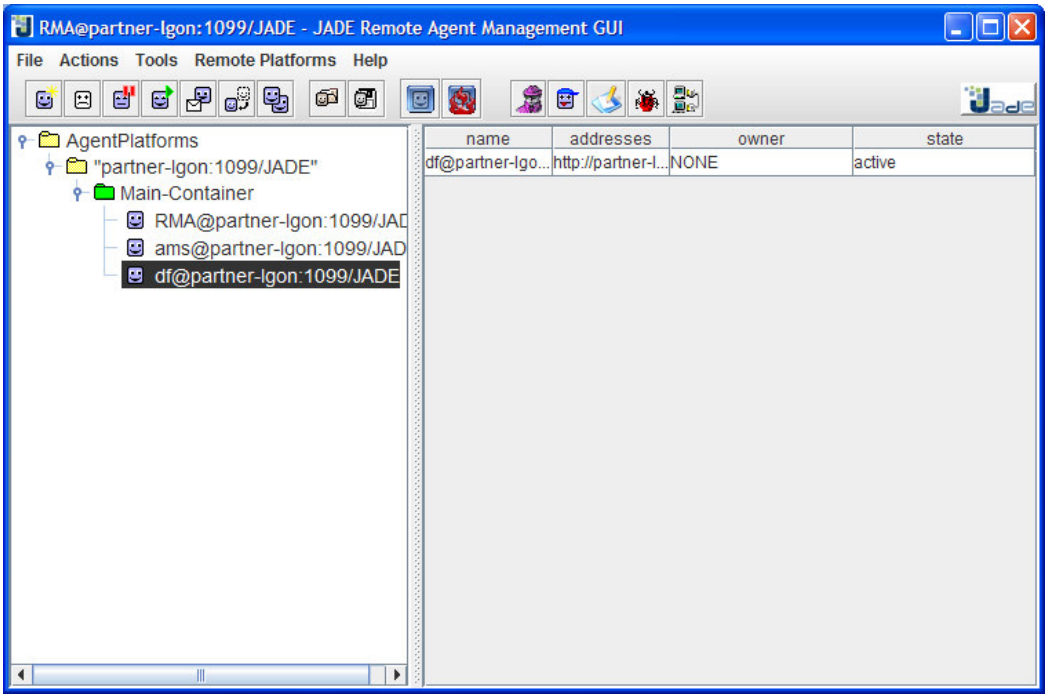

Figura 3-11: Interface gráfica do Gestor de Agentes do JADE

3.3 XML como base de dados

3.3.1 Definições

Uma base de dados é composta por uma colecção de registos estruturados, guardados num computador, disponíveis para pessoas ou sistemas. A estrutura dos dados arquivados é feita de acordo com o modelo de base de dados utilizado. O modelo mais comum é o modelo relacional. Outros modelos, tais como hierárquico ou de rede fazem uma representação das relações mais explicitamente.

O XML descrito em (Bray, Paoli et al. 2008) é um subconjunto do SGML (*"Standard Generalized Markup Language"*) tendo como objectivo tornar o SGML genérico, para que possa ser distribuído, recebido e processado na internet, da mesma forma que é possível com o HTML (*"Hypertext Markup Language"*) (Raggett, Hors et al. 24 December 1999).

O XML foi desenhado tendo em conta os seguintes objectivos:

- O XML deve ser simples de implementar e utilizável através da internet;
- Deverá suportar uma extensa variedade de aplicações;
- Deverá ser compatível com SGML e HTML;
- Deverá ser simples escrever programas que processem documentos XML;
- Os documentos XML deveram ser legíveis para um humano e interpretados pelo computador;
- O número de recursos opcionais em XML deve ser mantido a um mínimo absoluto, idealmente zero;
- Um projecto XML deverá ser preparado de uma forma rápida;
- O projecto XML deverá ser formal e conciso;
- Os documentos XML deveram ser simples de criar;

Os documentos em XML são compostos por um conjunto de identificadores designados por *"tags"*, que suportam os tipos mostrados na Figura 3-12, já o nome atribuído aos identificadores ou aos atributos não têm qualquer tipo de restrição, ficando ao critério do utilizador, a única restrição existente é a delimitação de uma entidade entre identificadores com o mesmo nome.

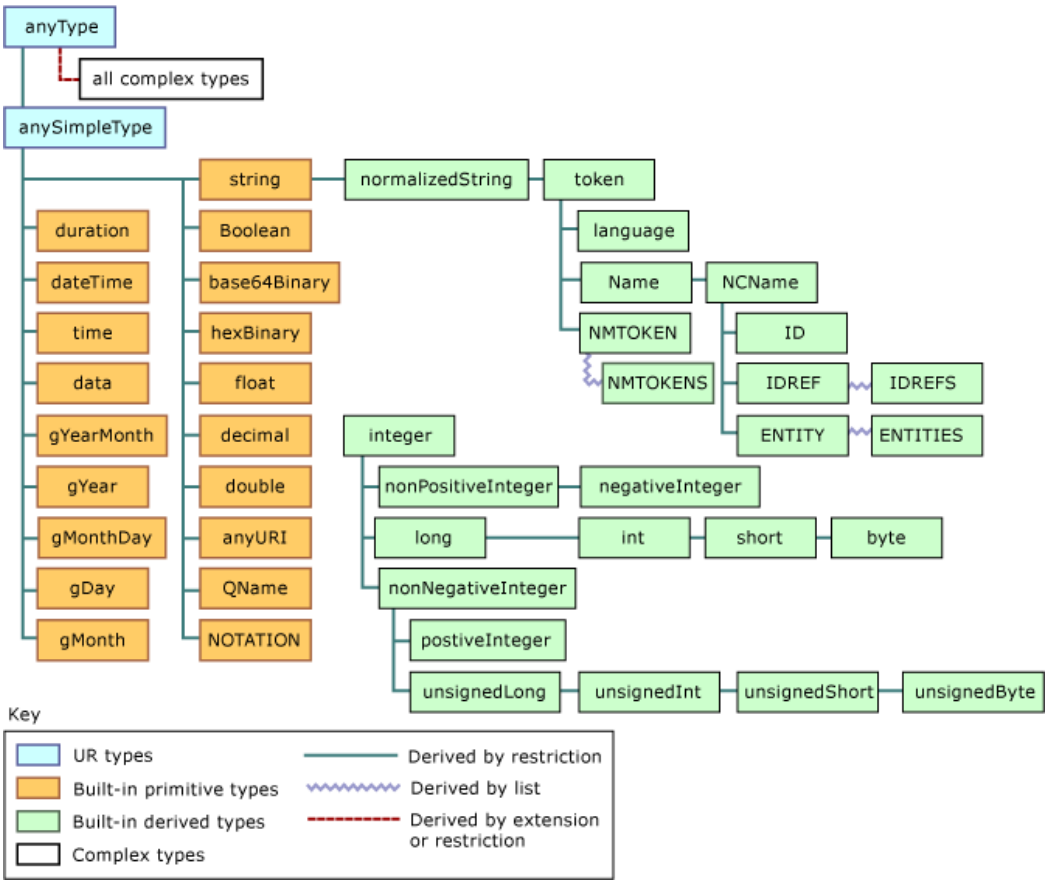

Figura 3-12: Tipos de dados suportados no XML. (Biron and Malhotra 2004)

3.3.2 Semelhanças e diferenças entre XML e base de dados

De uma forma sucinta serão descritas algumas semelhanças e diferenças entre bases de dados e ficheiros XML.

O armazenamento de dados em ambos é feito de uma forma estruturada, nas bases de dados cada registo é inserido numa linha de uma tabela e cujas colunas dessa tabela identificam cada um dos valores do registo, em XML cada registo é um conjunto de identificadores os quais, à semelhança das colunas nas tabelas, contêm os valores.

Ambos possuem linguagens para a validação dos dados (*"Schemas"*), no XML para além deste método existem ainda os DTD (*"Document Type Definition"*).

Para ambos existem interfaces de programação, para as bases de dados o JDBC (*"Java Database Connectivity"*), e para o XML o SAX (*"Simple API for XML"*), DOM (*"Document Object Model"*), entre outros.

Em relação as diferenças temos, os mecanismos de armazenamento e eventos, em que nas bases de dados existem os *"storage procedures"* e os *"triggers"*, os quais permitem de uma forma rápida inserir um novo registo ou responder a um determinado evento, em XML não se consegue ter estes mecanismos, por exemplo para a inserção de um novo registo é necessário ter o documento em memória e conhecer a estrutura do mesmo, para que o registo seja inserido de forma correcta.

Nas bases de dados consegue-se definir níveis de segurança, isto é, restringir ou dar acesso aos dados que podem ser vistos ou manipulados por um ou vários utilizadores.

O acesso concorrencial aos dados apenas é suportado nas bases de dados, bem como a selecção de um registo recorrendo a várias variáveis.

3.3.3 Vantagens e Aplicações do XML

As vantagens e aplicações dos ficheiros XML são várias, sendo as mais relevantes enumeradas de seguida:

- Consultas avançadas a bases de dados;
- Desenvolvimento de aplicações Web flexíveis;

- Suporta a integração de dados de diferentes fontes;
- Comporta dados oriundos de múltiplas aplicações;
- Escalabilidade, isto é, transferência dos dados para o lado do cliente, reduzindo o tráfego no servidor e o tempo de resposta do browser;
- Recorre à compressão dos dados, que pode ser tanto maior ou menor consoante a quantidade de dados a ser transferida entre cliente e servidor;
- Facilidade de programação nas linguagens baseadas neste formato, pois a programação dos documentos pode ser facilmente substituída por um editor;
- É extensível, ou seja, possui a capacidade de criar novos componentes essenciais ao desenvolvimento da interface;
- Rapidez no tempo de resposta;
- É portátil, ou seja, um documento pode ser executado em diferentes sistemas operativos;
- Suporta a integração com outras tecnologias;
- Possui a capacidade de aceder a dados remotos.

3.3.4 Desvantagens do XML

Por outro lado, a referida linguagem também apresenta um vasto conjunto de desvantagens, de entre as quais de destacam as seguintes:

- Quando o programa é extenso em termos de código, a sua interpretação, por parte do usuário, torna-se complicada;
- Ocupa um espaço em memória considerável;
- Por vezes, apresenta um elevado tempo de processamento;
- Apresenta uma complexidade mais elevada que as bases de dados;

Capítulo 4. **Arquitectura**

No decorrer deste capítulo pretende-se mostrar a arquitectura proposta como solução aos problemas mencionados anteriormente no mundo da manufactura.

A arquitectura baseia-se principalmente na arquitectura do CoBASA (Barata 2003), uma vez que todos os módulos presentes no sistema são representados por agentes com comportamentos de colaboração, capazes de oferecer funcionalidades ao sistema. Funcionalidades essas que neste contexto representam acções que cada um dos móludos é capaz de executar.

Difere no entanto em alguns dos módulos definidos, bem como na sua inicialização e modo de funcionamento.

A principal diferença é a substituição do Broker Agent (BA), cuja definição se pode encontrar em (Barata 2003), por um automatismo que lança dinamicamente um DCA (*"Dynamic Coalition Agent"*) e que é capaz, também de forma automática, encontrar um conjunto de agentes que satisfaçam um determinado requisito do sistema.

A introdução das tabelas de encaminhamento para o sistema de transportes foi também uma melhoria relevante em relação ao sistema que foi tomado por base, uma vez que se ganhou liberdade na adição, remoção e alteração da disposição dos módulos.

Deste modo a arquitectura é a apresentada na Figura 4-1, e que irá ser detalhada de seguida.

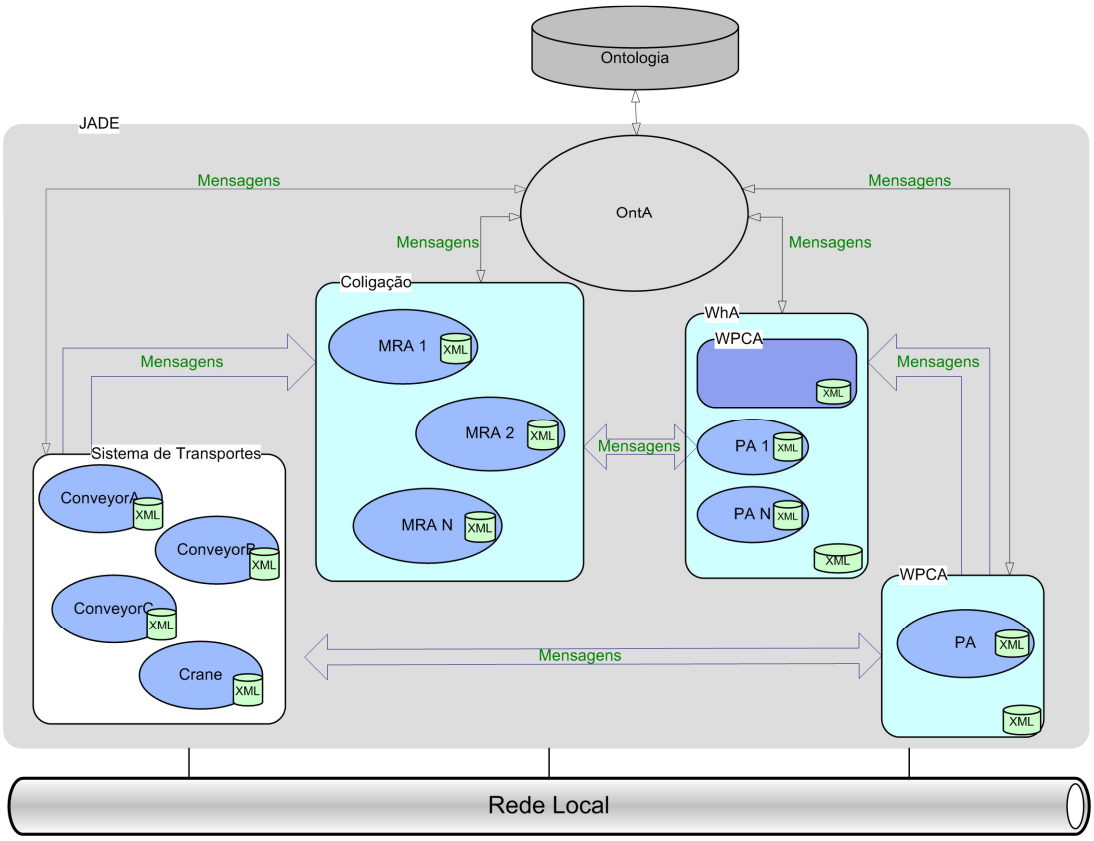

Figura 4-1: Arquitectura proposta.

Os módulos apresentados com cores são os propostos para implementação nesta tese, os restantes módulos foram utilizados sem que se tivesse de efectuar qualquer tipo de alteração. Note-se que o sistema de transportes está representado por uma cor diferente da coligação ou do armazém, isto deve-se ao facto do sistema de transportes não ser um agente propriamente dito, é apenas um encapsulamento de um grupo de agentes do mesmo tipo, por outro lado as coligações são representadas por agentes e são constituidas por outros agentes que representam os módulos fisicos, já o armazém é um agente que representa um módulo fisico do sistema, não sendo no entanto composto por outros agentes, a representação desse agente na figura anterior pretende evidenciar apenas o armazenamento de outros módulos existentes no sistema.

Toda a arquitectura tal como dito anteriormente é suportada pela plataforma distribuída JADE, a qual por sua vez suporta as interações entre os agentes. As

interacções entre os agenes são efectuadas através da troca de mensagens suportada por uma rede local.

4.1 Auto-organização

O conceito de auto-organização será conseguido através da formação dinâmica de coligações. O processo contendo as interacções necessárias para a formação de uma coligação segue uma abordagem *"bottom-up"* o qual é apresentado de uma forma abstracta na Figura 4-2, e detalhado de seguida.

O agente produto irá efectuar a pesquisa habitual à ontologia, procurando quem consegue realizar uma determinada acção, no entanto o agente que a oferece tem um ou vários requisitos para que consiga realizar a acção pretendida, este requisito irá despoletar o lançamento de uma coligação, a qual será responsável por encontrar agentes que satisfação os requisitos de todos os agentes que irão sendo adicionados. No final o agente coligação será responsável por orquestrar o funcionamento de todos os agentes e pela ordem correcta.

O objectivo para a auto-organização do sistema através de coligações dinâmicas prende-se com o facto de se efectuar as melhores escolhas e de uma forma mais rápida sobre os agentes a serem incluídos na coligação, uma vez que essa escolha irá ser efectuada através de software e suportada por ficheiros de configuração, os quais se encontram descritos mais à frente.

O risco de incompatibilidades entre os diversos agentes é menor, uma vez que podem ser comparadas mais características e mais rapidamente, tais como as dimensões, as interfaces eléctricas, magnéticas ou mecânicas, peso, entre outras.

O processo de reengenharia, quando existe uma modificação no *"layout"* de produção ou quando o próprio produto a fabricar é alterado, é mais rápido, visto o sistema suportar uma aproximação de *"Plug and Play"*.

É feita uma melhor reutilização dos recursos existentes no sistema, uma vez que estes podem participar em diversas coligações e cujo seu funcionamento é reservado e coordenado por cada agente coligação.

Consegue-se ainda gerir dinamicamente as limitações e graus de liberdade do sistema, através da união ou intersecção das características e limitações de cada agente.

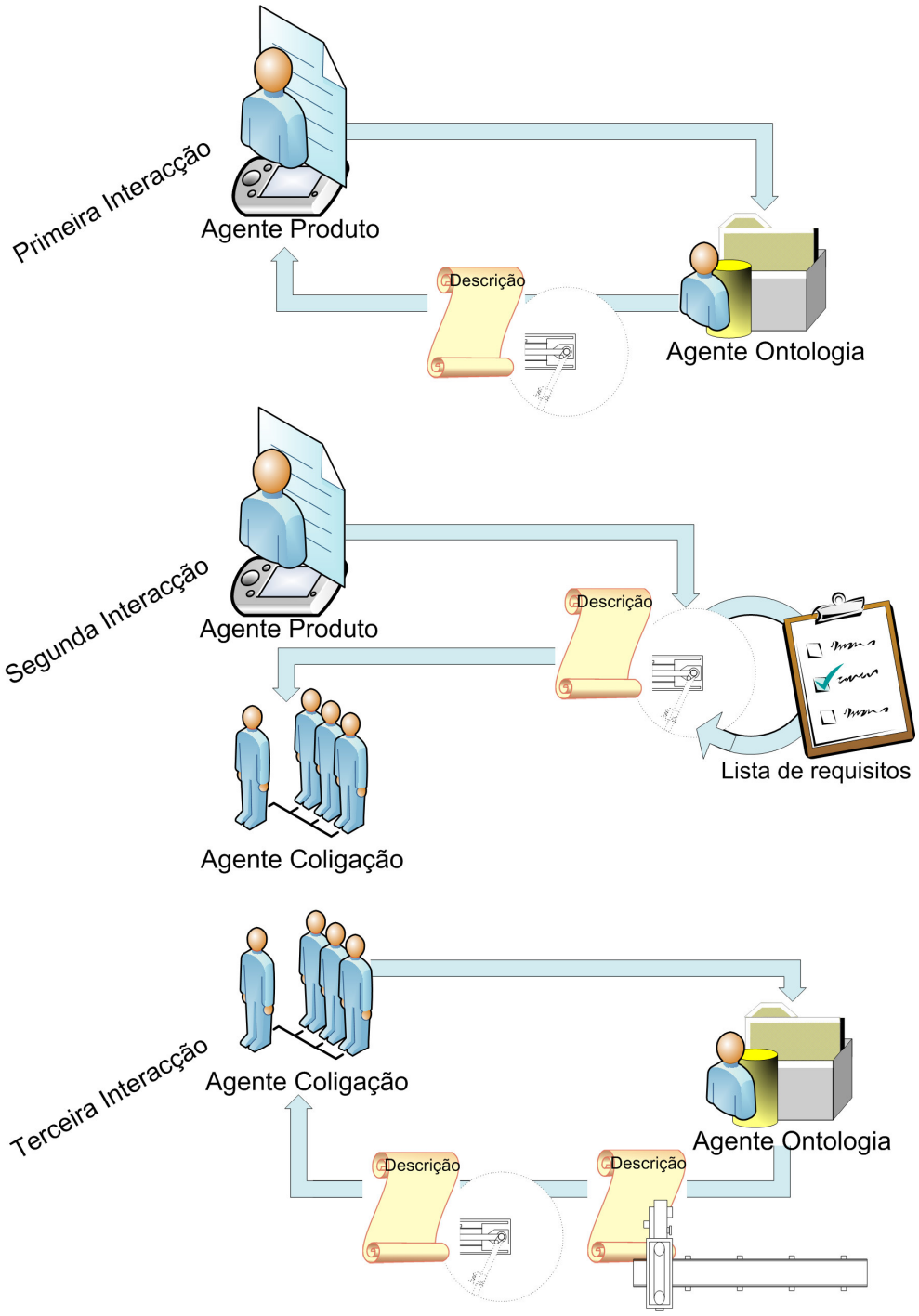

Figura 4-2: Passos na formação dinâmica de uma coligação.

No capítulo de implementação e resultados será apresentado de uma forma mais detalhada a formação das coligações bem como a evolução da implementação destas ao longo da tese apresentada.

4.2 Descrição dos Módulos

De seguida é feita a apresentação dos módulos presentes na arquitectura, sendo mencionado as características mais relevantes, é ainda mostrada uma representação gráfica das interacções que iniciam, com os restantes agentes do sistema.

- **MRA** (*"Manufacturing Resource Agent"*) – É o agente que representa um determinado módulo de manufactura (robô, *"gripper"*, eixo, etc.), oferecendo as funcionalidades desse mesmo módulo ao sistema. Tem ainda a capacidade de participar em coligações.

Na Figura 4-3 é feita uma representação abstracta do processo de lançamento de uma coligação.

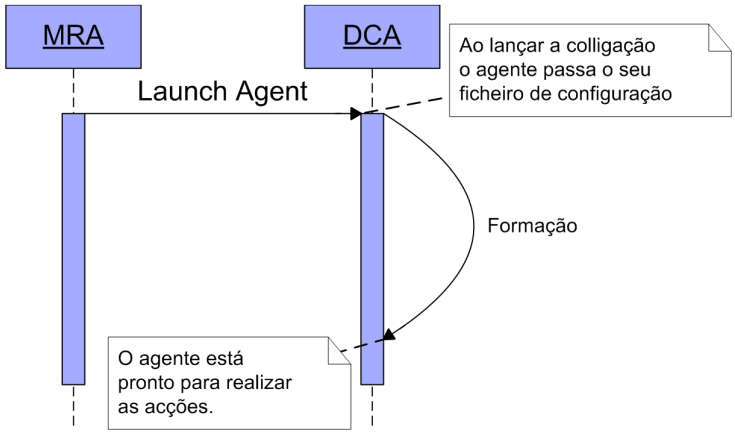

Figura 4-3: Representação abstracta do lançamento de uma coligação.

- **TA** (*"Transport Agent"*) – Transporta material entre as diferentes posições do sistema. Cada agente de transporte mantém uma tabela de encaminhamento permitindo as paletes atingirem qualquer posição no sistema. O sistema

utilizado nas tabelas de encaminhamento segue o mesmo princípio das tabelas de encaminhamento utilizadas nas redes de telecomunicações.

Na Figura 4-4, são ilustradas de uma forma abstracta as interacções que podem ocorrer entre agentes do mesmo tipo, sendo neste caso os transportes.

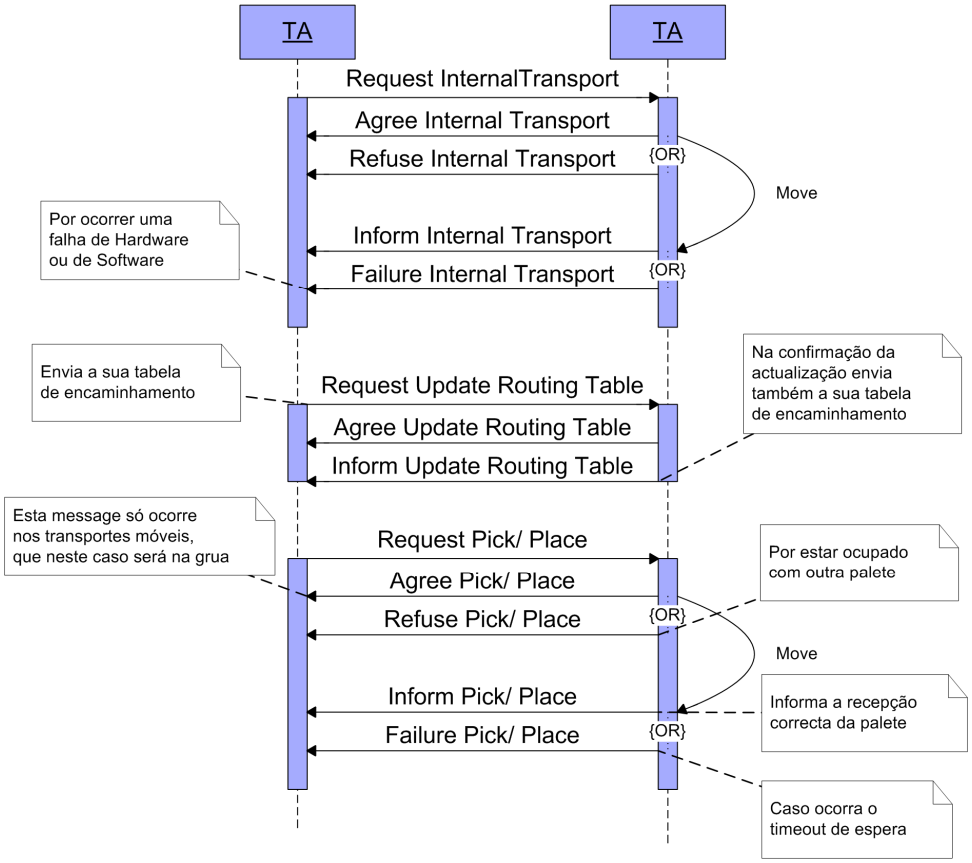

Figura 4-4: Possíveis interacções entre dois transportes.

- **DCA** (*"Dynamic Coalition Agent"*) – Este agente é responsável por formar uma coligação sempre que exista um módulo que tenha requisitos por preencher, é ainda responsável por reservar e coordená-los. Este agente é lançado no sistema sempre pelo módulo que tem a necessidade de um requisito e pode ser reutilizado.

Uma representação gráfica das interacções iniciadas pela coligação tendo como destino os módulos é apresentada na Figura 4-5, note-se que a pesquisa de

agentes que irão compor a coligação pode é efectuada de duas formas, através de um pedido directo a um módulo já adicionado (uma vez que podem oferecer mais do que uma funcionalidade), ou um pedido a todos os agentes que possam oferecer a funcionalidade desejada.

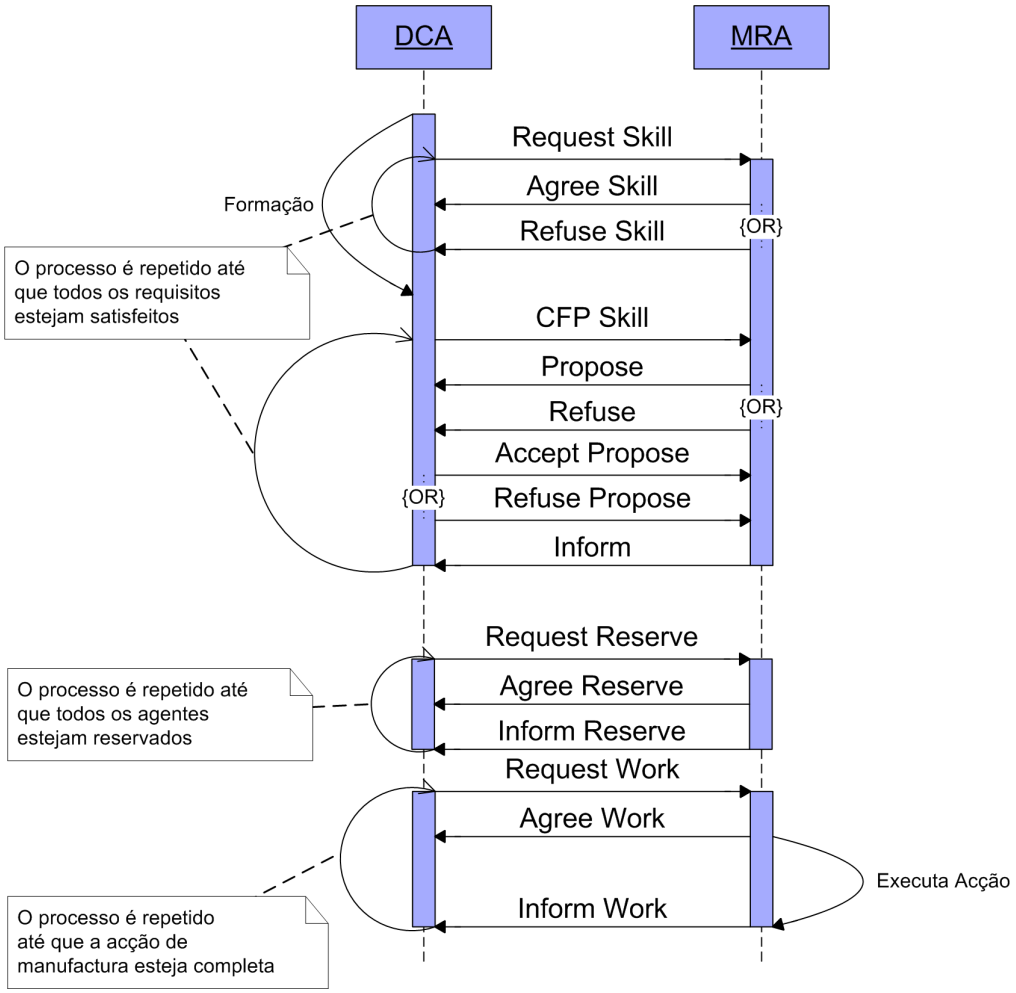

Figura 4-5: Interacções entre coligações e módulos.

- **WPCA** (*"Work Piece Carrier Agent"*) – A palete é responsável por viajar com o produto ao longo do sistema de transporte, permitindo assim que sejam executadas as operações necessárias sobre o produto.

Na Figura 4-6 é feita uma representação abstracta do pedido de transporte ao sistema de transportes realizada por uma palete.

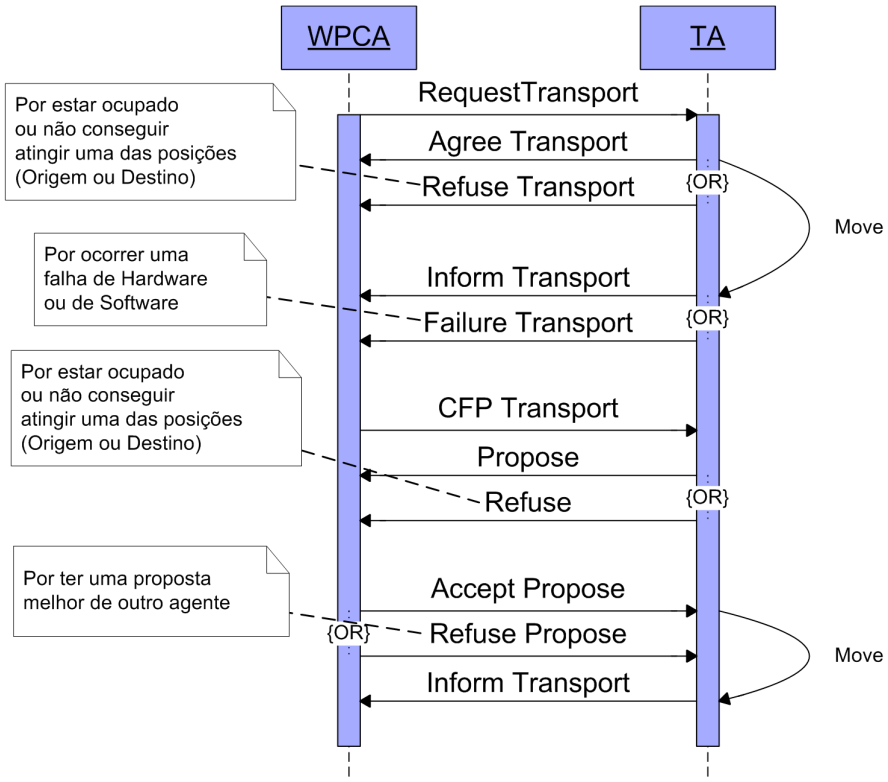

Figura 4-6: Possibilidades de interacção entre a paletes e os transportes.

- **PA** (*"Product Agent"*) – Cada agente "produto" inclui um plano de processo, o qual deverá ser executado através de interacções com os restantes módulos do sistema. Note-se que este plano de processo não refere como é que o sistema deve de executar cada acção, mas apenas a identificação da funcionalidade pretendida.

Na Figura 4-7 é feita uma representação abstracta do pedido de deslocação realizada por um produto à sua respectiva palete.

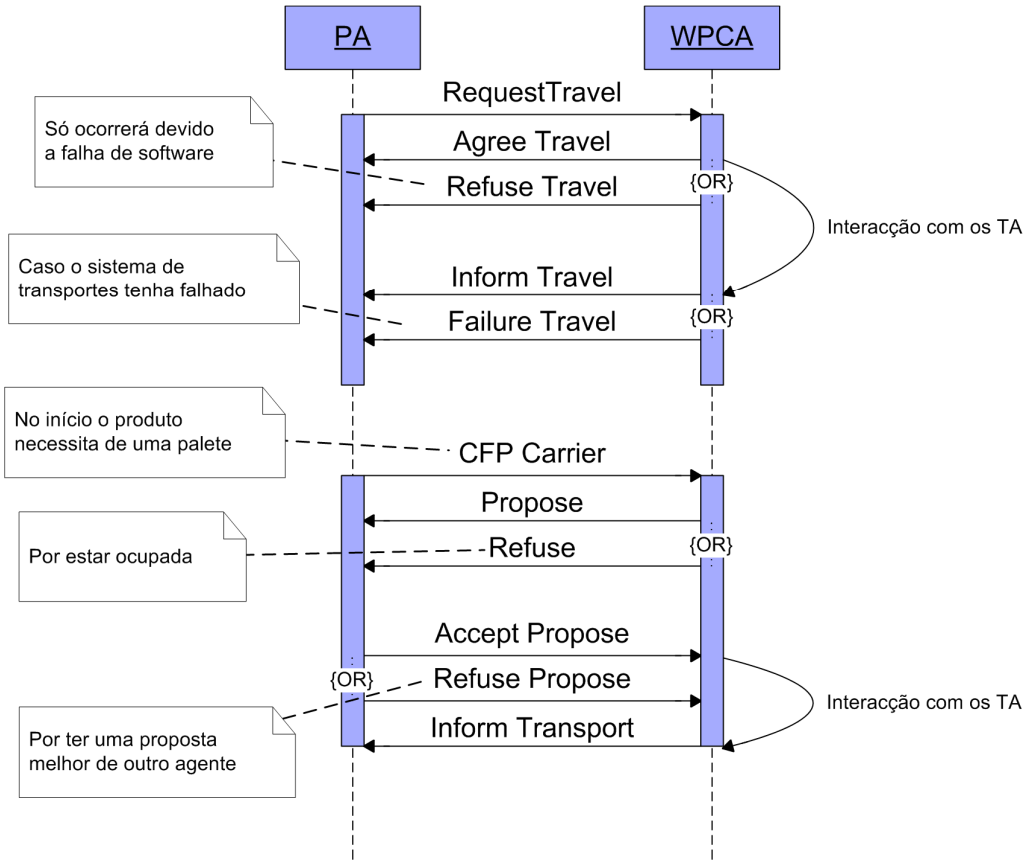

Figura 4-7: Interacção de um produto com a sua palete.

- **OntA** (*"Ontology Agent"*) – Oferece um acesso à ontologia, bem como a capacidade de filtrar a lista de agentes que oferecem uma determinada funcionalidade, através de determinadas características e/ou limitações.

Pode-se encontrar uma descrição detalhada da ontologia e do agente que a representa em (Ribeiro, Barata et al. 2008).

- **WhA** (*"Warehouse Agent"*) – O armazém tem a capacidade de armazenar paletes (com ou sem material), e produtos por processar ou já processados.
- **OA** (*"Order Agent"*) – Oferece ao utilizador uma interface gráfica que permite de uma forma rápida iniciar a actividade de vários produtos iguais, bastando para tal o utilizador fornecer o numero de produtos a serem produzidos e o plano de produção.

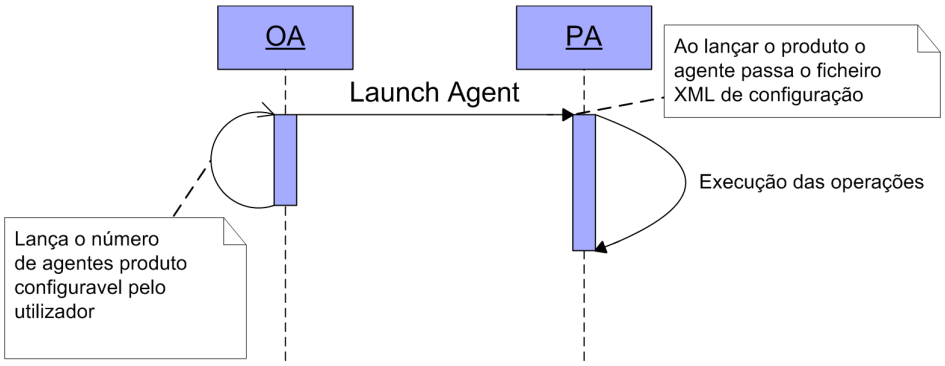

Figura 4-8: Representação abstracta do lançamento de produtos.

- **FA** (*"Feeder Agent")* – Este agente dispõe o material ainda não tratado ao restante sistema, de modo a que seja tratado sempre que o produto o requer. Este tipo de "alimentador" de material tem a característica de se encontrar numa posição fixa do sistema.
- **PSA** (*"Palette Storage Agent"*) – Este agente representa uma palete, que à semelhança do FA, dispõe o material ainda não tratado ao restante sistema, a diferença é que esta palete pode circular num sistema de transporte (principal ou num auxiliar). A ideia nesta situação é não ser o produto a ir ao encontro das partes que necessita mas sim o contrário.

4.3 Interacções suportadas

Na Tabela 4-1 são mostrados os tipos de mensagens trocadas entre os diversos módulos existentes no sistema.

Note-se que para além das mensagens do tipo *"Request"* e *"Call for Proposals"*, existe um *"Launch"*, que não é propriamente uma mensagem, é sim uma forma de designar o lançamento de um agente a partir de outro, durante esse lançamento é possível passar informação, daí ter sido incluído nesta tabela de interacções.

Destino / Origem	MRA	TA	DCA	WPCA	PA	OntA	WhA	OA
MRA	---	---	*"Launch"*	---	---	Request	---	---
TA	---	Request	---	---	---	Request	Request	---
DCA	Request CFP	---	Request CFP	---	---	Request CFP	---	---
WPCA	---	Request CFP	---	---	---	Request CFP	Request	---
PA	Request CFP	Request CFP	Request CFP	Request CFP	---	Request CFP	---	---
OntA	---	---	---	---	---	---	---	---
WhA	---	---	---	---	---	Request	---	---
OA	---	---	---	---	*"Launch"*	Request	---	---

Tabela 4-1: Tabela de interacção entre os diversos agentes existentes no sistema.

4.4 Ciclo de vida das coligações dinâmicas

O ciclo de vida de uma coligação é muito importante, uma vez que está directamente relacionado com a agilidade no sistema de produção, na medida em que podem ser adicionados ou removidos diversos módulos permitindo gerar diferentes funcionalidades que podem ser oferecidas ao sistema.

O ciclo de vida de uma coligação encontra-se definido no CoBASA (Barata 2003), e consiste em quatro fases: formação, operação, modificação e dissolução tal como apresentada na Figura 4-9. Uma vez que algumas fases são suportadas na presente tese, estas serão então detalhadas evidenciando as suas diferenças.

A fase de criação irá ocorrer quando um módulo (MRA) tiver um requisito para uma funcionalidade que lhe esta a ser requisitada por um produto (PA). No CoBASA esta fase era executada manualmente pelo utilizador.

A fase de operação refere-se à produção propriamente dita do produto, no âmbito das coligações, refere-se portanto ao funcionamento dos módulos que a constituem sobre um determinado produto. No âmbito desta tese esta fase será então dinâmica.

A fase de modificação ocorre aquando da adição de elementos existentes no sistema ou da sua remoção de uma coligação, esta fase não foi contemplada durante a implementação, no entanto como é uma fase com alguma relevância, formas de implementação, as suas vantagens e desvantagens, serão descritas nas perspectivas de trabalho futuro.

A dissolução ocorre quanto todos os elementos são removidos da coligação. Também nesta fase não foi efectuado qualquer tipo de análise para a sua implementação uma vez que o objectivo desta tese é mostrar formas de auto-organização, formando-se coligações, e não dissolvendo-as.

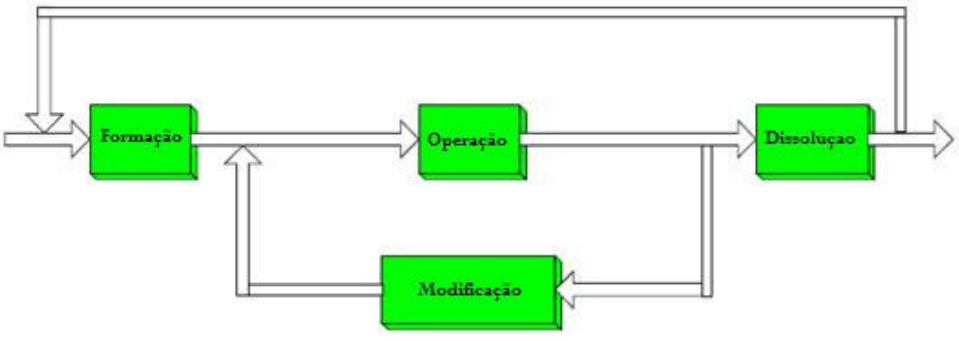

Figura 4-9: Ciclo de vida de uma coligação. [adaptada de (Barata 2003)]

O processo de formação de uma coligação, de uma forma genérica, consiste num conjunto de interacções com outros módulos do sistema tal como é apresentado na Figura 4-10.

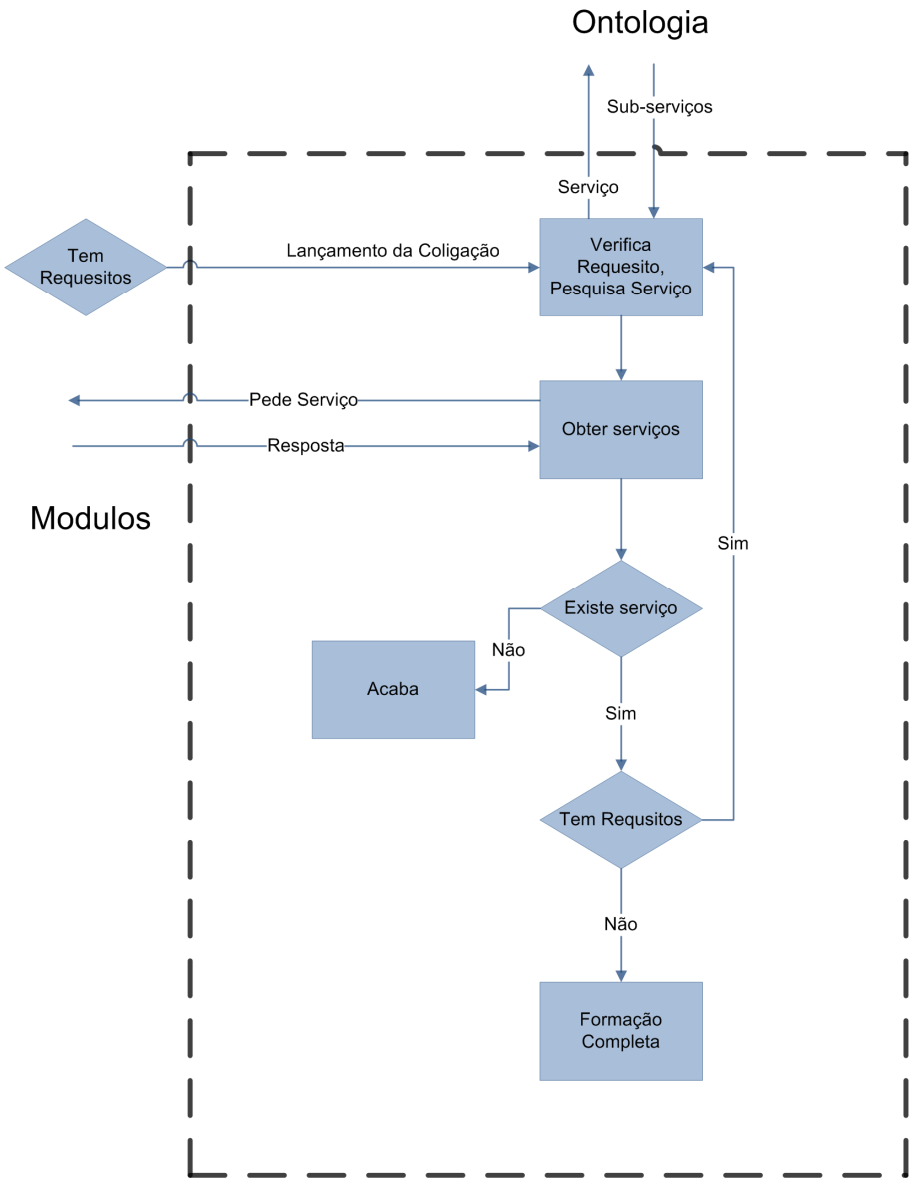

Figura 4-10: Fluxograma da formação de uma coligação.

No final do processo de formação da coligação teremos um conjunto de agentes que representam módulos físicos e irão ser orquestrados por um agente composto unicamente por *"software"*. Um exemplo concreto de formação irá ser mostrado e detalhado no capítulo da implementação e apresentação de resultados.

Capítulo 5. **Implementação e Resultados**

Pretende-se com este capítulo mostrar todo o trabalho prático desenvolvido seguindo a arquitectura descrita no capítulo anterior. Começar-se-á por descrever o caso de estudo, indicando de seguida os agentes intervenientes e respectivas funcionalidades implementadas, seguindo-se para a forma desses mesmos agentes serem inicializados e a descrição das suas interfaces gráficas. No final é apresentado um exemplo e demonstração de funcionamento.

5.1 Caso de estudo

De modo a se provar o conceito de auto-organização através da formação de coligações dinâmicas foi previamente preparado e configurado um ambiente de testes, este ambiente é um kit de manufactura didáctico como se pode ver na Figura 5-1 composto pelos seguintes módulos: uma grua com uma garra electromagnética, quatro tapetes bidireccionais, um armazém, uma plataforma giratória e outra deslizante, e quatro robôs com as respectivas pontas de trabalho.

Um dos trabalhos iniciais de preparação foi a identificação das posições de (X, Y, Z) de todos os módulos existentes no sistema. Foi ainda necessária a criação dos ficheiros XML e XSD para cada um dos módulos que iriam ser inicializados.

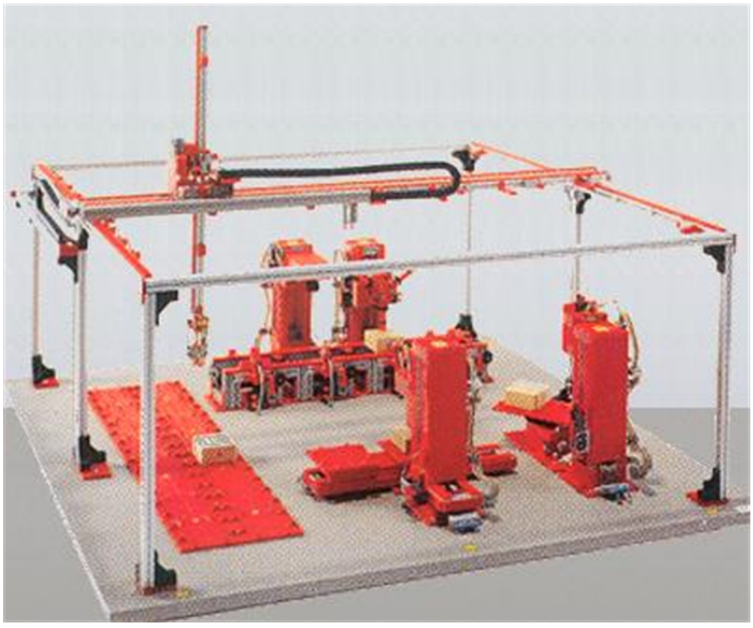

Figura 5-1: Kit didáctico Mofa France.

De seguida foi definido um *"layout"* e as coordenadas (x, y, z) de cada um dos módulos dentro desse mesmo *"layout"*, como se pode observar na Figura 5-2. Os módulos que não contêm posições definidas não foram contemplados para o caso de estudo.

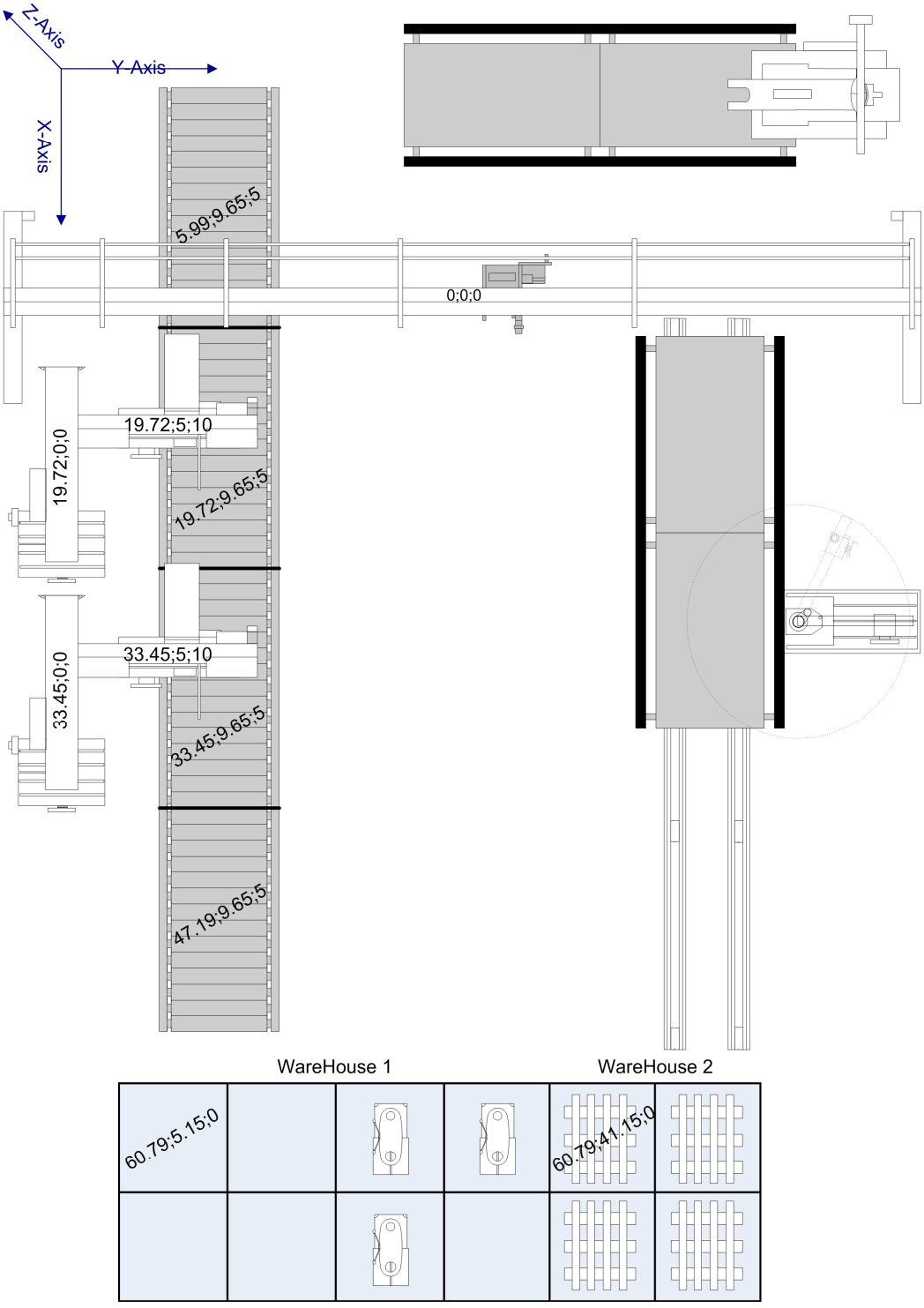

Figura 5-2: Layout.

De modo a que composição do *"layout"* apresentado anteriormente seja correctamente interpretada devem ser tidas em atenção algumas considerações iniciais.

O armazém é único, no entanto a nível de software foi dividido em dois armazéns. Tendo um sido inicializado para as paletes e o outro foi utilizado directamente pelos produtos, desta forma conseguimos observar como se comportava o sistema de transportes tendo sido uma posição relativa (dentro do armazém) transformada numa posição absoluta e sendo a outra logo uma posição absoluta no sistema.

Os dois robôs que se encontram mais à esquerda também são compostos fisicamente por um único módulo, no entanto de modo a que se pudesse observar a formação de coligações este foi dividido em dois agentes, em que um tem como requisito o outro, apesar de se ligarem ao mesmo AMI (*"Agent Manufacturing Interface"*), cada um representa um conjunto de funcionalidades diferentes.

A grua pertence ao sistema de transportes e como tal necessita de uma posição para que possa ser identificada na ontologia, de modo a que seja adicionada as tabelas de encaminhamento e de vizinhos directos dos restantes módulos pertencentes ao sistema de transporte, assim a posição da grua foi definida como sendo as coordenadas (0,0,0), esta é ainda indicativa de que o módulo consegue alcançar qualquer ponto do sistema, dentro dos seus limites.

Na Figura 5-3 pode-se observar a grua a remover uma palete do respectivo armazém.

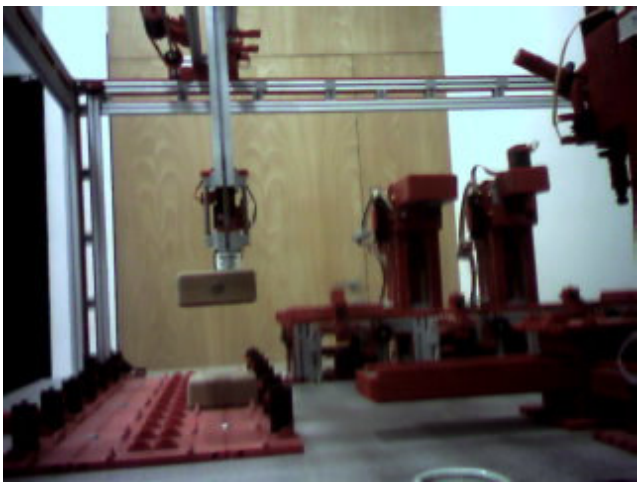

Figura 5-3: Foto da grua removendo uma palete do armazém.

5.2 Funcionalidades

De acordo com os tipos de módulos já descritos no capítulo anterior, são agora apresentadas na Tabela 5-1, os nomes dos módulos, as respectivas funcionalidades implementadas em cada um dos tipos de módulos instanciados para o caso de estudo apresentado.

Tipo	Módulo	Funcionalidades
MRA	Drill Chunk Tool Machine 1	Furar Hold
TA	Conveyor Belt 1 Conveyor Belt 2	Mover para Frente/Trás Mover para Frente/Trás
	Crane	Pick, Place, GotoPosition
DCA	Módulo Virtual	
PA	Producto	
WPCA	Palete	Transporte de Material
WhA	Armazém	Armazenamento/Buffer
OntA	Ontologia	Registo/Pesquisa

Tabela 5-1: Módulos e Funcionalidades implementadas para o caso de estudo.

5.3 Inicialização dos Agentes

Inicialmente todos os agentes são carregados com dois ficheiros: um em XSD (W3C 2004) e um em XML (Bray, Paoli et al. 2008). O ficheiro XSD servirá apenas para validar o XML de modo a que a posterior leitura do mesmo seja feita de uma forma correcta, o ficheiro de XML é a inicialização do agente, contendo toda a informação acerca das funcionalidades que é capaz de oferecer, as suas características e as suas limitações.

Na Figura 5-4 é apresentada a primeira parte do ficheiro XSD pertencente ao módulo *"DrillChunk"*.

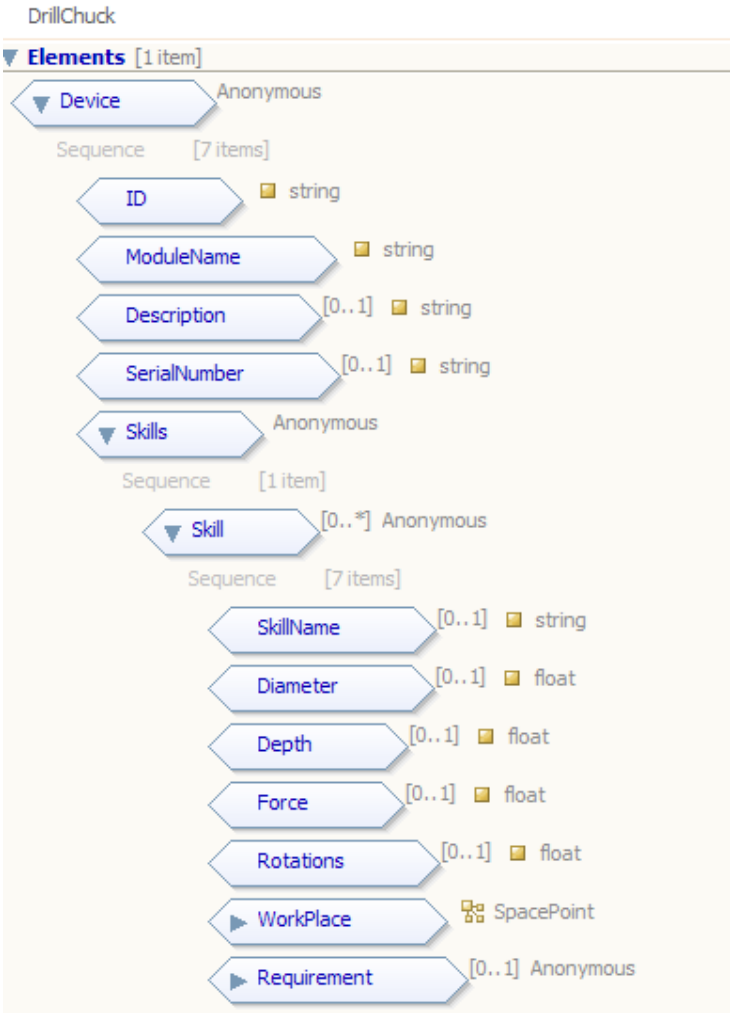

Figura 5-4: Primeira parte da estrutura do *"schema"*.

Os primeiros identificadores referem-se apenas à identificação do módulo, bem como uma breve descrição do mesmo. O utilizador tem a liberdade de adicionar ou remover os identificadores que pretender.

O identificador *"Skills"* refere-se às funcionalidades que o módulo pode oferecer ao sistema, este pode conter uma ou mais funcionalidades. As propriedades apresentadas são apenas uma ilustração do que o utilizador pode definir, à excepção das últimas duas, designadas por *"Workplace"* e *"Requirement"*, e que devem sempre acompanhar cada bloco referente a uma funcionalidade, isto deve-se ao facto de a primeira ser utilizada para que o produto possa alcançar as coordenadas onde será executada a operação, e a segunda é utilizada para que o módulo possa

lançar a coligação sempre que necessário. Estas coordenadas referem-se à posição absoluta no sistema.

A restante descrição (propriedades genéricas e limitações) do módulo é apresentada na Figura 5-5.

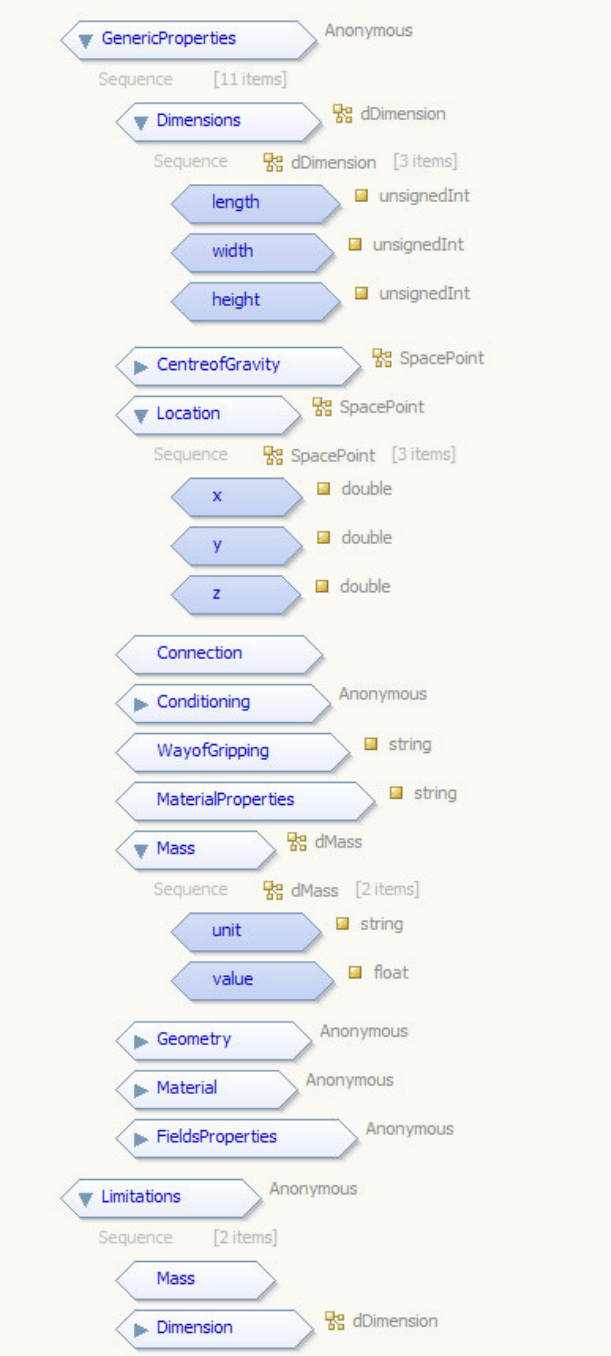

Figura 5-5: Segunda parte da estrutura do *"schema"*.

As propriedades genéricas servem para caracterizar o módulo físico, neste exemplo foram consideradas as dimensões (comprimento, largura e altura), centro de gravidade, localização física, tipo de conexão e de armazenamento, propriedades do material por que é composto, a sua massa, a geometria, tipo de material e propriedades de campos (eléctrico, magnético) caso existam.

Como limitações foram apenas definidas a massa e as dimensões máximas suportadas.

O ficheiro XML deverá obedecer à estrutura do ficheiro XSD apresentado anteriormente, podemos observar um exemplo do mesmo módulo na Figura 5-6.

```
<Device xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
        xmlns='DrillChuck'
        xsi:schemaLocation='DrillChuck  DrillChuck.xsd'>
    <ID>D20to30mm</ID>
    <ModuleName>Drill Chuck</ModuleName>
    <Description>Drill holes between 20mm and 30mm</Description>
    <SerialNumber>sn000dt001</SerialNumber>
    <Skills>
        <Skill>
            <SkillName>Drill</SkillName>
            <Diameter>25</Diameter>
            <Depth>25</Depth>
            <Rotations>2000</Rotations>
            <WorkPlace>
                <x>33.45</x>
                <y>9.65</y>
                <z>5.0</z>
            </WorkPlace>
            <Requirement>
                <SkillName>Gripper Holder</SkillName>
                <Time/>
            </Requirement>
        </Skill>
        <Skill>
    </Skills>
    <GenericProperties>
        <Dimensions>
            <length>10</length>
            <width>10</width>
            <height>15</height>
        </Dimensions>
        <CentreofGravity>
        <Location>
            <x>33.45</x>
            <y>5.0</y>
            <z>10.0</z>
        </Location>
        <Connection>AFDE200</Connection>
        <Conditioning>
        <WayofGripping>Magnetic</WayofGripping>
        <MaterialProperties>Rigidity</MaterialProperties>
        <Mass>
            <unit>Kg</unit>
            <value>1</value>
        </Mass>
        <Geometry>
        <Material>
        <FieldsProperties>
            <magnetic></magnetic>
        </FieldsProperties>
    </GenericProperties>
    <Limitations>
        <Mass>
            <unit></unit>
            <value></value>
        </Mass>
        <Dimension>
    </Limitations>
</Device>
```

Figura 5-6: Exemplo de um ficheiro XML de configuração dos agentes.

No início foi definido um identificador único do módulo, o nome, uma pequena descrição do que o módulo físico é capaz de oferecer ao sistema e ainda um número de série. De seguida temos a descrição das funcionalidades do módulo, neste exemplo foram definidas duas, mas apenas uma é mostrada em detalhe, a funcionalidade é composta pelo nome *"Drill"* e por algumas propriedades (tais como o diâmetro do furo, a sua profundidade e o número de rotações a que o módulo efectua a operação), estas características poderão influenciar na escolha ou não deste módulo. Ainda dentro da funcionalidade é indicada a posição de trabalho, esta propriedade foi definida como pertencendo à funcionalidade e não ao módulo uma vez que este pode oferecer mais do que uma funcionalidade em localizações distintas. O referido módulo tem ainda um requisito a ser satisfeito no caso de lhe ser pedida esta funcionalidade, o requisito é um outro módulo e que tenha a capacidade de o segurar e o mover para a posição onde efectua a operação.

As propriedades genéricas definidas, também podem ser condicionantes ou um factor de escolha quando o módulo está em selecção para pertencer a uma coligação, ou para uso directo num produto (caso o módulo não possua requisitos) deste modo foram definidas as seguintes propriedades:

- As dimensões físicas do módulo;
- A localização (33.45;**5.0**;10.0) é uma localização entre o módulo que irá responder ao seu requisito (33.45;**0**;0) e a posição de trabalho deste módulo (33.45;**9.65**;5.0);
- De seguida temos a definição do tipo de ligação (meramente ilustrativa);
- A forma de o segurar é magneticamente,
- A propriedade do material que o constitui é rígida;
- O módulo tem a massa de um quilograma;

O centro de gravidade, tipo de acondicionamento, a geometria, o tipo de material, e o campo magnético não foram definidos.

Por fim teríamos as limitações do módulo, no entanto aparentemente não possuía nenhuma.

5.4 Interfaces gráficas

Cada agente dispõe de uma componente gráfica de modo a que se possa de uma forma visual identificar algumas propriedades relevantes durante o seu funcionamento. As interfaces são mostradas e detalhadas de seguida.

5.4.1 MRA (Módulo)

Na interface gráfica do MRA (Figura 5-7) pode-se observar um botão (*"Load XSD and XML"*) que permite carregar os dois tipos de ficheiros apresentados anteriormente inicializando assim o agente. Durante esta inicialização são alimentados automaticamente os seguintes campos: *"LocalPos", "Working Place" e "Part Name".*As duas primeiras propriedades são uma posição tridimensional e referem-se às coordenadas absolutas no sistema, a primeira é a posição inicial do módulo e a segunda é a posição onde é executada a operação, como se pode observar esta última posição não foi inicializada uma vez que o módulo possuía duas funcionalidades distintas, tendo sido detectada aqui um ponto de melhoria da interface gráfica, este tópico será mais detalhado nas conclusões e perspectivas de trabalho futuro. A outra propriedade inicializada automaticamente é um nome simbólico do módulo (*"Drill Tool"*).

O custo apresentado pode ser um custo definido em tempo e serve para se efectuar a escolha de um módulo sempre que existam mais do que um a oferecer a mesma funcionalidade.

Existe ainda um espaço para se indicar o AMI deste agente, isto é o agente com que se deve comunicar para efectuar uma acção física no módulo (movimento, operação, etc.).

Existem dois pequenos painéis, com a designação de *"Busy"* e *"Reserve"*, e que servem respectivamente para indicação se o módulo está a executar alguma operação e se o módulo está reservado para alguma coligação.

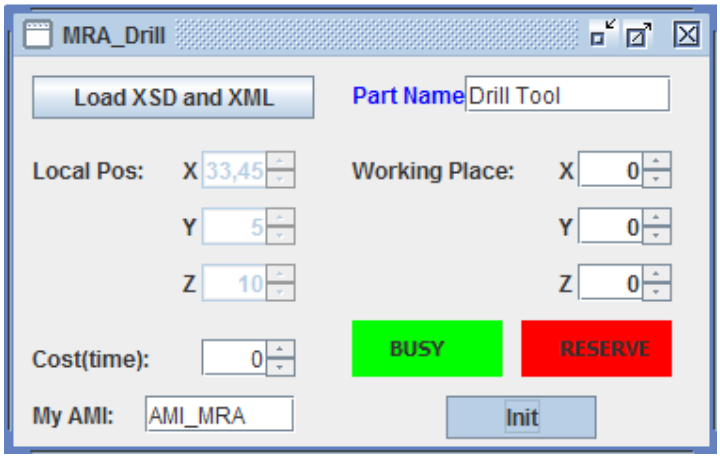

Figura 5-7: Interface gráfica de um agente módulo após a inicialização.

5.4.2 WhA (Armazém)

À semelhança da interface apresentada anteriormente, a interface do armazém também dispõe do botão que permite a sua inicialização, sendo os seguintes campos afectados automaticamente, *"Name"* e *"Position"*, que tal como os nomes indicam o nome simbólico do agente e a posição tridimensional do módulo, a posição mais uma vez refere-se à posição absoluta no sistema.

Esta interface dispõe ainda de dois campos *"Cost"* e *"Stored"*, o último dá ao utilizador a indicação do número de posições ocupadas, a propriedade *"cost"* tem um sentido um pouco diferente das outras interfaces uma vez que aqui não fará sentido ter o tempo como custo de operação, mas sim um custo inversamente proporcional ao número de posições livres, quer isto dizer que quanto mais ocupado este estiver mais "caro" será a ocupação de uma posição.

Existe ainda nesta interface uma tabela que é configurada automaticamente através da capacidade definida no ficheiro XML de inicialização, e que tem como objectivo a visualização das posições ocupadas, bem como o nome do agente que a ocupa.

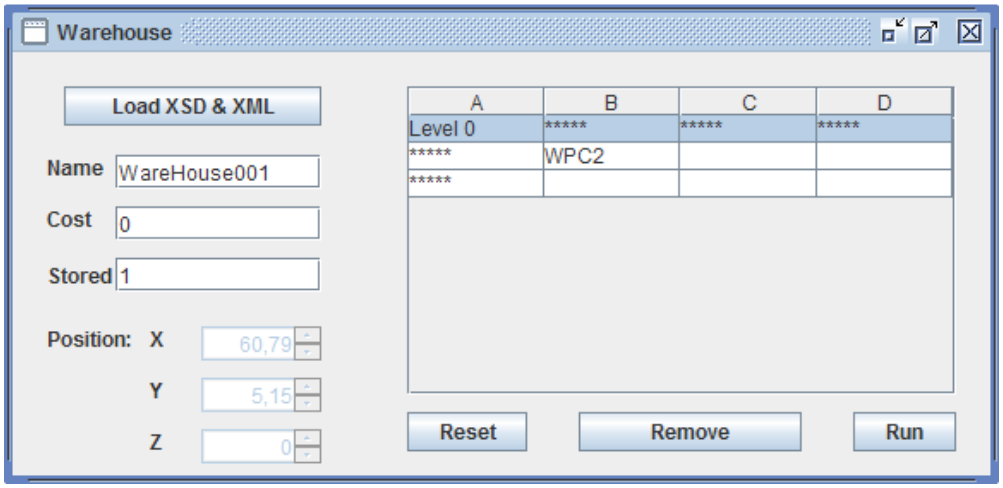

Figura 5-8: Interface gráfica de um agente armazém após a inicialização.

A implementação do armazém conta ainda com uma particularidade que se reflecte na simplicidade de utilização. O utilizador quando pretende registar uma nova palete no armazém não tem a necessidade de conhecer a posição absoluta de um determinado espaço não ocupado em relação ao sistema, basta introduzir a posição relativa que a palete ocupa dentro do armazém e este conhecendo a sua posição absoluta em relação ao sistema, irá então calcular e informar a palete da sua posição absoluta.

5.4.3 WPCA (Palete)

A palete dispõe da interface gráfica apresentada na Figura 5-9, exibindo duas partes já mencionadas e explicadas anteriormente, o botão de inicialização e a indicação da posição absoluta no sistema. Existem no entanto três outras partes, o botão *"GetWareHouses"*, e duas listas *"Initial WareHouse"* e *"Final WareHouse"*.

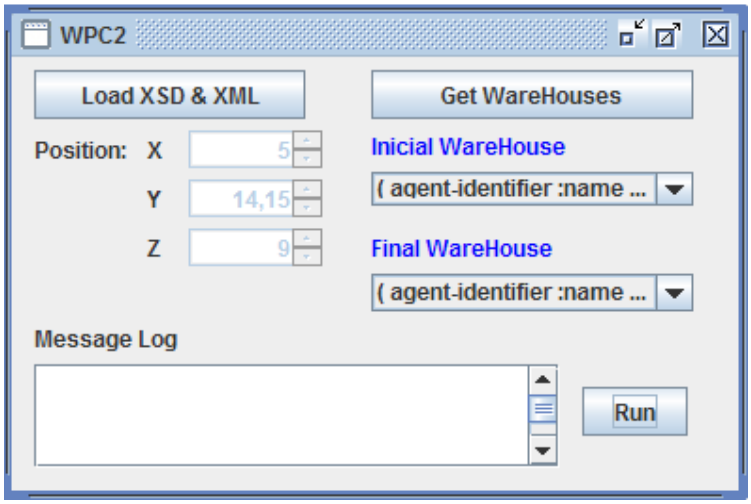

Figura 5-9: Interface gráfica de um agente palete após a inicialização.

O botão "*GetWareHouses*", tal como o nome indica, obtém a lista de armazéns, esta pesquisa é efectuada com recurso a uma mensagem enviada à ontologia indicando o que se pretende, neste caso é a funcionalidade *"storage"*, a lista de agentes que é retornada é colocada nas duas listas mencionadas anteriormente de modo a que o utilizador possa escolher os armazéns onde se encontra a palete e onde deseja que esta seja colocada no final das operações que serão efectuadas no produto que irá transportar.

5.4.4 TA (Transporte)

Os agentes que representam os transportes são os que possivelmente possuem a interface gráfica mais complexa como se pode observar na Figura 5-10.

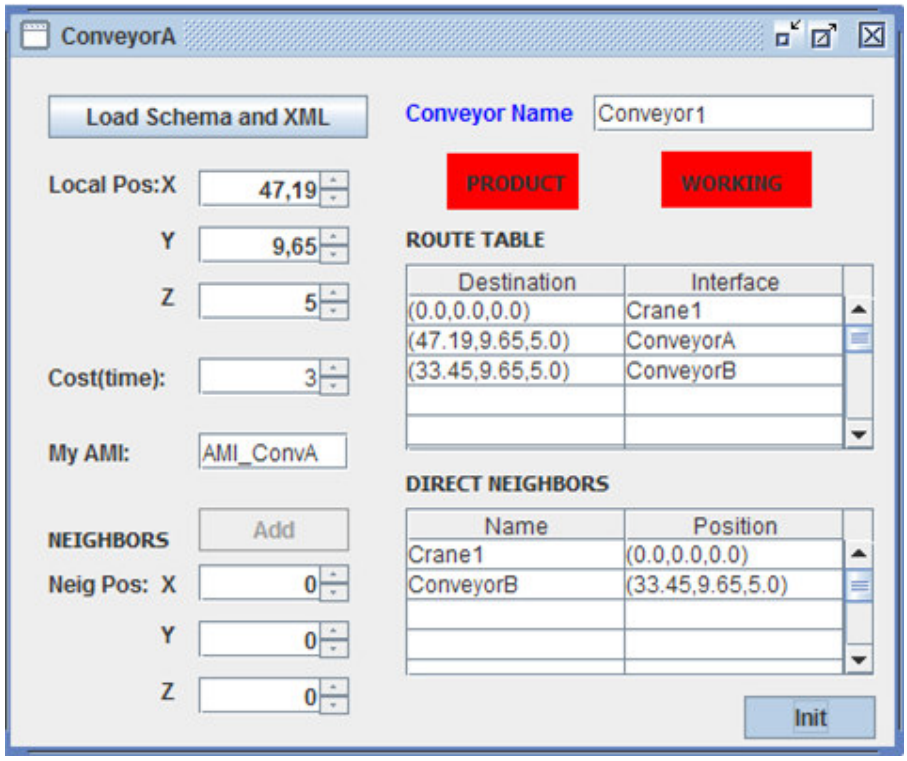

Figura 5-10: Interface gráfica de um agente de transporte após a inicialização.

Esta interface é composta pelo botão de inicialização, posição de base, custo em tempo e o nome do "AMI" já mencionados anteriormente. Possui ainda duas tabelas, designadas por *"Route Table"* e *"Direct Neighbors"* e que mostram respectivamente a sua tabela de encaminhamento, e tabela de vizinhos directos.

Começando pela tabela de vizinhos directos, esta é composta pelo nome do agente que ocupa uma determinada posição, e na segunda coluna a referida posição física. O nome do agente é obtido através de uma interacção com a ontologia, em que passando a posição do agente a ontologia envia uma mensagem informando o nome do agente que a ocupa ou vazio se não existir nenhuma correspondência para a posição indicada.

Após a adição de um vizinho directo na tabela o agente irá enviar para o agente adicionado uma mensagem contendo a sua tabela de encaminhamento e recebendo como resposta a tabela de encaminhamento desse agente, após esta troca inicial de tabelas de encaminhamento é feito um reenvio periódico de trinta em trinta segundos. O tempo definido pode parecer exagerado, no entanto temos de

ter em conta a aplicação final deste sistema, que é a indústria de manufactura, onde as alterações à plataforma fabril não acontecem de uma forma rápida e constante.

A tabela de encaminhamento como mencionado anteriormente é semelhante às tabelas de encaminhamento encontradas nas redes de telecomunicações (Figura 5-11), no entanto possui menos parâmetros e com algumas modificações. O destino no sistema de transportes não é um endereço de IP (*"Internet Protocol"*) mas sim uma posição física no sistema. A interface pelo qual o pedido deve de ser caminhado também não é o IP mas sim o nome do agente que ocupa a próxima posição, neste último campo optou-se por colocar o nome do agente e não a posição desse agente para que quando se tenha de enviar um produto para esse destino não se tenha de consultar novamente a ontologia perguntando quem é que ocupa essa posição.

Por fim, temos três painéis *"Product"* e *"Working"*, e que representam respectivamente o estado do sensor do tapete, isto é se tem uma palete ou não; e o estado físico propriamente dito do tapete, ou seja se este se encontra ou não em movimento.

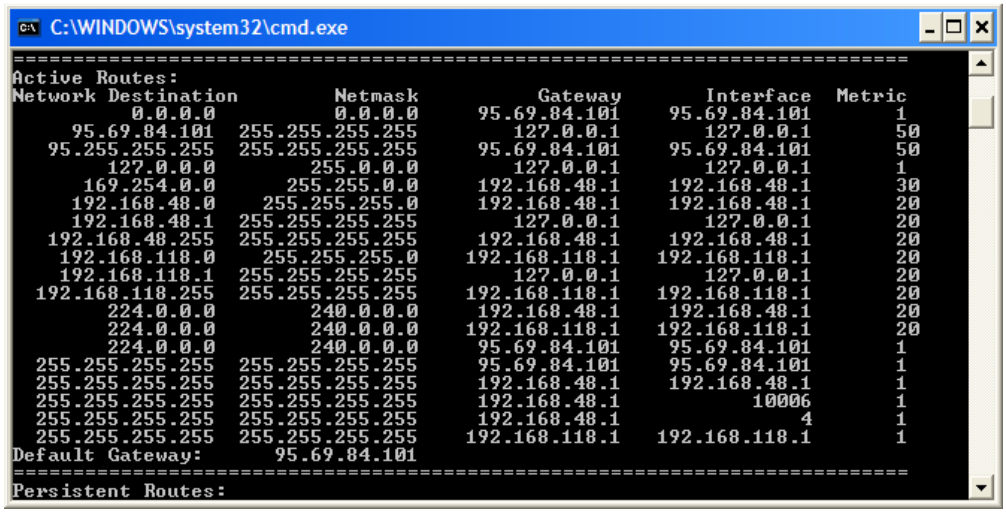

Figura 5-11: Tabela de encaminhamento numa rede de telecomunicações.

5.4.5 DCA (Coligação)

A interface gráfica do agente que representa a coligação Figura 5-12 é apenas composta por uma tabela com três colunas sendo assim de todas a mais simples, no entanto não a menos importante, como veremos de seguida.

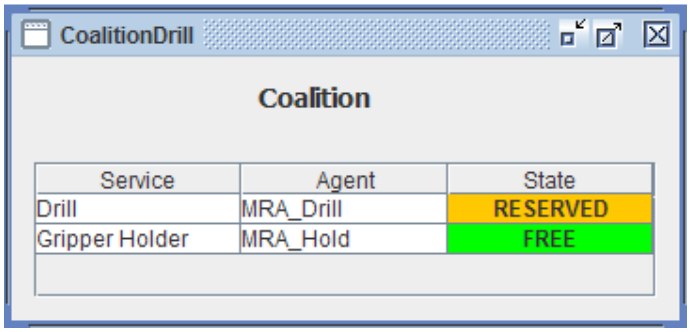

Figura 5-12: Interface gráfica da coligação após o seu lançamento.

A primeira coluna mostra ao utilizador o serviço que a coligação pretende obter do módulo físico, uma vez que o módulo pode oferecer ao sistema mais do que uma funcionalidade. A segunda coluna é o nome do respectivo agente que oferece a funcionalidade, e por último o estado em que se encontra cada agente no momento. Existem três estados, livre (fundo a verde), reservado (fundo laranja) e em funcionamento (fundo vermelho).

Note-se que a interface gráfica deste agente não dispõe de um botão, à semelhança dos restantes, para efectuar o *"Load"* dos ficheiros de inicialização e respectivo ficheiro de validação, este facto deve-se à herança das propriedades dos módulos que o compõem, quer isto dizer que as funcionalidades, da coligação para o restante sistema serão uma união das funcionalidades de cada um dos módulos. Em relação às propriedades será uma intersecção destas.

5.4.6 PA (Produto)

Por fim, temos a interface gráfica do produto, sendo através desta que todo o sistema começa a trabalhar.

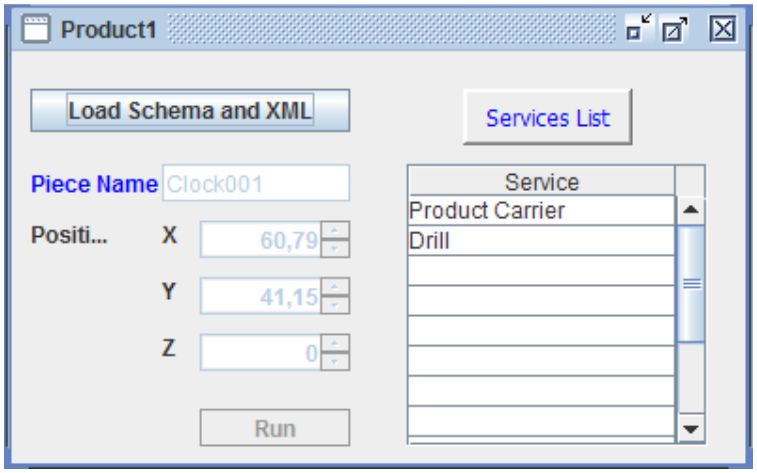

Figura 5-13: Interface gráfica do agente produto.

A semelhança de interfaces já explicadas anteriormente, esta dispõe de um botão de inicialização do agente, no entanto apresenta uma particularidade, uma vez que não se pode prever a composição da lista de tarefas a efectuar sobre o produto. Isto é dentro de uma funcionalidade pode ser introduzido determinados valores como máximos ou mínimos, por exemplo ao se definir uma funcionalidade de "Perfurar", o utilizador pode necessitar de especificar o número de rotações máximo ou a profundidade mínima que o furo deverá ter. Desta forma, não se pode introduzir um ficheiro XSD para a validação dos dados, assim a inicialização deverá ser efectuada apenas com um ficheiro XML. Esta abordagem levanta o problema de um determinado agente "Produto" ser mal inicializado e consequentemente ter um comportamento indesejado. A solução para este problema encontra-se detalhada nas conclusões e perspectivas de trabalho futuro.

5.5 Exemplo de Funcionamento

5.5.1 Formação da Coligação

Numa descrição anterior deste trabalho foi mostrado como é que uma coligação era formada e como esta funcionava (Frei, Ferreira et al. 2008), no entanto durante a implementação surgiu uma nova versão com alterações significativas e importantes, que irão de seguida ser descritas e explicadas. A primeira grande alteração foi no modo como se percebia que era necessário lançar uma coligação e quem a lançava, na versão anterior esta acção era realizada pelo agente que requisitava a funcionalidade que ainda não se encontrava disponível na sua forma mais simples, neste caso era o agente "produto". Nesta nova versão o agente "produto" não necessita de ter conhecimento se o agente MRA consegue ou não realizar uma acção sozinho ou necessita de algo mais, a alteração foi então no sentido de passar essa decisão ao MRA, uma vez que possui informação de modo a saber se necessita de algo mais para realizar a acção.

Na Figura 5-14 é mostrado um diagrama de sequência UML (*"Unified Modeling Language"*) com a troca de mensagens entre os diversos agentes intervenientes na formação de uma coligação.

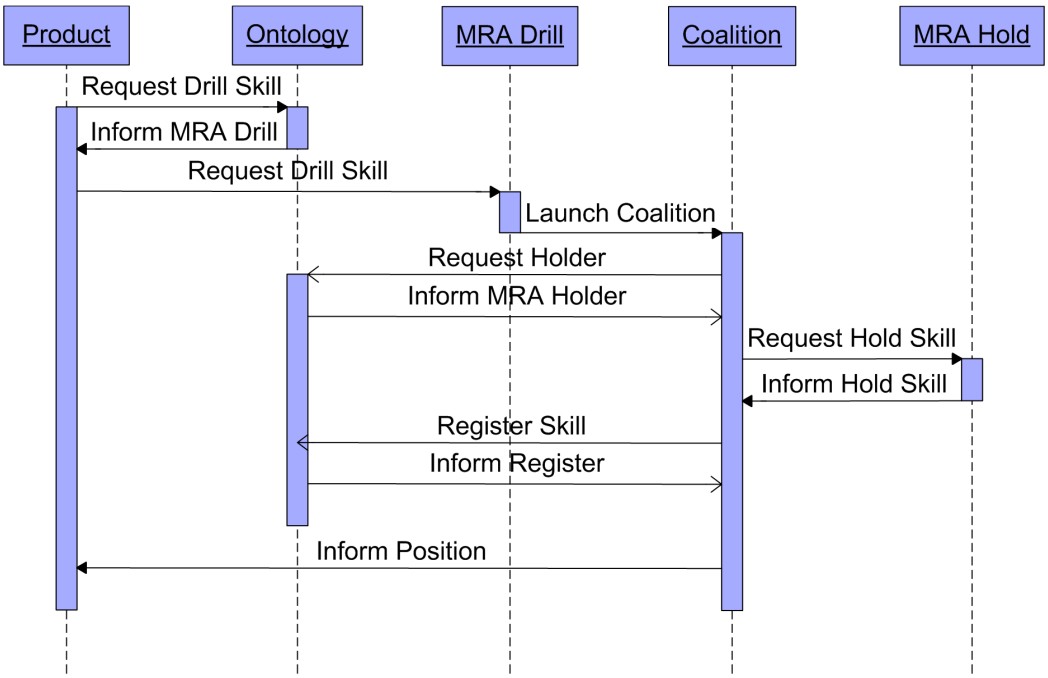

Figura 5-14: Diagrama de sequência UML da formação de uma coligação.

O processo é iniciado pelo produto que envia uma mensagem de *"request"* à ontologia perguntando quem oferece a funcionalidade *"drill"*, a ontologia responde que a MRA Drill consegue realizar, o produto requer então a acção ao agente que oferece a funcionalidade pretendida, de seguida esse agente analisa internamente se existe algum requisito para a acção requisitada, se não existir o agente informa que consegue realizar a acção informando a localização onde a executa, se existir algum requisito este lança um agente coligação passando-lhe informação sobre ele próprio, ou seja o seu ficheiro de inicialização. A coligação é neste momento composta por um agente que é o agente básico da coligação, isto significa que é quem realmente consegue realizar a acção e será esta mesma capacidade que a coligação vai registar na ontologia. O agente de coligação analisa novamente qual é o requerimento (neste caso é um suporte) e pergunta à ontologia quem consegue realizar esta acção/ funcionalidade após a resposta da ontologia a coligação pede a esse agente essa acção, na mensagem de *"inform"* o MRA Hold envia toda a informação sobre ele tal como o MRA Drill o fez, o agente de coligação repete o passo de análise para verificar se existe mais algum requisito, e que neste caso já não existe.

Os passos finais na formação de uma coligação são o registo na ontologia e o envio de uma mensagem de *"inform"* para o produto indicando o local onde executará a acção. Após estes passos a coligação está formada e pronta a funcionar.

O caso que aqui estamos a demonstrar é um caso especial porque o MRA Drill não necessita de mais nada para realizar a sua acção, isto é para furar, no entanto não consegue alcançar o ponto onde é suposto realizar a acção e neste caso precisa de um suporte para o segurar e o fazer atingir a posição desejada, portanto o MRA Drill lançou a coligação para encontrar essa capacidade e orquestra-la na forma correcta.

A outra diferença é que o transporte não faz mais parte da coligação devido ao decréscimo de gracilidade da coligação nos artigos anteriores (Frei, Ferreira et al. 2008; Ribeiro, Barata et al. 2008) considerou-se cada MRA como o mais pequeno módulo no sistema, no entanto nesta arquitectura/ implementação decompôs-se cada MRA em dois MRAs diferentes e consequentemente mais pequenos (Drill e Hold), os quais funcionam sobre o mesmo transporte, no entanto pretende-se no futuro voltar a incluir o sistema de transporte de forma a criar coligações entre diferentes conjuntos de módulos.

5.5.2 Transporte de um produto para uma posição de trabalho

Na sequência da formação da coligação, e desta ter enviado a sua posição de trabalho, existe então a necessidade de transportar o produto através do sistema de transportes. Na Figura 5-15 é mostrado um diagrama de sequência referente à troca de mensagens entre os agentes envolvidos no transporte do produto.

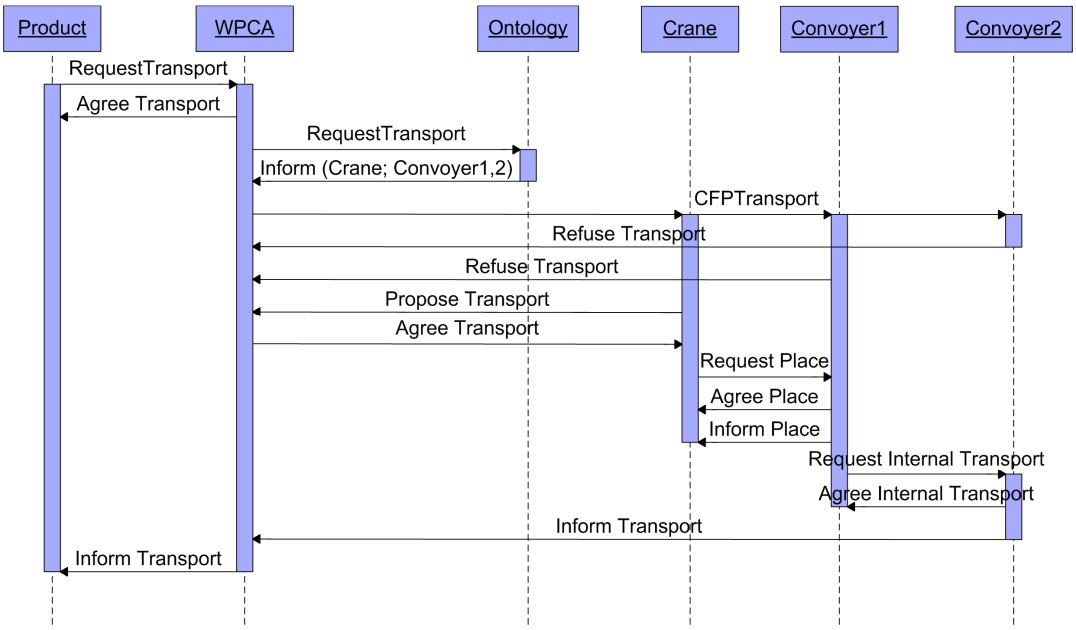

Figura 5-15: Diagrama de sequência UML da realização de um transporte.

O produto é quem despoleta a acção de transporte através de um pedido de transporte à sua palete, caso não tenha ocorrido nenhum problema de hardware ou software esta irá "concordar" em fazer o transporte. Deste modo começa por fazer uma pesquisa na ontologia por transportes, uma vez que não foi indicado nenhum constrangimento por parte da palete a ontologia irá fornecer a lista de todos os módulos que têm a funcionalidade de transporte, a palete envia de seguida uma mensagem do tipo *"Call For Proposals"* para os agentes indicados pela ontologia, esta mensagem já inclui a posição onde a palete se encontra e a posição final que pretende atingir. Uma vez que os tapetes estão numa posição fixa, estes recusam o transporte, ficando apenas a proposta da grua e consequentemente é a que é aceite.

A grua começa por analisar internamente, com recurso às suas tabelas de encaminhamento e vizinhos directos, se a posição de origem e a posição de destino pertencem a algum agente, caso pertençam esta terá de efectuar um pedido para respectivamente retirar ou colocar a palete, na figura anterior é mostrado o caso em que a posição de destino pertence ao *"Covoyer1"*, assim a grua após retirar a palete do armazém envia uma mensagem para o *"Covoyer1"* perguntando se pode colocar a peça, à qual recebe resposta positiva. Após a grua deixar a palete é notificada com uma mensagem de *"inform"*.

O *"Covoyer1"* analisa se a posição desejada pelo produto é a sua, e uma vez que não é irá então analisar as suas tabelas de encaminhamento e de vizinhos directos de modo a saber por onde deve encaminhar a palete, após ter um outro agente de transporte para encaminhar a palete é efectuado um pedido de transporte interno. A palete ao chegar à posição desejada é notificada pelo último transporte utilizado, e notifica então o produto que transporta sobre o resultado do transporte.

Note-se que a grua também pertence ao grupo do sistema de transportes, no entanto possui algumas mensagens diferentes, o que se deve ao facto de esta ser um transporte móvel.

O produto ao ser informado que chegou à posição onde irá ser processado, enviará uma mensagem de trabalho ao módulo respectivo desencadeando assim o restante processo, esta interacção e as consequentes são ilustradas na Figura 5-16.

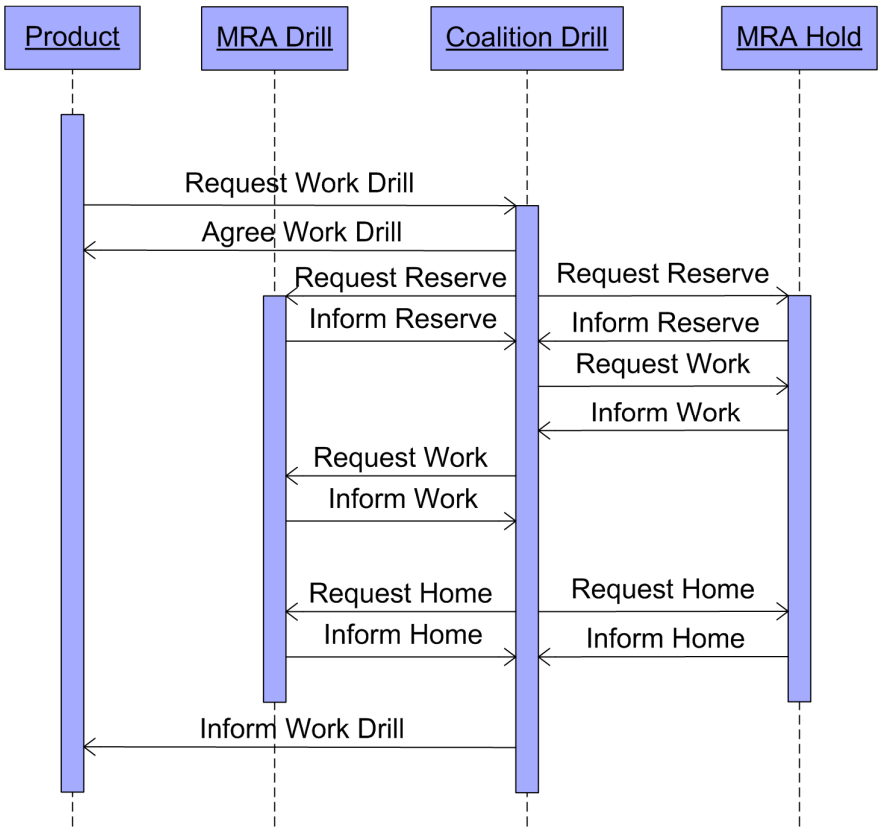

Figura 5-16: Diagrama de sequência UML do funcionamento de uma coligação.

A coligação após receber a mensagem de trabalho começa por reservar todos os agentes que a compõem, após o sucesso desta operação a coligação começa por orquestrar os agentes para realizar as acções de uma forma organizada e pela ordem correcta, esta ordem é dinâmica, isto é os agentes são adicionados a coligação por serem o requisito do agente posterior, por isso o primeiro agente que irá executar a acção é o que não possui requisitos, ou seja, o último que foi adicionado à coligação. No final a coligação informa o produto sobre o resultado da operação ou conjunto de operações.

5.6 Demonstração de Funcionamento

Um vídeo de demonstração da tese elaborada está disponível em http://www.archive.org/details/DynamicCoalitions-Video.

Capítulo 6. **Conclusões e Perspectivas**

6.1 Conclusões

Deste trabalho resultaram três artigos publicados, onde ao longo destes artigos é possível acompanhar não só o trabalho que foi desenvolvido mas também as modificações que foram sido efectuadas até se chegar ao presente trabalho.

As melhorias em relação a trabalhos anteriores é significativa e relevante, uma vez que trás um maior dinamismo às plataformas fabris. A forma de implementação de uma forma geral é inovadora.

Uma melhoria relevante, em relação ao CoBASA ou ao ProPlanT foi a alteração da forma de lançamento das coligações, a qual passou a ser efectuada de forma automática. Também a sua formação e forma de funcionamento são efectuadas de forma autónoma conseguindo-se assim provar que o conceito de auto-organização entre agentes de forma a preencher requisitos do sistema é possível.

A introdução das tabelas de encaminhamento nos sistemas de transporte foi também uma grande ajuda e um passo importante para que se conseguisse atingir um sistema dinâmico através do conhecimento local.

Um resumo das vantagens sobre a formação dinâmica de coligações é enumerado de seguida:

- Durante a formação das coligações são escolhidos os melhores módulos e mais rapidamente;
- O possível aparecimento de erros devido a incompatibilidades é menor;
- Torna o processo de reengenharia mais rápido;
- É feita uma melhor reutilização dos recursos existentes no sistema;
- Limitações e graus de liberdade do sistema são geridos de forma dinâmica e automática.

De uma forma sucinta provou-se ser possível tornar um sistema complexo, num sistema dinâmico exibindo comportamentos de auto-organização, decompondo-o em partes mais pequenas, isto é, um nível de graularidade adequado, interagindo apenas com conhecimento local e dando também uma descrição adequada a cada um dos módulos que o constituiem.

6.2 Perspectivas de Trabalho Futuro

Como não poderia deixar de ser, um trabalho numa área em constante evolução deixa sempre pontos em aberto, os quais serão evidenciados de seguida bem como uma forma de os atingir.

De modo a que o utilizador tenha um apoio na criação e edição dos ficheiros XML de inicialização dos agentes, deve ser elaborada uma interface gráfica que apresente todas as funcionalidades e respectivos agentes, deveram ainda ser incluídas as propriedades genéricas e as limitações. De modo a que esta interface siga a mesma filosofia do restante trabalho, esta deverá suportar novas versões do documento XML, isto é a alteração, remoção ou adição de identificadores, assim a interface tem de ser a mais genérica possível.

Como se pode prever num sistema deste tipo, o agrupamento de módulos não se irá manter por um período de tempo ilimitado, podem ocorrer avarias nos módulos deixando-os indisponíveis, ou simplesmente existir um novo módulo que substitua vários. Neste sentido foram identificadas duas abordagens (Tabela 6-1) para a auto-reconfiguração das coligações, bem como a identificação do tipo de sistema em que deve de ser aplicada cada uma.

	Vantagens	Desvantagens
De tempo a tempo	Aumenta a eficiência, (isto porque quando tiver de ser utilizada está o mais actualizada possível)	Aumenta o número de interacções e mensagens na rede.
Antes da reutilização da coligação	Diminui o número de interacções e mensagens na rede	Requer mais tempo no processo inicial da coligação

Tabela 6-1: Vantagens e desvantagens na auto-reconfiguração das coligações.

A primeira abordagem (tempo a tempo), será a ideal para sistemas pequenos, isto é com poucos módulos, uma vez que o número de mensagens existentes na rede durante o seu funcionamento normal será reduzido, deixando assim uma maior largura de banda disponível para as mensagens de auto-reconfiguração das coligações.

No entanto esta abordagem também poderá ser utilizada num sistema de grandes dimensões sempre que se preveja um baixo número de interacções entre os diversos módulos.

A segunda abordagem será oposta à primeira, uma vez que o número de mensagens na rede durante o funcionamento normal do sistema será maior, devendo-se assim reduzir o número de auto-reconfigurações das coligações de modo a não sobrecarregar a rede.

Na interface gráfica do MRA, se este tiver mais que uma “skill”, deverá aparecer os diferentes workplaces, isto é um por cada funcionalidade apresentada. Ainda na interface gráfica do MRA, se este estiver reservado para uma coligação deverá ter a indicação do nome dessa coligação.

Capítulo 7. **Referências Bibliográficas**

Auyang, S. Y. (1998). Foundations of Complex-system Theories.

Barata, J. (2003). Coalition Based Approach For ShopFloor Agility. Electrical and Computer Science Engineering. Monte da Caparica, Universidade Nova de Lisboa. **PhD**.

Barata, J. and L. M. Camarinha-Matos (2003). "Coalitions of manufacturing components for shoop floor agility - The CoBASA architecture." pp. 26.

Barata, J., G. Cândido, et al. (2007). A multiagent-based control system applied to an educational shop floor. Information Technology For Balanced Manufacturing Systems, Springer Boston. **220/2006:** pp. 119-128.

Barata, J. and M. Onori (2006). Evolvable Assembly and Exploiting Emergent Behaviour. ISIE.

Barata, J., M. Onori, et al. (2007). Evolvable Production Systems: Enabling Research Domains. 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production. CARV. Toronto, Canada.

Barata, J., P. Santana, et al. (2006). Evolvable Assembly Systems: A Development RoadMap. IFAC.

Bellifemine, F., G. Caire, et al. (18-June-2007). "JADE programmer's guide." from http://jade.tilab.com.

Bellifemine, F., A. Poggi, et al. (1999). JADE - A FIPA-compliant agent framework. PAAM'99. London**:** 97-108.

Biron, P. V. and A. Malhotra. (2004). "XML Schema Part 2: Datatypes Second Edition." 2008, from http://www.w3.org/TR/xmlschema-2/.

Bray, T., J. Paoli, et al. (2008). "Extensible Markup Language (XML), Version 1.0." Fifth Edition. 2009, from http://www.w3.org/TR/REC-xml.

Brussel, H. V., J. Wyns, et al. (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers In Industry, special issue on intelligent manufacturing systems. **Vol. 37, No. 3:** pp. 255 - 276.

Érdi, P. (2007). Complexity Explained, Springer.

Ferreira, B., R. Frei, et al. (2009). Implementing Self-Managent for Evolvable Assembly Systems (under review). ICAC. Spain**:** pp. 10.

FIPA. (2002a). "FIPA Request Interaction Protocol Specification." 2008, from http://www.fipa.org/specs/fipa00026/SC00026H.html.

FIPA. (2002b). "FIPA Query Interaction Protocol Specification." 2008, from http://www.fipa.org/specs/fipa00027/SC00027H.html.

FIPA. (2002c). "FIPA Contract Net Interaction Protocol Specification." from http://www.fipa.org/specs/fipa00029/SC00029H.html.
FIPA. (2002d). "FIPA ACL Message Structure Specification." from http://www.fipa.org/specs/fipa00061/SC00061G.html.
Franklin, S. and A. Graesser (1996). Is It an agent, or just a program?: A taxonomy for autonomous agents Third International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag. **Vol. 1193/1997:** pp. 21-35.
Frei, R., J. Barata, et al. A Complexity Theory Approach to Evolvable Production Systems. ICINCO**:** pp. 10.
Frei, R., B. Ferreira, et al. (2008). Dynamic Coalitions for Self-Organizing Manufacturing Systems (in press). CIRP-ICME Naples, Italy**:** pp. 6.
Gruver, W. A., D. B. Kotak, et al. (2003). Holonic Manufacturing Systems: Phase II. HoloMAS**:** pp. 1-14.
Hall, K. H., R. J. Staron, et al. (2005). Experience with Holonic and Agent-Based Control Systems and Their Adoption by Industry. Holonic and Multi-Agent Systems for Manufacturing, Springer Berlin / Heidelberg. **3593/2005:** 1-10.
Hoda, E. (2006). "Flexible and reconfigurable manufacturing systems paradigms." International Journal of Flexible Manufacturing Systems **17(4)**(Reconfigurable Manufacturing Systems): 261-276.
Koren, Y., U. Heisel, et al. (1999). Reconfigurable Manufacturing Systems. Annals of the CIRP. **48(2):** pp. 527-540.
Maraldo, T., M. Onori, et al. (2006). Evolvable Assembly Systems: Clarifications & developmants to Date. International Workshop on Emergent Synthesis IWES. Kashiwa, Japan.
Mařík, V. and J. Lazansky (2004). Industrial applications of agent technologies. Control Engineering Practice, Special Issue on Manufacturing Plant Control: Challenges and Issues - INCOM. **15(11):** pp. 1364-1380

Mařík, V., P. Vrba, et al. (2005). Rockwell automation agents for manufacturing. International Conference on Autonomous Agents. Netherlands**:** pp 107 - 113
Mehrabi, M. G., A. G. Ulsoy, et al. (April 2002). "Trends and perspectives in flexible and reconfigurable manufacturing systems " Journal of Intelligent Manufacturing **13(2)**: pp. 135-146.
Monostori, L., J. Váncza, et al. (2006). "Agent-Based Systems for Manufacturing." Annals of the CIRP **Vol. 55/2/2006**: pp. 697-720.
Nwana, H. S. (1996). Software Agents: An Overview. K. E. Review. **Vol. 11, No 3:** pp. 205-244.
Onori, M., J. Barata, et al. (2006). Evolvable Assembly Systems - Basic Principles. BASYS.
Pěchouček, M., V. Mařík, et al. (2000). Coalition Formation in Manufacturing Multi-Agent Systems. 11th International Workshop on Database and Expert Systems Applications, London, UK.
Raggett, D., A. L. Hors, et al. (24 December 1999). "HTML 4.01 Specification." 2009, from http://www.w3.org/TR/1999/REC-html401-19991224.
Ribeiro, L., J. Barata, et al. (2008). An Architecture for a Fault Tolerant Highly Reconfigurable Shop Floor. INDIN Daejeon, Korea**:** pp. 1194-1199.
Ribeiro, L., J. Barata, et al. (2008). OWL Ontology to support Evolvable Assembly Systems. IMS'08**:** pp. 6.

Sandholm, T. W. and V. R. Lesser (1997). "Coalitions among Computationally Bounded Agents." Artificial Intelligence, Special Issue on Economic Principles of Multi-Agent Systems **Vol: 94, Num: 1**: pp. 99 - 137.

Serugendo, G. D. M., M.-P. Gleizes, et al. (2005a). "Self-Organisation in MAs." The Knowledge Engineering Review **Vol. 00:0, 1-24**.

Serugendo, G. D. M., M.-P. Gleizes, et al. (2005b). "Self-Organisation and Emergence in MAS: An Overview." Informatica **30**: pp. 10.

Valckenaers, P., H. V. Brussel, et al. (2005). Holonic Manufacturing Execution Systems. CIRP Annals - Manufacturing Technology. **Volume 54, Issue 1:** pp. 427-432.

W3C. (2004). "XML Schema." Second Edition. 2009, from http://www.w3.org/TR/xmlschema-0/.