PLATAFORMA DE NEGOCIAÇÃO DE COMPONENTES MULTIMÉDIA

Luís Miguel Geada Ventura de Sousa

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
2012

Este relatório satisfaz, parcialmente, os requisitos que constam da Unidade Curricular de Tese/Dissertação do 2º ano do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Luís Miguel Geada Ventura de Sousa, N.º 1900503, 1900503@isep.ipp.pt
Orientação científica: Professora Doutora Maria Benedita Campos Neves Malheiro, mbm@isep.ipp.pt

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
23 de Novembro de 2012

À minha Família.

Agradecimentos

Queria agradecer à minha esposa Teresa, pelo seu amor, carinho, paciência e generosidade por sempre me apoiar e incentivar em todas as fases deste projecto.

Aos meus pais Alfredo e Emília, irmãos Fernando, Paulo e Mariana, cunhados Dalila, João, e Alexandre, sobrinhos Rodrigo, Sofia, Gil, André, Nuno e Susana pelas reuniões semanais de família e pelo relacionamento fantástico e clima de paz e afecto que sempre proporcionam.

À minha orientadora Professora Doutora Benedita Malheiro pela sua disponibilidade incondicional, determinação e partilha de conhecimentos o meu muito obrigado.

Por fim a todos os meus amigos que com a sua presença complementam a minha felicidade.

"*Almost everything you do will seem insignificant, but it is important that you do it*"

Mahatma Gandhi

Resumo

O objectivo da tese é demonstrar a adequação do paradigma dos mercados electrónicos baseados em agentes para transaccionar objectos multimédia em função do perfil dos espectadores. Esta dissertação descreve o projecto realizado no âmbito da plataforma de personalização de conteúdos em construção. O domínio de aplicação adoptado foi a personalização dos intervalos publicitários difundidos pelos distribuidores de conteúdos multimédia, *i.e.*, pretende-se gerar em tempo útil o alinhamento de anúncios publicitários que melhor se adeqúe ao perfil de um espectador ou de um grupo de espectadores.

O projecto focou-se no estudo e selecção das tecnologias de suporte, na concepção da arquitectura e no desenvolvimento de um protótipo que permitisse realizar diversas experiências nomeadamente com diferentes estratégias e tipos de mercado.

A arquitectura proposta para a plataforma consiste num sistema multiagente organizado em três camadas que disponibiliza interfaces do tipo serviço *Web* com o exterior. A camada de topo é constituída por agentes de interface com o exterior. Na camada intermédia encontram-se os agentes autónomos que modelam as entidades produtoras e consumidoras de componentes multimédia assim como a entidade reguladora do mercado. Estes agentes registam-se num serviço de registo próprio onde especificam os componentes multimédia que pretendem negociar. Na camada inferior realiza-se o mercado que é constituído por agentes delegados dos agentes da camada superior. O lançamento do mercado é efectuado através de uma interface e consiste na escolha do tipo de mercado e no tipo de itens a negociar.

Este projecto centrou-se na realização da camada do mercado e da parte da camada intermédia de apoio às actividades de negociação no mercado. A negociação é efectuada em relação ao preço da transmissão do anúncio no intervalo em preenchimento. Foram implementados diferentes perfis de negociação com tácticas, incrementos e limites de variação de preço distintos. Em termos de protocolos de negociação, adoptou-se uma variante do *Iterated Contract Net* – o *Fixed Iterated Contract Net*. O protótipo resultante foi testado e depurado com sucesso.

Palavras-Chave

Serviços Web (XML, WSDL, SOAP, UDDI), JADE, FIPA ACL e Contract Net, Sistemas Multiagente, Ontologias.

Abstract

The aim of this MSc. thesis is to demonstrate the suitability of the agent-based paradigm to select media objects according to the profile of the viewers by means of a multiagent negotiation platform. The application domain is the personalisation of the programme intervals transmitted by media distributors, *i.e.*, the goal is to generate a timely alignment of the advertisements that best fits the profile of a viewer or a group of viewers for a programme interval.

The project involved the selection of the supporting technologies, the design of the platform architecture and the development of a prototype to perform experiments with different types of negotiation protocols and tactics.

The proposed architecture consists of a multi-agent system organized in three layers. The top layer – the interface layer – contains agents that interface with the external producers and distributors by exposing Web services. The intermediate layer – the enterprise layer – holds a collection of autonomous agents that model the producers and distributors of media objects as well as the market regulator. These agents register in a service registry to publish the multimedia components they are willing to trade. The lower layer – the market layer – consists of delegated agents of the enterprise layer agents that are dedicated to negotiate individual media objects, *i.e.*, buy and sell timeslots of programme intervals. The user specifies the type of market and the items to be negotiated through a dedicated interface.

This project focused on the development of the market layer as well as on the functionalities of the intermediate layer required to support the trading activities of the market layer. The market negotiation, which is carried around the price of the interval timeslots, is conducted by the delegate agents using different negotiation profiles, *i.e.*, price constraints and price adaption tactics. In terms of negotiation protocols, the market implements variant of the Iterated Contract Net negotiation protocol – the Fixed Iterated Contract Net. The resulting prototype was successfully tested and debugged.

Keywords

Web Services (XML, WSDL, SOAP, UDDI), JADE, and FIPA ACL Contract Net, Multiagent Systems, Ontologies.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Excertos de Código

Acrónimos

.NET	–	*dot Net*
ACL	–	*Agent Communication Language*
AMS	–	*Agent Management System*
API	–	*Application Programming Interface*
B2B	–	*Business-to-Business*
B2C	–	*Business-to-Consumer*
BDI	–	*Belief-Desire-Intention*
CDR	–	*Common Data Representation*
CLSID		*CLasS ID*
CMML	–	*Continuous Media Markup Language*
COM	–	*Component Object Model*
COMM	–	*Core Ontology for Multimedia*
CORBA	–	*Common Object Request Broker Arquitecture*
CRM	–	*Content Reference Model*
CSIRO	–	Commonwealth Scientific and Industrial Research Organisation
DAML	–	DARPA *Agent Markup Language*
DARPA	–	Defense Advanced Research Projects Agency
DAWG	–	RDF Data Access Working Group

DCE/RPC – *Distributed Computing Environment/Remote Procedure Calls*

DCMI – *Dublin Core Metadata Initiative*

DCOM – *Distributed Component Object Model*

DDL – *Description Definition Language*

DEE – Departamento de Engenharia Electrotécnica

DF – *Directory Facilitator*

DOLCE – *Description Ontology for Linguistic and Cognitive Engineering*

DOM – *Document Object Model*

DS-MIRF – *Domain-Specific Multimedia Indexing, Retrieval and Filtering*

DTD – *Document Type Definition*

ESWC – European Semantic Web Conference

FIPA – Foundation for Intelligent Physical Agents

FIPA OS – FIPA *Open Source*

FOAF – *Friend Of A Friend*

GIOP – *General Inter*-ORB *Protocol*

GUI – *Graphical User Interface*

HTML – *HyperText Markup Language*

HTTP – *HyperText Transfer Protocol*

IaaS – *Infrastructure-as-a-Service*

IDE – *Integrated Development Environment*

IDL – *Interface Definition Language*

IEC – International Electrotechnical Commission

IIOP – *Internet Inter*-ORB *Protocol*

IIOP – *Internet Inter-Object Request Broker Protocol*

ISEP – Instituto Superior de Engenharia do Porto

ISO – International Organization for Standardization

ISWC – International Semantic Web Conference

iTV – *Interactive Television*

JADE – *Java Agent DEvelopment Framework*

JESS – *Java Expert System Shell*

KB – *Knowledge Base*

KIF – *Knowledge Interchange Format*

KML – *Keyhole Markup Language*

KOS – *Knowledge Organization System*

KQML – *Knowledge Query Manipulation Language*

KSE – *Knowledge Sharing Effort*

MDS – *Multimedia Description Schemes*

MPEG-4 – ISO/IEC *international standard 14496*

MPEG-7 – *Multimedia Content Description Interface*

MTS – *Message Transport Service*

OIL – *Ontology Inference Layer*

OMG – Object Management Group

ORB – *Object Request Broker*

OSF – Open Source Foundation

OWL – *Web Ontology Language*

PaaS – *Platform-as-a-Service*

PDA – *Personal Digital Assistants*

PDF – *Portable Data Format*

PHP – PHP *Hypertext Preprocessor*

RDF – *Resource Description Framework*

RDFDB – RDF *Database*

RDQL – RDF *Data Query Language*

REST – *Representational State Transfer*

RMI – *Remote Method Invocation*

RPC – *Remote Procedure Calls*

SaaS – *Software-as-a-Service*

SACI – *Simple Agent Communication Infrastructure*

SCM – *Service Control Manager*

SeRQL – *Sesame* RDF *Query Language*

SGML – *Standard Generalized Markup Language*

SKOS – *Simple Knowledge Organization System*

SL – FIPA *Semantic Language*

SMA – Sistemas MultiAgente

SMIL – *Synchronized Multimedia Integration Language*

SOAP – *Simple Object Access Protocol*

SPARQL – SPARQL *Protocol and* RDF *Query Language*

SWC Semantic Web Conference Ontology

SWDWG – Semantic Web Deployment Working Group

TCP/IP – *Transmission Control Protocol/Internet Protocol*

UDDI – *Universal Description Discovery and Integration*

URI – *Universal Resource Identifier*

URL – *Uniform Resource Locator,*

VoD – *Video on Demand*

WAR – *Web Application Archive*

WS – *Web Services*

WSDC – *Web Service Dynamic Client*

WSDL – *Web Service Description Language*

WSIG – *Web Service Integration Gateway*

WWW – World Wide Web

XHTML – *eXtensible* HTML

XM – *eXperimental Model*

XML – *eXtensible Markup Language*

XPath – XML *Path Language*

XQuery – XML *Query Language*

XSD – XML *Schema Definition*

XSL – XML *Stylesheet Language*

XSLT – XSL *Transformations*

1. INTRODUÇÃO

Na última década tem-se assistido a uma evolução rápida e constante da capacidade, velocidade e flexibilidade das infra-estruturas de comunicação. A Internet afirmou-se como um suporte fiável e economicamente interessante para a interligação de organizações e pessoas. É o caso do *e-Banking*, que proporciona às instituições financeiras uma enorme economia de recursos humanos e aumento da disponibilidade de serviço, do *e-Government*, que permite ao Estado disponibilizar vários serviços aos cidadãos de forma cómoda, rápida e confortável, e do *e-Business*, que permite às empresas vender e publicitar produtos e serviços de forma rápida, fiável e barata.

Para dar resposta a estas solicitações, têm vindo a ser desenvolvidas especificações e tecnologias destinadas a disponibilizar e prestar serviços electrónicos assim como ambientes de execução de agentes de *software* capazes de negociar automaticamente produtos e serviços. É nesta abordagem que se insere este trabalho.

1.1. CONTEXTO

Para negociar bens ou serviços através da Internet é necessário, por um lado, que o comprador encontre com facilidade e a bom preço o que procura e, por outro lado, que o vendedor afixe e publicite os seus produtos da forma o mais apelativa possível em locais com boa visibilidade e disseminação entre os potenciais interessados.

Apesar de, actualmente, os motores de busca como o Google ou Yahoo terem atingido um grau de sofisticação e eficiência bastante elevado [94], no que diz respeito à procura de componentes multimédia como imagens, vídeos e áudios encontra-se ainda num estado bastante incipiente dada a complexidade e grau de subjectividade envolvidos. Tomemos como exemplo a imagem da Figura 1. Este objecto multimédia, que representa uma garrafa, se estiver inserido junto a um ecoponto, indica genericamente que se trata de um depósito de vidro para reciclagem. No entanto, noutro contexto, pode representar uma preciosa garrafa de um coleccionador de vinhos. Isto quer dizer que uma mesma imagem poderá ter várias leituras, dependendo das motivações do observador, do objectivo a que se destina e do contexto em que se insere.

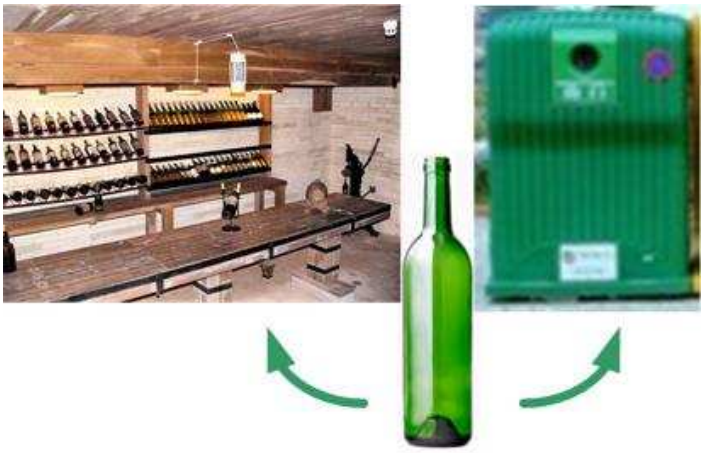


Figura 1 Componentes Multimédia

Para ajudar a resolver esta tarefa de elevada complexidade, têm sido adoptadas diversas abordagens. Ao nível da representação de conhecimento, foram propostas inúmeras classificações, taxonomias e ontologias de descrição de conteúdos multimédia [92]. Ao nível da anotação de componentes multimédia, têm sido desenvolvidas múltiplas ferramentas semi-automáticas de descrição/anotação, estando-se, contudo, longe de se alcançar um mecanismo satisfatório de classificação automática de objectos multimédia. Ao nível da escolha automática de produtos ou serviços, diversas plataformas de intermediação e negociação automática têm sido propostas, nomeadamente, plataformas de negociação baseadas em agentes.

Com a introdução de novas tecnologias como o *Interactive TV* (iTV) alterou-se o paradigma do relacionamento entre os consumidores e publicidade, que passou a ser mais

individualizado e focado nos interesses, necessidades e receptividade do espectador [2]. No entanto, o sucesso deste novo paradigma depende de vários factores [90]:

- Recolha dos elementos que possam caracterizar os gostos e desejos do consumidor assim como dados circunstanciais (hora do dia, dia da semana, época, estado do tempo, local, *etc*.) em que a programação está a ser visionada;
- Caracterização dos componentes multimédia (anúncios e programas);
- Existência de um sistema de integração dos componentes multimédia (anúncios) nos intervalos dos programas que o consumidor se encontre a visualizar;
- Capacidade de encontrar parceiros de negócio aptos a negociar automaticamente anúncios que contenham as características apropriadas ao preenchimento de intervalos de programação;

1.2. MOTIVAÇÃO

O tema desta tese enquadra-se num projecto mais abrangente que está a ser desenvolvido em parceria entre o ISEP e a Birmingham City University com o objectivo de definir e testar um modelo de negócio virtual de componentes multimédia baseado em agentes de *software* [1] [2]. A Birmingham City University é responsável pelo estudo e desenvolvimento de ferramentas de codificação/descodificação de vídeos MPEG-4 e a respectiva inserção/extracção de componentes multimédia de acordo com os metadados armazenados no formato *Multimedia Content Description Interface* (MPEG-7). O Instituto Superior de Engenharia do Porto (ISEP), por seu lado, é responsável pelo estudo, especificação e teste de um modelo de mercado virtual baseado em agentes para a selecção e transacção dos componentes multimédia a inserir de forma automática. Durante a definição da solução, concluiu-se que o tema era demasiado vasto para se enquadrar numa única tese e que, dadas a arquitectura e tecnologias adoptadas, a própria solução se prestava a uma divisão de tarefas. Assim, a parte do projecto a cargo do ISEP, foi dividida em três propostas de tese.

1.3. OBJECTIVOS

Pretende-se com esta tese estudar, propor e realizar um modelo distribuído de negócio de componentes multimédia descritos no formato MPEG-7, usando para o efeito um ambiente virtual onde concorrem agentes de *software*. Do ponto de vista da plataforma global de transacção de componentes multimédia, este projecto foca-se no desenvolvimento da

terceira camada – camada de mercado – e das funcionalidades da camada intermédia necessárias ao funcionamento do mercado.

A prossecução do trabalho foi organizada em três fases:

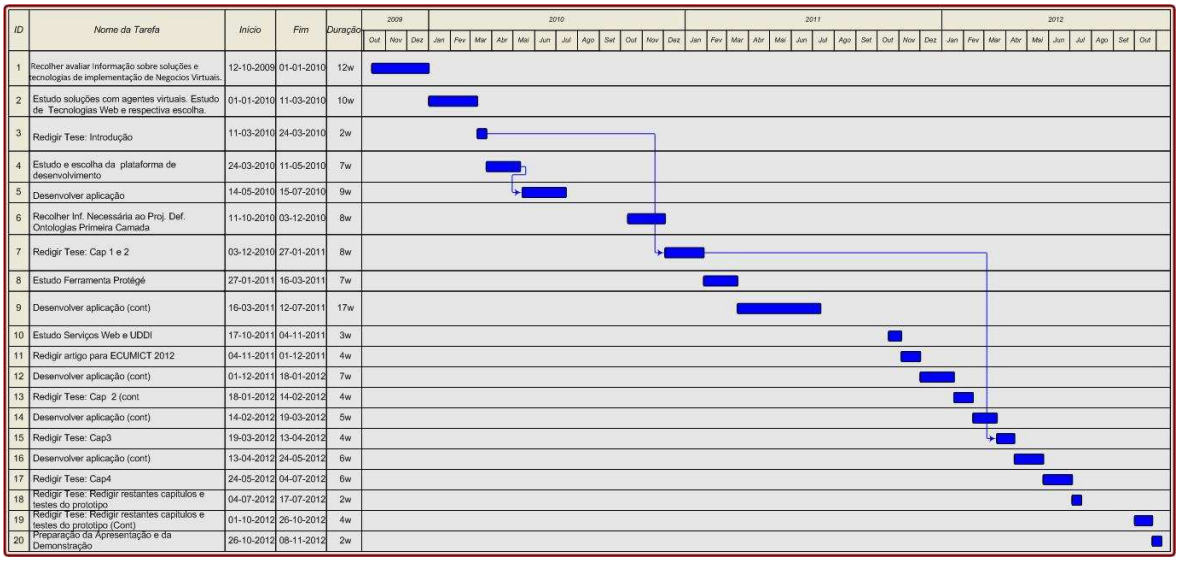
(i) Estudo e levantamento do estado da arte da transacção electrónica de componentes multimédia e na familiarização com as tecnologias de *software* envolvidas nos negócios virtuais entre empresas – *business-to-business* (B2B);

(ii) A especificação das características do protótipo e a escolha das linguagens, tecnologias e ferramentas de desenvolvimento a adoptar;

(iii) O desenvolvimento, teste e depuração do protótipo.

1.4. CALENDARIZAÇÃO

Este projecto foi organizado através de um conjunto de tarefas calendarizadas de acordo com o cronograma que se apresenta na Figura 2.



ID	Nome da Tarefa	Início	Fim	Duração
1	Recolher avaliar informação sobre soluções e tecnologias de implementação de Negocios Virtuais.	12-10-2009	01-01-2010	12w
2	Estudo soluções com agentes virtuais. Estudo de Tecnologias Web e respectiva escolha.	01-01-2010	11-03-2010	10w
3	Redigir Tese: Introdução	11-03-2010	24-03-2010	2w
4	Estudo e escolha da plataforma de desenvolvimento	24-03-2010	11-05-2010	7w
5	Desenvolver aplicação	14-05-2010	15-07-2010	9w
6	Recolher Inf. Necessária ao Proj. Def. Ontologias Primeira Camada	11-10-2010	03-12-2010	8w
7	Redigir Tese: Cap 1 e 2	03-12-2010	27-01-2011	8w
8	Estudo Ferramenta Protégé	27-01-2011	16-03-2011	7w
9	Desenvolver aplicação (cont)	16-03-2011	12-07-2011	17w
10	Estudo Serviços Web e UDDI	17-10-2011	04-11-2011	3w
11	Redigir artigo para ECUMICT 2012	04-11-2011	01-12-2011	4w
12	Desenvolver aplicação (cont)	01-12-2011	18-01-2012	7w
13	Redigir Tese: Cap 2 (cont	18-01-2012	14-02-2012	4w
14	Desenvolver aplicação (cont)	14-02-2012	19-03-2012	5w
15	Redigir Tese: Cap3	19-03-2012	13-04-2012	4w
16	Desenvolver aplicação (cont)	13-04-2012	24-05-2012	6w
17	Redigir Tese: Cap4	24-05-2012	04-07-2012	6w
18	Redigir Tese: Redigir restantes capitulos e testes do protótipo	04-07-2012	17-07-2012	2w
19	Redigir Tese: Redigir restantes capitulos e testes do protótipo (Cont)	01-10-2012	26-10-2012	4w
20	Preparação da Apresentação e da Demonstração	26-10-2012	08-11-2012	2w

Figura 2 Calendarização

1.5. ORGANIZAÇÃO DO RELATÓRIO

Esta dissertação está organizada em seis capítulos:

- **Capitulo 1** – Efectua a introdução ao projecto e a toda a sua envolvente, incluindo o problema, o contexto, os objectivos e a organização desta dissertação;

- **Capitulo 2** – Apresenta o estado da arte das principais tecnologias que vão ser utilizadas no projecto e que se perspectiva que venham a ser usadas na solução;
- **Capitulo 3** – Apresenta as ferramentas e instalação do ambiente de desenvolvimento;
- **Capitulo 4** – Descreve o desenvolvimento do protótipo
- **Capitulo 5** – Detalha os testes efectuados e os resultados obtidos;
- **Capitulo 6** – Retira as principais conclusões e sugere desenvolvimentos futuros;

Cada capítulo possui um sumário inicial e termina com uma breve conclusão.

2. ESTADO DA ARTE

Neste capítulo apresenta-se o estado da arte no contexto da modelação de mercados virtuais e respectivas tecnologias de suporte.

2.1. NEGÓCIOS E MERCADOS VIRTUAIS

A disponibilização por parte dos operadores de telecomunicações de infra-estruturas de comunicação rápidas e eficientes associadas à ubiquidade e massificação do acesso à rede mudou o paradigma das transacções comerciais e do relacionamento institucional no mundo dos negócios. É o caso, por exemplo, das operações de Bolsa que, hoje em dia, podem ser realizadas em tempo quase real a partir de qualquer local através de portais especializados que permitem consultar a informação das empresas e, de seguida, comprar e vender acções. Outro exemplo consiste na reserva e compra electrónica de bilhetes de autocarro, comboio ou avião, incluindo a escolha do lugar e da classe pretendida, sem se sair de casa. Por outro lado, a evolução tecnológica permitiu a personalização dos produtos, transformando o *marketing* tradicional baseado na segmentação do mercado num *marketing* especializado e individualizado [22].

Uma vez que a competitividade é global, a volatilidade de preços nos negócios virtuais tende a aumentar. O equilíbrio entre a oferta e procura sofre ajustes quase instantâneos e as

margens de lucro tendem a diminuir, fazendo da rapidez e eficiência das transacções um elemento crucial neste tipo de mercados.

É também importante investir no aspecto gráfico dos portais, na qualidade da informação que disponibilizam assim como na facilidade e simplicidade de preenchimento dos formulários, uma vez que o relacionamento vendedor – cliente neste domínio é obviamente impessoal. No entanto, essa interacção electrónica permite aos fornecedores recolher de forma automática informação dos potenciais clientes potenciando a personalização dos produtos oferecidos [21].

Por último, um dos aspectos essenciais do desenvolvimento dos mercados virtuais é a fiabilidade das transacções realizadas [6], *i.e.*, para além da confiança subjectiva que o serviço possa inspirar ao consumidor, as transacções têm de decorrer de acordo com os preceitos técnicos e legais apropriados.

2.1.1. B2B E B2C

Existem dois segmentos distintos no mundo dos mercados virtuais: o B2B e o *Business-to-Consumer* (B2C). Constituindo ambos áreas do negócio electrónico, possuem objectivos e estilos distintos.

No B2C o principal objectivo é angariação de compradores (clientes finais) da forma mais agressiva e consistente possível [23] [24]. Recorre extensivamente a técnicas de *merchandising*, *e.g.*, com recurso a mostradores publicitários, à construção de montras de loja, *etc.*, no sentido de seduzir o público-alvo a comprar o produto e a campanhas de comercialização, *e.g.*, campanhas de descontos ou de atribuição de vales (*vouchers*) de curta duração, dado que se pretende captar rapidamente o interesse do consumidor. Neste tipo de negócio o objecto a transaccionar é um bem ou um serviço final.

No B2B, apesar do objectivo principal ser também a construção de uma carteira sólida de clientes, o processo tem mais etapas e envolve vários responsáveis, sendo necessariamente mais longo [23] [24]. O *marketing* é usado para construir relações que poderão ser reforçadas durante os ciclos de negócio. É um processo cooperativo baseado na confiança mútua que necessita de informação técnica detalhada dos produtos a transaccionar. O objectivo é proceder à contratação externa de produtos ou serviços, *i.e.*, *outsourcing*, de forma a tornar a empresa mais lucrativa e competitiva.

A marca é importante em ambos os tipos de mercado [92], mas por diferentes razões. Em B2C, uma marca forte poderá encorajar o consumidor a comprar, manter a sua lealdade e pagar um preço potencialmente mais elevado. Nos mercados B2B, a marca ajuda a credibilizar a empresa, mas não significa por si só que venha a ser escolhida.

2.2. INTEROPERABILIDADE

No âmbito dos sistemas distribuídos a interoperabilidade entre módulos é um aspecto essencial. A primeira especificação que teve por objectivo assegurar a interoperabilidade entre módulos distribuídos heterogéneos foi o *Common Object Request Broker Arquitecture* (CORBA) e foi seguida, um pouco mais tarde pelo *Distributed Component Object Model* (DCOM) desenvolvido pela Microsoft. Até então os ambientes de computação distribuída eram do tipo *Remote Procedure Calls* (RPC). Actualmente as tecnologias associadas à especificação de serviços *Web* permitem a comunicação entre aplicações instaladas em diferentes plataformas e também a descoberta e a interacção automática de serviços no contexto global da *Web*.

A questão da interoperabilidade coloca-se actualmente ainda com mais acuidade dado o mais recente paradigma da computação distribuída: a computação em nuvem – *cloud computing*. A computação em nuvem obriga todos os provedores a sérios esforços de padronização ao nível das interfaces que a nuvem oferece para acesso às infra-estruturas – *Infrastructure-as-a-Service* (IaaS), plataformas – *Platform-as-a-Service* (PaaS) – ou aplicações – *Software-as-a-Service* (SaaS). A computação em nuvem adopta frequentemente interfaces do tipo serviço *Web*.

2.2.1. DCE/RPC

A especificação *Distributed Computing Environment/Remote Procedure Calls* (DCE/RPC) foi definido pela Open Source Foundation (OSF) e faz parte de um conjunto de tecnologias para executar procedimentos de aplicações remotas a partir de computadores ligados em rede. Esta tecnologia cliente – servidor é inspirada no conceito convencional de chamada de procedimentos dentro da mesma aplicação/máquina [100]. Quando uma aplicação realiza um RPC, passa os argumentos ao procedimento remoto e fica a aguardar pelos resultados [95]. Assim, a DCE/RPC define um formato normalizado de troca de argumentos entre o servidor e o cliente. Cada procedimento é identificado através de um

conjunto de três elementos: identificação do programa, versão e identificação do procedimento.

2.2.2. RMI

A metodologia Java *Remote Method Invocation* (RMI) sucedeu ao RPC e permite a invocação de métodos de objectos localizados numa máquina remota. Tipicamente, é necessário um servidor que cria os objectos e gera as respectivas referências, registando-as no serviço de nomes para que esses objectos fiquem acessíveis remotamente. Os clientes que pretendam invocar os métodos acessíveis para esses objectos deverão, primeiro, obter a respectiva referência procurando-a no serviço de nomes. A forma como o servidor e cliente comunicam é totalmente transparente para o programador que faz as invocações remotas de forma semelhante a invocações regulares [3]. Essa transparência é conseguida através dos adaptadores *skeleton* (lado servidor) e *stub* (lado cliente) que são criados aquando da criação do objecto/método e que servem para empacotar e desempacotar as mensagens recebidas da rede e associar a mensagem à respectiva operação de invocação remota – ver Figura 3.

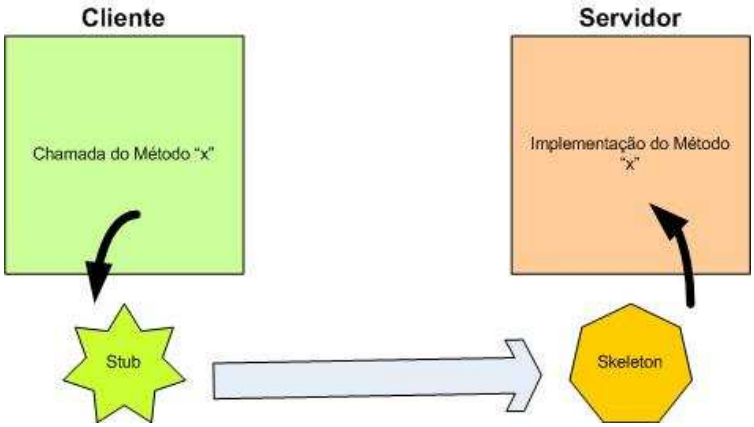


Figura 3 Arquitectura RMI

O RMI apresenta vantagens como a portabilidade entre plataformas, *i.e.*, o código poder ser executado nas JVM remotas, e a possibilidade de se adaptar aos sistemas existentes. Por outro lado, tem a desvantagem de obrigar as plataformas a suportarem a tecnologia Java, apresenta alguns problemas de segurança e não oferece suporte para sistemas proprietários [101].

2.2.3. CORBA

CORBA é o acrónimo de *Common Object Request Broker Architecture* que é uma norma desenvolvida no início dos anos 90 pelo Object Management Group (OMG). Surge da necessidade de assegurar a interoperabilidade entre aplicações, independentemente dos sistemas operativos, das linguagens de programação ou das redes onde se encontram instaladas [5] [99]. O CORBA especifica o *Object Request Broker* (ORB) que permite às aplicações comunicarem entre si [4]. A partir de 1994, com o CORBA 2.0, a interoperabilidade entre ambientes heterogéneos foi alcançada e, consequentemente, passou a permitir a sua utilização na Internet, fazendo uso do protocolo padrão *Internet Inter-ORB Protocol* (IIOP) que é gerido pelos ORB de forma transparente para o programador [25]. A interacção com o IIOP é feita de forma indirecta usando um protocolo abstracto chamado *General Inter-ORB Protocol* (GIOP). O GIOP é constituído por um conjunto de regras de formatação de dados – *Common Data Representation* (CDR) – compatíveis com os dados suportados pela *Interface Definition Language* (IDL) do CORBA e por um conjunto de mensagens tipo que suportam toda a semântica ORB definida na especificação do CORBA. Apesar das mensagens GIOP poderem ser virtualmente enviadas sobre qualquer protocolo de transporte (TCP/IP, Novell SPX, *etc.*), para se assegurar a interoperabilidade entre ORB distintos no contexto da Internet, é necessário que o protocolo de transporte seja o *Transmission Control Protocol/Internet Protocol* (TCP/IP), uma vez que é este o protocolo usado pela rede. A utilização do CORBA tem-se resumido à interligação de plataformas proprietárias ou aplicações legadas e é uma tecnologia bastante complexa e pesada em termos de recursos [97]. Por estas razões não será utilizada no nosso projecto.

2.2.4. DCOM

O DCOM é uma tecnologia desenvolvida pela Microsoft bastante semelhante ao CORBA [98] destinada ao desenvolvimento de aplicações distribuídas. Trata-se de um ambiente de computação distribuída – *Distributed Computing Environment* (DCE) – que usa RPC como uma extensão do *Component Object Model* (COM) para permitir a criação de objectos que residem no servidor remoto [102]. As classes COM dentro do servidor são representadas por um identificador único designado *CLasS ID* (CLSID) que as máquinas cliente devem conhecer para além do nome do servidor. Essa informação reside do lado do cliente nas bibliotecas COM que, quando pretende criar um objecto na máquina remota (servidor),

consulta a sua biblioteca COM e solicita a criação via o *Service Control Manager* (SCM) que existe de ambos os lados. Devido a problemas de compatibilidade, sempre que um componente DCOM do lado do servidor é actualizado, a informação desse componente é alterada, sendo necessário propagar essa informação por todas as máquinas de cliente. Por outro lado, apresenta também alguns problemas com a utilização de *firewalls*. O DCOM é apenas utilizável em plataformas Microsoft, é de difícil implementação e requer bastante manutenção manual [26].

2.2.5. MICROSOFT .NET FRAMEWORK

A tecnologia. *dot Net FrameWork* (.NET) surgiu da necessidade dos produtos Microsoft se adaptarem às necessidades do mercado dos ambientes distribuídos. O objectivo desta tecnologia é suportar a comunicação e expor serviços num contexto multiplataforma. Assim, o .NET adopta as tecnologias padrão dos serviços *Web* possibilitando a utilização multiplataforma [103]. É possível definir ligações sobre *HyperText Transfer Protocol* (HTTP) e TCP. No caso do HTTP, a informação circula em formato SOAP, permitindo ao cliente invocar métodos de objectos remotos que não utilizam necessariamente a *framework* .NET, *i.e.*, permite disponibilizar e consumir serviços *Web*. No caso do TCP, a informação é seriada e circula no formato de sequências de bits. Os formatadores permitem a codificação e seriação dos dados do lado do emissor e o processo inverso do lado do receptor. Na tecnologia .NET encontram-se disponíveis duas soluções distintas para desenvolver aplicações distribuídas: (*i*) a .NET *Remoting*; e (*ii*) a .NET *Web Services* [27] [29].

A .NET Remoting encontra-se optimizada para aplicações que usam a tecnologia .NET. É a sucessora do DCOM e pretende ultrapassar as principais limitações que este último apresentava. Permite a execução assíncrona das aplicações, *i.e.*, permite fazer chamadas assíncronas a métodos de classes sem ter de esperar que o método responda, diminuindo desta forma os tempos de execução da aplicação. Suporta uma grande quantidade de protocolos e, quando associada ao protocolo TCP, apresenta um desempenho e eficiência acrescidos, sendo indicada para a transferência de elevada quantidade de dados.

A solução .NET Web Services permite interagir com interfaces do tipo serviço *Web*. A informação circula encapsulada em mensagens do protocolo *Simple Object Access Protocol* (SOAP). Apesar da transferência de dados ser mais lenta, o SOAP é um protocolo

aberto, permitindo a comunicação com organizações externas ou que não usam a tecnologia. NET. Na secção seguinte efectua-se uma descrição mais detalhada da tecnologia das interfaces do tipo serviço *Web*.

As tecnologias DCOM e .NET não são interessantes no âmbito dos objectivos propostos por serem proprietárias.

2.2.6. SERVIÇOS *WEB*

Os serviços *Web* – *Web Services* (WS) – foram desenvolvidos para assegurar uma maior interoperabilidade entre sistemas e organizações e, consequentemente, permitir a troca informação de uma forma mais rápida, cómoda, flexível e transparente. Essa necessidade de interoperabilidade entre sistemas que disponibilizam serviços através da *Web* obriga a um esforço de normalização ao nível da comunicação. O W3C [29] e a OASIS [30] definiram a especificação do serviço *Web* enquanto a *Apache Software Foundation* desenvolvia tecnologias do lado do servidor que suportam os serviços *Web* (Axis2). Este esforço de padronização foi apoiado por grandes empresas como a IBM e a Microsoft, contribuindo fortemente para a afirmação e consolidação desta tecnologia no mercado.

As empresas são, quer na vertente B2B, quer na vertente B2C, as grandes beneficiárias desta nova abordagem porque lhes permite publicitar e disponibilizar os seus produtos e serviços a um conjunto cada vez mais alargado de clientes. Permite procurar, encontrar e encomendar produtos e serviços de forma automática, consultar o estado do tempo de qualquer parte do globo, efectuar (quase) qualquer tipo de operação bancária, utilizar recursos multimédia, *etc.* Neste cenário, a quantidade e qualidade das oportunidades de negócio multiplicam-se. Assim, a tecnologia dos serviços *Web* veio permitir a criação de serviços que integram serviços de terceiros de forma transparente. O conhecimento já não reside apenas num ponto, mas encontra-se geograficamente distribuído e disponível para utilização a partir de qualquer local.

Os serviços *Web* usam extensivamente a *eXtensible Markup Language* (XML) para: (*i*) a descrição do serviço – *Web Service Description Language* (WSDL) [31]; (*ii*) a representação de dados; e (*iii*) a troca de dados de forma estruturada – SOAP [32]. Esta opção é natural face ao requisito primordial dos serviços *Web* de assegurar a interoperabilidade entre sistemas. Para o transporte dos dados utiliza-se frequentemente, apesar de não ser obrigatório porque a norma não especifica o protocolo de transporte, o

HTTP. Os dados são então transferidos no formato XML – encapsulados em SOAP – e enviados tipicamente em mensagens HTTP. A Figura 4 representa a arquitectura dos serviços *Web*.

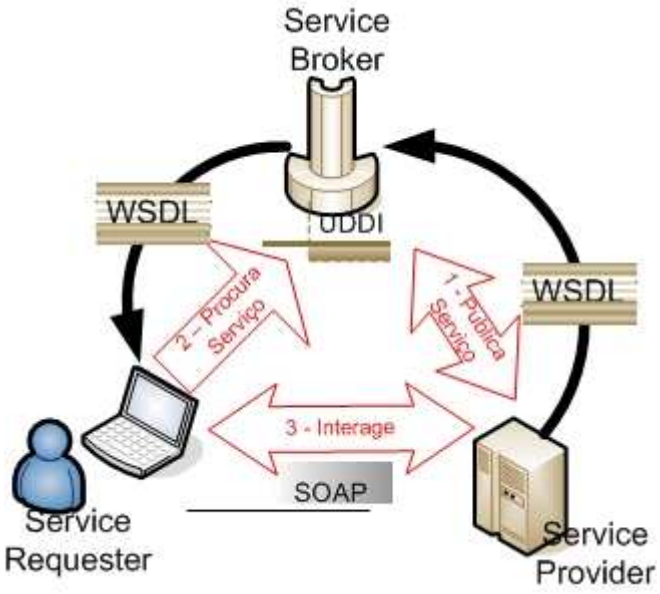


Figura 4 Arquitectura dos Serviços *Web*

Existem na *Web* serviços destinados à publicação de descrições de serviços *Web* – serviços de registo *Universal Description Discovery and Integration* (UDDI) [33] – que mantêm a informação relevante sobre os provedores, as implementações e os metadados dos serviços registados. O potencial utilizador recorre ao serviço de registo UDDI para encontrar o serviço desejado. Caso o serviço pretendido esteja lá registado, descarrega a respectiva descrição em WSDL que especifica como interagir (tipo de interfaces, métodos disponibilizados e parâmetros de entrada) com o serviço.

Na prática, quando um provedor – *service provider* – pretende disponibilizar um serviço *Web* a terceiros (pago ou gratuito), publica a sua descrição WSDL num serviço de registo UDDI. O utilizador procura o serviço no UDDI, descarrega e processa a sua descrição WSDL e interage com o serviço usando o protocolo SOAP. Cada serviço *Web* é referenciado através de um identificador único chamado *Universal Resource Identifier* (URI) que consta do documento WSDL e que, naturalmente, se tem de especificar em todas as mensagens SOAP enviadas ao serviço.

Mais recentemente, para além dos serviços *Web* baseados em SOAP, surgiu a modalidade baseada em *Representational State Transfer* (REST) – os serviços RESTful. Os serviços

Web baseados em SOAP são suportados por RPC, *i.e.*, os clientes invocam o serviço *Web* através de RPC. Os serviços *Web* RESTful adoptam a filosofia dos servidores HTTP, *i.e.*, os clientes interagem com o serviço *Web* através das operações padrão de HTTP como o GET, POST, *etc.*

2.3. DESCRIÇÃO DE OBJECTOS MULTIMÉDIA

Existem diversas especificações para a catalogação, sincronização, indexação e descrição de objectos multimédia digitais. Neste contexto referem-se brevemente a *Synchronized Multimedia Integration Language* (SMIL) e a Annodex e, mais detalhadamente, a norma MPEG-7. No Anexo A apresentam-se também algumas ferramentas de anotação em MPEG-7 dado que os objectos multimédia a transaccionar estarão descritos neste formato. É de referir que, apesar de o objectivo ser a transacção de componentes multimédia descritos em MPEG-7, para a camada de negociação os seus descritores não serão usados uma vez que – como irá ser mostrado mais à frente – serão apenas necessários parâmetros inerentes à própria negociação.

2.3.1. SYNCHRONIZED MULTIMEDIA INTEGRATION LANGUAGE

A *Synchronized Multimedia Integration Language* (SMIL) é uma recomendação W3C desde Junho de 1998 [34]. Trata-se de uma linguagem baseada em XML que permite a criação e descrição de apresentações multimédia envolvendo áudio, vídeo, texto, *etc.* A SMIL disponibiliza assim um conjunto de anotações que permitem referenciar os elementos multimédia de forma sequencial ou paralela. As apresentações são criadas descrevendo o comportamento ao longo do tempo da apresentação, associando hiperligações a objectos multimédia e descrevendo a sua apresentação no ecrã.

2.3.2. ANNODEX

A Annodex [35] é uma especificação de XML desenvolvida pela Commonwealth Scientific and Industrial Research Organisation (CSIRO) em 2001 que se destina a permitir a indexação e anotação de áudio e vídeo. Utiliza a *Continuous Media Markup Language* (CMML) para subdividir o componente audiovisual em secções temporais – *clips* – e incorporar anotações [36]. A CMML define um conjunto de âncoras que podem ser adicionadas ao ficheiro de vídeo e que apontam para outros recursos relacionados. Esta

especificação também definiu uma forma de encapsular o documento CMML no ficheiro de vídeo, dando origem a um único ficheiro com formato Annodex [37].

2.3.3. MPEG-7

O MPEG-7 é uma norma conjunta da International Organization for Standardization (ISO) e da International Electrotechnical Commission (IEC) – ISO/IEC [38]. Destina-se à descrição e pesquisa de conteúdos vídeo e áudio desenvolvido pelo *Moving Picture Experts Group* (MPEG). O MPEG é constituído por especialistas das mais diversas áreas como gestores, criadores de conteúdos, editores, gestores de direitos de propriedade intelectual, prestadores de serviços de telecomunicações, construtores de equipamento electrónico, profissionais da área da difusão (*broadcast*) de TV e rádio e investigadores de universidades.

Oferece um conjunto de ferramentas que permite descrever objectos audiovisuais através da definição de estruturas de metadados e das suas relações de forma normalizada através de descritores e esquemas de descritores. Tem como objectivo principal descrever o conteúdo de ambientes multimédia para possibilitar a procura, filtragem, indexação e o acesso à informação audiovisual (imagens, figuras, gráficos, modelos 3D, áudio, vídeo) de forma rápida e eficaz [15] [38]. Dado que o MPEG-7 pretende ser o mais abrangente possível, as ferramentas padrão que disponibiliza são genéricas não sendo dirigidas a uma aplicação específica ou domínio de aplicações [14]. O padrão permite efectuar descrições com diferentes granularidades (que devem ter significado no contexto de uma determinada aplicação), tornando possível obter diferentes níveis de discriminação. Esta particularidade é importante porque permite adaptar o conteúdo audiovisual ao domínio de aplicação de cada utilizador, uma vez que o mesmo conteúdo audiovisual poderá ter interpretações diferentes que dependem dos interesses e condicionalismos dos destinatários.

Por exemplo, em relação a uma imagem estática ou em movimento, podem-se obter descritores para vários níveis de abstracção que estão relacionados com a maneira como a informação é extraída. O nível mais baixo, que é constituído pelos descritores que caracterizam elementos como formato, tamanho, textura, cor, trajectória e posição dos objectos na cena, são passíveis de extracção e processamento automático. No entanto, à medida que se vai subindo no nível de abstracção, torna-se mais difícil descrever a cena automaticamente. No nível mais alto, onde se utilizam múltiplos descritores com esquemas

de descritores e se explicitam as suas relações, pretende-se obter informação de cariz semântico. Este tipo de descrição requer a intervenção humana apesar de, em determinadas circunstâncias, ser possível utilizar ferramentas automáticas de extracção de informação semântica.

Os principais elementos do MPEG-7 são [15]:

- As ferramentas de descrição de conteúdos pré-definidas que são constituídas pelo conjunto dos *Descriptors* (D) e dos *Description Schemes* (DS) que especificam a estrutura (esquema) e semântica dos seus elementos;
- A linguagem de definição de descritores – *Description Definition Language* (DDL) – que permite a criação ou alteração de descritores e esquemas de descritores;
- As ferramentas de sistema que definem a codificação binária apropriada para a transmissão e armazenamento eficiente da informação. Contém também descritores de multiplexagem, sincronização das descrições com os conteúdos, protecção da propriedade intelectual, *etc.*;

Com estes elementos será possível descrever e transportar um componente multimédia no formato MPEG-7. As descrições de conteúdo do MPEG-7 deverão conter informação acerca de [38]:

- Descrição dos processos de criação e produção do conteúdo (director, título, resumo curto do filme);
- Utilização do conteúdo (apontadores de direitos de autor, histórico de utilização, agendamento de *broadcast*);
- Armazenamento do conteúdo (formato de gravação, codificação);
- Descrição dos processos de criação e produção do conteúdo (director, título, resumo curto do filme) e da estrutura dos componentes espácio-temporais do conteúdo (cortes de cena, regiões de segmentação, de monitorização de movimento);
- Características de baixo nível dos conteúdos (cores, texturas, timbres de som, descrição de melodias);
- Cariz conceptual (eventos dos objectos, interacções entre objectos);
- Navegação no conteúdo de forma eficiente (sumários, variações, sub-bandas de frequência);
- Colecções de objectos;

- Interacção do utilizador com o conteúdo (escolhas do utilizador, históricos de utilização).

Uma descrição gerada pelas ferramentas de descrição do MPEG-7 deve, naturalmente, estar relacionada com o conteúdo para permitir a eficiente procura e filtragem do vídeo pretendido. A descrição textual do conteúdo é efectuada em XML para facilitar a interoperabilidade com metadados gerados por outras aplicações. No padrão MEPG-7 não estão incluídos quaisquer mecanismos ou algoritmos de extracção automática ou semi-automática de conteúdos, motores de busca, agentes de filtragem ou qualquer outra aplicação que faça uso da descrição. A interoperabilidade entre aplicações está garantida pela própria linguagem usada nas descrições: XML. Por outro lado, o facto de estes algoritmos não estarem definidos [41], estimula a criatividade e competitividade entre as empresas da área apesar de também aumentar a complexidade e dificultar a interoperabilidade entre as diversas soluções encontradas.

Na Figura 5 podemos identificar os principais elementos constituintes do MPEG-7 e a forma como se inter-relacionam. O elemento central é constituído pela *Description Definition Language* (DDL) que permite definir descritores e esquemas de descritores. De seguida, os descritores são convertidos em XML por um processo de instanciação de acordo com regras definidas pela DDL.

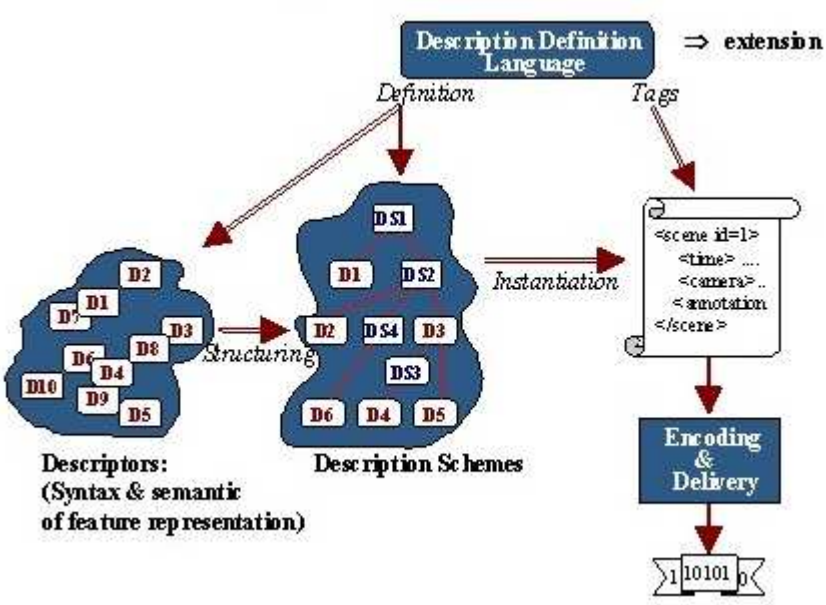


Figura 5 Componentes do MPEG-7[38]

A norma MPEG-7 pode ser dividida em dez áreas principais [38]:

1 *Systems* – Providencia as ferramentas necessárias ao transporte e arquivo em formato binário das descrições de conteúdo e define a arquitectura do terminal.

2 *Description Definition Language* – Define a sintaxe que permite criar descritores e esquemas de descritores. A DDL, que se baseia em XML *Schema Language*, inclui algumas adaptações para permitir a descrição de conteúdos audiovisuais.

3 *Visual* – Disponibiliza ferramentas para lidar com descritores de imagem (com ou sem movimento) e inclui descritores para cor, textura, forma, movimento, localização e reconhecimento facial.

4 *Audio* – Disponibiliza estruturas constituídas por um conjunto de descritores de baixo nível (funções espectrais, paramétricas e temporais do sinal) para aplicações gerais e descritores de alto nível (reconhecimento de sons, descritores de timbre instrumental para melodias, *etc*.) para aplicações mais específicas.

5 *Multimedia Description Schemes* (MDS) – Disponibiliza ferramentas para lidar com funções genéricas dos descritores de multimédia.

6 *Reference Software: eXperimental Model* (XM) – Define as normas gerais que se aplicam ao desenvolvimento de *software* para componentes do padrão MPEG-7. As aplicações XM são divididas em dois tipos de aplicações: (*i*) o servidor para extracção de metadados; e (*ii*) o cliente para as funções de procura, filtragem e codificação.

7 *Conformance testing* – Define os procedimentos e linhas gerais para testar a conformidade das implementações MPEG-7.

8 *Extraction and use of* MPEG-7 *descriptions* – Material informativo (na forma de relatório técnico) sobre a utilização de alguns descritores.

9 *Profiles and Levels* – Fornece orientações e define perfis padrão.

10 *Schema definition* – Especifica, usando a DDL, os esquemas de descritores.

2.4. REPRESENTAÇÃO DE CONHECIMENTO

Hoje em dia, milhões de pessoas acedem e partilham enormes quantidades de dados a velocidades ainda há poucos anos inimaginável. Nunca na história da humanidade esteve tanta e tão variada informação disponível. No entanto, esta heterogeneidade acarreta problemas ao nível da procura, interpretação e processamento, confrontando o utilizador com três tipos de problemas [50]:

1 Sintácticos – devido à heterogeneidade do formato dos dados;
2 Estruturais – provocados pela existência de homónimos, sinónimos ou atributos diferentes nas bases de dados;
3 Semânticos – resultantes dos diversos significados atribuíveis a um termo.

A heterogeneidade do formato dos dados na World Wide Web (WWW) conduziu ao desenvolvimento e adopção de padrões de anotação como o HTML e o *eXtensible* HTML (XHTML) para disponibilizar documentos que possam ser facilmente visualizados por intermédio de um navegador e o XML para a troca de dados entre máquinas que, devido à sua versatilidade e simplicidade, tem vindo a ser adoptado como parte integrante no desenvolvimento de aplicações e serviços nas mais variadas áreas.

O problema torna-se mais complexo com a heterogeneidade ao nível da estrutura e semântica da informação. O conteúdo e estrutura das bases de dados das empresas contêm informação estratégica que é considerada confidencial. Por outro lado, as bases de dados são desenvolvidas para satisfazer as necessidades específicas de funcionamento e de negócio de cada empresa e não as necessidades de um negócio à escala global, que requer a adopção de padrões para assegurar a interoperabilidade.

Com a globalização dos mercados e o crescimento exponencial das transacções B2B e B2C, as empresas aperceberam-se que a *Web* poderia ser o veículo de eleição para publicitar e comercializar os seus produtos e serviços. Foi então necessário encontrar abordagens que, não colocando em causa a confidencialidade da informação contida nas bases de dados das organizações, permitissem trocar a informação necessária num formato padrão inteligível entre os intervenientes. É neste contexto que as ontologias, enquanto formas de representação do conhecimento de um domínio, se revelam interessantes.

2.4.1. ONTOLOGIAS

A definição de ontologia não é consensual. No entanto, em [51], e no contexto das tecnologias de informação é proposta uma definição simples e concisa: "Uma ontologia é a descrição explícita e formal dos conceitos de um domínio de conhecimento". Trata-se, então, de uma forma de representação de conhecimento que é constituída pelos seguintes blocos [91]:

- Classes e conceitos organizados hierarquicamente;
- Propriedades que descrevem as várias características e atributos de cada conceito;
- Restrições.

O conjunto formado por uma ontologia e por um conjunto de instâncias das classes da ontologia constitui uma base de conhecimento.

Guarino em [52] sugere que as ontologias podem ser tipificadas em ontologias de topo, de domínio, de tarefas e de aplicação de acordo com o seu grau de generalização. Por outro lado, Leo Orbst [17] sugere que com o aumento da complexidade dos sistemas, aliado às distâncias cada vez maiores que os separa, será cada vez mais necessário adoptar tecnologias com nível semântico crescente, *i.e.*, que se aproximam do nível semântico da comunicação humana. Quanto maior o nível semântico, maior será a complexidade das tecnologias que suportam a representação do conhecimento – ver Figura 6.

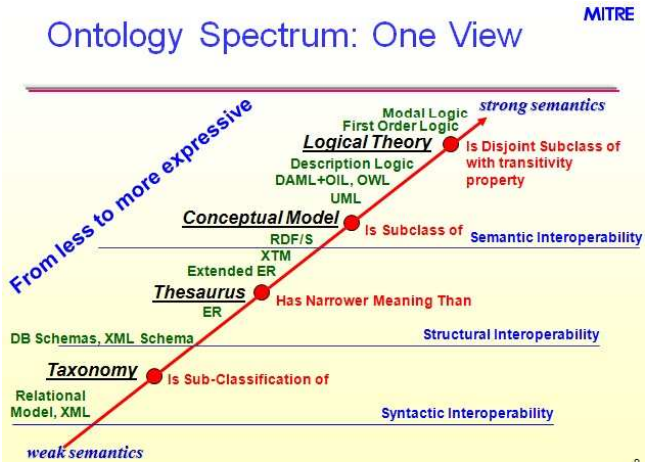


Figura 6 Ontolology Spectrum: One View [17]

Se cada empresa criar um agente virtual onde o conhecimento relativo aos produtos ou serviços a transaccionar estiver representado de acordo com ontologias conhecidas do seu domínio de negócio, continua a não partilhar com terceiros o conteúdo da sua base de

dados, mas conseguirá interagir com os seus pares. Por uma questão de eficiência e da redução do tempo de troca de informação entre agentes seria desejável que todos os intervenientes do negócio usassem as mesmas ontologias, mas, no mundo real, não é possível definir-se uma ontologia universal e normalizada. Isto faz com que, actualmente, estes agentes tenham de saber lidar com diferentes ontologias ou, então, recorrer a serviços externos de tradução ou mapeamento entre ontologias.

As ontologias, no contexto das tecnologias multimédia, são linguagens que usam um esquema de anotações (o mais comum é XML) para codificar o conhecimento sobre os objectos audiovisuais. Pertencem à família das linguagens de representação de conhecimento da Inteligência Artificial e surgiram da necessidade de promover a interoperabilidade entre sistemas, tendo já sido desenvolvidos esforços consideráveis na construção de uma ontologia MPEG-7 assim como na sua integração com outras ontologias.

Nas secções seguintes irá ser feita uma breve descrição de algumas destas tecnologias.

2.4.2. FORMATOS DE REPRESENTAÇÃO DE ONTOLOGIAS

A necessidade de desenvolver agentes que possam negociar de forma inteligente, fazendo depender as suas acções da análise semântica do conteúdo da informação recebida, obrigou ao desenvolvimento de novas tecnologias de suporte.

As características da linguagem XML tornam-na particularmente adequada para a transacção de informação entre plataformas e sistemas, mas o XML, por si só, não permite representar a componente semântica da informação [50] É nesse contexto que surgiram formatos de representação de conhecimento baseados em XML com capacidade de representação semântica. Nos parágrafos seguintes procura-se descrever resumidamente os principais formatos em uso.

2.4.2.1. RDF

O *Resource Description Framework* (RDF) foi inicialmente criado (em 1999) para descrever recursos, tendo sido concebido para processamento exclusivamente automático. Em Fevereiro de 2004 passou a fazer parte integrante das especificações do W3C [54], transformando-se numa linguagem genérica de representação da informação na *Web*. Tem como objectivo principal facilitar a interoperabilidade e troca de informação entre

aplicações obtendo-se assim nova informação (partilhada) para ser processada. Pretende ser uma língua franca para o tratamento automatizado de informação por agentes de *software* em processos cooperativos [53].

O RDF utiliza uma sintaxe baseada em XML (RDF/XML) para guardar e partilhar informação [54]. É uma linguagem baseada em declarações que descrevem os recursos através de expressões triplas do tipo Sujeito – Predicado – Objecto. Um conjunto de triplas constitui um modelo que pode ser visualizado sob a forma de grafo. Assim, uma declaração (ou tripla) pode ser representada por [58]:

- Um nó para o sujeito;
- Um nó para o objecto;
- Um arco para o predicado (seta no sentido do sujeito para o objecto).

A Figura 7 apresenta uma tripla RDF no formato de grafo.

Figura 7 Declaração RDF (adaptado de [58])

É possível agrupar triplas que contenham o mesmo sujeito e nesse caso ficaria Sujeito – Predicado – Objecto; Predicado – Objecto; Predicado – Objecto. Esta abordagem torna a leitura mais fácil, uma vez que se fica com uma noção da forma como determinado objecto se encontra relacionado.

O sujeito representa o recurso, o objecto pode conter dados ou representar outro recurso e o predicado (também conhecido como propriedade) define a forma como o sujeito se encontra relacionado com o objecto. O predicado é também considerado como um recurso e, tal como outros recursos em RDF, é identificado através de um *Universal Resource Identifier* (URI). O tipo de URI mais usado é o chamado *Uniform Resource Locator* (URL) [56]. Propriedades como título e autor podem ser usadas de forma recursiva com outras triplas. É também possível definir nós vazios – *blank nodes* ou *bNodes* – que também são conhecidos por nós anónimos – *anonymous nodes* [59]. Caso se trate de um objecto, substitui-se o nó vazio por '[]' ou '_:*x*'; caso se trate de um sujeito, fecha-se a tripla com parêntesis recto da seguinte forma: '[Predicado – Objecto]'.

O domínio de aplicabilidade do RDF é vasto, *e.g.*, é usado no melhoramento dos motores de busca de informação, na catalogação da informação de bibliotecas digitais, na descrição de conteúdos e suas relações de páginas e portais de informação. Um exemplo bastante conhecido da utilização do RDF é o vocabulário *Friend of a Friend* (FOAF) [60] que faz a descrição de pessoas e suas relações. O W3C disponibiliza uma ferramenta *on-line* para efectuar a validação de ficheiros RDF [61]. Em [62] encontra-se uma lista bastante exaustiva de ferramentas para manipular, gerir e interpretar documentos RDF desenvolvidos em diversas linguagens de programação (Java, PHP, Prolog, C++, Perl, Python, *etc.*).

2.4.2.2. DAML+OIL

A DAML+OIL é uma norma do W3C que resultou da união de esforços de dois projectos: o Norte-Americano DARPA *Agent Markup Language* (DAML) e o Europeu *Ontology Inference Layer* (OIL). A DAML+OIL é uma linguagem de anotação de recursos da rede semântica que incrementa as potencialidades do RDF e RDF *Schema* (RDFS) [110]. Trata-se de uma linguagem de definição de ontologias para a *Web* semântica que foi, entretanto, substituída pela OWL. Uma ontologia DAM+OIL consiste num conjunto de cabeçalhos (*headers*), seguido por um conjunto de classes, propriedades e instâncias [63]. É possível definir conjuntos de operações lógicas chamadas *class constructors* para fazer a intersecção, união e negação de classes. Suporta um conjunto de axiomas que permite relacionar as classes e/ou propriedades (equivalência, disjunção, igualdade, *etc.*) [110]. Nos cabeçalhos pode conter informação sobre a versão utilizada (`daml:versionInfo`) assim como as importações das definições de outras ontologias DAM+OIL que se pretendam utilizar nessa ontologia.

2.4.2.3. OWL

A *Web Ontology Language* (OWL) é uma evolução dos formatos RDF e DAM+OIL. Foi concebida como linguagem de definição de ontologias para a *Web* e pertence ao conjunto de recomendações do W3C para a *Web* semântica. O conjunto da OWL é constituído por:

- XML que fornece uma sintaxe para documentos estruturados, mas que não impõe restrições semânticas quanto ao seu significado.
- XML *Schema* que é a linguagem que permite restringir a estrutura dos documentos XML e definir tipos de dados em XML.

- RDF que é um modelo de dados para objectos e suas relações que podem ser representados em XML e que possui uma semântica simples.
- RDF *Schema* que é um vocabulário para descrever propriedades e classes de recursos RDF com uma semântica para hierarquias de generalização dessas propriedades e classes.
- OWL que contribui com mais vocabulário para descrever propriedades e classes, nomeadamente relações entre classes (*e.g.*, disjunções), cardinalidade, igualdade, características de propriedades (*e.g.*, simetria) e classes enumeráveis.

A OWL serve para descrever as classes e suas relações sob ponto de vista das aplicações e documentos *Web*. Existem três versões da linguagem com níveis de profundidade e complexidade crescente [3]:

- OWL *Lite*: destina-se a utilizadores que necessitam apenas de um conjunto limitado de funcionalidades.
- OWL DL (*Description Logics*): desenhada para suportar a lógica de descritores associada ao comércio electrónico (B2B, B2C, *etc*.). Permite obter a máxima expressividade sem perder capacidade computacional.
- OWL *Full*: destina-se a utilizadores que pretendem obter a totalidade da expressividade e liberdade sintáctica associadas ao RDF, mas que não necessitam de assegurar as condições para o seu tratamento computacional. Na verdade, é altamente improvável que exista algum *software* que suporte todas as funcionalidades do OWL *Full*.

É de salientar que cada uma destas sublinguagens é uma extensão da que a precede, o que quer dizer que as seguintes relações são válidas, mas o inverso não:

- Toda a Ontologia OWL *Lite* válida é uma ontologia OWL DL válida.
- Toda a Ontologia OWL DL válida é uma ontologia OWL *Full* válida.

Esta característica permite que se possa começar por uma ontologia com menor expressividade e, caso a aplicação venha a necessitar de maior expressividade, se migre para uma ontologia com maior capacidade de expressão.

As ontologias OWL-DL permitem realizar inferências sobre a informação que se encontra implicitamente representada nas ontologias. Neste caso podem ser usados DL *Reasoners* [64] que realizam verificações de consistência, equivalência, subsunção e instanciação.

A maioria das ontologias usadas nas descrições MPEG-7 são ontologias OWL, *i.e.*, foram definidas em OWL.

2.4.2.4. SKOS

Em Agosto de 2009 o W3C anunciou uma nova norma chamada *Simple Knowledge Organization System* (SKOS), desenvolvida pelo Semantic Web Deployment Working Group (SWDWG) que veio substituir a anterior (SKOS 2005) [72]. Esta norma permite estabelecer uma ponte entre os sistemas de organização de conhecimento que compreendem, entre outros, esquemas de classificação, taxonomias e *thesaurus*, e os *frameworks* usados na rede semântica. Trata-se, portanto, de uma forma normalizada de utilizar o RDF na representação de sistemas de organização de conhecimento – *Knowledge Organization Systems* (KOS) – contribuindo para uma maior interoperabilidade entre aplicações [70].

O SKOS aposta numa maior simplicidade face ao OWL, que é vocacionado para aplicações onde não é necessária uma elevada complexidade semântica. No entanto, paradoxalmente, o próprio modelo de dados é descrito através de uma ontologia OWL-*Full*, sendo cada KOS representado através de uma instância dessa ontologia [71].

2.4.3. CONSULTA DE ONTOLOGIAS

A SPARQL *Protocol and* RDF *Query Language* (SPARQL) é simultaneamente uma linguagem de consulta [73] [74] e um protocolo de acesso a documentos RDF. Foi desenvolvida pelo RDF Data Access Working Group (DAWG) do W3C com o objectivo de suprir as necessidades identificadas na consulta e acesso aos dados em formato RDF. Baseia-se em linguagens de consulta RDF pré-existentes como RDF *Database* (RDFDB), RDF *Data Query Language* (RDQL) e *Sesame* RDF *Query Language* (SeRQL), adicionando novos recursos [75]. Pode ser usada para consultar fontes de dados no formato RDF ou que sejam vistas como RDF através de um *middleware* apropriado.

Uma consulta (*query*) SPARQL geralmente envolve um conjunto de triplas. Estas triplas são semelhantes às triplas RDF descritas na secção anterior com a diferença de que qualquer um dos seus elementos (sujeito, predicado e objecto) pode representar a variável a encontrar. As consultas SPARQL são constituídas por duas partes [73]: (*i*) a identificação das variáveis desejadas; e (*ii*) a especificação da condição a satisfazer. A identificação das

variáveis é efectuada através de declarações do tipo `SELECT`, `CONSTRUCT`, `ASK` ou `DESCRIBE` e a especificação da condição de busca recorre à declaração `WHERE`. O Excerto de Código 1 apresenta a estrutura genérica de uma consulta SPARQL.

```
# prefix declarations PREFIX foo: <http://example.com/resources/>
…
# result clause SELECT … (ou CONSTRUCT)
# dataset definition FROM …
# query pattern WHERE … UNION …
# query modifiers ORDER BY …
```

Excerto de Código 1 Estrutura de uma Interrogação SPARQL

Os recursos a consultar são representados através do seu URI, que pode ser abreviado usando declarações do tipo `PREFIX`. Os resultados podem ser restringidos, filtrados ou ordenados usando operadores e funções específicas como `FILTER regex()`, que permite restringir a lista de resultados aqueles que contenham determinada palavra ou conjunto de letras, ou recorrendo ao `ORDER BY`, que permite que a lista de resultados seja ordenada de forma ascendente ou descendente. É também possível obter a junção de duas triplas através da declaração `UNION`, cujo resultado reflecte as soluções parciais obtidas a partir de cada tripla. De uma consulta pode resultar um, vários ou nenhum conjunto de dados [76].

2.4.4. VOCABULÁRIOS NORMALIZADOS

Nas ontologias utilizam-se vocabulários para descrever os conceitos e suas relações. Para assegurar que todos entendem da mesma forma os conceitos e relações, é desejável que utilizem o mesmo vocabulário. Tem sido investido um esforço considerável no desenvolvimento de vocabulários controlados universalmente aceites dentro dos diversos domínios do conhecimento [135]. Nesta secção descrevem-se alguns dos vocabulários mais conhecidos.

2.4.4.1. DUBLIN CORE

A *Dublin Core Metadata Initiative* (DCMI) é uma organização internacional sem fins lucrativos que tem como principal objectivo o desenvolvimento de normas simples que facilitem a procura, partilha e gestão da informação [66]. Para além da definição de termos para metadados (*i.e.*, DCMI *Metadata Terms*), também desenvolve guias e procedimentos para ajudar na implementação do *Dublin Core Metadata* em aplicações de *software*. A norma Dublin Core (DC) inclui dois níveis de filtragem: o simples e o avançado ou qualificado [89]. O *Simple* Dublin Core define os seguintes quinze elementos [66] [67]:

- *Title* – Nome dado ao recurso, *i.e.*, o nome com que o recurso é formalmente conhecido, *e.g.*, *Title* = "Artigos Multimédia".
- *Subject* – Assunto e palavras-chave que definem o conteúdo do recurso, *e.g.*, *Subject* = "Fotografias de Animais".
- *Description* – Descrição ou resumo do conteúdo do recurso para facilitar a obtenção dos resultados desejados.
- *Type* – Natureza, tipo ou género do conteúdo do recurso. Para se garantir interoperabilidade recomenda-se a utilização de pelo menos um termo da lista *DCMIType vocabulary*, *e.g.*, *Type* = "*Image*", *Type* = "*Sound*".
- *Source* – Referência a um recurso do qual o recurso actual deriva, *e.g.*, *Source* = "Imagem tirada da página 62 do manual da disciplina de SADIT no ano lectivo 2009-2010".
- *Relation* – Referência a um recurso relacionado.
- *Coverage* – Extensão ou âmbito do recurso, *e.g.*, *Coverage* = "2009-2010" ou *Coverage* = "Instituto Superior de Engenharia".
- *Creator* – Principal entidade responsável pelo conteúdo do recurso. As entidades poderão ser pessoas, organizações ou serviços.
- *Publisher* – A entidade responsável por tornar o recurso disponível. Também aqui poderão ser consideradas pessoas, organizações ou serviços.
- *Contributor* – Entidade responsável por contribuições para o conteúdo do recurso. Esta entidade é considerada a mais geral de todas as entidades que contribuem para o conteúdo do recurso, pelo que, só deverá ser usada quando não existirem outras ou estas sejam irrelevantes.
- *Rights* – Trata-se de informação relativa aos direitos que se aplicam ao recurso. Esses direitos poderão ser de propriedade intelectual, direitos de autor e outros direitos de propriedade.
- *Date* – Data associada a algum evento ou fase da vida de um recurso. Tipicamente é usada a data de criação ou de publicação. O W3C aconselha a utilização de formatos normalizados de acordo com a norma ISO 8601 [68].
- *Format* – O formato do recurso define características coerentes com o tipo (*Type*) de recurso. No caso de um objecto físico, o formato poderá conter informações como o volume, peso e outras características físicas. No caso de um recurso digital, *e.g.*,

imagem, vídeo, *etc*., o formato poderá incluir características como tamanho (em KiB), duração, tipo de codificação, entre outras.

- *Identifier* – Trata-se da referência que identifica inequivocamente o recurso num determinado contexto, *e.g.*, *Universal Resource Identifier* (URI), *International Standard Book Number* (ISBN), *Digital Object Identifier* (DOI), *etc.*
- *Language* – Língua do conteúdo intelectual do recurso.

O *Qualified* Dublin Core contém mais três elementos (*Audience*, *Provenance* e *RightsHolder*) e inclui um conjunto de qualificadores que refinam a semântica associada aos elementos que poderão tornar a procura de recursos bastante mais eficiente. Existem duas classes de termos designadas elementos (ou nomes) e qualificadores (adjectivos) que, quando combinadas, constituem declarações. Os qualificadores servem para ajudar a especificar um recurso e não para alargar o âmbito semântico de uma propriedade [69]. Para que tal aconteça a *Dublin Core* define um princípio popularmente conhecido como *dumb-down*. De acordo com esta regra, um cliente deve ser capaz de ignorar qualquer qualificador e utilizar o valor como se fosse absoluto. Apesar de se perder alguma especificidade, o elemento por si só deverá ser correcto e útil para a descoberta. O DC define um conjunto de recomendações e regras de boas práticas na utilização dos elementos que incluem vocabulários controlados e anotações de análise formal ou regras [66]. Cada elemento é opcional e pode ser repetido sempre que for necessário.

2.4.4.2. FRIEND OF A FRIEND

O *Friend Of A Friend* (FOAF) é um vocabulário controlado para descrever pessoas e as suas relações, *e.g.*, nome e endereço *email* do próprio e dos seus conhecidos [77]. Foi criado em 2000 por Dan Brickley e Libby Miller ao abrigo do projecto "*experimental linked information Project*". Todos os documentos FOAF devem ser documentos RDF bem formados [79], o que os torna compatíveis com as tecnologias da rede semântica. A ferramenta (FOAF-a-matic), que se encontra em [78], permite gerar documentos com descrições pessoais através da introdução dos dados num formulário – ver Excerto de Código 2.

```
…
</foaf:PersonalProfileDocument>
<foaf:Person rdf:ID="me">
<foaf:name>Luis Sousa</foaf:name>
<foaf:title>Aspirante a Mh</foaf:title>
<foaf:givenname>Luis</foaf:givenname>
<foaf:family_name>Sousa</foaf:family_name>
<foaf:nick>LuisSousa</foaf:nick>
<foaf:mbox rdf:resource="mailto:1900503@isep.ipp.pt"/>
<foaf:homepage rdf:resource="http://dee.isep.ipp.pt"/>
<foaf:depiction rdf:resource="NA"/>
<foaf:phone rdf:resource="tel:222222222"/>
<foaf:workplaceHomepage rdf:resource="NA"/>
<foaf:workInfoHomepage rdf:resource="NA"/>
<foaf:schoolHomepage rdf:resource="http://dee.isep.ipp.pt"/>
…
```

Excerto de Código 2 Excerto Documento Gerado pela Ferramenta FOAF-a-matic [78]

O vocabulário FOAF é usado para assegurar a interoperabilidade de dados nas redes sociais [80].

2.4.4.3. DBPEDIA

Consiste na versão RDF da informação contida na Wikipedia que contava em Setembro de 2011 com mais de 1,89 biliões de triplas que descrevem cerca de 10,3 milhões de coisas em mais de 111 línguas diferentes [81]. A Dbpedia é uma das maiores ontologias multidomínio existentes e encontra-se inserida numa rede de 295 ontologias interligadas que compreendem mais de 31 biliões de triplas RDF e cerca de 504 hiperligações RDF de interligação [82].

2.4.4.4. JAMENDO

Trata-se de uma colecção de músicas sob licença gerada gratuitamente pela *Creative Commons* [83]. Continha em 2008 dados de milhares de artistas, dezenas de milhar de álbuns, cerca de 100 000 faixas e possui mais de 0,61 milhões de triplas [84].

2.4.4.5. GOVTRACK

O GovTrack [85] contém informação sobre o congresso dos Estados Unidos da América, perfazendo em 2008 cerca de 1012 milhões de triplas [84] sobre legisladores, contas e votos.

2.4.4.6. SEMANTIC WEB CONFERENCE

A Semantic Web Conference (SWC) *ontology* contém informação no formato RDF sobre *workshops*, agendas e apresentadores de conferências patrocinadas pela International Semantic Web Conference (ISWC) e pela European Semantic Web Conference (ESWC) [86]. Os dados são apresentados com recurso a outros vocabulários como o FOAF, *Semantic Web for Research Communities* (SWRC) ou o iCal/RDF *Calendar*.

2.4.4.7. GOODRELATIONS

Trata-se de um vocabulário para comércio electrónico de utilização livre compatível com as normas W3C da rede semântica mais relevantes (RDF e OWL) [87]. Trata-se de uma ontologia descrita no formato OWL DL que disponibiliza um vocabulário para anotação de venda de produtos e outros aspectos relacionados com o *e-commerce*.

2.4.5. ONTOLOGIAS MPEG-7

Uma das principais vantagens do MPEG-7 é a flexibilidade com que os descritores podem ser associados a qualquer segmento multimédia com qualquer nível de granularidade e usando diferentes níveis de abstracção. No entanto, esta característica resulta numa maior complexidade e ambiguidade nas descrições. Encontram-se definidos nos esquemas XML do MPEG-7 1182 elementos, 417 atributos e 377 tipos complexos de dados, o que torna o padrão MPEG-7 muito difícil de gerir [4]. Por outro lado, a utilização de esquemas XML implica que a maior parte da semântica se mantém implícita. O facto da norma não permitir a definição de uma semântica formal, pode levar uma variação da sintaxe a causar sérios problemas de interoperabilidade no processamento multimédia e nas trocas de descrições entre agentes. Para tentar resolver essa falta de semântica formal nas descrições MPEG-7, foram propostas quatro ontologias descritas em OWL que cobrem todos os aspectos do padrão da norma MPEG-7: as ontologias de Hunter, DS-MIRF, Rhizomik e COMM.

2.4.5.1. ONTOLOGIA HARMONY

A Ontologia proposta por Hunter em 2001 surge no âmbito do projecto Harmony. A linguagem original de desenvolvimento da ontologia foi o RDFS que, mais tarde, foi convertida e adaptada para DAML+OIL. Por fim, foi codificada no formato OWL *Full* [104], contendo classes que definem diversos tipos de objectos multimédia (áudio,

audiovisual, imagem, multimédia e vídeo) e de decomposições a partir dos esquemas de descrição multimédia – *Multimedia Description Schemes* (MDS). A sua aplicação mais corrente é na descrição de imagens e dos seus descritores visuais. Foi possível harmonizar esta ontologia com outra ontologia de nível superior (ABC) para permitir interrogações, usando conceitos mais abstractos como subclasses, eventos ou agentes, para obter objectos multimédia ou segmentos desses objectos [88]. Hunter refere em [106] que uma das vantagens do uso de uma ontologia de topo é promover uma maior interoperabilidade entre ontologias heterogéneas mas que partilham os mesmos conceitos de topo, permitindo também a integração ou fusão de descrições complementares baseadas em metadados assim como obter diferentes pontos de vista dos metadados da mesma descrição de acordo com as preferência particulares, perspectivas e requisitos dos utilizadores.

2.4.5.2. ONTOLOGIA DS-MIRF

Em 2004 Tsinaraki propôs a ontologia *Domain-Specific Multimedia Indexing, Retrieval and Filtering* (DS-MIRF) que captura, no formato OWL DL, a semântica do MPEG-7 MDS e os esquemas de classificação [88]. Foi feita uma tradução manual e directa de todos os descritores do MPEG-7 para a ontologia que, apesar de ter clarificado e explicitado um pouco mais alguns elementos do XML *Schema* original, acabou por herdar as mesmas ambiguidades semânticas [92]. Esta ontologia tem sido usada em diversas aplicações que incluem bibliotecas digitais de audiovisual e de ensino à distância.

2.4.5.3. ONTOLOGIA RHIZOMIK

Garcia e Celma em 2005 apresentaram a ontologia de Rhizomik [123] que consiste no mapeamento automático de todos os elementos do XML *Schema* em OWL-DL [92]. Usa duas ferramentas de mapeamento XSD2OWL (transforma XML *Schema* numa ontologia OWL) e XML2RDF (transforma XML em RDF) que combinadas permitem fazer a tradução completa do MPEG-7 *Schema* [88]. A intervenção humana resume-se à resolução de alguns conflitos ao nível da ontologia de domínio. Alguns exemplos de utilização compreendem a integração e extracção dos metadados de música [92].

2.4.5.4. ONTOLOGIA COMM

A ontologia *Core Ontology for Multimedia* (COMM) constitui a proposta mais recente (2007) de anotação de componentes multimédia descritos em MPEG-7 [92] . Utiliza uma

ontologia de topo chamada *Descriptive Ontology for Linguistic and Cognitive Engineering* (DOLCE) que, sendo independente do vocabulário das ontologias do domínio, contém definições formais de categorias como processos e objectos físicos. A COMM encontra-se descrita no formato OWL DL e permite traduzir os descritores mais importantes do MPEG-7 usados na descrição da estrutura e do conteúdo dos documentos multimédia [88].

2.5. SISTEMAS MULTIAGENTE

Os Sistemas Multiagente (SMA) constituem um paradigma computacional particularmente apropriado para a modelação de problemas de natureza inerentemente distribuída. Cada elemento é capaz de, individualmente, procurar alcançar os seus objectivos e, ao fazê-lo, contribuir para a satisfação dos objectivos da comunidade. Os agentes correm num determinado espaço virtual de suporte que fornece serviços que os agentes poderão requerer para facilitar as tarefas que têm de desempenhar. A maior ou menor facilidade e disponibilidade desses serviços assim como o seu grau de sofisticação ajudam a determinar as opções de escolha do ambiente de desenvolvimento e execução.

Existem inúmeras formas de classificar agentes de acordo com a forma como se comportam no meio [89]. Essa classificação é baseada na presença em maior ou menor grau de atributos que foram definidos nos agentes durante a sua concepção. A Tabela 1 descreve resumidamente esses atributos:

Tabela 1 **Atributos dos Agentes de *Software* [89]**

Atributo	Descrição
Autonomia	Capacidade de executar tarefas e interagir com o meio sem a intervenção humana.
Reactividade	Capacidade de o agente reagir e responder de acordo com as alterações que percepciona.
Pro-actividade	Capacidade de o agente tomar a iniciativa de realizar determinada tarefa.
Persistência	Capacidade de levar as tarefas até ao fim de acordo com os objectivos estabelecidos.
Sociabilidade	Capacidade estabelecer diálogos com outros agentes ou utilizador de forma a perceber e executar a suas tarefas

Mobilidade	Capacidade de um agente transferir o código e estado em que se encontra para outra máquina para aí continuar a execução a partir do ponto onde se encontrava.
Intencionalidade	Capacidade de tomar decisões (ou acções) racionais decorrentes da sua base de conhecimento e das interacções com o meio.
Aprendizagem	Capacidade de construir uma base de conhecimento sobre a qual irá tomar as suas decisões racionais e de ir actualizando essa base de conhecimento de acordo com o que vai percepcionando.
Cooperação	Capacidade de os agentes se agruparem e darem os seus contributos com vista a atingir um objectivo comum.

2.5.1. AMBIENTES DE DESENVOLVIMENTO DE AGENTES

Existem diversos ambientes de desenvolvimento e execução de sistemas multiagente que oferecem serviços e ferramentas de suporte à realização das tarefas habituais de criação, depuração, lançamento e monitorização de uma comunidade de agentes. Neste trabalho restringiu-se o estudo a ambientes conformes com normas e padrões reconhecidas neste âmbito, *e.g.*, as especificações da *Foundation for Intelligent Physical Agents* (FIPA). O modelo de referência FIPA é descrito pelo FIPA *Agent Management Specification* [124], que obriga à existência de três serviços obrigatórios:

- *Agent Management System* (AMS) – Serviço de páginas brancas para registo dos agentes na plataforma.
- *Message Transport Service* (MTS) – Serviço que define o método padrão de comunicação dos agentes na mesma plataforma ou entre plataformas diferentes
- *Directory Facilitador* (DF) – Serviço de páginas amarelas que funciona de ponto de encontro entre os agentes da plataforma.

2.5.1.1. AMBIENTE SACI

O *Simple Agent Communication Infrastructure* (SACI) é uma ferramenta desenvolvida em linguagem Java para a construção de sistemas multiagente desenvolvida na Universidade de S. Paulo, Brasil. O ambiente oferece dois recursos principais: (*i*) uma *Application Programming Interface* (API) para definir, enviar e receber mensagens entre agentes no formato *Knowledge Query Manipulation Language* (KQML); e (*ii*) um conjunto de

ferramentas vocacionadas para o desenvolvimento de aplicações em ambientes distribuídos. Os agentes são identificados pelo nome e agrupados em sociedades. Existe um agente facilitador – *Facilitator* – que disponibiliza um serviço de páginas amarelas e brancas que fornece um serviço de registo e procura dos agentes activos. Possui também um componente denominado de ProtocolBox que permite definir protocolos de interacção [38].

2.5.1.2. AMBIENTE FIPA OS

O FIPA *Open Source* (FIPA OS) [125] é um *software* de código aberto desenvolvido pela Nortel Networks para suportar a construção de SMA que seguem as normas FIPA. Foi desenvolvida integralmente em linguagem Java (JDK1.1) e possui duas versões: (*i*) a versão padrão que inclui todas as funcionalidades e (*ii*) a versão leve denominada de *Micro*-OS para correr em *Personal Digital Assistants* (PDA) Linux ou PocketPC. Possui uma classe para a implementação de agentes e uma extensão para tirar partido do motor de inferência *Java Expert System Shell* (JESS) [111]. Também permite incluir agentes de teste para depuração. A codificação das mensagens ACL pode ser feita em ASCII ou em XML [116]. Tem sido usado em vários domínios como o desenvolvimento de plataformas de marcação de reuniões ou em redes privadas de provisionamento.

2.5.1.3. AMBIENTE JADE

O *Java Agent DEvelopment Framework* (JADE) é um ambiente de desenvolvimento totalmente implementado em linguagem Java. A escolha da linguagem Java deveu-se às características da programação por objectos em ambientes heterogéneos distribuídos como a independência da plataforma, a serialização de objectos ou a API de RMI [5]. Trata-se de um *middleware* para desenvolvimento de sistemas multiagente onde os agentes comunicam entre si de acordo com as especificações FIPA e que disponibiliza um conjunto de ferramentas de suporte à gestão de agentes e análise de erros. O mecanismo que usa para o transporte adapta-se de forma transparente a cada situação de acordo com os protocolos disponíveis (Java RMI, *event-notification*, HTTP e *Internet Inter-Object Request Broker Protocol* (IIOP)) [5]. O JADE é composto pelos seguintes componentes principais:

- `Jade.core`: Implementa o *kernel* do sistema e inclui a classe `Agent` que deverá ser especializada pelas aplicações para a especificação de agentes. Os comportamentos dos

agentes, tais como tarefas e intenções, estão definidos na classe `Behaviour` contida no pacote `jade.core.behaviours`.

- `Jade.content`: Contém as classes que suportam a inclusão das ontologias definidas pelo utilizador.
- `jade.domain`: Contém todas as classes que representam as entidades de gestão dos agentes definidas no padrão FIPA: o AMS responsável pela gestão da plataforma e que disponibiliza o serviço de páginas brancas e o agente DF que, de acordo, com a arquitectura FIPA disponibiliza o serviço de páginas amarelas. O pacote `FIPAAgentManagement` contém a ontologia FIPA de gestão dos agentes e todas as classes que representam os seus conceitos. O pacote `JADEAgentManagement` contém a ontologia e as classes que permitem fazer a gestão dos agentes (*e.g.*, fazer *sniffing* de mensagens, controlar o tempo de vida dos agentes, *etc.*). Para fazer a ligação entre as ferramentas do JADE e o *kernel* são usadas classes contidas na biblioteca `jade.domain.introspector`. O pacote `mobility` contém todas as classes necessárias à mobilidade dos agentes.
- `jade.gui`: Inclui um conjunto de classes que permite criar *Graphical User Interfaces* (GUI), editar as descrições e identificadores dos agentes, visualizar as mensagens *Agent Communication Language* (ACL) trocadas entre os agentes, *etc*.
- `jade.mpt`: Contém as classes que definem *o* MTS do sistema, *i.e.*, o conjunto de protocolos do sistema.

O JADE permite ter vários ambientes de execução designados contentores (*Container*) activos em simultâneo. Uma plataforma é formada por conjuntos de contentores que podem estar ou não a ser executados na mesma máquina – ver Figura 8.

Em cada plataforma deverá existir apenas um contentor principal, que contém os agentes DF e AMS, onde os restantes contentores se devem registar. Cada agente tem um identificador único na rede constituído por `<nickname>@<nome-da-plataforma>` gerido pelo AMS da plataforma correspondente (que se encarrega de assegurar que cada agente tem um identificador único). Quando um agente pretende contactar outro agente, recorre ao serviço de páginas amarelas disponibilizado pelo agente DF.

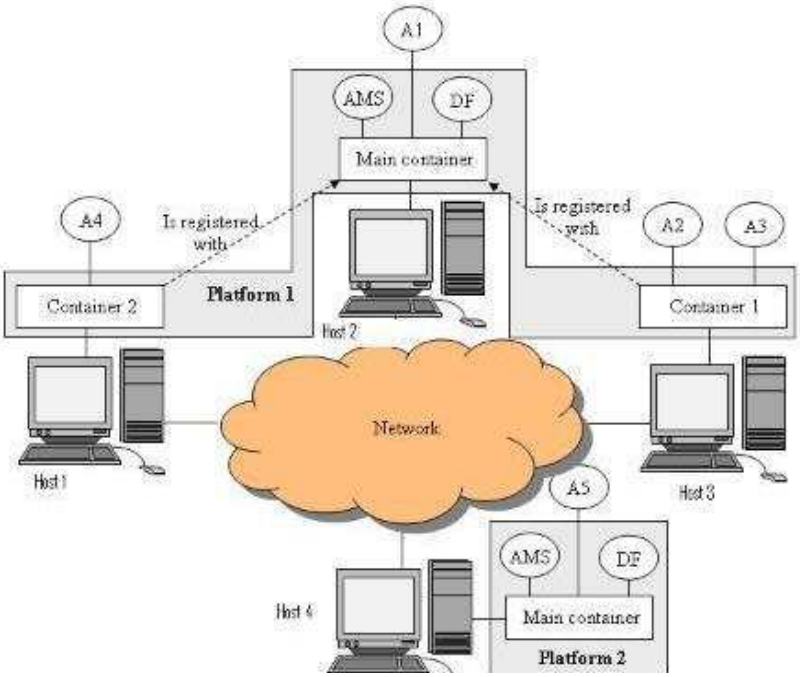


Figura 8 Arquitectura JADE [126]

2.5.2. LINGUAGENS DE COMUNICAÇÃO

Existe um conjunto de linguagens e protocolos que permite aos agentes de *software* aceder, manipular e comunicar a informação mantida nas suas bases de conhecimento. Deste conjunto, apenas as especificações padronizadas são relevantes para este projecto.

2.5.2.1. KQML

A KQML foi desenvolvida nos anos 90 pela Defense Advanced Research Projects Agency (DARPA) – *Knowledge Sharing Effort* (KSE) e tinha como objectivo o desenvolvimento de técnicas para construção de bases de conhecimento de larga escala partilháveis e reutilizáveis. O KQML tanto pode ser usado para uma aplicação interagir com um sistema inteligente como para sistemas inteligentes interagirem entre si. Tem como objectivo disponibilizar um protocolo de transporte de conhecimento baseado numa ou mais ontologias que são inteligíveis pelos agentes intervenientes nessa comunicação. Dessa maneira, o KQML não analisa a linguagem de conteúdo utilizada, que pode ser *Knowledge Interchange Format* (KIF), RDF, FIPA *Semantic Language* (SL) ou outra, nem o mecanismo de especificação das ontologias que representam o modelo mental [122]. Trata-se de uma linguagem baseada em actos de fala (performativas) que representam intenções de realização de determinadas acções.

2.5.2.2. FIPA-ACL

A FIPA *Agent Communication Language* (FIPA–ACL) é uma especificação da *Foundation for Intelligent Physical Agents* (FIPA) desde Dezembro de 2002 [114] e tem os mesmos objectivos que a KQML. No entanto, é mais leve em termos de número de actos de fala (performativas) definidos. Foi desenvolvida para trabalhar com qualquer linguagem de conteúdo e qualquer abordagem que utilize ontologias. Na arquitectura FIPA foi definida a utilização de um AMS para gerir o registo de agentes na plataforma, evitando a ambiguidade de nomes entre agentes num mesmo ambiente de execução. A comunicação entre agentes é feita com recurso a um conjunto de actos de comunicação que podem ser divididos em cinco grupos funcionais: passagem de informação (Tabela 2), solicitação de informação (Tabela 3), negociação (Tabela 4), acção (Tabela 5) e gestão de erros (Tabela 6) [114] .

Tabela 2 **ACL: Mensagens de Passagem de Informação**

Performativa	**Descrição**
confirm	Usada quando um agente pretende informar outro agente de que determinada proposição é verdadeira.
disconfirm	Usada quando um agente acredita que outro agente poderá estar convencido da veracidade de determinada proposição e tenta convencer este que a mesma é falsa.
inform	Usada quando um agente acredita que determinada proposição é verdadeira e pretende convencer outro agente dessa veracidade. A diferença relativamente à mensagem *confirm* é que neste caso, o agente (que envia) não acredita que o outro agente tenha conhecimento da proposição.
inform-if *(macro act)*	Usada quando o agente crê que a proposição é verdadeira mas admite que o agente de destino possa discordar, pelo que poderá receber do agente destino a sua crença. Basicamente pretende obter do agente de destino a confirmação ou não se determinada proposição é verdadeira (ou falsa).
inform-ref *(macro act)*	Usada quando agente interlocutor questiona o agente de destino acerca da resposta a determinada questão e espera receber uma mensagem *inform* com a resposta (ou então *refuse* caso não saiba ou não esteja em condições de responder)

Tabela 3 **ACL: Mensagens de Solicitação de Informação**

Performativa	**Descrição**
query-if	Usada para perguntar a outro agente se, por exemplo, já efectuou determinada acção. Este deverá responder (com *inform*) se sim ou não.
query-ref	Usada quando a resposta que se pretende obter do agente de destino é o um conjunto de objectos que satisfaçam a questão que foi solicitada.
subscribe	Usada quando se pretende implementar uma versão persistente da mensagem *query-ref*, *i.e*., o agente que recebe o pedido de *subscribe* vai responder com um *inform* contendo o objecto solicitado e irá, posteriormente, reportar (com mais mensagens *inform*) sempre que houver alterações nesse objecto.

Tabela 4 **ACL: Mensagens de Negociação**

Performativa	**Descrição**
accept-proposal	Mensagem de aceitação a uma proposta previamente recebida (tipicamente pela mensagem *propose*)
cfp	Mensagem usada para iniciar um processo de negociação com agentes que reúnam as condições especificadas (mesmo protocolo, use a mesma ontologia, que corresponda aos parâmetros solicitados, *etc*.)
propose	Mensagem do agente proponente com uma proposta (com determinadas condições) para o agente de destino
reject-proposal	Resposta de recusa à mensagem *proposal* durante um processo de negociação. Poderá também ser indicado o motivo dessa recusa embora não seja obrigatório.

Tabela 5 **ACL: Mensagens de Acção**

Performativa	**Descrição**
agree	Reflecte a aceitação de realização de determinada acção. Trata-se de uma resposta à mensagem *request*.
cancel	Usado quando um agente pretende cancelar uma acção que foi previamente solicitada a outro agente e que acredita que ainda não foi executada. Caso já tenha sido executada deverá ser recebida uma mensagem de recusa ou *refuse*.

refuse	Usada quando o agente pretende recusar a execução de determinada acção indicando o respectivo motivo. Acontece quando não consegue assegurar todos os requisitos da acção ou não tem privilégios suficientes para a sua execução.
request	Mensagem usada quando um determinado agente pretende que outro execute determinada acção.
request-when	Mensagem idêntica à anterior mas neste caso é dada uma proposição e é pedido ao agente de destino que apenas execute a acção quando a proposição indicada se tornar verdadeira.
request-whenever	Neste caso a acção deverá ser executada sempre que a proposição indicada for verdadeira.

Tabela 6 **ACL: Mensagens de Gestão de Erros**

Performativa	**Descrição**
failure	Mensagem usada informar outro agente que tentou executar determinada acção mas que, por alguma razão, falhou.
not-understood	Mensagem usada para informar o agente de destino que a mensagem recebida não foi entendida.

As mensagens ACL trocadas entre agentes contêm um conjunto de parâmetros. Apesar de apenas o parâmetro *performative* ser obrigatório, é comum conter igualmente os parâmetros *sender*, *receiver*, *content* e *ontology* que identificam o nome do agente que envia, o que recebe, o conteúdo da mensagem e, por fim, o nome da ontologia usada. No Excerto de Código 3 apresenta-se uma mensagem ACL com todos estes parâmetros.

```
(inform
   :sender agent1
   :receiver hlp-auction-server
   :content (price(bid good02) 150)
   :in-reply-to round-4
   :reply-with bid04
   :language sl
   :ontology hpl-auction
)
```

Excerto de Código 3 Mensagem ACL

2.5.3. COMUNICAÇÃO

A troca de mensagens entre agentes segue um padrão bem definido designado protocolo. No contexto dos SMA, os protocolos controlam, usando um conjunto de regras, o diálogo entre agentes para alcançar um determinado objectivo. Na realidade, a adopção de protocolos facilita bastante o desenvolvimento de agentes porque fica assegurado o sucesso

da comunicação. A FIPA [114] define um conjunto de protocolos que podem ser utilizados pelos agentes, bastando indicar no campo *protocol* da mensagem o protocolo que se pretende usar. Para descrever o conteúdo das mensagens ACL é usada a linguagem FIPA *Semantic Language* (SL).

No âmbito deste projecto adoptou-se o protocolo FIPA *Iterated Contract Net* (ICNet) [13], mais concretamente definiu-se uma variante do ICNet designada *Fixed* ICNet. Trata-se de um protocolo em tudo idêntico ao ICNet apenas diferindo num aspecto: o número de iterações é pré-determinado e é fixo, *i.e.*, uma ronda negocial tem sempre o número de iterações que se pré-definiu.

2.6. CONCLUSÃO

As plataformas de comércio electrónico oferecem actualmente às empresas um conjunto de vantagens como o acesso ao mercado global, a redução de tempos de transacção, o aumento da qualidade (e quantidade) dos serviços oferecidos, a capacidade de adaptação dos produtos às necessidades do cliente, a redução de custos e, potencialmente, o aumento do volume de vendas [118].

Dada a natureza intrinsecamente distribuída deste domínio, é frequente adoptar-se o modelo da computação baseada em agentes para a modelação das entidades envolvidas. Os agentes para negociar de forma flexível e autónoma, devem possuir bases de conhecimento correctamente estruturadas e instanciadas e adoptar protocolos de negociação padrão.

A representação de conhecimento através da definição de ontologias ou da utilização de ontologias padrão do domínio e da adopção de taxonomias de classificação padrão permite reforçar a interoperabilidade. As ontologias permitem aos agentes, tanto internos quanto externos de um sistema, partilhar o conhecimento sobre a informação utilizada e promover a interoperabilidade entre sistemas [121]. Em termos de formatos de representação, a OWL apresenta-se como uma das principais linguagens de representação de ontologias.

A abertura proporcionada pela norma MPEG-7 que não especifica propositadamente quaisquer esquemas de extracção automática das descrições, dificultou o aparecimento de representações universais de descrição de componentes multimédia como bases de conhecimento – *Knowledge Bases* (KB) – normalizadas e correspondentes taxonomias, ontologias, thesaurus, *etc*. A hipótese de definição de uma ontologia única global para a

descrição de objectos multimédia esbarra em dois grandes obstáculos: (*i*) a dificuldade em convencer todos os intervenientes a utilizá-la; e (*ii*) o desafio que constitui a especificação de uma ontologia que cubra satisfatoriamente as necessidades das diversas aplicações [92]. Assim, deve-se prever a coexistência de um conjunto heterogéneo de ontologias que obriga a utilizar esquemas de mapeamento entre conceitos. No entanto, dado que objectivo deste projecto não é o mapeamento entre ontologias, optou-se pelo uso de ontologias homogéneas no âmbito da plataforma de negociação de componentes multimédia.

A selecção da plataforma de desenvolvimento de agentes recaiu sobre o JADE e baseou-se nos seguintes critérios. (*i*) ser de código e distribuição livre; (*ii*) disponibilizar protocolos e serviços padrão, (*iii*) oferecer suporte, documentação e actualizações frequentes.

3. AMBIENTE DE DESENVOLVIMENTO

Neste capítulo apresenta-se o ambiente de desenvolvimento adoptado e referem-se as linguagens, ferramentas e bibliotecas utilizadas.

3.1. LINGUAGEM DE PROGRAMAÇÃO

A linguagem de desenvolvimento escolhida foi a linguagem de programação Java. Trata-se de uma linguagem de programação orientada por objectos, independente da plataforma, baseada na partilha e reutilização de código e onde impera a filosofia do código aberto. Encontra-se suportada por um vasto conjunto de ferramentas e API, além de existirem várias comunidades de programadores nas diversas áreas activas e solícitas.

A linguagem Java é a escolha natural para o desenvolvimento de aplicações com elevado nível de portabilidade e interoperabilidade.

3.2. LINGUAGENS DE ANOTAÇÃO

A *Web* transformou-se num mundo de oportunidades para o comércio, publicidade e os negócios. É neste contexto que surge a necessidade da troca de informação entre

plataformas de uma forma rápida e padronizada, tendo surgido diversas linguagens de anotação.

As linguagens de anotação são usadas em documentos de texto e têm a particularidade de recorrerem a anotações para comentar o texto de base e lhe conferir determinadas características. Resultaram da necessidade da partilha de conteúdos de forma cómoda, rápida entre plataformas e, sobretudo, permitem o processamento automático. Estas linguagens definem um conjunto de anotações de maior ou menor complexidade. As mais complexas derivam, na sua maioria, da metalinguagem de anotação XML.

Para gerar e interpretar correctamente as anotações (como um conjunto de identificadores com características específicas e não como um simples texto), foram criadas ferramentas adequadas.

Um caso paradigmático das linguagens de anotação é a *HyperText Markup Language* (HTML) que foi desenvolvida para a anotação de documentos de texto, conferindo-lhes uma estrutura, estilos de apresentação e interactividade, e que é interpretada pelos navegadores dos utilizadores.

3.2.1. XML

A linguagem XML deriva da *Standard Generalized Markup Language* (SGML) (ISO 8879) e consiste numa versão mais simplificada e funcional. Surgiu para promover a troca, de uma forma prática, rápida e segura, de dados entre computadores e aplicações de características diferentes [12]. Trata-se de uma metalinguagem de anotação desenvolvida pelo W3C que permite definir novos dialectos de XML, *i.e.*, novos conjuntos de anotações organizados hierarquicamente que se destinam a descrever e a armazenar dados de forma semi-estruturada. Os documentos XML são documentos de texto autodocumentados e legíveis quer por aplicações, quer por humanos. A autodocumentação é obtida através de documentos do tipo *Document Type Definition* (DTD) ou XML *Schema Definition* (XSD) que especificam os elementos (anotações) e as regras de validação do dialecto XML em causa. Os documentos XML são ainda *case-sensitive, i.e.*, `<item>` é diferente de `<Item>`. O principal objectivo dos documentos XML é assegurar a troca e não a apresentação e formatação dos dados armazenados.

Existe um vasto conjunto de ferramentas especializadas que permitem definir novos dialectos XML, criar e verificar a validade e a correcta estruturação dos documentos, escrever e ler dados assim como extrair e converter a informação desejada para o formato de saída especificado, *e.g.*, XML, HTML, XHTML, *Portable Data Format* (PDF), *etc*. Estas operações são suportadas por um conjunto bastante alargado de linguagens e tecnologias que derivaram do XML: XML *Schema*, XML *Path Language* (XPath), XML *Query Language* (XQuery) XML *Stylesheet Language* (XSL) e XSL *Transformations* (XSLT). A especificação de dialectos XML tem proliferado exponencialmente nos domínios onde a troca de informação entre instituições e aplicações distintas é essencial. É o caso da linguagem *Keyhole Markup Language* (KML) utilizada pelo Google Earth para definir coordenadas geográficas ou da WSDL para descrever serviços *Web*, *etc*.

No Excerto de Código 4 apresenta-se um exemplo de um documento XML que descreve um conjunto de dados relativos a um objecto multimédia.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<items>
   <item>
  <ref>0123455</ref>
  <type>vídeo</type>
  <size>350</size>
  <code>MPEG-4</code>
  <duration>1000</duration>
  <quality>HD</quality>
  <class name="automobile">
     <maker>Toyota</maker>
     <model>Lexus IS</model>
     <year>2008</year>
    </class>
   </item>
</items>
```

Excerto de Código 4 Exemplo de Documento XML

A primeira linha indica qual a versão de XML e a codificação dos caracteres (ISO-8859-1) utilizada no documento XML. O atributo `standalone` é usado para indicar se o documento partilha (`no`) ou não (`yes`) dados com outros documentos XML. Caso haja partilha, o documento encontra-se dividido em entidades e inclui ficheiros externos.

A estrutura hierárquica dos dados depende do dialecto usado. Cada dialecto define o conjunto de anotações (elementos) e seus atributos e as regras da sua utilização [12]. No entanto, todo o documento XML tem uma anotação principal ou elemento raiz (*root element*) que classifica o tipo dos dados armazenados no documento. Neste exemplo, a anotação principal é `items`. Desse elemento derivam outros sub-elementos chamados elementos filho (*child elements*). É de notar que qualquer elemento poderá conter

elementos filhos, texto simples ou a mistura dos dois. Daqui resulta que os dados estão organizados em árvore – ver Figura 9.

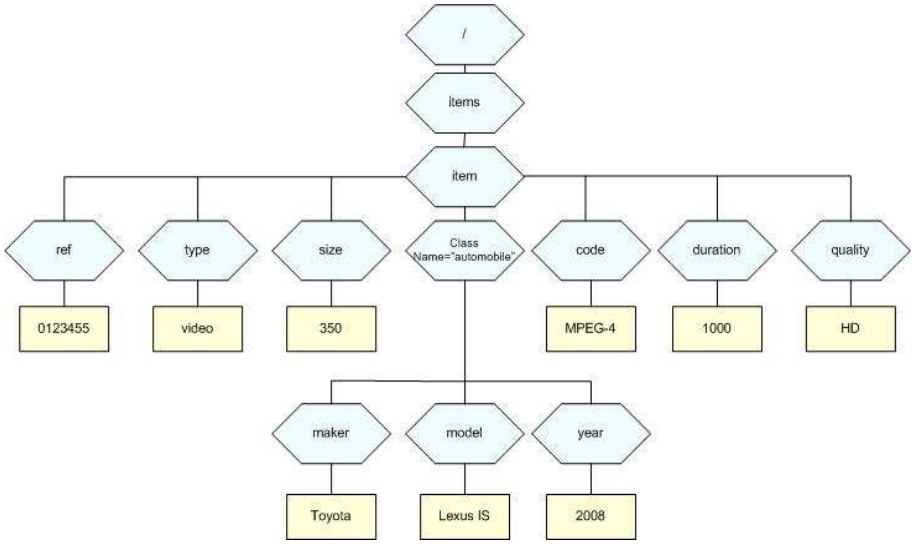


Figura 9 Exemplo de Árvore XML

Quando se afirma que um documento XML é bem formado significa que obedece a um conjunto de regras simples de sintaxe onde se incluem [12]:

- A abertura e fecho de todas as anotações utilizadas;
- A delimitação dos valores dos atributos das anotações com aspas simples ou duplas. Quando o valor do atributo inclui uma destes caracteres deve-se usar o outro tipo como delimitador;
- A exclusão de caracteres numéricos e de pontuação, do tipo *underscore*, hífen e do conjunto de caracteres XML da designação das anotações;
- A correcta ordenação das anotações, o que significa que devem ser fechadas pela ordem inversa à da sua abertura.

O processamento automático de um documento XML consiste na sua submissão a um analisador sintáctico (*parser*) que lê o documento, identifica e extrai os seus elementos constituintes (elementos de início e fim, atributos, conteúdos, *etc.*). De entre as diversas implementações de analisadores sintácticos de XML destacam-se a API *Document Object Model* (DOM) e a *Simple* API *for* XML (SAX).

3.2.2. SOAP

SOAP é um protocolo de comunicação baseado em XML que permite às aplicações trocarem informação com serviços *Web*. As mensagens SOAP são encapsuladas habitualmente em mensagens do protocolo HTTP que é suportado por todos os navegadores e servidores [113] daí a grande vantagem em termos de interoperabilidade que este protocolo pode oferecer à comunicação entre aplicações. Uma mensagem SOAP constitui a unidade de comunicação básica entre a aplicação cliente e o serviço *Web*. Cada um destes módulos deverá conter uma unidade lógica de comunicação denominada de SOAP *Node* para interpretar as mensagens SOAP recebidas (*receiver*) e enviar novas mensagens.

Uma mensagem SOAP é constituída por um elemento principal obrigatório designado por *Envelope* que é constituído por um cabeçalho opcional (*Header*); e um corpo obrigatório (*Body*) que contém a informação propriamente dita (solicitações e respostas). O cabeçalho pode ser simples ou composto por vários cabeçalhos (*HeaderBlock*) que fornecem indicações aos nós SOAP sobre como processar a mensagem (se se destina a esse nó, que parte da mensagem deve ser lida e interpretada, *etc*.). O corpo da mensagem pode comportar, para além dos dados trocados, um elemento destinado a reportar falhas (*Fault*). No Excerto de Código 5 encontra-se um exemplo de uma mensagem SOAP.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
   <soap:Header>
    …
   </soap:Header>
   <soap:Body>
    …
   <soap:Fault>
    …
   </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Excerto de Código 5 Mensagem SOAP [127]

3.3. UDDI

O *Universal Description Discovery and Integration* (UDDI) no âmbito do nosso trabalho irá servir de interface entre o agente da camada intermédia e os da camada inferior permitindo a interoperabilidade entre diferentes plataformas dos clientes com a plataforma JADE.

Por uma questão de facilidade, funcionalidade e comodidade, o serviço de registo do UDDI foi instalado no servidor HTTP do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia do Porto (DEE -ISEP), ficando desta maneira acessível de qualquer lado desde que se disponha de ligação à rede.

3.3.1. MySQL

A base de dados usada no âmbito deste trabalho serviu para suportar o funcionamento do UDDI. É nesta Base de dados que é mantida a informação relativa aos serviços e servidor onde se encontram disponibilizados.

Esta base de dados pertence ao serviço de registo do UDDI e comporta a descrição das empresas e serviços *Web* que são publicados para descoberta e consumo.

3.3.2. Instalação

Para instalar foram seguidos os passos sugeridos no manual do utilizador do jUDDI [117].

Para aceder à base de dados MySQL do UDDI foi usada a interface gráfica phpMyAdmin que é um programa desenvolvido em PHP *Hypertext Preprocessor* (PHP) para gerir bases de dados MySQL remotamente com recursos a um navegador. O acesso à Base de Dados do UDDI é feita pelo endereço `http://phpmyadmin.dee.isep.ipp.pt.`

3.3.3. Registo da Empresa

Para que cada agente possa registar os seus serviços no UDDI, tem de se registar uma empresa. A entidade que fizer esse registo irá ser a proprietária dos serviços *Web* oferecidos pelas entidades que desejem.

Esta operação é importante a nível da camada intermédia para permitir aos agentes encontrar os interlocutores adequados assim como os produtos que pretendem negociar.

O registo da empresa no UDDI foi feito automaticamente pelo *add-on* do JADE *Web Service Integration Gateway* (WSIG) que se encontra descrito na secção 3.8. Durante este processo é gerada a chave da empresa ou *business key* que neste caso foi "`7A7B3E00-00C5-11E1-BE00-979858722BD2`".

Trata-se de uma chave única para toda a solução e que os agentes das diversas plataformas irão usar para expor os serviços que disponibilizam. Assim, só será necessário efectuar este registo aquando a criação do Mercado por parte de uma Agência de Negociação.

3.4. JADE

Como foi já dito, escolheu-se a plataforma JADE para suportar o ambiente de execução dos agentes. A versão que foi usada nesta dissertação foi a última disponível em Julho de 2012 (Ver. 4.2.0) e para instalar esta plataforma foram seguidos os seguintes passos:

- Descarga do ficheiro `JADE-all-4.2.0.zip` a partir da página oficial [130]. Este ficheiro comprimido inclui a plataforma JADE já compilada (`JADE-bin-4.2.0.zip`), o código fonte (`JADE-src-4.2.0.zip`), documentação (`JADE-doc-4.2.0.zip`) e vários exemplos (`JADE-examples-4.2.0.zip`).
- Descompactação ficheiro `JADE-bin-4.2.0.zip` para uma pasta do disco.

Para desenvolver os agentes no IDE é necessário importar a biblioteca *jade.jar* para o NetBeans selecionando *Tools*, *Libraries*, *New Library ...*, "JADE", *Add JAR/Folder*, `jade.jar`.

3.5. NETBEANS

O NetBeans é um ambiente integrado de desenvolvimento – *Integrated Development Environment* (IDE) – desenvolvido pela Sun em Java pelo que, à partida, limitaria a escolha das tecnologias *Web* às ferramentas baseadas em Java. No entanto, a filosofia deste IDE é incluir as diversas tecnologias de suporte do tipo código livre. Assim, para além de desenvolvimento em Java, suporta também desenvolvimento em C/C++. Na área das linguagens dinâmicas, para além de Java Servlets e JavaServer Pages (JSP), oferece PHP, JavaScript, Ruby, Groovy, Python. Quanto ao desenvolvimento de serviços Web, permite desenvolver serviços baseados em SOAP ou RESTful e inclui acesso a diversas API RESTful, *e.g.*, Google, Facebook, Yahoo, flickr, Amazon, Twitter, *etc.*, e baseadas em SOAP, *e.g.*, o StrikeIron. As possibilidades de desenvolvimento do ambiente e destas ferramentas são imensas. A sua aplicabilidade a este trabalho advém das seguintes características:

- A área de trabalho é-nos apresentada de uma forma muito organizada e modular o que permite uma rápida identificação de todos as partes e segmentos constituintes da

aplicação que estamos a implementar. Após se ter efectuado a depuração do código, gera-se através do comando `build` um ficheiro comprimido com extensão `war` – *Web Application Archive* (WAR). Este ficheiro que, contém toda a aplicação *Web*, serve para fazer o *deployment* no servidor HTTP. No caso do servidor de aplicações Apache Tomcat, basta copiá-lo para a pasta `webapps` porque, ao ser detectado, é automaticamente descomprimido gerando toda a estrutura de directórios da aplicação *Web*. E de notar que esta árvore de directórios, entretanto criada, apresenta-se devidamente estruturada e inclui todos recursos e configurações necessárias ao funcionamento do portal.

- Integra servidores de aplicações (Apache Tomcat, Glassfish, WEBrick), o servidor de base de dados Java DB (Apache Derby) e controladores de acesso para servidores de bases de dados MySQL, PostgreSQL e outros.
- A última versão do NetBeans já inclui o pacote Visual Web Pack que coloca à disposição do utilizador, um conjunto de componentes gráficos para a construção das páginas. De forma rápida e visual é possível arrastar os objectos pretendidos para as páginas e construir rapidamente o seu aspecto gráfico. Esses objectos possuem propriedades manipuláveis e é possível associar-lhes um determinado comportamento, associando-os a JavaBeans (de aplicação, sessão, requisição ou página) dependendo do âmbito desse comportamento.
- Possui um editor de código muito poderoso, fornecendo ajuda à medida que se vai escrevendo o código Java.
- Possui ferramentas de validação de formulários bastante práticas, permitindo poupar tempo de desenvolvimento.

3.5.1. INSTALAÇÃO

A última versão estável disponível [8] do NetBeans IDE é a 7.1.2. Cada versão inclui um vasto conjunto de pacotes de tecnologias que podem ser escolhidos de acordo com as necessidades. Apesar de ter sido instalado o pacote que inclui todas as tecnologias disponíveis (`netbeans-7.1.2-ml-windows.exe`), este projecto necessita apenas do pacote da linguagem Java e do servidor de aplicações Tomcat. A instalação do IDE tem como pré-requisito a instalação prévia do Java Development Kit (JDK) versão 1.6.0 ou superior [7].

É necessário proceder a algumas configurações para compilar e correr o ambiente de execução JADE a partir do IDE NetBeans. A partir do item principal dentro do separador *Projects*, selecciona-se *Properties* e *Run*, surgindo o formulário da Figura 10.

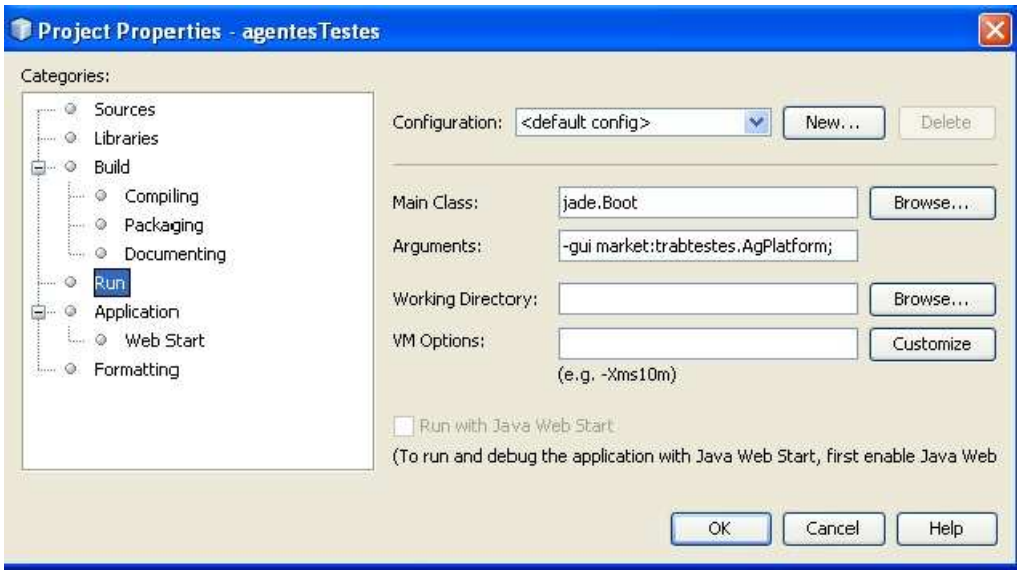


Figura 10 Execução da Plataforma JADE no NetBeans

No campo *Arguments* especificam-se os parâmetros do comando de execução do JADE. O significado dos parâmetros é o seguinte:

- `-gui` indica que é para lançar a GUI da plataforma JADE;
- `market` é o nome do agente a lançar na plataforma;
- `trabtestes` é o nome do projecto de desenvolvimento;
- `AgPlatform` é o nome da classe de Java que corresponde ao agente `market`.

Depois de instalado e configurado o NetBeans, foi necessário instalar um conjunto de bibliotecas necessárias ao desenvolvimento do trabalho.

3.6. JFREECHARTS

O JFreecharts disponibiliza um conjunto de bibliotecas que permite traçar vários tipos de gráficos.Para adicionar o JFreecharts ao projecto, bastou incluir a biblioteca `jfreechart-1.0.11.jar` [128] à pasta *Libraries* do projecto.

3.7. CHART2D

O chart2D disponibiliza um conjunto de bibliotecas de suporte a gráficos 2D [129] para traçar e configurar gráficos lineares, tarte, barras, *etc*. Depois de descarregar o chart2D, descomprime-se o ficheiro `jar`, fazendo `jar –xf Chart2D_1.9.6k.jar`, e importa-se o ficheiro `Chart2d.jar` da subpasta `net` para o projecto.

3.8. WSIG

Os serviços *Web* são uma forma padronizada de interligar diferentes aplicações. O *add-on* do JADE *Web Service Integration Gateway* (WSIG) permite aos agentes JADE criar, publicar e expor de serviços *Web*. A arquitectura do WSIG é apresentada na Figura 11.

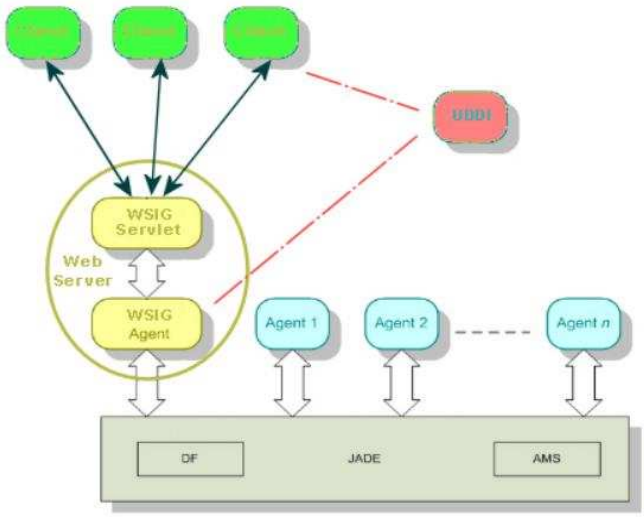


Figura 11 Arquitectura do WSIG [20]

O WSIG é composto por dois elementos principais:

- **WSIG Servlet** – Recebe as solicitações HTTP/SOAP, extrai a mensagem SOAP, prepara a invocação da acção do agente e entrega-a WSIG Agent. Uma vez executada, converte o resultado da acção numa mensagem SOAP e prepara o envio da mensagem ao cliente que solicitou o serviço.
- **WSIG Agent** – Encaminha as invocações de acções recebidas do WSIG Servlet aos agentes correspondentes e obtém as respectivas respostas, regista e desregista o serviço no DF e cria o WSDL correspondente ao serviço registado. Por fim, caso seja usado, publica o serviço no UDDI. A UDDI.org definiu um conjunto de linhas de orientação para a publicação de serviços no UDDI em [120].

3.8.1. INSTALAÇÃO

A instalação do WSIG [20] requer a pré-instalação do Java JRE v5.0, JADEv3.5 e de um *Servlet Container* como, por exemplo, o Apache Tomcat.

Depois de descarregar o WSIG a partir da página oficial do JADE [130], descomprime-se o ficheiro `wsigAddOn-2.3.zip` para a pasta `JADE/add-ons`.

3.8.2. CONFIGURAÇÃO

Para registar serviços no UDDI foi necessário configurar o WSIG para trabalhar com o UDDI e indicar ao WSIG Servlet os dados da empresa (nome e respectiva chave) através da edição do ficheiro de configuração WSIG `(webapps\wsig\conf\wsig.properties)` de cada plataforma e acrescentar as linhas:

```
uddi.enable=true
uddi.businessKey=7A7B3E00-00C5-11E1-BE00-979858722BD2
uddi.userName=wsig
```

Para o WSIG Servlet publicar e procurar serviços no UDDI acrescentaram-se as linhas:

```
uddi.queryManagerURL=http://ave.dee.isep.ipp.pt:8080/juddi/inquiry
uddi.lifeCycleManagerURL=http://ave.dee.isep.ipp.pt:8080/juddi/publish
```

Para assegurar que são gerados os ficheiros *Web Service Description Language* (WSDL) respectivos aos serviços de cada agente e respectivo `endPoint` é necessário incluir:

```
wsdl.style=document
wsdl.localNamespacePrefix=impl
wsdl.writeEnable=true
wsdl.directory=wsdl
wsig.servicesURL=http://localhost:8080/wsig/ws
```

Finalmente, para que o WSIG saiba qual a ontologia adoptada para representar os serviços e acções dos agentes, é necessário acrescentar:

```
onto.NegPub=brokerage.NegPub.ontology.NegPubOntology
```

Segundo o manual de utilização, o WSIG consegue utilizar apenas uma única ontologia [20].

3.9. WSDC

O JADE *Web Service Dynamic Client* (WSDC) *add-on* é uma API desenvolvida pela Telecom Itália que permite aos agentes invocar serviços expostos por outros agentes sem ter necessidade de gerar classes adicionais [10]. A partir do `endPoint` e do nome do serviço, o WSDC cria automaticamente o conjunto de classes de suporte (*stubs* e *skeletons*) para interagir directamente com os serviços *Web* disponibilizados. A WSDC permite de

uma maneira transparente interagir com o serviço e consultar, por exemplo, todo o conteúdo do ficheiro WSDL de descrição de um serviço *Web*.

3.9.1.1. INSTALAÇÃO

Para instalar assim como obter a descrição da API completa e manual, basta descarregar e descomprimir o ficheiro do WSDC [130] fornecido para uma pasta no computador. Todos os ficheiros `jar` devem ser importados para as bibliotecas do projecto do NetBeans.

3.10. TOMCAT

O servidor de aplicações escolhido foi o Apache Tomcat. O Tomcat disponibiliza as tecnologias Java Servlet e JavaSever Pages e aloja ainda, neste caso, as aplicações *Web* Axis2/Java, WSIG (*Servlet Agent* do WSIG) e jUDDI necessárias à interacção dos agentes JADE com o WSIG e o UDDI. Para o desenvolvimento foi usada a solução integrada de instalação do Tomcat com o IDE NetBeans que torna transparente a publicação (*deploy*) no servidor dos serviços.

3.10.1. AXIS2/JAVA

O Axis2/Java é um *framework* de processamento de serviços Web, SOAP e WSDL desenvolvido pela Apache de código livre que suporta serviços *Web* dos tipos RPC e REST. Inclui, para além da API de desenvolvimento, um conjunto de ferramentas específicas de apoio à criação e interacção com serviços *Web*.

3.10.1.1. INSTALAÇÃO

Por uma questão prática, optou-se pela integração do Axis2/Java no IDE NetBeans. Esta operação consistiu na:

- Descarga do ficheiro `axis2-1.5.4-war` da respectiva página oficial [112];
- Cópia do ficheiro `war` para a pasta `CATALINA_BASE/webapps`. Para se obter a localização da pasta de instalação do Apache Tomcat – `CATALINA_BASE` – selecciona-se o separador *Services* do NetBeans, *Servers*, Apache Tomcat 6.0.26 e com o botão direito do rato sobre o Apache Tomcat 6.0.26 selecciona-se *Properties*;
- Rearranque do servidor Tomcat para descomprimir o ficheiro `war` para a subpasta `CATALINA_BASE/Webapps/axis2`.

- Selecção no NetBeans do menu *Tools + Options* e, nas opções disponibilizadas para o Axis2, especificar o caminho `CATALINA_BASE/Webapps/axis2` onde irão ser guardados os ficheiros com extensão `aar`. De seguida, deve indicar-se o porto do servidor Tomcat (*e.g.*, `http://localhost:8084/axis2`) e, por fim, seleccionar-se *'Use Tomcat for Deployment'* e introduzir as credenciais de administração, evitando-se ter de rearrancar o servidor cada vez que se altera o ficheiro `axis2.war`.

A página principal do Axis2/Java fica disponível em `http://localhost:8084/axis2`.

3.11. PROTÉGÉ

Segundo a sua página oficial [119], o Protégé é uma ferramenta com interface gráfica, intuitiva, de fácil utilização e independente da plataforma para criar e editar ontologias e criar bases de conhecimento em variados formatos de representação. O Protégé foi desenvolvido em linguagem Java e disponibiliza um conjunto de bibliotecas que permitem, por exemplo, a uma aplicação externa aceder directamente ao conteúdo de uma ontologia OWL. Nesta ferramenta existem duas formas principais de definição de ontologias: o editor de *frames* ou o editor OWL. No primeiro caso, as ontologias resultantes incluem um conjunto de classes organizadas numa hierarquia de subordinação dos conceitos de um determinado domínio, um conjunto de *slots* associados às classes, que descrevem as suas propriedades e forma como se inter-relacionam, e por instâncias dos conceitos que possuem valores específicos nas suas propriedades. No caso do editor de OWL, as ontologias criadas incluem descrições de classes, propriedades e suas instâncias. Para criar ontologias *frame-based* deve-se usar a versão 3.4, uma vez que as versões 4.x deixaram de suportar este modelo. Para o desenvolvimento de ontologias OWL é recomendada a versão 4 [131].

O Protégé possui um conjunto alargado de extensões como o TagViz, OntoViz, BeanGenerator ou o OWLViz que são fruto do contributo de uma comunidade de programadores e investigadores bastante activa. A extensão mais utilizada é a OWLViz que permite a visualização gráfica de ontologias OWL. Em `http://protege.cim3.net/cgi-bin/wiki.pl?ProjectsThatUseProtege` apresenta-se uma lista de projectos que recorreram ao Protégé para criar e editar ontologias.

3.11.1. INSTALAÇÃO

A versão do Protégé utilizada foi a 3.1.1 (build 216) [132] por ser compatível com a versão do BeanGenerator usada. Após a instalação do Protégé, é necessário importar o modelo de referência de ontologias JADE que faz parte do *plug-in* BeanGenerator.

3.11.2. BEANGENERATOR

O *plug-in* OntologyBeanGenerator permite gerar o conjunto de classes Java de representação dos conceitos definidos numa ontologia *frame-based*. Este conjunto de classes pode, depois de compilado numa única biblioteca e criado o respectivo ficheiro `JAR`, ser utilizado para representar o conhecimento do domínio da ontologia por qualquer aplicação Java.

Instalou-se a versão 3.2.1 do OntologyBeanGenerator porque a versão 4 não é incompatível com a plataforma JADE – consultar a resposta dada por Caire Giovanni da Telecom Itália [133]. O ficheiro descarregado [134] contém as bibliotecas que permitem gerar as classes de Java. A importação do *plug-in* consiste em descomprimir o ficheiro descarregado para `…\Protege_3.3.1\plugins`.

Por uma questão de eficiência adoptou-se a seguinte metodologia de criação e actualização de ontologias:

- Definir ou alterar a ontologia no Protégé
- Gerar as classes através do *plug-in* BeanGenerator directamente para a pasta `src` de um projecto previamente criado no NetBeans:
  `…\Dropbox\Luis Sousa\Implementacao\Ontologia\NegPub\src` para a ontologia `NegPub`;
  `…\Dropbox\Luis Sousa\Implementacao\Ontologia\negMComp\src` para a ontologia `negMComp`.
- Compilar no NetBeans o projecto cada vez que se altera a Ontologia. O ficheiro `JAR` é automaticamente gerado e armazenado na pasta `dist`.
- Importar a biblioteca gerada para o projecto de trabalho, seleccionando *Libraries* e *Add JAR/Folder*.

3.12. CONCLUSÃO

O ambiente de desenvolvimento instalado é integralmente suportado pela linguagem de desenvolvimento Java. O ambiente de desenvolvimento integrado NetBeans permite

reutilizar o vasto conjunto de bibliotecas e ferramentas escolhidas. O *plug-in* BeanGenerator permite importar e utilizar as ontologias definidas no Protégé para a plataforma JADE de forma elegante e transparente.

Para assegurar o trabalho colaborativo, todas as bibliotecas, projectos, código fonte e informação relevante foi mantida numa pasta de acesso partilhado.

As tecnologias adoptadas para o desenvolvimento são gratuitas e *open source* e garantem um elevado grau de interoperabilidade, escalabilidade e adaptabilidade. Estas propriedades irão robustecer a plataforma, permitindo-lhe acompanhar evoluções futuras quer em capacidade, quer em complexidade.

4. DESENVOLVIMENTO

Neste capítulo descreve-se a etapa de desenvolvimento do sistema. Apresenta-se a arquitectura, descrevem-se os seus módulos constituintes e respectivas funcionalidades

4.1. ARQUITECTURA

Esta tese tem como objectivo propor um modelo de mercado virtual para a negociação automática de anúncios (componentes multimédia que deverão estar descritos no formato MPEG-7) em tempo útil de forma descentralizada e personalizada, *i.e.*, em função do perfil do espectador. Na Figura 12 apresentam-se as entidades envolvidas neste cenário.

Figura 12 Interlocutores na Negociação

Na proposta existem três tipos de interlocutores com interesses e objectivos diferentes na negociação:

- Produtores de Conteúdos: são entidades que detêm componentes multimédia e que pretendem negociar neste tipo de mercado. Poderão ser agências publicitárias ou de marketing que representam determinado artigo ou serviço. Obviamente, a categorização dos respectivos anúncios é da sua responsabilidade, *i.e.*, deverão pré-definir o tipo de público-alvo. Estas empresas pretendem adquirir tempo de transmissão para os seus anúncios e, portanto, desempenham o papel de compradoras no processo de negociação. Deverão previamente estabelecer um contrato com pelo menos uma Agência de Negociação.
- Distribuidores de Conteúdos: são empresas que distribuem (ou difundem) conteúdos. Dado que o objectivo final é seleccionar um conjunto de anúncios compatíveis com o perfil do espectador, esta proposta é mais adequada para empresas que suportam a difusão personalizada de conteúdos, *e.g.*, *Video on Demand* (VoD). Estas empresas pretendem vender os intervalos de publicidade (dos seus espectadores) às agências produtoras para serem ocupados por um alinhamento personalizado de anúncios e, portanto, desempenham o papel de vendedoras no processo de negociação. São responsáveis pela gestão e categorização dos intervalos em função do perfil de cada espectador. O perfil do espectador deverá ser construído com base em todas as interacções entre o espectador e o distribuidor, *e.g.*, escolha de programação, duração das visualizações, contexto temporal ou preferências expressas (um espectador pode indicar que sobre determinadas condições está disponível para receber determinado tipo de publicidade em troca de um benefício como a possibilidade de visualizar determinado filme ou programa). Deverão previamente celebrar um contrato com pelo menos uma Agência de Negociação.
- Agências de Negociação: são empresas que disponibilizam o suporte legal e tecnológico (*software*) ao processo de negociação e desempenham ainda o papel de moderação entre as partes envolvidas. Devem manter uma base de dados com toda a informação relevante dos componentes negociados e disponibilizar relatórios genéricos e estatísticos que permitam monitorizar o sistema e avaliar o desempenho do mercado. Estes relatórios, que são relevantes tanto para os Distribuidores de Conteúdos como para os Produtores de Conteúdos, deverão ser disponibilizados (ou eventualmente vendidos) por intermédio de um serviço *Web*. Estas agências por não terem um interesse

directo nas negociações também poderiam ser os proprietários das plataformas de suporte do sistema num contexto distribuído.

Entre cada um dos interlocutores empresariais deverá existir uma relação contratual que formaliza as operações efectuadas dentro da plataforma. A entidade responsável pelo controlo do mercado não tem interesse directo nas operações, mantém uma base de dados com o resultado de todas as negociações e disponibiliza relatórios gerais de negociação que podem ser consultados por terceiros. O modelo proposto pode ser dividido em três camadas funcionais com objectivos diferenciados – ver Figura 13.

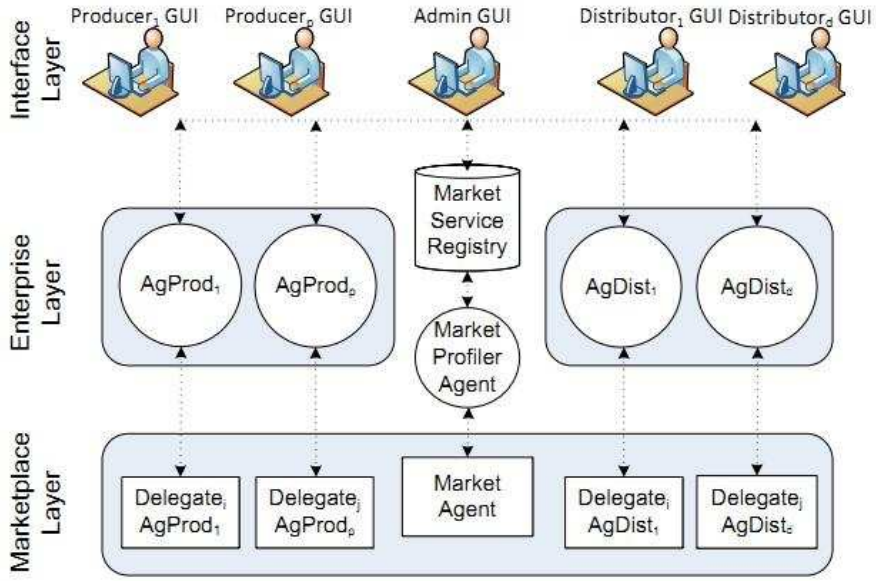


Figura 13 Arquitectura da Plataforma [2]

A **camada superior** – *Interface Layer* – é constituída pelas entidades interessadas na compra e venda dos componentes multimédia. Deverá ser disponibilizado um *front-end* apropriado para a interacção (homem – máquina) e introdução de toda a informação relevante à negociação.

A **camada intermédia** – *Enterprise Layer* – é constituída pelos agentes que representam as empresas distribuidoras e produtoras na plataforma. Estes agentes recebem a informação da camada superior, coordenam a sua actividade na plataforma e geram relatórios de resposta. É nesta camada que é mantida toda a informação da negociação e é efectuado o *matching* entre o perfil dos espectadores e as categorias dos anúncios dos produtores e a gestão dos contratos.

Finalmente, a **camada inferior** – *Marketplace Layer* – é onde a negociação decorre entre agentes delegados da camada intermédia.

É importante notar que a intenção de venda ou compra tem de ser transversal a todo o processo negocial, *i.e.*, só existe negociação se houver uma real intenção de compra e venda. Esta clarificação simplifica o processo de procura uma vez que reduz o universo de artigos disponíveis aos componentes multimédia anotados em MPEG-7, *i.e.*, devidamente caracterizados.

4.2. DOMÍNIO DE APLICAÇÃO

O objectivo deste projecto é o desenvolvimento da *Marketplace Layer*, incluindo a criação do agente de mercado e dos agentes delegados, a implementação dos protocolos de negociação, *etc*., e do processo de comunicação entre esta camada e a *Enterprise Layer*. De forma a demonstrar e fundamentar as funcionalidades, foram também desenvolvidos os agentes da *Enterprise Layer*, tendo sido criados as funcionalidades suficientes para a inserção de dados (necessários à *Marketplace Layer)*, a obtenção de resultados e a visualização e arquivamento das respectivas respostas.

Cada entidade faz-se representar na camada intermédia por um único agente. Estes agentes representam as empresas distribuidoras e produtoras na plataforma, mantêm a lista de itens (anúncios ou intervalos) que possuem em carteira e são responsáveis pela participação da sua empresa no mercado. Cada item possui um perfil de negociação específico que inclui os valores limite do preço, a táctica de negociação e outras características.

Os agentes distribuidores desencadeiam o processo de negociação sempre que se avizinhar o intervalo de um espectador. Nessa altura, procuram e convidam os agentes produtores com produtos compatíveis com o perfil do espectador para irem ao mercado negociar o preenchimento do intervalo.

Uma vez aceite o convite, cada um dos agentes da camada intermédia envolvidos lança um agente delegado no mercado. A negociação efectua-se em relação a uma única dimensão: o preço em €/s da transmissão durante o intervalo de um espectador. Cada agente delegado tem por objectivo a venda (delegado de distribuidor) ou compra (delegado de produtor) de um único artigo multimédia (porção de um intervalo ou anúncio) de acordo com as regras

do protocolo de negociação que o mercado disponibiliza e usando uma táctica e valores limites bem definidos.

4.2.1. BASE DE CONHECIMENTO

O conjunto de conceitos de um domínio pode ser definido e até armazenado em bases de conhecimento designadas ontologias. Cada agente tem acesso à sua ontologia privada assim como a outras que partilha com os outros agentes de forma a comunicar e a possibilitar o negócio. Uma vez que o objectivo deste trabalho é o da prova de conceito, isto é, mostrar que é possível adoptar este tipo de tecnologia para negociar artigos multimédia, irão ser usadas ontologias uniformes não havendo por isso, necessidade de efectuar tradução ou mapeamento entre ontologias diferentes. A base de conhecimento é constituída pelo conjunto de instâncias da ontologia do domínio adoptada.

O protocolo de comunicação entre agentes é suportado por estruturas de dados que foram especificadas usando a ferramenta Protégé V3.3.1 da Stanford Medical Informatics conjuntamente com o *plug-in* BeanGenerator V3.2.1 para gerar as classes de Java adequadas à sua utilização pelos agentes no ambiente de execução JADE. No entanto, para que estas classes possam ser utilizadas pelos agentes terão que ser subclasses da ontologia genérica *Content Reference Model* (CRM) do JADE chamada `abstractJadeOnt`.

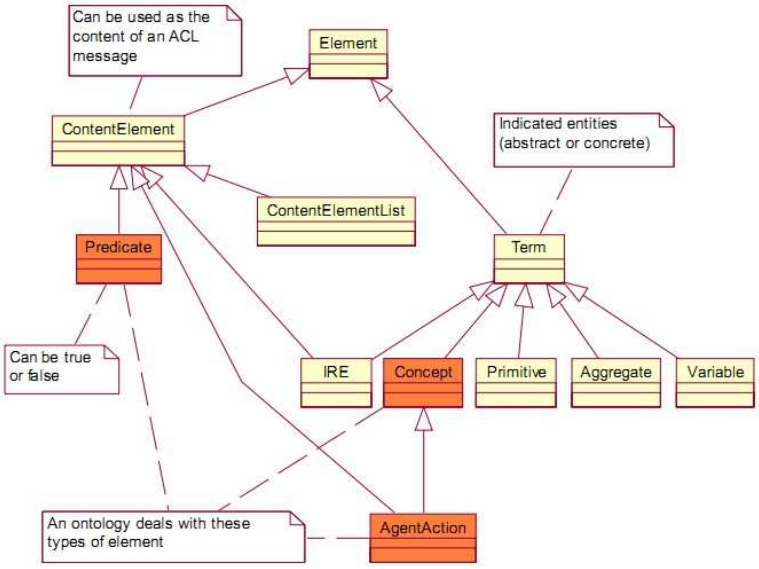


Figura 14 Ontologia *Content Reference Model* [18]

A Figura 14 contém os elementos e as relações disponíveis para os agentes utilizarem no conteúdo das suas mensagens. Os elementos base de definição das ontologias são:

- **Conceitos** – definições do conhecimento do domínio dos agentes;
- **Predicados** – servem avaliar e validar relações entre conceitos (booleanos);
- **Acções de agente** – representam as tarefas que os agentes podem realizar.

Não houve necessidade de validar as relações entre as expressões, fazendo-se apenas uso do conceito `AgentAction` para definir o processo de decisão e comportamento do agente. A definição dos conceitos permitiu a utilização reiterada das estruturas associadas.

Foram definidas duas ontologias com objectivos ligeiramente diferentes: a `NegMComp` (*Negotiate Multimedia Component*), que é usada na comunicação interna da camada inferior de mercado entre agentes delegados e a ontologia `NegPub` (*Negotiate Publicity*), que é usada na comunicação entre as camadas intermédia e de mercado. A Figura 15 ilustra a separação entre as duas ontologias.

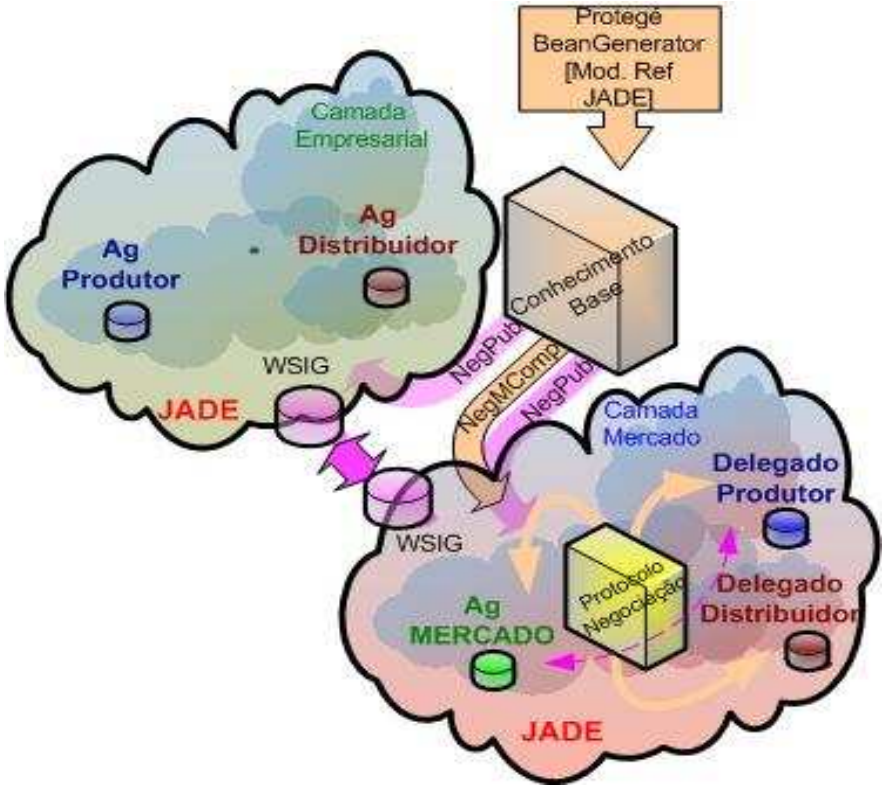


Figura 15 Gestão do Conhecimento

Os agentes delegados da camada inferior utilizam as duas ontologias uma vez que negoceiam entre si, usando uma variante do protocolo *Iterated Contract Net* – o *Fixed* ICNet – em conjunto com a ontologia `NegMComp` e, ao mesmo tempo, interagem com os agentes correspondentes da camada superior, usando a ontologia `NegPub`.

4.3. DEFINIÇÃO DAS ONTOLOGIAS

As ontologias foram desenvolvidas com a ferramenta Protégé. A hierarquia de conceitos definida é convertida, usando o *plug-in* BeanGenerator do JADE, em JavaBeans prontos a ser utilizados pelos agentes da plataforma. É o caso da ontologia `NegPub` que, ao ser partilhada por todos os agentes da plataforma, assegura que as mensagens trocadas entre agentes de camadas diferentes são correctamente interpretadas. Esta opção foi intencional para evitar recorrer a técnicas de mapeamento entre ontologias que tornaria este projecto mais complexo e não constituiria uma mais-valia para alcançar os objectivos propostos. Como já referimos foi proposto o uso de duas ontologias: Uma que irá conter os conceitos, estratégias e regras de mercado que os agentes usarão para efectuar os negócios, e outra para suportar todos os conceitos e características inerentes aos próprios artigos que se pretende negociar.

As duas ontologias foram definidas sobre a ontologia base do JADE chamada `SimpleJADEAbstractOntology` [134]. Esta ontologia define dois conceitos (`AID` e `AgentAction`) e uma classe especial `Predicate`. O `AID` representa os agentes e o `AgentAction` representa as acções que os agentes executam. A classe `Predicate` serve para testar relações entre conceitos.

Para tornar mais simples a descrição das ontologias desenvolvidas são apresentadas vistas parciais dos conceitos e aspectos mais relevantes, recorrendo ao *plug-in* OntoViz do Protégé. Os conceitos estão organizados hierarquicamente em classes e subclasses através de relações de herança, *i.e.*, as subclasses herdam todas as características da superclasse (*slots* e instâncias). Neste projecto a ontologia não é utilizada para armazenar a base de conhecimento mas para definir a estrutura do conhecimento. As instâncias que são criadas pelos agentes no curso da sua actividade são instâncias dos JavaBeans gerados através do *plug-in* BeanGenerator.

4.3.1. ONTOLOGIA NEGPUB

A `NegPubOntology` representa a estrutura do conhecimento da plataforma, incluindo a informação trocada entre as camadas inferior e intermédia. Nesta ontologia foram definidos 3 tipos de conceitos: `AgentType`, `AgentAction`, `AgentData`.

4.3.1.1. AGENTTYPE

O conceito `AgentType` apresentado na Figura 16 define o tipo de agente interveniente no processo. Este conceito é especializado para representar os agentes delegados da camada inferior (`DelegateAgent`), os agentes da camada intermédia (`EnterpriseAgent`), os agentes que responsáveis pelas camadas (`LayerAgent`) e, por fim, os agentes da camada superior (`InterfaceAgent`). Estes dois últimos estão fora do âmbito desta tese pelo que não irão ser mais detalhados.

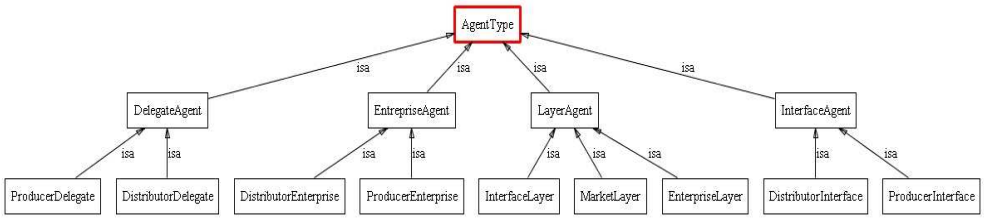


Figura 16 Conceito `AgentType`

O conceito `DelegateAgent` ilustrado na Figura 17 é especializado nos dois tipos de agentes que intervêm na negociação da camada inferior: `DistributorDelegate` e `ProducerDelegate`.

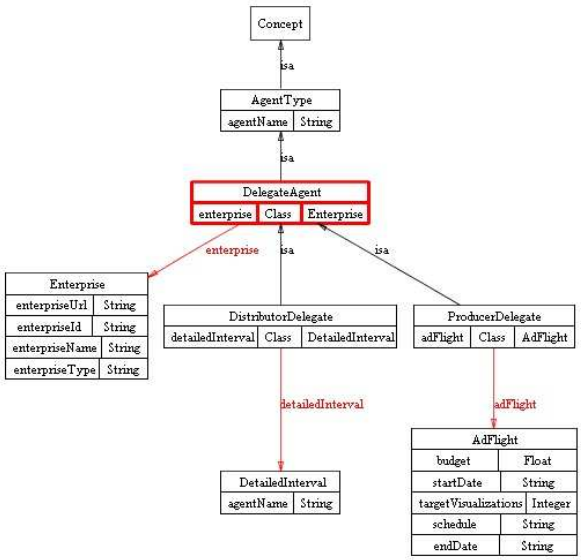


Figura 17 Conceito `DelegateAgent`

A Figura 17 apresenta ainda os conceitos e as propriedades (*slots*) correspondentes. As *slots* podem conter tipos primitivos de dados (`float`, `int`, *etc.*), mas também admitem outro

tipo de dados mais complexos como classes (*e.g.*, `DelegateAgent` contém uma única *slot* do tipo complexo `Enterprise` chamada `enterprise`).

Pode-se verificar que os agentes delegados dos distribuidores e dos produtores partilham o conceito `Enterprise`.

4.3.1.2. AGENTDATA

O conceito `AgentData` (Figura 18) representa os objectos que são criados e trocados pelos agentes: `Ad`, `Product`, `Interval`, `NegotiationProfile`, `NegotiationResult` e `Contract`. Alguns destes conceitos são ainda representados com maior detalhe, sendo especializados noutras classes: `AdFlight`, `DetailedProduct`, `DetailedInterval`, `AdProfile`, `IntervalProfile`, `IntervalResult` e `AdResult`.

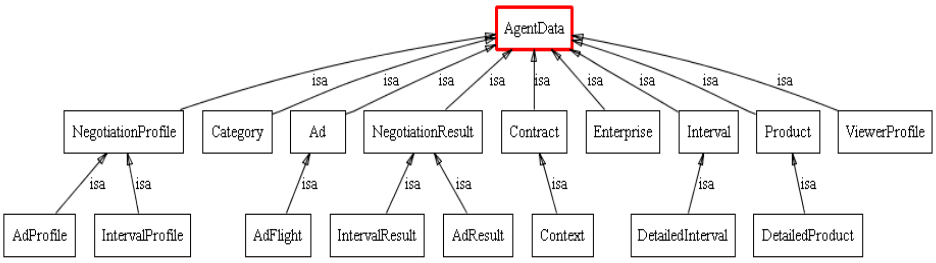


Figura 18 Conceito `AgentData`

4.3.1.3. AGENTACTION

O conceito `AgentAction` representado na Figura 18 agrupa as acções dos agentes. Nesta ontologia as acções dos agentes estão associadas aos serviços que cada agente disponibiliza e expõem no UDDI para consumo interno da plataforma. Por exemplo, a acção `GetProduct` é usada pelos agentes `DistributorEnterprise` para obter dos agentes `ProducerEnterprise` os produtos (anúncios) que possuam a determinadas características.

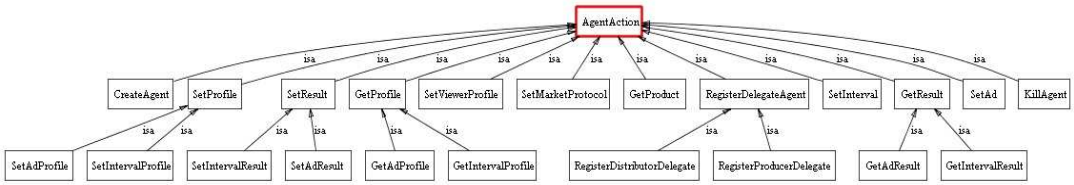


Figura 19 Conceito `AgentAction`

A execução de uma acção pressupõe uma resposta que pode ser um tipo primitivo de dados (`float`, `int`, *etc.*) ou complexo (*e.g.*, uma instância de classe) [19]. Assim, é necessário especificar na ontologia o tipo de dados retornado por cada acção. Dado que o Protégé não permite essa definição, editou-se o JavaBean da classe principal da ontologia e especificou-se o tipo de dados retornado. No caso da ontologia `NegPub` o JavaBean é a classe `NegPubOntology.java`. No Excerto de Código 6 apresenta-se a definição do tipo de dados retornado pelas acções `getProduct`, `getIntervalProfile`, `getAdProfile`, `registerDistributorDelegate` e `registerProducerDelegate`.

```
getProductSchema.setResult(adProfileSchema);
getIntervalProfileSchema.setResult(intervalProfileSchema);
getAdProfileSchema.setResult(adProfileSchema);
registerDistributorDelegateSchema.setResult(
          (TermSchema)getSchema(BasicOntology.STRING)
   );
registerProducerDelegateSchema.setResult(
          (TermSchema)getSchema(BasicOntology.STRING)
   );
```

Excerto de Código 6 Especificação do Resultado das Operações

Por exemplo, a acção `getIntervalProfile` tem como resultado uma instância `intervalProfile` cuja classe foi definida como um sub-conceito `NegotiationProfile` que por sua vez é um sub-conceito de `AgentData` – ver Figura 20.

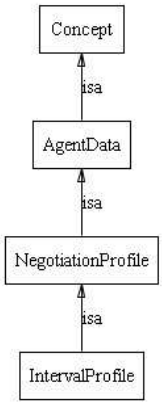


Figura 20 Hierarquia do Conceito `IntervalProfile`

Dado que cada plataforma JADE possui apenas um agente WSIG [20], pode apenas utilizar uma ontologia de definição dos conceitos envolvidos na exposição e consumo de serviços *Web*. Esta ontologia é partilhada por todos os agentes criados na plataforma, fazendo com que todos exponham as mesmas operações – é o caso da `NegPub`. No entanto, o JADE permite que cada tipo de agente decida que operações expor. Para tal é necessário incluir na ontologia uma classe adicional por cada tipo de agente onde se indicam as operações a

suprimir. Assim, foram acrescentadas à ontologia mais três classes (`NegPubOntoMapMarket`, `NegPubOntoMapDistributor`, `NegPubOntoMapProducer`). O Excerto de Código 7 apresenta a classe `NegPubOntoMapProducer` que corresponde aos agentes produtores da camada intermédia.

```
package brokerage.NegPub.ontology;
import com.tilab.wsig.store.SuppressOperation;
public class NegPubOntoMapProducer {
   @SuppressOperation
   public CreateAgent toCreateAgent() { return null;}
   …
}
```

Excerto de Código 7 Supressão de Operações

Este código exemplifica a supressão da operação `CreateAgent`, significando que os agentes produtores não criam agentes na camada intermédia.

4.3.2. ONTOLOGIA NEGMCOMP

Esta ontologia é usada na camada de mercado e define a estrutura da informação trocada entre os agentes delegados do mercado durante a negociação. Partilha alguns conceitos com a ontologia `NegPub` designadamente o perfil de negociação do componente multimédia (anúncio ou intervalo) que define o comportamento que o agente irá assumir durante a negociação. Esta ontologia comporta três tipos de conceitos ilustrados na Figura 21: `MultimediaComponent`, `Negotiation` e `AgentAction`.

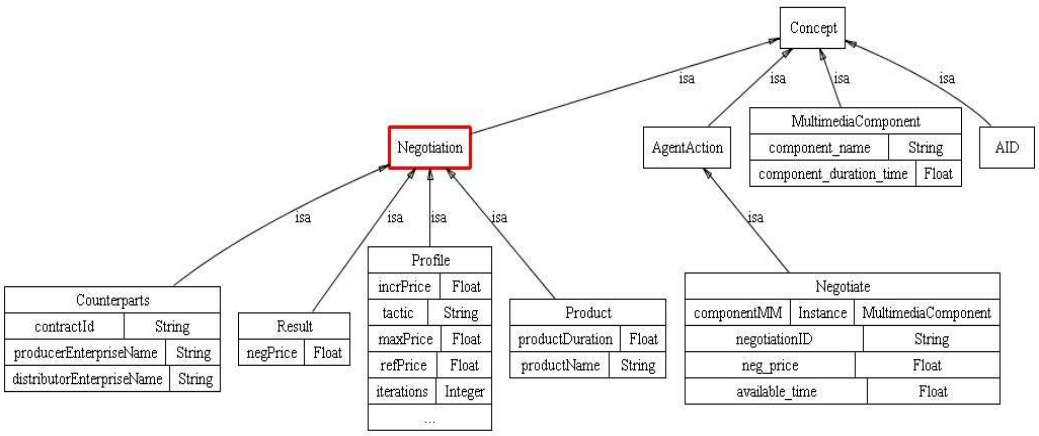


Figura 21 Conceito `AgentNegotiation`

O conceito `Negotiation` inclui alguns dos conceitos da ontologia da plataforma `NegPub`. Optou-se por não se reutilizar a `NegPub` para se isolar o processo de negociação que decorre mercado das restantes camadas, adicionando à `NegMComp` os elementos necessários como a

identificação dos intervenientes na negociação, as características do componente multimédia, o perfil de negociação e o tipo de resultado a reportar.

Esta ontologia é exclusiva dos agentes delegados da camada de mercado. Adicionalmente, utilizam a ontologia da plataforma `NegPub` para consumir os serviços expostos pelos agentes da camada intermédia.

4.4. INTERFACE GRÁFICA

Para testar o protótipo foi criada uma interface gráfica minimalista composta por um conjunto de janelas disponibilizadas pelos próprios agentes. Num futuro próximo esta interface será substituída por módulos específicos associados aos agentes da camada de topo.

4.4.1. AGENTE DE MERCADO

O agente de mercado, responsável pela camada inferior, é lançado conjuntamente com a própria plataforma. Este agente disponibiliza a interface de criação dos agentes da camada intermédia apresentada na Figura 22.

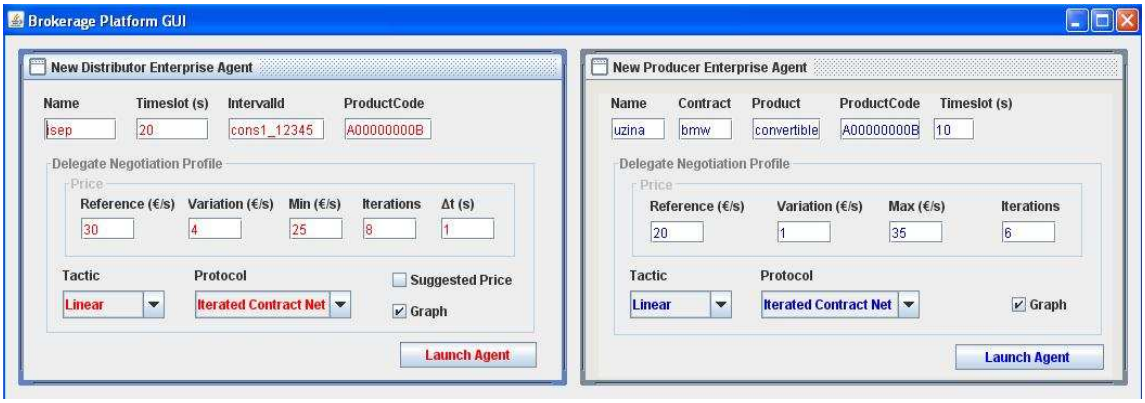


Figura 22 GUI do Agente de Mercado

Nesta interface especifica-se a identificação do agente e caracterizam-se os componentes a negociar. Os parâmetros de um produtor e de um distribuidor estão apresentados na Tabela 7 e Tabela 8, respectivamente.

Tabela 7 **Parâmetros do Distribuidor**

Parâmetro	**Significado**
`Name`	Nome da empresa que representa o distribuidor.
`Timeslot`	Duração do intervalo de tempo que o distribuidor disponibiliza para negociação.
`IntervalId`	Identificação do espectador dos anúncios. Esta identificação deve ser única.
`ProductCode`	Código de caracterização do perfil do intervalo.
`Reference (€/s)`	Valor mínimo do preço por segundo que o agente distribuidor pede pela ocupação do intervalo.
`Variation (€/s)`	Valor da variação do preço por ronda de negociação.
`Min (€/s)`	Valor mínimo do preço por segundo que irá ser considerado pelo distribuidor no ajuste de preço.
`Iterations`	Número de iterações (dentro da mesma ronda) que serão cumpridas na negociação.
`Δt (s)`	Intervalo de tempo entre iterações.
`Tactic`	Define a função de adaptação de preço que o agente *Enterprise* deve adoptar entre rondas de negociação.
`Protocol`	Representa o tipo de mercado da negociação.
`SuggestedPrice (€/s)`	Activar o mecanismo de *open outcry* durante a negociação.
`Graph`	Activa/desactiva a interface gráfica do agente delegado.

Tabela 8 **Parâmetros do Produtor**

Parâmetro	**Significado**
`Name`	Nome da empresa que representa o produtor.
`Contract`	Identificação do contracto celebrado entre o produtor e a agência de negociação.
`Product`	Identificação do componente multimédia (anúncio) a negociar.
`ProductCode`	Código de caracterização do perfil do componente (anúncio).
`Timeslot (s)`	Duração do componente (anúncio).
`Reference (€/s)`	Valor de mínimo do preço por segundo do componente (anúncio).
`Variation (€/s)`	Valor da variação do preço por iteração na táctica linear.
`Protocol`	Representa o tipo de mercado da negociação.
`Tactic`	Define a função de adaptação de preço que o agente delegado deve adoptar no mercado
`Max (€/s)`	Valor máximo de preço por segundo do componente (anúncio).
`Iterations`	Número de iterações considerado na definição do valor da variação nas tácticas quadrática e exponencial.
`Graph`	Activa/desactiva a interface gráfica do agente delegado.

Para que a negociação tenha sucesso, a duração (*timeslot*) do intervalo que o agente distribuidor disponibiliza tem que ser superior à duração (*timeslot*) do anúncio que o agente produtor pretende difundir.

O intervalo é caracterizado por um código (*i.e.*, o *Product Code* do intervalo) que representa o perfil do espectador no contexto actual (o programa que está ver, os seus gostos, preferências, *etc*.).

4.4.2. AGENTE PRODUTOR

O agente produtor da camada intermédia quando é lançado regista-se automaticamente no UDDI para ser convidado por potenciais agentes distribuidores a negociar. Na Figura 23 apresenta-se a GUI de um agente produtor.

Figura 23 GUI do Agente Produtor da Camada Intermédia

Os parâmetros que se visualizam neste formulário são os especificados na GUI aquando da criação do agente na plataforma, exceptuando o URL do ficheiro WSDL de descrição dos serviços disponibilizados pelo agente produtor. Contém, ainda, uma área de texto (*Result*) onde irão ser afixados os seus resultados de negociação.

4.4.3. AGENTE DISTRIBUIDOR

O agente distribuidor da camada intermédia (Figura 24) quando é lançado regista-se no UDDI e disponibiliza uma interface com o utilizador. A interface apresenta duas janelas internas, sendo uma de apresentação dos parâmetros específicos do distribuidor (identificação, URL dos serviços *Web* expostos e características do intervalo) a negociar e outra que se destina a desencadear o processo de negociação e que inclui a procura no

UDDI e o convite de produtores e o lançamento no mercado dos delegados dos produtores e do distribuidor.

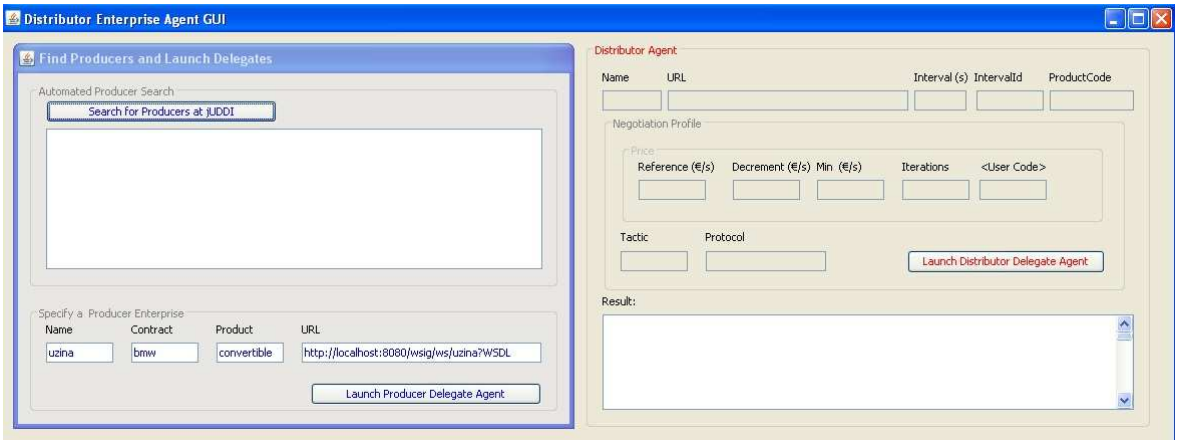


Figura 24 GUI do Agente Distribuidor da Camada Intermédia

O agente distribuidor ajusta no final de cada ronda negocial o preço (de referência) na camada intermédia, *i.e.*, se teve insucesso, diminui-o até valor mínimo que foi definido no seu perfil de negociação. O agente produtor ajusta o preço durante a ronda negocial por intermédio dos seus delegados da camada de mercado.

O agente distribuidor pode também decidir adoptar o mecanismo de *open outcry* durante a negociação de um intervalo. Este mecanismo é activado seleccionando a opção *suggested price* na definição do perfil de negociação (ver Figura 22). Neste caso, durante a negociação, o agente delegado do distribuidor envia aos agentes delegados dos produtores mensagens *Call For Proposal* (CFP) com o valor do preço mais elevado atingido na iteração anterior. Assim, os agentes delegados dos produtores ao receberem o preço sugerido efectuam as suas propostas com base nesse valor e de acordo com a sua táctica de negociação. O preço que é enviado pelo delegado do agente distribuidor na primeira iteração é o preço de referência definido no seu perfil.

4.4.4. CÓDIGO DE CARACTERIZAÇÃO

Para tentar alinhar o tipo de publicidade com as características do programa corrente e o perfil do espectador definiu-se um código denominado `ProductCode`. Este código é formado por um vector de 10 caracteres alfanuméricos que representam dez características. Cada posição corresponde a uma característica do componente multimédia. O valor de cada característica pode variar de "`0`" a "`Z`", definindo a uma escala de 34 níveis de

interesse. O carácter "`0`" significa que a característica é indiferente para o espectador. Por exemplo, se um espectador está a ver um programa de desenhos animados às 09:00 de Domingo e o seu perfil acaba de ser criado, a publicidade a inserir durante o intervalo deve excluir anúncios para adultos. Se na característica associada a público adulto se encontrasse "`0`", significava que o intervalo de programação podia ser preenchido com anúncios para adultos. O código de caracterização de um intervalo ou de um anúncio `ProductCode` corresponde ao perfil do intervalo ou do anúncio.

Os perfis dos intervalos de programação e dos anúncios são definidos através dos respectivos códigos de caracterização. Os códigos de caracterização são utilizados mas não são gerados no âmbito deste projecto.

Na Figura 25 ilustra-se o processo de emparelhamento dos componentes multimédia (anúncios e intervalos de programação) através da análise dos `ProductCode` correspondentes. Este cenário pressupõe que: (*i*) todos os componentes multimédia em jogo (programas que estão a ser visionados, intervalos de programação e anúncios) possuem o seu `ProductCode`; (*ii*) os anúncios têm igual duração (*timeslot*); e (*iii*) o intervalo tem uma duração (*timeslot*) múltipla da duração dos anúncios, sendo preenchido na totalidade.

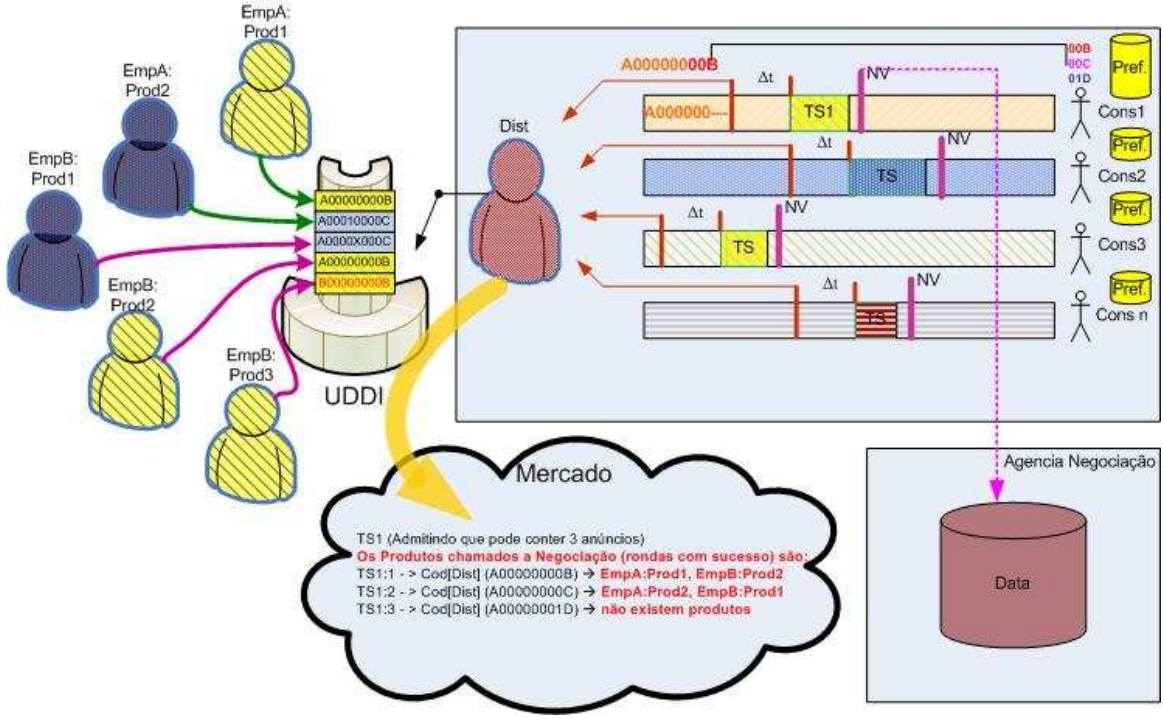


Figura 25 Escolha de Agentes por Código de Caracterização

É de notar que não é obrigatório que durante o período de publicidade haja necessariamente a interrupção da programação em curso [2].

4.5. PLATAFORMA DE NEGOCIAÇÃO

A implementação da plataforma multiagente foi suportada pelo JADE. A plataforma destina-se a negociar automaticamente, em tempo útil e em função do perfil e contexto do espectador a publicidade a transmitir nos intervalos de programação. Na Figura 26 apresenta-se uma visão do funcionamento global do sistema, incluindo a arquitectura e as funcionalidades.

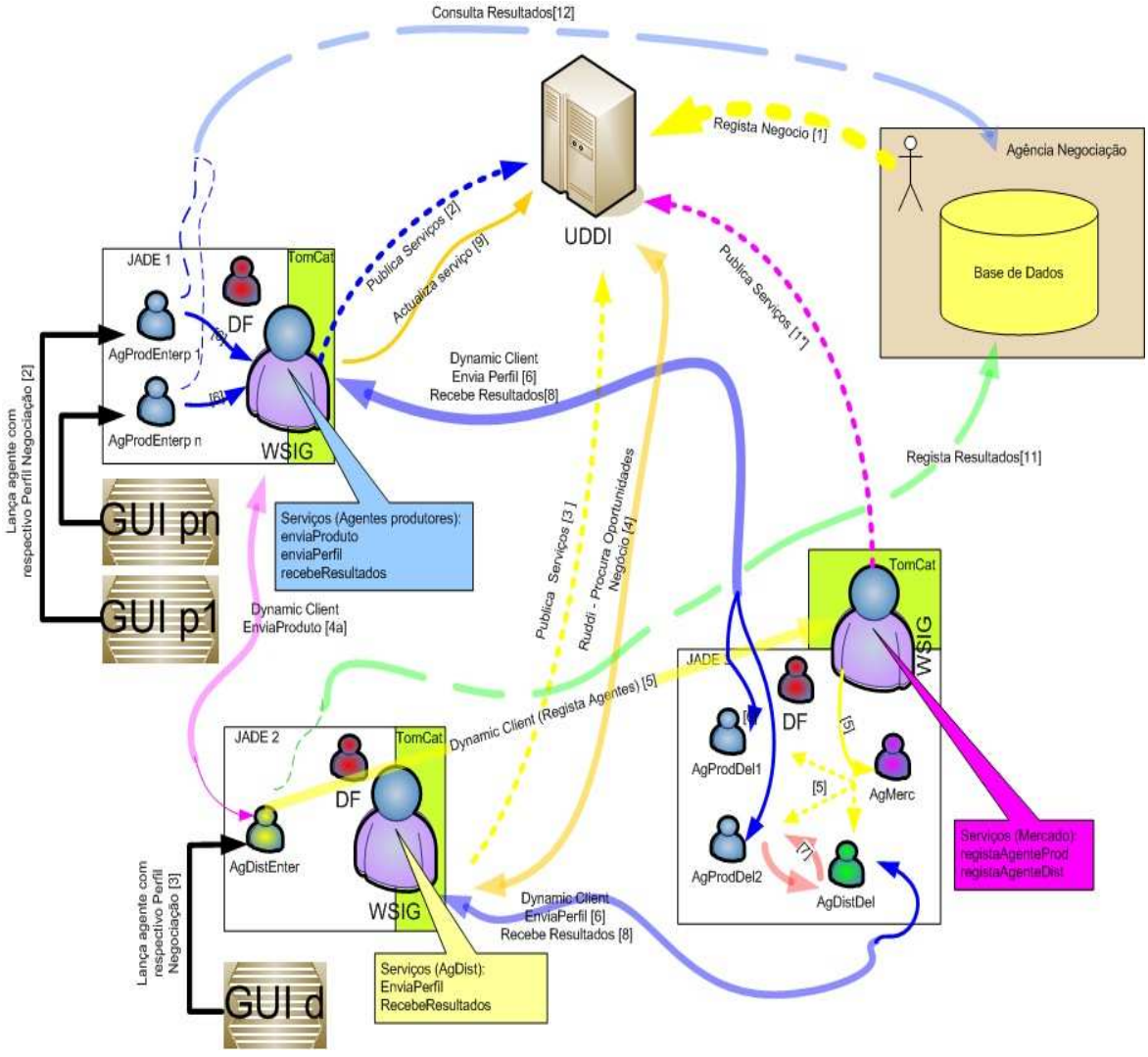


Figura 26 Funcionamento Global do Sistema

As funcionalidades incluem: (1) o registo de empresas; (2) e (3) a publicação de serviços; (4) a procura de parceiros de negócio; (4A) a obtenção de lista de componentes; (5) a

criação de agentes delegados; (6) a obtenção do perfil de negociação; (7) a negociação; e (8) o reporte de resultados.

Trata-se de uma solução distribuída composta pela plataforma de negociação de componentes, o UDDI e a base de dados. A própria plataforma pode, por sua vez, ser distribuída por diferentes máquinas ficando, por exemplo, os agentes da camada intermédia alojados em plataformas JADE das próprias empresas produtoras e distribuidoras e apenas a camada de mercado ficaria alojada numa plataforma central.

A configuração adoptada ao longo do desenvolvimento recorreu a duas plataformas físicas distintas que alojaram, respectivamente, o UDDI e a plataforma de negociação. Estas diferentes configurações são possíveis porque a comunicação entre agentes de plataformas JADE diferentes é efectuada através de serviços *Web*, usando recursos como o *add-on* do JADE WSDC e as bibliotecas UDDI4J e RUDDI para questionar o UDDI.

Em relação ao sistema global da Figura 26 há funcionalidades que se encontram foram do âmbito do projecto e que não foram implementadas. É o caso do registo automático das empresas no UDDI, que actualmente é efectuado de forma manual, e a interacção dos agentes da camada intermédia com a base de dados que não está implementada.

4.5.1. REGISTO DA EMPRESA

A empresa que representa a plataforma deve-se registar no UDDI para obter a sua chave de registo no UDDI. Essa chave vai ser partilhada com os seus clientes após a formalização contratual da sua relação. Estes clientes são as empresas de produção de conteúdos e empresas de distribuição desses conteúdos. Também deverá gerar a plataforma da camada inferior com o respectivo *Web Service Integration Gateway* (WSIG) devidamente configurado.

A chave do negócio deverá ser conhecida por todos os WSIG de todas as plataformas que desejem entrar neste mercado para que possam procurar e interagir com os respectivos serviços. Assume-se que o WSIG se encontra devidamente registado como uma empresa no UDDI e que a sua chave de negócio é partilhada com os agentes da plataforma.

4.5.2. PUBLICAÇÃO DOS SERVIÇOS NO UDDI

Todos os agentes que disponibilizam serviços para consumo deverão registá-los no UDDI usando a chave do WSIG. A publicação dos serviços no UDDI é feita de forma automática pelo agente WSIG plataforma de execução correspondente.

Durante o processo de publicação dos serviços são geradas e enviadas pelo UDDI as chaves de autorização necessárias à publicação dos serviços (`authToken`). Por fim, publica-se o `endPoint` do serviço, usando-se essa chave.

Na Figura 27 apresenta-se a mensagem SOAP da publicação do serviço *Web* exposto pelo agente de mercado (`market`) que foi enviada pelo WSIG Servlet ao UDDI. Para mais detalhes consultar, por favor, o Anexo A.

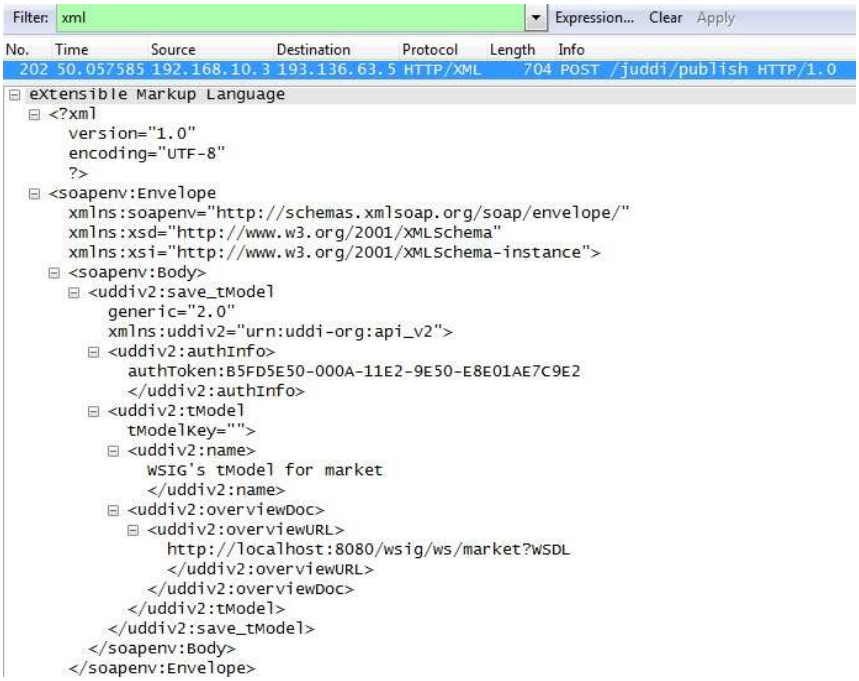


Figura 27 Mensagem SOAP de Publicação do Serviço do agente `market`

4.5.3. PROCURA PARCEIROS DE NEGÓCIO NO UDDI

A procura de parceiros de negócio é desencadeada pelo agente distribuidor quando se aproximam os intervalos de programação dos espectadores. O agente distribuidor questiona o UDDI em busca de produtores, sendo conveniente que existam agências produtoras registadas. Esta operação é efectuada através da API RUDDI e solicitando o URL de todos os serviços da empresa WSIG.

4.5.4. OBTENÇÃO DOS PRODUTOS A NEGOCIAR

O agente distribuidor contacta cada um dos agentes produtores da camada intermédia encontrados e interroga-os acerca dos produtos que pretendem negociar (`GetProduct`). Esta operação utiliza a API WSDC que permite a um agente consumir serviços disponibilizados por outro agente, necessitando apenas do URL do serviço obtido no UDDI.

4.5.5. CRIAÇÃO AGENTES NA CAMADA INFERIOR

Uma vez obtida a descrição dos anúncios dos produtores, o agente distribuidor contacta o agente de mercado da camada inferior e solicita a criação dos agentes delegados (`RegisterDistributorDelegate`, `RegisterProducerDelegate`) dos produtores com anúncios compatíveis com o perfil do intervalo e do seu próprio delegado.

4.5.6. OBTENÇÃO DO PERFIL

Os agentes delegados, assim que são lançados no mercado, contactam o agente da camada intermédia correspondente solicitando o envio do perfil de negociação (`SetAdProfile`, `SetIntervalProfile`).

4.5.7. NEGOCIAÇÃO

Assim que o delegado do distribuidor é lançado na plataforma, inicia-se a negociação propriamente dita. O mercado aplica o protocolo de negociação escolhido. Com o intuito de manter uma lógica de escalabilidade da solução optou-se pelo desenvolvimento de agentes "leves" que conhecem apenas as regras de um único tipo de mercado (protocolo). Assim, quando se pretende negociar um produto usando um protocolo diferente, serão lançados agentes delegados que utilizam apenas esse protocolo.

O protocolo de negociação implementado é uma variante do FIPA *Iterated Contract Net* (ICNet) [13]. Trata-se do *Fixed* ICNet que só termina a negociação após ter ocorrido uma ronda negocial com um número de iterações pré-determinado.

O agente delegado de um distribuidor (`AgDistDel`) procura no DF os agentes produtores delegados disponíveis para participar na negociação e envia um pedido de envio de proposta a cada um – mensagem *Call For Proposal* (CFP). Cada agente delegado de um produtor (`AgProdDel`) verifica, quando recebe este pedido, se a duração do produto que tem para vender se enquadra no tempo disponível e, em caso afirmativo, responde enviando

uma proposta de acordo com o seu perfil de negociação – mensagem `PROPOSE`. Assim que se esgota o número de iterações que foram definidas por ronda, o `AgDistDel` envia ao último `AgProdDel` que efectuou a proposta com valor mais elevado uma mensagem de aceitação (mensagem `ACCEPT-PROPOSAL`) e aos restantes `AgProdDel` envia uma mensagem de rejeição das respectivas propostas (mensagem `REJECT-PROPOSAL`).

4.5.8. RECEBE RESULTADOS

Quando a negociação termina, os agentes participantes reportam os resultados aos agentes correspondentes da camada intermédia (`setAdResult`, `setIntervalResult`).

Caso a negociação não tenha sucesso por não se ter atingido o valor mínimo de referência especificada pelo agente distribuidor, este reajusta o preço de acordo com a táctica definida e reinicia o processo de negociação na camada inferior com o novo preço de referência. Apesar de não ter sido implementado, o agente distribuidor poderá utilizar evolução do valor das propostas da negociação falhada para redefinir um preço de referência com maior probabilidade de sucesso. No entanto, dado que os agentes produtores também poderão alterar as suas estratégias, o resultado é imprevisível.

4.6. CONCLUSÃO

A arquitectura da solução proposta apresenta um elevado grau de flexibilidade e adaptabilidade. Essa adaptabilidade foi conseguida com a divisão em camadas funcionais e com o desenvolvimento de agentes especializados, aumentando a modularidade do sistema e facilitando a implementação de agentes com novas funcionalidade. A camada inferior contém o agente de mercado (`AgMerc`) que presta serviços como lançar os agentes delegados com o protocolo seleccionado.

Os agentes da camada intermédia registam-se no UDDI, expõem serviços de suporte à negociação e procuram parceiros de negócio. Os parceiros são seleccionados através do código de caracterização. Os agentes da camada inferior implementam o protocolo de negociação implementado e interagem com os agentes da camada superior consumindo os serviços disponibilizados para obter o perfil de negociação e reportar o resultado.

5. TESTES E RESULTADOS

Neste capítulo caracterizam-se os testes efectuados e analisam-se os resultados obtidos.

5.1. TESTES

Os testes efectuados visaram analisar: (*i*) a comunicação entre os agentes da camada intermédia suportada pelos mecanismos de publicação e pesquisa de serviços *Web* no serviço de registo UDDI e pela exposição e consumo de serviços *Web*; (*ii*) a comunicação entre os agentes das camadas inferior e intermédia alicerçada no mecanismo de exposição e consumo de serviços *Web*; e (*iii*) a comunicação entre os agentes da camada inferior de mercado constituída pelo protocolo de negociação.

Para estes testes foram usadas as seguintes ferramentas.

- WireShark – Análise das mensagens SOAP trocadas entre a plataforma JADE e o UDDI a correr no servidor do ISEP.
- JavaSniffer da Rockwell Automation – Análise das mensagens trocadas entre agentes da plataforma JADE inter e intracamada.
- Janelas GUI dos agentes – Análise do protocolo de negociação.

De forma a facilitar a descrição dos testes apresenta-se a na Tabela 9 e na Tabela 10 a terminologia adoptada.

Tabela 9 **Parâmetros dos Perfis de Negociação**

Parâmetro	**Descrição**	**Terminologia**
Name	Nome Agente (Distribuidor)	NAdst
Timeslot (s)	Duração do Intervalo (Distribuidor)	DIdst
IntervalId	Código do Consumidor (Distribuidor)	CCst
ProductCode	Código do Intervalo (Distribuidor)	CPdst
Reference (€)	Preço de Referência (Distribuidor)	PRdst
Decrement (%)	Decremento de Preço (Distribuidor)	DPdst
Min (€)	Preço Mínimo (Distribuidor)	PMdst
Iterations	Número de Iterações (Distribuidor)	NIdst
Tactic	Táctica de Negociação (Distribuidor)	TNdst
Protocol	Protocolo de Negociação (Distribuidor)	PNdst
SuggestedPrice (€)	Preço Sugerido (Distribuidor)	PSdst
Name	Nome do Agente (Produtor)	NAprd
Contract	Identificação do Contracto (Produtor)	ICprd
Product	Nome do Produto (Produtor)	NPprd
ProductCode	Código do Produto (Produtor)	CPprd
Timeslot (s)	Duração do Anúncio (Produtor)	DAprd
Reference	Preço de Referência (Produtor)	PRprd
Increment (%)	Incremento do Preço (Produtor)	IPprd
Max (€)	Valor Máximo Preço (Produtor)	MPprd
Iterations	Número de Iterações (Produtor)	NIprd
Tactic	Táctica Negociação (Produtor)	TNprd
Protocol	Protocolo de Negociação (Produtor)	PNprd

Tabela 10 **Designação dos Agentes**

Descrição	**Terminologia**
Agente Distribuidor (Camada Intermédia)	AgDst
Agente Produtor (Camada Intermédia)	AgPrd
Agente Distribuidor Delegado (Camada Inferior)	AgDstd
Agente Produtor Delegado (Camada Inferior)	AgPrdd
Agente de Mercado (Camada Inferior)	AgMrc

Os testes são organizados em cinco cenários:

- 1.º Cenário – Demonstração das funcionalidades genéricas do protótipo e o mecanismo de *open outcry* (suggestedPrice).
- 2.º Cenário – Ilustração da influência das tácticas de variação de preço dos perfis de negociação dos anúncios no resultado da negociação e analisa-se o protocolo *Fixed Iterated Contract Net*.

- 3º Cenário – Exemplificação da influência dos limites de variação de preço dos perfis de negociação dos anúncios no resultado da negociação.
- 4º Cenário – Demonstração do funcionamento com diferentes protocolos de negociação.
- 5º Cenário – Demonstração da influência do perfil do intervalo (código de caracterização) na escolha dos agentes a negociação.

A fim de se evitar a repetição da descrição, apenas serão apresentadas as diferenças entre cada cenário.

5.1.1. 1.º CENÁRIO – FUNCIONALIDADES BÁSICAS

Pretende-se demonstrar o funcionamento do protocolo de comunicação entre as camadas intermédia e inferior e o protocolo de negociação que decorre na camada inferior de mercado. Os parâmetros iniciais dos agentes são apresentados na Tabela 11 e na Tabela 12.

Tabela 11 **1.º Cenário – Parâmetros do Agente Distribuidor**

AgDst											
Índice	**NAdst**	**DIdst**	**CCst**	**CPdst**	**PRdst**	**DPdst**	**PMdst**	**NIdst**	**TNdst**	**PNdst**	**PSdst**
1	isep	20	cons1_12345	A00000000B	30	4	25	8	Linear	IC	false

Tabela 12 **1.º Cenário – Parâmetros do Agente Produtor**

AgPrd											
Índice	**NAprd**	**ICprd**	**NPprd**	**CPprd**	**DAprd**	**PRprd**	**IPprd**	**MPprd**	**NIprd**	**TNprd**	**PNprd**
1	uzina	bmw	convertible	A00000000B	10	20	1	35	6	Linear	IC

Nestas condições prevê-se que haja sucesso na negociação, uma vez que a duração do intervalo é maior que a duração do anúncio e os códigos dos produtos (intervalo e anúncio) são iguais. Contudo, só há sucesso na segunda ronda negocial porque, na primeira ronda, o valor da proposta final do produtor no final das três iterações definidas é 27 que é inferior ao valor mínimo que o agente distribuidor aceita na primeira ronda.

Na segunda ronda, o agente distribuidor ajusta o seu preço máximo para 26. Embora não tenha sido implementado neste trabalho, o agente produtor poderia igualmente ajustar o seu perfil entre rondas de forma a tirar partido da margem de negociação que tenha.

Contudo, na versão actual da plataforma, os agentes produtores da camada intermédia não são informados acerca do valor atingido na ronda anterior.

Os agentes da camada intermédia afixam no WSIG e registam no UDDI os seus serviços. A Figura 28 apresenta a lista dos serviços WSIG dos três agentes em execução na plataforma (AgDst – `dst_isep`; AgPrd – `uzina`; AgMrc – `market`) que expõem serviços para o exterior.

Figura 28 Serviços Expostos pelos Agentes

Cada agente é registado no UDDI como um serviço e cada serviço disponibiliza um conjunto de operações. A Figura 29 mostra as operações do serviço `uzina`.

Name:	uzina
Prefix:	-
Mapper class:	brokerage.NegPub.ontology.NegPubOntoMapProducer
Hierarchical complex type:	false
Jade ontology:	wsig_NegPub
Jade agent:	uzina@10.0.2.15:1099/JADE
UDDI service key:	E86268E0-22D8-11E2-A4DB-8A5F31B80CB1
WSDL url:	http://localhost:8080/wsig/ws/uzina?WSDL
Operations:	GetAdProfile SetAdResult GetProduct GetProfile

Figura 29 Operações do Serviço `uzina`

Os delegados dos agentes produtores, assim que são criados, entram automaticamente em contacto com os seus pares da camada intermédia para solicitar o envio do seu perfil de negociação. Quando um serviço é consumido, o WSIG Servlet recebe a mensagem SOAP com a solicitação e entrega-a ao WSIG Agent que, por sua vez, a converte numa mensagem FIPA ACL e remete ao agente que presta o serviço. A resposta é processada e encaminhada da mesma forma mas no sentido inverso. Na Figura 30 apresenta-se a

sequência de mensagens FIPA ACL do pedido/resposta do perfil do anúncio que foi efectuado pelo delegado do produtor e respondido pelo agente produtor.

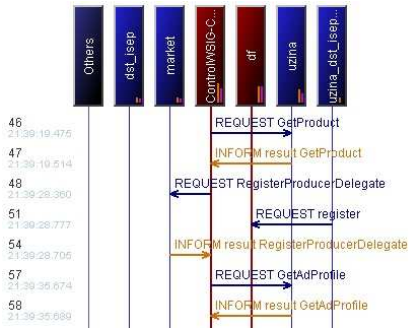


Figura 30 Criação e Configuração do Agente Delegado do Produtor

Quando o agente delegado do distribuidor é lançado na camada de mercado inicia-se a negociação de acordo com o protocolo definido. Como não houve sucesso na primeira ronda negocial, o agente distribuidor adapta o seu preço mínimo e desencadeia uma nova ronda voltando a convidar os agentes produtores da camada intermédia com anúncios compatíveis. Nesta fase o agente distribuidor poderia, para além de alterar o preço mínimo, readaptar o pedido, alterando o código de caracterização, *etc*. No entanto, como essa modalidade não está implementada, os parceiros para a segunda ronda são os mesmos da primeira ronda – ver Figura 31.

```
enterpriseURL: http://localhost:8080/wsig/ws/uzina?WSDL
Producer ProductCode: A00000000B
productName: convertible
contractId: bmw
Negotiation Protocol: Iterated Contract Net
entity: uzina

enterpriseURL: http://localhost:8080/wsig/ws/uzina?WSDL
Producer ProductCode: A00000000B
productName: convertible
contractId: bmw
Negotiation Protocol: Iterated Contract Net
entity: uzina
```

Figura 31 Resultado da Procura de Parceiros no UDDI

No final de cada ronda negocial os resultados são reportados aos agentes correspondentes da camada intermédia (`SetAdResult`, `SetIntervalResult`).

Figura 32 Reporte de Resultados da Negociação

5.1.1.1. RESULTADOS DO 1.º CENÁRIO

A Figura 33 apresenta os resultados enviados pela camada inferior aos agentes `AgDst` e `AgPrd`. Cada agente recebe os resultados da negociação (`Price Proposals; Negotiated Price`) do seu delegado. No caso do `AgDst`, vê-se claramente o ajuste do preço de referência da primeira ronda para a segunda ronda (`DPdst=4`). Por outro lado, verifica-se que as propostas (`Price Proposals`) do delegado do produtor apresentam uma evolução linear (`IPprd=1`; `TNprd=Linear`).

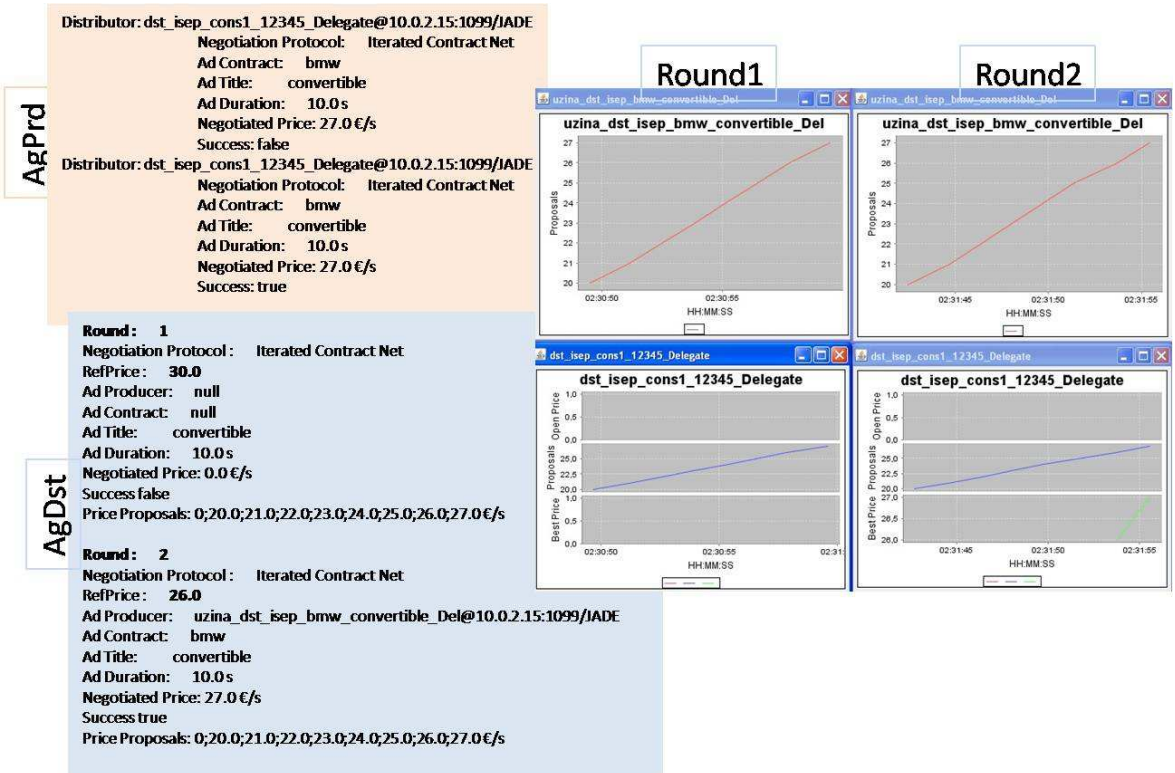


Figura 33 1.º Cenário – Resultados

5.1.1.2. INFLUÊNCIA DO *OPEN OUTCRY*

Este teste ilustra a adopção do mecanismo de *open outcry*. O `AgDst` comunica aos agentes produtores (`AgPrd`) o preço recomendado, *i.e.*, especifica o menor valor (€/s) que ele aceita para transmitir um anúncio no intervalo em preenchimento. É assumido que os agentes não fazem *bluff*.

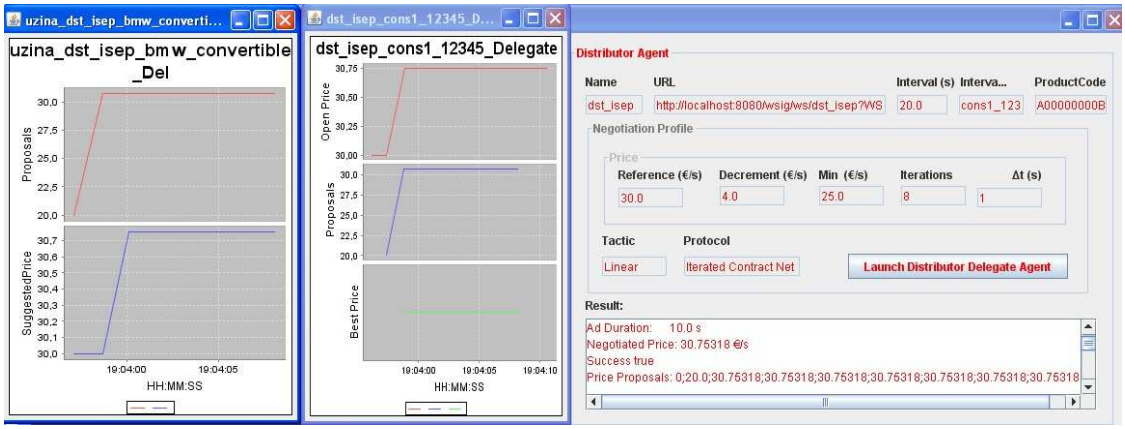


Figura 34 1º Cenário – Resultados com *open outcry*

A Figura 34 mostra que, quando o `AgPrd` recebe uma sugestão de preço do `AgDst` mais elevada do que o seu preço mínimo, oferece na iteração seguinte um valor ligeiramente acima. Como na iteração seguinte o preço sugerido pelo `AgDst` é igual ao valor que o `AgPrd` propôs na iteração anterior, manteve o valor da sua proposta anterior. Dado que neste cenário não existem outros agentes produtores, o preço proposto pelo `AgPrd` manteve-se inalterável até ao final da ronda.

5.1.2. 2.º CENÁRIO – PROTOCOLO E DIFERENTES TÁCTICAS

Neste cenário envolve cinco produtores com anúncios compatíveis com o perfil do espectador do intervalo com diferentes tácticas e os mesmos limites de variação do preço. As condições iniciais do teste são apresentadas na Tabela 13 e na Tabela 14.

Tabela 13 **2.º Cenário – Parâmetros do Agente Distribuidor**

AgDst											
Índice	**NAdst**	**DIdst**	**CCst**	**CPdst**	**PRdst**	**DPdst**	**PMdst**	**NIdst**	**TNdst**	**PNdst**	**PSdst**
1	isep	20	cons1_12345	A00000000B	30	4	25	8	Linear	IC	false

Tabela 14 **2.º Cenário – Parâmetros dos Agentes Produtores**

AgPrd											
Índice	**NAprd**	**ICprd**	**NPprd**	**CPprd**	**DAprd**	**PRprd**	**IPprd**	**MPprd**	**NIprd**	**TNprd**	**PNprd**
1	uzina	bmw	convertible	A00000000B	10	20	1	35	6	Linear	IC
2	uzina	bmw	convertible	A00000000B	10	20	1	35	6	Quadratic	IC
3	uzina	bmw	convertible	A00000000B	10	20	1	35	6	Exponencial	IC
4	uzina	bmw	convertible	A00000000B	10	20	1	35	6	Random	IC
5	uzina	bmw	convertible	A00000000B	10	20	1	35	6	None	IC

Neste cenário pretende-se demonstrar que as propostas dos agentes produtores estão conformes com a respectiva táctica.

O protocolo de negociação é o *Fixed* ICNet. Na Figura 27 apresenta-se o conjunto das mensagens do protocolo *Fixed* ICNet trocadas numa ronda negocial envolvendo os seis agentes.

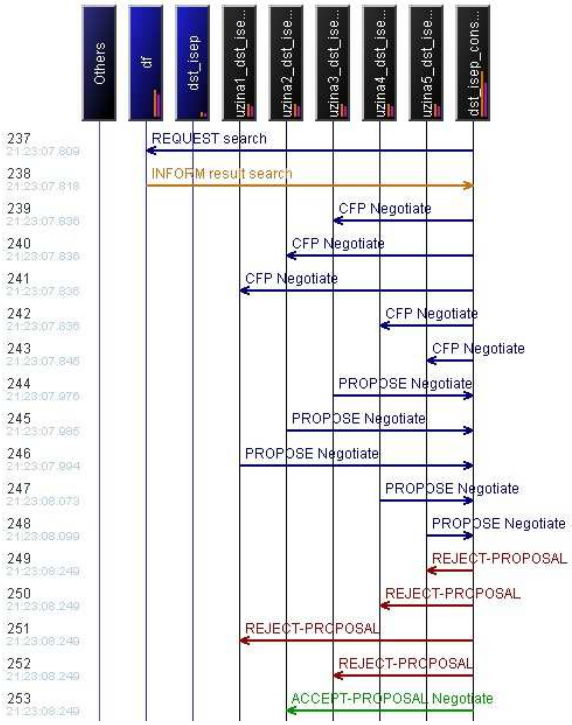


Figura 35 ***Fixed Iterated Contract Net***

O agente delegado de um distribuidor (`AgDistDel`) procura no *Directory Facilitator* (DF) os agentes produtores delegados disponíveis para participar na negociação e envia um pedido de envio de proposta a cada um – mensagem *Call For Proposal* (CFP). Cada agente

delegado de um produtor (`AgProdDel`) verifica, quando recebe este pedido, se a duração do produto que tem para vender se enquadra no tempo disponível e, em caso afirmativo, responde enviando uma proposta de acordo com o seu perfil de negociação – mensagem `PROPOSE`. Assim que se esgota o número de iterações que foram definidas por ronda, o `AgDistDel` envia ao último `AgProdDel` que efectuou a proposta com valor mais elevado uma mensagem de aceitação (mensagem `ACCEPT-PROPOSAL`) e aos restantes `AgProdDel` envia uma mensagem de rejeição das respectivas propostas (mensagem `REJECT-PROPOSAL`).

O *Fixed* ICNet permite garantir que ganha o agente produtor que, em cada ronda, apresentar a proposta mais elevada e não o primeiro agente produtor que apresente uma proposta superior ao preço de referência do distribuidor (`PRdst`). Isto permite que o agente distribuidor da camada intermédia adapte o seu perfil à forma como deseja atribuir o tempo restante do seu intervalo (`DIdst`). Dado que não é indiferente o impacto que a publicidade pode ter na audiência do programa em curso [90], esta abordagem permitira controlar a duração do intervalo publicitário.

5.1.2.1. RESULTADOS 2.º CENÁRIO

A Figura 36 ilustra as tácticas que os agentes produtores usaram nesta ronda. O agente que consegue colocar o seu anúncio no intervalo não é necessariamente o agente que ofereceu primeiro o valor mais alto, mas sim aquele que ofereceu o valor mais alto da última iteração.

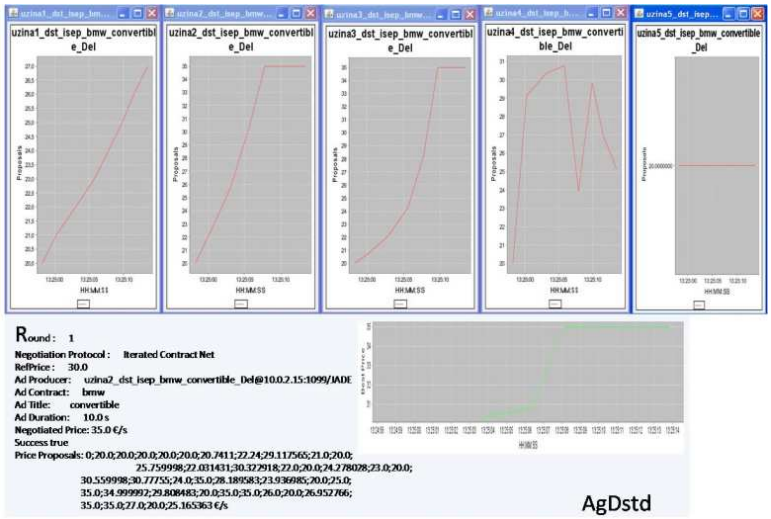


Figura 36 2.º Cenário – Resultados

Neste contexto, a táctica *Random* não faz sentido pois apenas interessa o valor que foi atingido na última iteração. Por outro lado, tem também um efeito indesejável quando se aplica essa táctica juntamente com *open outcry*.

5.1.2.2. *OPEN OUTCRY*

Pretende-se verificar o efeito do mecanismo de *open outcry*, que consiste no envio pelo agente distribuidor do preço sugerido (`PSdst`), numa ronda negocial. O valor preço máximo do produtor (`MPprd`) também foi aumentado para 100 para facilitar a visualização. A Figura 37 demonstra que a convergência dos valores das propostas é muito mais rápida.

	Tácticas				
Niteração	Linear	Quad	Exp	Random	None
1	20.00	20.00	20.00	20.00	20.00
2	26.00	37.05	33.44	48.94	20.00
3	49.94	71.88	91.11	49.74	20.00
4	92.11	100.00	91.40	51.45	20.00
5	100.00	100.00	100.00	41.51	20.00
6	100.00	100.00	100.00	36.00	20.00
7	100.00	100.00	100.00	31.53	20.00
8	100.00	100.00	100.00	67.87	20.00

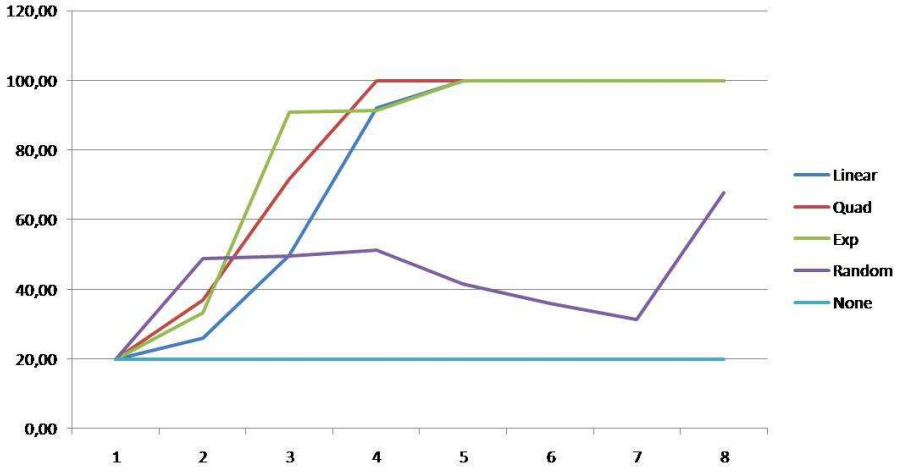


Figura 37 2.º Cenário – Resultados com *Open Outcry*

Se tomarmos como referência a táctica linear, verificamos que a evolução tenta acompanhar o valor mais alto da última iteração. Apesar de neste exemplo a táctica aleatória ter influenciado pouco o resultado final (apenas na segunda iteração teve o valor mais alto), pode-se concluir que se a evolução fosse diferente, (valores muito altos nas iterações iniciais), obrigaria os outros participantes a aumentar o preço para os seus valores

máximos. Esta táctica tem o efeito de fazer aumentar os preços do mercado (nas condições descritas) e pouca probabilidade de sucesso.

5.1.3. 3.º CENÁRIO – LIMITES DE VARIAÇÃO DE PREÇO DISTINTOS

No terceiro cenário apresentam-se cinco agentes produtores com anúncios que encaixam no perfil do espectador do intervalo e com as mesmas tácticas de adaptação de preço e limites de variação de preço diferentes. Os parâmetros iniciais dos agentes são apresentados na Tabela 15.

Tabela 15 **3.º Cenário – Parâmetros dos Agente Produtores**

AgPrd											
Índice	**NAprd**	**ICprd**	**NPprd**	**CPprd**	**DAprd**	**PRprd**	**IPprd**	**MPprd**	**NIprd**	**TNprd**	**PNprd**
1	uzina	bmw	convertible	A00000000B	10	20	2	35	6	Linear	IC
2	uzina	bmw	convertible	A00000000B	10	2	1	40	6	Linear	IC
3	uzina	bmw	convertible	A00000000B	10	5	6	45	6	Linear	IC
4	uzina	bmw	convertible	A00000000B	10	35	2	45	6	Linear	IC
5	uzina	bmw	convertible	A00000000B	25	55	2	60	6	Linear	IC

Neste cenário foi a `uzina4` quem colocou o anúncio no intervalo porque, apesar do declive da sua recta de evolução de propostas ser inferior ao do `uzina3`, parte de um valor mais elevado.

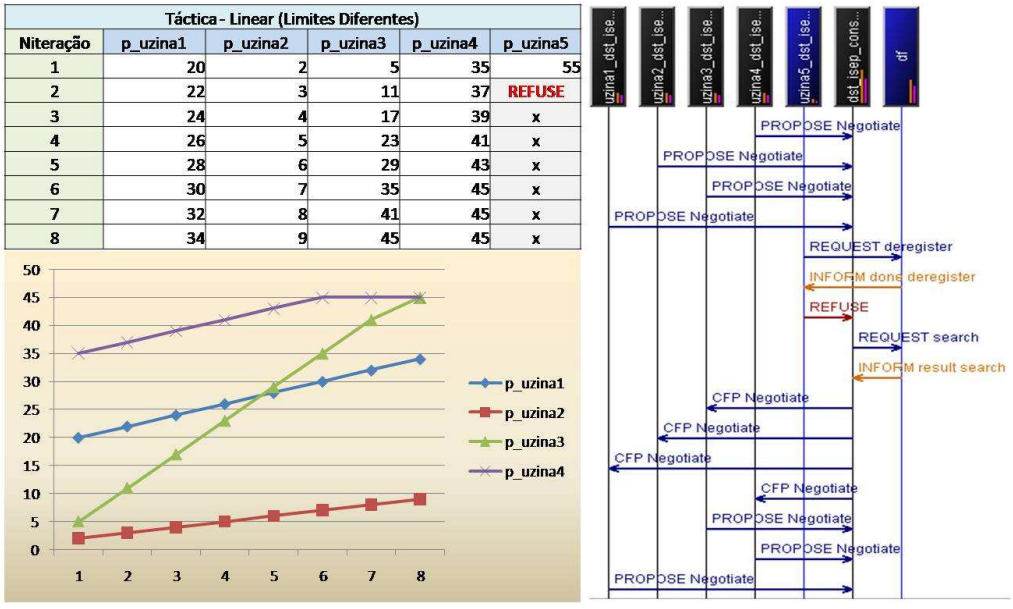


Figura 38 3.º Cenário – Resultados

Quando a duração do anúncio (`DAprd=25 s`) é superior à duração do intervalo (`DIdst=20 s`), o agente responde ao CFP (*Call for Proposal*) com um *REFUSE*. É o caso do `uzina5` que, em consequência, envia uma solicitação ao DF para se desregistar do mercado. A partir desse momento, o `AgDstd` deixa de lhe enviar o CFP porque o `uzina5` já não se encontra nas páginas amarelas do DF.

5.1.4. 4.º CENÁRIO – PROTOCOLOS E LIMITES DE VARAÇÃO DE PREÇO DIFERENTES

Este cenário envolve quatro agentes produtores divididos em dois grupos com tácticas de variação de preço idênticas mas com limites e protocolos diferentes dentro do mesmo grupo. Pretende-se demonstrar neste ensaio na mesma plataforma podem coexistir protocolos diferentes: IC – *Fixed Iterated Contract Net* e o IC_P – IC_Protótipo). O IC_Protótipo é um protocolo idêntico ao *Fixed Iterated Contract Net* com uma pequena alteração que é perceptível nos resultados: quando o agente produtor atinge o seu valor máximo retira-se da negociação.

As condições iniciais do teste são apresentadas nas Tabela 16 e Tabela 17.

Tabela 16 **4.º Cenário – Parâmetros dos Agentes Distribuidores**

AgDst											
Índice	**NAdst**	**DIdst**	**CCst**	**CPdst**	**PRdst**	**DPdst**	**PMdst**	**NIdst**	**TNdst**	**PNdst**	**PSdst**
1	isep	20	cons1_12345	A00000000B	30	4	25	8	Linear	IC	true
2	isep	20	cons1_12345	A00000000B	30	4	25	8	Linear	IC_P	true

Tabela 17 **4.º Cenário – Parâmetros dos Agentes Produtores**

AgPrd											
Índice	**NAprd**	**ICprd**	**NPprd**	**CPprd**	**DAprd**	**PRprd**	**IPprd**	**MPprd**	**NIprd**	**TNprd**	**PNprd**
1	uzina	bmw	convertible	A00000000B	10	20	5	35	6	Linear	IC
2	uzina	bmw	convertible	A00000000B	10	20	3	60	6	Linear	IC
3	uzina	bmw	convertible	A00000000B	10	20	5	35	6	Linear	IC_P
4	uzina	bmw	convertible	A00000000B	10	20	3	60	6	Linear	IC_P

A Figura 39 sintetiza os resultados da experiência: (*i*) a GUI da plataforma JADE apresenta os agentes registados; (*ii*) no topo esquerdo e direito são expostos os resultados do UDDI; e (*iii*) na base apresentam-se os gráficos com os resultados da negociação.

Pode-se verificar que o agente `isep1` convocou o `uzina1` e `uzina2` e o `isep2` convocou o `uzina3` e `uzina4` como foi previsto e configurado. Nos gráficos da negociação distingue-se a diferença dos dois protocolos. No caso do protocolo IC_Protótipo, os agentes produtores retiram-se da negociação assim que atingem o seu preço máximo; no caso do protocolo IC, os agentes permanecem em negociação até ao final da ronda.

Figura 39 4.º Cenário – Resultados

5.1.5. 5.º CENÁRIO – TÁCTICAS E LIMITES DE VARIAÇÃO DE PREÇO DIFERENTES

Este teste junta cinco agentes produtores com tácticas de adaptação e limites de variação de preço diferentes e com anúncios com perfis diversos: uns compatíveis e outros incompatíveis com o perfil do intervalo do agente distribuidor. Cria-se ainda um segundo agente distribuidor com um intervalo com um perfil (código de caracterização) diferente.

Neste cenário pretende-se demonstrar a selecção dos anúncios por código de caracterização assim como que agentes distribuidores diferentes podem coexistir na mesma plataforma, cada um dialogando com os seus parceiros.

Os parâmetros iniciais dos agentes são apresentados na Tabela 18 e Tabela 19.

Tabela 18 **5.º Cenário – Parâmetros dos Agentes Distribuidores**

AgDst											
Índice	**NAdst**	**DIdst**	**CCst**	**CPdst**	**PRdst**	**DPdst**	**PMdst**	**NIdst**	**TNdst**	**PNdst**	**PSdst**
1	isep	20	cons1_12345	A00000000B	30	4	25	8	Linear	IC	false
2	isep	20	cons1_76543	U000X0000B	30	4	25	8	Linear	IC	true

Tabela 19 **5º Cenário – Parâmetros dos Agentes Produtores**

AgPrd											
Índice	**NAprd**	**ICprd**	**NPprd**	**CPprd**	**DAprd**	**PRprd**	**IPprd**	**MPprd**	**NIprd**	**TNprd**	**PNprd**
1	uzina	bmw	convertible	A00000000B	10	20	2	35	6	Linear	IC
2	uzina	bmw	convertible	A000X0000B	10	2	1	40	6	Quadratic	IC
3	uzina	bmw	convertible	A0000000B	10	5	6	45	6	Exponencial	IC
4	uzina	bmw	convertible	A00100000C	10	35	2	45	6	Random	IC
5	uzina	bmw	convertible	U000X0Z00B	5	55	2	60	6	None	IC

A Figura 40 inclui, do lado esquerdo, os agentes registados na plataforma e, do lado direito, os gráficos de resultados dos agentes referentes a esta negociação.

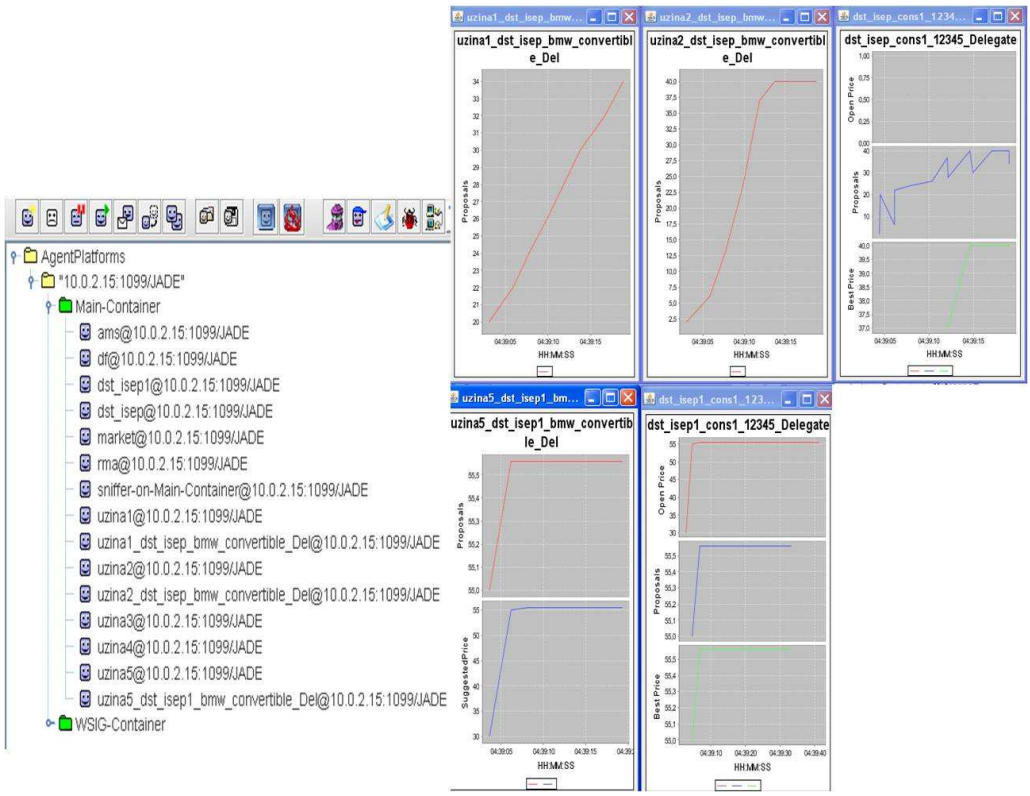


Figura 40 5.º Cenário – Resultados

Neste caso, o `isep1` só convoca para negociar o `uzina1` e o `uzina2` porque, segundo as regras definidas, `A00000000B` emparelha com `A00000000B` e `A000X0000B`. O `isep2` apenas convoca o `uzina5` porque é o único produtor com código compatível. O `uzina3` nunca irá ser convidado porque tem um código de caracterização incompleto. O `uzina4` terá de esperar por uma oportunidade de negociação futura.

5.2. CONCLUSÃO

Os resultados apresentados demonstram que a solução proposta permite negociar componentes multimédia, mais especificamente anúncios comerciais a integrar nos intervalos de programação. Na definição da solução houve uma preocupação constante na utilização de tecnologias *freeware* e com a escalabilidade do sistema. Os cenários apresentados mostraram a enorme flexibilidade da arquitectura face as necessidades actuais e futuras. Essa flexibilidade foi conseguida em grande parte pela adopção do JADE enquanto ambiente de execução dos agentes.

6. CONCLUSÕES

Neste capítulo apresenta-se o balanço do projecto realizado e propõem-se sugestões de desenvolvimento futuro.

6.1. BALANÇO

O objectivo deste projecto foi a especificação, desenvolvimento, teste e depuração de um mercado virtual de transacção de componentes multimédia. No âmbito desta tese e em relação à plataforma global de transacção de componentes multimédia, desenvolveram-se todas as funcionalidades da camada de mercado assim como as funcionalidades da camada intermédia necessárias ao funcionamento do mercado.

O desenvolvimento do protótipo envolveu um esforço relevante na selecção, implantação e utilização de várias tecnologias diferentes exclusivamente suportado por código aberto.

6.1.1. REPRESENTAÇÃO DE CONHECIMENTO

Foram criadas duas ontologias: (*i*) a `NegPub` é usada na comunicação intercamada entre o agente que modela a empresa e os seus agentes delegados; e (*ii*) `NegMComp` é usada na comunicação interna entre os agentes delegados da camada de mercado. A primeira ontologia representa os conceitos do domínio: agentes (`AgentType`), acções dos agentes (`AgentAction`) e os tipos de dados processados pelos agentes (`AgentData`). A segunda

ontologia define os conceitos associados à negociação de componentes multimédia (`Negotiation`) que ocorre na camada de mercado.

6.1.2. AGENTES

Foram desenvolvidos os agentes da camada intermédia que modelam as empresas distribuidoras e produtoras da camada intermédia (`AgDistributorEnterprise`, `AgProducerEnterprise`) e os agentes delegados dos agentes da camada intermédia (`AgDistributorDelegate`, `AgProducerDelegate`) da camada de mercado. Por último, foi implementado o agente responsável pela camada inferior de mercado (`AgPlatform`) que disponibiliza a interface de interacção com a plataforma. Esta interface permite criar agentes produtores e distribuidores da camada intermédia e definir as características dos objectos multimédia que pretendem negociar (intervalos de espectadores e anúncios). Os agentes da camada intermédia apresentam interfaces que permitem desencadear e reportar os resultados de rondas negociais na camada inferior de mercado.

Os agentes da camada intermédia registam automaticamente os seus serviços no UDDI usando a chave de negócio do WSIG. A Tabela 20 apresenta a lista de serviços *Web* criados que são destinados à comunicação entre as camadas intermédia e inferior.

Tabela 20 **Serviços *Web* Criados**

Serviço *Web*	Operações
`AgPlatform`	`RegisterDistributorDelegate` `RegisterProducerDelegate`

`AgProducerEnterprise`	`GetAdProfile`
	`SetAdResult`
	`GetProduct`

`AgDistributorEnterprise`	`GetIntervalProfile`
	`SetIntervalResult`
	`GetProduct`

6.1.3. PESQUISA DE PARCEIROS

O `AgDistributorEnterprise` desencadeia o processo de pesquisa de parceiros de negociação sempre que pretende preencher o intervalo de um espectador. Os intervalos estão caracterizados através de um código de caracterização que representa o perfil do espectador e que vai ser utilizado para seleccionar apenas os anúncios com as

características desejadas. O agente distribuidor procura no UDDI agentes produtores com anúncios compatíveis e convida-os a negociar na camada de mercado. Caso o convite seja aceite, os parceiros de negociação lançam os seus agentes delegados na camada inferior.

6.1.4. NEGOCIAÇÃO

Os delegados solicitam aos agentes da camada intermédia correspondentes o envio das características dos objectos a negociar e desencadeiam o processo de negociação de acordo com o protocolo de negociação do mercado. Cada agente delegado negocia apenas um objecto multimédia e comporta-se no mercado de acordo com o perfil de negociação do objecto. O perfil de negociação do objecto define a táctica de adaptação e os limites variação de preço no mercado. No final reportam os resultados aos agentes correspondentes da camada intermédia e terminam a execução.

Foram definidas tácticas simples de adaptação do preço (*Linear*, *Quadratic*, *Exponential*, *Random* e *None*) que são usadas pelos agentes delegados para ajustar o preço que atribuem ao produto durante a negociação.

Foi implementada uma variante do protocolo FIPA *Iterated Contract Net*. – o *Fixed* ICNet. A implementação de rondas de negociação possibilita uma maior convergência de preços no mercado no caso de haver uma discrepância entre o valor mínimo que o distribuidor está disposto a vender e o valor máximo que os produtores estão dispostos a oferecer. Assim, entre rondas, haverá um período de "reflexão" em que o agente distribuidor poderá ajustar o seu preço para um mais alinhado com os preços de mercado. Para aumentar a convergência de preços é também possível configurar o agente distribuidor da camada intermédia para enviar valor mais alto da oferta anterior (começando por enviar o seu preço mínimo) para que os agentes produtores possam enquadrar as suas ofertas nesse valor de referência.

Na fase de testes foi implementado um segundo protocolo que demonstrou a flexibilidade e escalabilidade da solução.

6.2. DESENVOLVIMENTOS FUTUROS

No futuro próximo perspectiva-se o desenvolvimento continuado da plataforma de negociação de componentes multimédia que decorre, não só, do enriquecimento do mercado desenvolvido no âmbito desta tese, mas também das teses complementares em

curso. Neste último contexto enquadra-se o desenvolvimento da camada superior da plataforma e das suas repercussões na camada intermédia assim como do módulo de recomendação de conteúdos baseado no contexto e perfil do espectador. É de referir que, apesar de este projecto ter implementado as características e funcionalidades da camada intermédia relevantes para o funcionamento da camada inferior de mercado, nomeadamente, a interface de comunicação entre estas duas camadas, a especificação, desenvolvimento e teste das restantes funcionalidades da camada não fazem parte deste projecto, assim como, toda a problemática associada à formalização das negociações assim como o garante dos direitos e segurança.

6.2.1. LIMITAÇÕES E SUGESTÕES

Por omissão, o número de agentes que o DF consegue gerir é limitado (cerca de 100), o que quer dizer que cada plataforma JADE está limitada a cerca 100 agentes em execução simultânea. No entanto, o problema não se coloca neste limite, mas sim no enorme fluxo de mensagens que o WSIG processa durante a execução do protocolo de comunicação entre camadas. Para tentar resolver esse problema é necessário distribuir os agentes por várias plataformas, cada uma com o seu agente WSIG. Neste cenário quem teria que gerir o processo de divisão de carga seria o agente distribuidor da camada intermédia. Idealmente seria desenvolver um módulo que permitisse gerir as negociações entre plataformas que reduzisse ao máximo as mensagens que são trocadas entre elas, com respectivos melhoramentos no referido protocolo.

O comportamento dos agentes delegados no mercado é baseado no perfil de negociação estipulado pelos respectivos agentes da camada intermédia. Poderia ser interessante desenvolver os agentes delegados do tipo *Belief-Desire-Intention* (BDI) que, além de agirem de acordo com o perfil estipulado, poderiam também adaptar o seu comportamento de modo a maximizar os seus objectivos. Este novo tipo de agente poderia competir e partilhar o mesmo ambiente de execução dos agentes delegados desenvolvidos. No caso de ocorrer um empate entre o valor da melhor proposta efectuada por dois ou mais delegados de produtor, o delegado do distribuidor poderia efectuar a selecção do melhor anúncio a inserir no intervalo através da determinação da similaridade entre os códigos de caracterização dos anúncios e do intervalo.

Poderia ser estudada a possibilidade do agente de mercado ser enriquecido com novas funções como a selecção automática dos protocolos de interacção cuja lógica estaria guardada numa ontologia em formato RDF a consultar pelos agentes delegados usando o SPARQL.

Para reduzir o tamanho das mensagens entre camadas poder-se-ia reestruturar a ontologia de forma às mensagens incluírem apenas as *slots* necessárias à comunicação. Por outro lado, o agente delegado do distribuidor poderia enviar ao agente distribuidor da camada intermédia os valores médio, máximo, mínimo e do desvio padrão das propostas recebidas durante uma ronda negocial em vez de enviar a lista completa das propostas.

De igual modo, também poderiam ser implementados agentes BDI na camada superior com os mesmos objectivos assim como para o sistema de recomendação de conteúdos baseado nas descrições dos conteúdos e no contexto e perfil dos espectadores. A plataforma JADE oferece alguns recursos para implementação deste tipo de solução, nomeadamente, recorrer ao motor de inferência JESS [107], ao conjunto de bibliotecas JENA, que permite executar comandos SPARQL para extrair elementos de documentos RDF/OWL [73] e à biblioteca JADE *Semantics Add-On* (JSA), que permite o desenvolvimento de agentes capazes de interpretar o significado das mensagens FIPA-ACL trocadas entre agentes [108]. O JADE também tem a possibilidade de implementar agentes BDI usando a JADE *eXtension* (JADEX).

Seria também interessante implementar mais protocolos e tácticas de negociação e efectuar um estudo dos protocolos e tácticas que melhor se adaptam às diversas necessidades de carga, resposta temporal, *etc.* Se, por um lado, o protocolo de negociação define as regras de funcionamento do mercado, a táctica de negociação estabelece a forma como cada agente delegado se adapta dinamicamente às condições do mercado. Como actualmente o protocolo de negociação do mercado é definido pelo agente responsável pelo mercado, não se antevê qualquer vantagem em criar agentes com a capacidade implementar simultaneamente todos os protocolos. Alternativamente, sugere-se a criação de diferentes modelos de agente delegado, correspondendo um modelo de agente a cada protocolo de negociação distinto. Ao nível dos agentes da camada intermédia, já é desejável que tenham conhecimento do conjunto de protocolos de negociação disponíveis para, poderem, em tempo útil e com base nos resultados obtidos até ao momento, decidir que tipo de negociação melhor se adapta ao seu produto.

A problemática associada à formalização das negociações assim como o garante dos direitos e segurança não foram contemplados.

Referências Documentais

[1] B. Malheiro, J. Foss, J. C. Burguillo, A. Peleteiro, F. A. Mikic (2011), “Dynamic Personalisation of Media Content”, Proceedings of the Sixth International Workshop on Semantic Media Adaptation and Personalization (SMAP 2011), ISBN 978-0-7695-4524-0, 21-26, Vigo, Spain. Acedido em Novembro 2011.

[2] B. Malheiro, J. Foss (2010), “A Proposal for Media Component Brokerage”, Proceedings of the Fourth International European Conference on the Use of Modern Information and Communication Technologies (ECUMICT 2010), Ed. Lieven de Strycker, Nevelland v.z.w., ISBN 978-9-08-082555-5, 389-403, Gent, Belgium. Acedido em Outubro de 2011.

[3] Oracle, The Java Tutorials (2012), “An Overview of RMI Applications”. Acedido em Agosto 2012. Disponível em http://docs.oracle.com/javase/tutorial/rmi/overview.html.

[4] D. Curtis, Christopher Stone, Mike Bradley (2012), “IIOP: OMG's Internet Inter-ORB Protocol – A Brief Description” Acedido em Julho 2012. Disponível em http://www.omg.org/library/iiop4.html.

[5] Object Management Group (2012), "CORBA BASICS". Acedido em Agosto 2012. Disponível em http://www.omg.org/gettingstarted/specintro.htm#CORBA.

[6] J. C. de Carvalho e L. Encantado (2006), “Logística e Negócio Electrónico”, Editora SPI – Sociedade Portuguesa de Inovação. Acedido em Dezembro 2009. Disponível em http://web.spi.pt/negocio_electronico/documentos/manuais_PDF/Manual_VI.pdf.

[7] ORACLE (2012), "Java SE Downloads". Acedido em Junho de 2012. Disponível em http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[8] NetBeans (2007), “NetBeans IDE Docs & Support Resources”. Acedido em Março 2010. Disponível em http://www.netbeans.org/kb/index.html.

[9] O'Reilly Media (2007), “O'Reilly Network – Developers' Hub”. Acedido em Agosto 2012. Disponível em http://www.oreillynet.com.

[10] E. Scagliotti e G. Caire (2010) “Web Services Dynamica Client Guide”.. Acedido em Novembro 2011. Disponível em http://jade.tilab.com/doc/tutorials/DynamicClientGuide.pdf.

[11] Apache Software Foundation (2007), “The Apache Tomcat”. Acedido em Maio 2010. Disponível em http://tomcat.apache.org/.

[12] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan (2006), "Extensible Markup Language (XML) 1.1 (Second Edition)". Acedido em Fevereiro de 2010. Disponível em http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf.

[13] FIPA TC C (2002), “FIPA Iterated Contract Net Interaction Protocol Specification”. Acedido em Outubro 2011. Disponível em http://www.fipa.org/specs/fipa00030/index.html.

[14] J. M. Martínez, R. Koenen, and F. Pereira (2002), “MPEG-7: the generic Multimedia Content Description Standard”. Acedido em Janeiro 2010. Disponível em http://www.chiariglione.org/mpeg/tutorials/papers/IEEEMM_mp7overview_withcopyrigth.pdf.

[15] J. Hunter (2001). “Adding Multimedia to the Semantic Web – Building an MPEG-7 Ontology”. Acedido em Fevereiro 2010. Disponível em http://www.itee.uq.edu.au/~jane/jane-hunter/swws.pdf.

[16] P. Schallauer, W. Bailer, G. Thallinger (2006). “A Description Infrastructure for Audiovisual Media Processing Systems Based on MPEG-7”. Acedido em Março 2010. Disponível em http://prestospace.org/training/images/jukm.pdf.

[17] L. Obrst (2004), “Ontologies for Semantically Interoperable Systems”. Acedido em Março 2010. Disponível em http://ontolog.cim3.net/file/resource/tutorial/OntologiesForSemanticallyInteroperableSystems-ONTOLOG–LeoObrst_20040115b.htm.

[18] G. Caire (2010), “JADE Tutorial Application-defined Content Languages and Ontologies”. Acedido em Outubro 2010. Disponível em http://jade.cselt.it/doc/index.html.

[19] G. Caire, F. Bellifemine e D. Greenwood (2004), “Developing Multi-Agent Systems with JADE”, Editora Wiley. Acedido em Outubro 2010. Disponível em https://sisis.rz.fhtw-berlin.de/inh2008/12361265.pdf.

[20] JADE Board (2012), “Jade Web Services Integration Gateway (WSIG) Guide”, Telecom Italia. Acedido em Agosto 2012. Disponível em http://jade.tilab.com/doc/tutorials/WSIG_Guide.pdf.

[21] R. Ferreira e C. Cunha (2006), “Estratégia e Negócio Electrónico”. Acedido em Janeiro 2010. Disponível em http://spi.pt/negocio_electronico/documentos/manuais_PDF/Manual_III.pdf.

[22] C. Brito e C. Carvalho (2006), “Parcerias no Negócio Electrónico”. Acedido em Janeiro 2010. Disponível em http://web.spi.pt/negocio_electronico/documentos/manuais_PDF/Manual_IV.pdf.

[23] APEXTWO, CRM & Marketing Automation Experts (2012), “B2B vs Marketing – The Difference and Why it Matters”. Acedido em Agosto 2012. Disponível em http://apextwo.com/b2b-vs-b2c-marketing-difference-why-it-matters/.

[24] D. Murphy (2007), “Marketing for B2B vs B2C – Similar but Different”. Acedido em Janeiro 2011. Disponível em http://masterful-marketing.com/marketing-b2b-vs-b2c/.

[25] Z. Yang e K. Duddy (1996), “CORBA: A Platform for Distributed Object Computing”, ACM Operating System Review, Vol. 30, No. 2, 4-31. Acedido em Janeiro 2012. Disponível em http://www.itu.dk/~oladjones/masters%20thesis/materials%20from%20portals/p4-yang.pdf.

[26] M. Wasznicky, Microsoft Corporation, (2002), “Using Web Services Instead of DCOM”. Acedido em Fevereiro 2012. Disponível em http://msdn.microsoft.com/en-us/library/aa302336.aspx#webservicesdcom_topic1.

[27] J. D. Meier, S. Vasireddy, A. Babbar e A. Mackman, Microsoft Corporation, (2004), “Improving .NET Application Performance and Scalability”. Acedido em Janeiro 2012. Disponível em http://msdn.microsoft.com/en-us/library/ms998565.aspx#scalenetchapt11_topic5.

[28] Microsoft Corporation, (2012), “Choosing Communication Options in .NET”. Acedido em Agosto 2012. Disponível em http://msdn.microsoft.com/en-us/library/vstudio/8119f66k%28v=vs.100%29.aspx.

[29] W3C (2012), “Web of Services”. Acedido em Setembro de 2012. Disponível em http://www.w3.org/standards/webofservices/.

[30] OASIS (2012), “OASIS Committee Categories: Web Services”. Acedido em Setembro de 2012. Disponível em https://www.oasis-open.org/committees/tc_cat.php?cat=ws.

[31] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana (2001), “Web Services Description Language (WSDL) 1.1”. Acedido em Fevereiro de 2012. Disponível em http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[32] N. Mitra, Y. Lafon, (2007), “SOAP Version 1.2 Part 0: Primer (Second Edition)”. Acedido em Junho 2012. Disponível em http://static-71-166-250-129.washdc.east.verizon.net/eLibrary/TECH/W3C/W070427M.pdf.

[33] OASIS (2004), “Introduction to UDDI: Important Features and Functional Concepts”. Acedido em Janeiro 2012. Disponível em http://xml.coverpages.org/ni2005-02-02-a.html.

[34] W3C (1998), “Synchronized Multimedia Integration Language (SMIL) 1.0 Specification”. Acedido em Janeiro 2011. Disponível em http://www.w3.org/TR/REC-smil/c.

[35] S. Pfeiffer, C. Parker (2005), “The Annodex exchange format for time-continuous bitstreams, Version 3.0”. Acedido em Janeiro 2011. Disponível em http://www.annodex.net/TR/draft-pfeiffer-annodex-02.htmlx.

[36] S. Pfeiffer, C. Parker, A. Pang (2002), “The Continuous Media Markup Language (CMML), Version 2.0”. Acedido em Fevereiro 2011. Disponível em http://tools.ietf.org/html/draft-pfeiffer-cmml-01.

[37] S. Pfeiffer (2007), “Architecture of a Video Web – Experience with Annodex”, W3C Video Workshop. Acedido em Janeiro 2011. Disponível em www.w3.org/2007/08/video/positions/annodex.pdf.

[38] ISO/IEC JTC1/SC29/WG11N6828 (2004), “MPEG-7 Overview (version 10)” (2004). Acedido em Dezembro 2010. Disponível em http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm.

[39] I. Hirata, J. F. Hübner, J. S. Sichman (2004), “Implementação de Protocolos de Interação no Ambiente SACI”. Acedido em Fevereiro de 2011. Disponível em http://www.inf.furb.br/seminco/2005/artigos/106-vf.pdf.

[40] J. Hunter (1999), “MPEG-7 Behind the Scenes”, D-Lib Magazine, Vol. 5, N. 9. Acedido em Janeiro 2012. Disponível em http://www.dlib.org/dlib/september99/hunter/09hunter.html.

[41] P. Salembier (2002), “Overview of theMPEG-7 Standard and of Future Challenges for Visual Information Analysis”. Acedido em Janeiro 2012. Disponível em https://imatge.upc.edu/web-gps-tsc/pub/ps/JASP02_Salembier.pdf.

[42] G. Valkanas, V. Tsetsos, S. Hadjiefthymiades (2008), “The POLYSEMA MPEG-7 Video Annotator”. Acedido em Janeiro 2012. Disponível em http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-300/p08.pdf.

[43] POLYSEMA Página oficial (2007), Acedido em Janeiro 2012. Disponível em http://polysema.di.uoa.gr/index_en.html.

[44] IBM Research (2002), “VideoAnnEx Annotation Tool – User Manual”. Acedido em Janeiro 2012. Disponível em http://www.research.ibm.com/VideoAnnEx/usermanual.html.

[45] R. Schroeter, J. Hunter, D. Kosovic (2003), “Vannotea – A Collaborative Video Indexing, Annotation and Discussion System For Broadband Networks”. Acedido em Fevereiro 2012. Disponível em http://eprint.uq.edu.au/archive/00004535/01/schroeter-kcap03.pdf.

[46] M. Spaniol, R. Klamma (2005), “MEDINA: A Semi-Automatic Dublin Core to MPEG-7 Converter for Collaboration and Knowledge Management in Multimedia Repositories”. Acedido em Janeiro 2012. Disponível em http://i-know.tugraz.at/wp-content/uploads/2008/11/17_medina.pdf.

[47] J. Kunze, T. Baker (2007), “The Dublin Core Metadata Element Set” RFC5013. Acedido em Janeiro de 2012. Disponível em http://tools.ietf.org/html/rfc5013.html.

[48] C. Saathoff, N. Timmermann, S. Staab, K. Petridis, D. Anastasopoulos, Y. Kompatsiaris (2006). “M-OntoMat-Annotizer: Linking Ontologies with Multimedia Low-Level Features for Automatic Image Annotation”. Acedido em Janeiro de 2012. Disponível em http://hnsp.inf-bb.uni-jena.de/proceedings/eswc2006/proceedings-posters-demos/eswc2006-poster-demo-proceedings.pdf#page=19.

[49] aceMedia (2007), “aceMedia Annual Report – November 2007”. Acedido em Janeiro de 2012. Disponível em ftp://ftp.cordis.europa.eu/pub/ist/docs/kct/acemedia-annual-report-2007.pdf.

[50] H. Stuckenschimidt, F. van Harmelen (2003), “Information Sharing on the Semantic Web”, Editora Springer. Acedido em Fevereiro 2012. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.4702&rep=rep1&type=pdf.

[51] N. F. Noy, D. L. McGuinness (2012), “Ontology Development 101: A Guide to Creating Your First Ontology”. Acedido em Janeiro de 2012. Disponível em http://protege.stanford.edu/publications/ontology_development/ontology101.html.

[52] N. Guarino (1998), “Formal Ontology and Information Systems”. Acedido em Março de 2012. Disponível em http://www.loa-cnr.it/Papers/FOIS98.pdf.

[53] G. Klyne, J. J. Carroll, B. McBride (2004), “Resource Description Framework (RDF): Concepts and Abstract Syntax”, W3C Rec. Acedido em Junho de 2o12. Disponível em http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-Introduction.

[54] F. Manola, E. Miller, B. McBride (2004), “RDF Primer”, W3C Rec. Acedido em Março de 2012. Disponível em http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[55] D. Brickley, R. V. Guha (2000), “Resource Description Framework (RDF) Schema Specification 1.0”, W3C Candidate Rec. Acedido em Março 2012. Disponível em http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[56] T. Berners-Lee, J. Hendler, O. Lassila (2001), “The Semantic Web” Revista SCIENTIFIC AMERICAN.com. May 17, 2001. Acedido em Janeiro 2012. Disponível em

http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

[57] L. Ding, T. Finin (2006), “Characterizing the Semantic Web on the Web” 5th International Semantic Web Conference. Acedido em Fevereiro 2012. Disponível em http://aisl.umbc.edu/resources/295.pdf.

[58] G. Klyne, J. J. Carroll, B. McBride (2004), “Resource Description Framework (RDF): Concepts and Abstract Syntax” W3C Rec. Acedido em Fevereiro 2012. Disponível em http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-URI-reference.

[59] J. Tauberer (2008), “What is RDF and what is it good for?”. Acedido em Janeiro de 2012. Disponível em http://rdfabout.com/intro/?section=contents.

[60] D. Brickley, L. Miller (2010), “FOAF Vocabulary Specification 0.98”. Acedido em Março de 2012. Disponível em http://xmlns.com/foaf/spec/.

[61] E. G. Prud’Hommeaux (2007), “W3C RDF Validation Service”. Acedido em Março de 2012. Disponível em http://www.w3.org/RDF/Validator/.

[62] RDF Working Group (2004), “Resource Description Framework (RDF)”. Acedido em Janeiro de 2012. Disponível em http://www.w3.org/RDF/.

[63] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein (2001),“DAML+OIL (March 2001) Reference Description”, W3C Note . Acedido em Janeiro de 2011. Disponível em http://www.w3.org/TR/daml+oil-reference.

[64] N. Drummond, M. Horridge, H. Knublauch (2005), "Protégé-OWL Tutorial – 8th International Protégé Conference". Acedido em Março de 2011. Disponível em http://protege.stanford.edu/conference/2005/slides/T2_OWLTutorialI_Drummond_final.pdf.

[65] M. K. Smith, C. Welty, D. L. McGuinness (2004), "OWL Web Ontology Language Guide". Acedido em Fevereiro de 2012. Disponível em http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[66] DCMI (2012), "Dublin Core Metadata Initiative". Acedido em Abril de 2012. Disponível em http://dublincore.org/.

[67] J. Hunter (2002), "An Application Profile which combines Dublin Core and MPEG-7 Metadata Terms for Simple Video Description". Acedido em Novembro de 2010. Disponível em http://itee.uq.edu.au/~eresearch/papers/2002/video_appln_profile.pdf.

[68] K. Dubost (2003), "Use international date format (ISO)" W3C Tips for Webmasters. Acedido em Janeiro de 2001. Disponível em http://www.w3.org/QA/Tips/iso-date.

[69] T. Baker (2000), "A Grammar of Dublin Core", D-Lib Magazine, Vol. 6 No. 10. Acedido em Abril de 2012. Disponível em http://dlib.org/dlib/october00/baker/10baker.html.

[70] A. Isaac, E. Summers (2009), "SKOS Simple Knowledge Organization System Primer". Acedido em Abril de 2012. Disponível em http://www.w3.org/TR/skos-primer/.

[71] S. Jupp and S. Bechhofer, R. Stevens (2008), "SKOS with OWL: Don’t be Full-ish!" (2008). Acedido em Abril de 2012. Disponível em http://owl1-1.googlecode.com/svn-history/r552/trunk/www.webont.org/owled/2008/papers/owled2008eu_submission_22.pdf.

[72] A. Miles, S. Bechhofer (2009), "SKOS Simple Knowledge Organization System Reference". Acedido em Abril de 2012. Disponível em http://www.w3.org/TR/skos-reference/.

[73] E. Prud’hommeaux, A. Seaborne (2008)"SPARQL Query Language for RDF". Acedido em Abril de 2012. Disponível em http://www.w3.org/TR/rdf-sparql-query/.

[74] K. G. Clark, L. Feigenbaum, E. Torres (2008), "SPARQL Protocol for RDF". Acedido em Março de 2012. Disponível em http://www.w3.org/TR/rdf-sparql-protocol/.

[75] P. McCarthy (2005), "Search RDF data with SPARQL" (2005). Acedido em Março de 2012. Disponível em http://www.ibm.com/developerworks/xml/library/j-sparql/.

[76] Cambridge Semantics (2011), "SPARQL by Example". Acedido em Março de 2012. Disponível em http://www.cambridgesemantics.com/2008/09/sparql-by-example/#(4).

[77] FOAF (2000), "The Friend of a Friend (FOAF) project". Acedido em Abril de 2012. Disponível em http://www.foaf-project.org/.

[78] L. Dodds (2003), "FOAF-a-Matic". Acedido em Abril de 2012. Disponível em http://www.ldodds.com/foaf/foaf-a-matic.

[79] D. Brickley, L. Miller (2010), "FOAF Vocabulary Specification 0.98". Acedido em Abril de 2012. Disponível em http://xmlns.com/foaf/spec/.

[80] C. Au Yeung, I. Liccardi, K. Lu, O. Seneviratne, T. Berners-Lee (2008), "Decentralization: The Future of Online Social Networking". Acedido em Fevereiro de 2012. Disponível em http://www.w3.org/2008/09/msnws/papers/decentralization.pdf.

[81] DBpedia (2012), "The DBpedia Knowledge Base". Acedido em Abril de 2012. Disponível em http://wiki.dbpedia.org/About.

[82] W3C SWEO Community Project (2012), “SweoIG/TaskForces / CommunityProjects / LinkingOpenData – W3C Wiki”. Acedido em Abril de 2012. Disponível em http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData#Tutorials_and_Overview_Articles.

[83] DBTune (1999), “Jamendo RDF Server”. Acedido em Abril 2012. Disponível em http://dbtune.org/jamendo/.

[84] M. Hausenblas, W. Halb, Yves Raimond (2008), "What is the Size of the Semantic Web?". Acedido em Abril de 2012. Disponível em http://i-know.tugraz.at/wp-content/uploads/2008/11/1_what-is-the-size-of-the-semantic-web.pdf.

[85] Govtrack.us (2012), “Developer Documentation, “Acedido em Abril de 2012. Disponível em http://www.govtrack.us.

[86] K. Möller, S. Bechhofer, T. Heath (2009) "Semantic Web Conference Ontology". Consultado em Janeiro de 2012. Disponível em http://data.semanticweb.org/ns/swc/swc_2009-05-09.html.

[87] GoodRelations (2012), “Who uses GoodRelations?”. Acedido em Maio de 2012. Disponível em http://www.heppnetz.de/projects/goodrelations/.

[88] R. Troncy, O. Celma, S. Little, R. Garcia, C. Tsinaraki (2007), "MPEG-7 based Multimedia Ontologies: Interoperability Support or Interoperability Issue?". Acedido em Janeiro de 2012. Disponível em http://rhizomik.net/html/~roberto/papers/mareso-2007.pdf.

[89] V. M. B. Gonçalves (2007), "A Web Semântica no Contexto Educativo", Tese Doutoramento, Universidade do Porto. Acedido em Abril de 2012. Disponível em http://repositorio-aberto.up.pt/bitstream/10216/11104/2/Texto%20integral.pdf.

[90] K. C. Wilbur, M. S. Goeree, G. Ridder (2009), "Effects of Advertising and Product Placement on Television Audiences". Institute for Empirical Research in Economics University of Zurich, Working Paper Series ISSN 1424-0459. Acedido em Abril de 2012. Disponível em www.iew.uzh.ch/wp/iewwp449.pdf.

[91] N. A. P. Silva (2004), “Multi-dimentional Service-oriented ontology mapping” Tese Doutoramento – UTAD. Acedido em Março de 2012. Disponível em www.dei.isep.ipp.pt/~nsilva/R&D/PhD/PhDThesis.pdf.

[92] S. Dasiopoulou, V. Tzouvaras, I. Kompatsiaris, M. G. Strintzis (2010), "Enquiring MPEG-7 based multimedia ontologies". Multimed Tools Appl (2010) 46:331–370 DOI 10.1007/s11042-009-0387-4, Springer Science + Business Media, LLC 2009. Acedido em Abril de 2012. Disponível em http://3dlife-noe.eu/iti/files/document/mpeg_7_ontologies.pdf.

[93] A. Gordon (2006), "Business To Business Marketing – B2B – All You Need To Know". Acedido em Dezembro de 2010. Disponível em http://ezinearticles.com/?Business-To-Business-Marketing---B2B---All-You-Need-To-Know&id=396214.

[94] Najjar, K. Osama (2008), "Yahoo vs Google". Acedido em Dezembro de 2010. Disponível em http://dx.doi.org/10.2139/ssrn.1375406.

[95] H. Shekhar (2010), "RPC vs RMI". Acedido em Janeiro de 2011. Disponível em http://itgs.tistory.com/attachment/1406296529.pdf.

[96] K.Qureshi, H. Rashid (2005), "A PERFORMANCE EVALUATION OF RPC, JAVA RMI, MPI AND PVM" Malaysian Journal of Computer Science, Vol. 18 No. 2, December 2005, pp. 38-44. Acedido em Dezembro de 2010. Disponível em http://biomed2011.um.edu.my/filebank/published_article/2013/339.pdf.

[97] M. Henning “The rise and fall of CORBA” (2006), Journal Queue, ACM, Vol4, Nº5, pag 28 a 34. Disponível em http://dl.acm.org/citation.cfm?id=1378718.

[98] R. A. Goulart, C. Geyer (2000), "Um estudo comparativo sobre os modelos de objetos distribuídos CORBA, DCOM e RMI" Dissertação Universidade Federal do Rio Grande do Sul. Acedido em Dezembro de 2010. Disponível em http://www.inf.ufrgs.br/gppd/disc/cmp167/trabalhos/mp2000-1/RicardoGoulart/index.html.

[99] T. Scallan (2012), "A CORBA PRIMER". Acedido em Abril de 2012. Disponível em http://www.omg.org/news/whitepapers/index.htm

[100] B. H. Tay, A. L. Ananda (1990), "A survey of remote procedure calls". ACM SIGOPS Operating Systems Review, Vol.24,Nº3, p.68-79. Acedido em Abril de 2012. Disponível em http://dl.acm.org/citation.cfm?id=382832.

[101] A.Satpute (2008), "What is RMI?". Acedido em Fevereiro 2011. Disponível em http://www.careerride.com/RMI-Defined.aspx#architecture%20end.

[102] P. E. Chung, Y Huang, S. Yajnik, D. Liang, J. C. Shih, C. Wang (1997), "DCOM and CORBA Side by Side, Step by Step, and Layer by Layer". Acedido em Fevereiro de 2011. Disponível em http://ece842.com/S12/readings/raj2003.pdf.

[103] N. Kumar (2009), "DCOM vs .Net Remoting". Acedido em Fevereiro de 2011. Disponível em http://www.careerride.com/Remoting-VS-DCOM.aspx.

[104] R. Garcia, Celma (2005), "Semantic integration and retrieval of multimedia metadata". 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media (p. 69-80) (2005). Acedido em Janeiro de 2011. Disponível em http://ceur-ws.org/Vol-185/semAnnot05-07.pdf.

[105] T. Athanasiadis, V. Tzouvaras, K. Petridis, F. Precioso, Y. Avrithis, Y. Kompatsiaris (2005), "Using a multimedia ontology infrastructure for semantic annotation of multimedia content", Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005) at the 4th International Semantic Web Conference (ISWC 2005). Acedido em Janeiro de 2011. Disponível em http://frederic.precioso.free.fr/documents/semannot05.pdf.

[106] J. Hunter (2001), "Adding Multimedia to the Semantic Web: Building an MPEG-7 ontology", SWWS, p261-p283. Acedido em Fevereiro de 2011. Disponível em http://espace.uq.edu.au/eserv/UQ:7845/swws.pdf.

[107] H. L. Cardoso (2007), "Integrating JADE and Jess". Acedido em Abril de 2012. Disponível em http://jade.tilab.com/doc/tutorials/jade-jess/jade_jess.html.

[108] V. Pautret (2006), "Jade Semantics Add-on Programmer's guide". Disponível em http://jade.tilab.com/doc/tutorials/SemanticsProgrammerGuide.pdf.

[109] Versatile Delivery Systems (2006), “Frameline 47 Video Annotation”. Acedido em Setembro 2011. Disponível em http://www.frameline.tv/

[110] I. Horrocks (2002), "DAML+OIL: a Description Logic for the Semantic Web". Acedido em Abril de 2012. Disponível em http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1039835.

[111] R. M. França, T. D. Moreira, S. Labidi (2008), "Análise Comparativa entre as Plataformas Multiagentes JADE e FIPA-OS". Acedido em Abril de 2012. Disponível em http://www.sirc.unifra.br/artigos2008/40105_1.pdf.

[112] Apache Sogftware Foundation (2012), “Apache Axis2 Releases”. Acedido em Abril 2012. Disponível em http://axis.apache.org/axis2/java/core/download.cgi.

[113] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. F. Nielsen, A. Karmarkar, Y. Lafon, (2007), "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) W3C Recommendation". Acedido em Outubro de 2011. Disponível em http://www.w3.org/TR/2007/REC-soap12-part1-20070427/#soapenv.

[114] FIPA – Foundation For Intelligent Physical Agents (2002), “A description of the structure of FIPA ACL”. Acedido em Abril de 2012. Disponível em http://www.fipa.org/specs/fipa00061/SC00061G.html.

[115] D. Grimshaw (2010), "JADE Administration Tutorial". Acedido em Março de 2012. Disponível em http://jade.tilab.com/doc/tutorials/JADEAdmin/jadeArchitecture.html.

[116] P. Buckle, R. Hadingham, S. Poslad (2000), "The FIPA-OS agent platform: Open Source for Open Standards" Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents Vol. 355 p. 368 (2000). Acedido em Ferereiro de 2012. Disponível em http://www.elec.qmul.ac.uk/people/stefan/publications/2000-paam2000-fipaos.pdf.

[117] T. Cunningham, K. Stam, J. Faath, The jUDDI Community (2010), “jUDDI User Guide – A guide to using jUDDI”. Acedido em Outubro de 2011. Disponível em http://juddi.apache.org/docs/3.x/userguide/pdf/userguide.pdf.

[118] Centro Estudos Tecnologias Informação e da Comunicação Brasil (2007), "TIC Empresas 2007". Acedido em Janeiro de 2011. Disponível em http://www.cetic.br/empresas/2007/tic-microempresas-2007.pdf.

[119] Stanford Center for Biomedical Informatics Research (2012), “Protégé Overview”. Acedido em Novembro de 2011. Disponível em http://protege.stanford.edu/overview/index.html.

[120] P. Brittenham, F. Cubera, D. Ehnebuske, S. Graham (2001), “How to publish and find WSDL service descriptions”. Acedido em Novembro de 2011. Disponível em http://www.ibm.com/developerworks/library/ws-wsdl/.

[121] T. Berners-Lee. L. Kagal (2008), "The Fractal Nature of the Semantic Web", AI Magazine Vol. 29, N. 3. Acedido em Fevereiro de 2011. Disponível em http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/2161.

[122] V. Vasudevan (1998), “Comparing agent communication languages” OBJS Technical Note. Acedido em Janeiro de 2011. Disponível em http://www.objs.com/agility/tech-reports/9807-comparing-ACLs.html.

[123] Rhizomic (2012), “MPEG-7Ontos”. Acedido em Janeiro de 2011. Disponível em http://rhizomik.net/ontologies/mpeg7ontos.

[124] Foundation for Intelligent Physical Agents (2004), “FIPA TC Agent Management”. Acedido em Feveiro de 2011. Disponível em http://www.fipa.org/specs/fipa00023/index.html.

[125] SourceForge.net (2003), “FIPA-OS Overview”. Acedido em Julho de 2012. Disponível em http://fipa-os.sourceforge.net/index.htm.

[126] D. Grimshaw (2010), “Tutorial 1: JADE Architecture Overview”. Acedido em Abril de 2011. Disponível em http://jade.tilab.com/doc/tutorials/JADEAdmin/jadeArchitecture.html.

[127] W3Schools (2012), “SOAP Syntax”. Acedido em Março de 2012. Disponível em http://www.w3schools.com/soap/soap_syntax.asp.

[128] Source Forge (2011), “JFreeChart Download”. Acedido em Novembro 2011. Disponível em http://sourceforge.net/projects/jfreechart/files.

[129] J. J. Simas (2011), “Chart 2d” (2011). Acedido em Outubro de 2011. Disponível em http://chart2d.sourceforge.net/index.php.

[130] Jade (2012),“Java Agent DEvelopment Framework “. Acedido em Julho de 2012. Disponível em http://jade.tilab.com.

[131] Protégé (2011), “Choosing between versions of Protege”. Acedido em Dezembro de2011. Disponível em http://protegewiki.stanford.edu/wiki/Protege4Migration.

[132] Protégé (2005), “Protege 3.1.1 (build 216)” (2005). Acedido em Julho de 2011. Disponível em http://protege.cim3.net/download/old-releases/3.1.1/full/#windows.

[133] C. Giovanni (2009), “Use bean generator plugin v4 interfaces with JADE” (2009). Acedido em Julho de 2011. Disponível em http://sharon.cselt.it/pipermail/jade-develop/2009q3/014276.html.

[134] C. J. van Aart (2009), “OntologyBeanGenerator”. Acedido em Janeiro de 2012. Disponível em http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator.

[135] Semantic Web (2012), "Ontology". Acedido em Fevereiro de 2012. Disponível em http://semanticweb.org/wiki/Main_Page.

Anexo A. Ferramentas de Anotação MPEG-7

Um vídeo é constituído por um conjunto de fotogramas (*frames*) que são apresentados sequencialmente a uma determinada cadência temporal (fotograma/s). Um conjunto de fotogramas inter-relacionados (no contexto das funcionalidades associadas às ferramentas de descrição) é chamado segmento. Geralmente, o segmento é representado por um fotograma (tipicamente o primeiro) que pode ser anotado com informação relevante.

A anotação de vídeo tem-se revelado uma tarefa extremamente complexa. Têm sido encetadas inúmeras tentativas de desenvolvimento de ferramentas de anotação, mas ainda nenhuma conseguiu implementar a totalidade da norma MPEG-7 ou satisfazer todas as necessidades do utilizador final. Dada a relevância que a anotação de vídeo tem no projecto global em que esta tese se inscreve, e apesar de não ser do âmbito específico deste trabalho, serão referidas nas secções seguintes algumas contribuições nesse domínio.

POLYSEMA

O POLYSEMA – do grego *poli* (muitos) e *sema* (significados) – é uma ferramenta *open source* de anotação de conteúdo audiovisual criada pela Universidade de Atenas em parceria com a Siemens e a Lumiere Cosmos Communications (LCC) para suportar o desenvolvimento de serviços de televisão interactiva – *interactive* TV (iTV) – baseados em ontologias [42]. Permite descrever objectos multimédia usando descritores de baixo nível e facilita a descrição de alto nível recorrendo a ontologias da rede semântica que combina com o Multimedia Description Schemes (MDS) da norma MPEG-7. Actualmente, para efectuar as anotações, a ferramenta necessita que se disponha de um documento MPEG-7 com informação básica acerca dos segmentos do vídeo (*i.e.*, início e duração). Esta informação poderá ser obtida a partir de um algoritmo ou ferramenta de segmentação automática. A anotação pode, então, ter início recorrendo a dois modos de operação:

- Modo de anotação completa de vídeo – o utilizador insere a informação básica como título, resumo, data e local de criação, classificação do tipo de conteúdo (*e.g.*, maiores

de 18 anos). Para facilitar o processo de anotação, não é aconselhável a utilização de qualquer ontologia de domínio.

- Modo de anotação de segmentos vídeo – o utilizador pode descrever segmentos de três formas: *FreeText*, *Structured* e *Keyword*. Nas descrições do tipo estruturado podem-se utilizar ontologias de domínio descritas em *Resource Description Framework Schema*/*Web Ontology Language* (RDFS/OWL) para controlo de vocabulário.

Na página oficial da ferramenta [43] pode ser consultada toda a informação disponível sobre este projecto assim como efectuar a descarga do POLYSEMA e desenvolvimento.

BM VIDEOANNEX TOOL

A VideoAnnex [44] é uma ferramenta desenvolvida pela IBM que permite efectuar a anotação de sequências de vídeo usando metadados MPEG-7 para a descrição de cenas, objectos-chave, descrição de eventos que podem ser guardadas num ficheiro XML. Permite também abrir os ficheiros MPEG-7 de forma a visualizar as anotações relativas a determinada sequência de vídeo. Apesar do léxico estar restringido apenas a estas 3 categorias poderão também ser usadas palavras-chave em formato de texto livre. Para proceder à anotação tem de se dispor do vídeo original em formato MPEG e de se segmentar o vídeo em pequenas parcelas para identificar as cenas. Essa segmentação poderá ser feita recorrendo a um programa externo ou usando uma ferramenta própria chamada IBM CueVideo Shot Detection Toolkit. Segundo [45], a interface de utilizador esta restrita a vídeos com características pré-definidas, *i.e.*, se um vídeo tiver um formato diferente não se consegue visualizar correctamente. Além disso a ferramenta não suporta a segmentação hierárquica de vídeo. Actualmente esta ferramenta encontra-se descontinuada.

FRAMELINE 47

Trata-se de uma ferramenta comercial de anotação de vídeo, desenvolvido pela empresa Versatile Delivery Systems e que recentemente se tornou *software* livre para o sistema operativo Macintosh OSX. Esta ferramenta disponibiliza descritores que podem ser aplicados a três níveis: ao ficheiro de vídeo, a cada segmento e a grupos de segmentos. Disponibiliza dez campos para a anotação do conteúdo:

1 *Who* – destina-se à anotação de pessoas ou personagens;

2 *Who Value* – serve para classificar o grau de importância do segmento (*Vital*, *Important*, *Helpful*, *Optional*, *No Value* e *Any Value*);
3 *What* – destina-se à anotação de objectos;
4 *Thread* – permite descrever o vídeo por cenas e episódios;
5 *Thread Value* – permite classificar o grau de importância das cenas e episódios;
6 *Thread Stage* – atribui um estado à narrativa de acordo com a sua estrutura (introdução, desenvolvimento, fim);
7 *Event* – permite descrever acontecimentos reais (explosão, beijo, palmas, *etc.*);
8 *Place* – serve para indicar locais (ISEP, Casa da Maria, *etc.*);
9 *Time* – descreve um determinado período de tempo (manhã, tarde, amanhecer, *etc.*);
10 *Value* – permite efectuar uma descrição genérica de um segmento.

A funcionalidade de auto-segmentação baseia-se num processo de procura de diferenças substanciais entre tramas consecutivas (*e.g.*, cortes de câmara). Quando uma ocorrência é detectada, é criado um ponto de segmentação que poderá posteriormente ser comentado.

Quando um vídeo não segmentado é aberto pela primeira vez no modo de edição, aparece um menu que permite configurar o grau de sensibilidade, o número mínimo de fotogramas por segmento e a porção do vídeo a que queremos aplicar a auto-segmentação. Cada segmento é representado por uma imagem – *Poster Image Frame* – que, em modo de edição, é retirada do primeiro quadro do segmento. Também permite definir um quadro alternativo de representação do segmento, bastando seleccionar o quadro pretendido dentro desse segmento. O *software* possui um conjunto bastante completo de funcionalidades que permitem gerir, editar, anotar, capturar, filtrar e reproduzir os ficheiros de vídeo.Apesar dos inúmeros esforços de desenvolvimento, a ferramenta não conseguiu tornar-se num produto comercial por não satisfazer os padrões de exigência do mercado. O seu desenvolvimento terminou na versão 3.2 dado ser considerado um produto inviável sob ponto de vista económico [109].

MEDINA

Trata-se de uma ferramenta desenvolvida pela RWTH Aachen University [46] e apresentada em Julho de 2005. Foi construída no âmbito do projecto MPEG-7 *Encoding of Dublin Core Information and Naming Application* (MEDINA) com o objectivo de converter o formato Dublin Core em MPEG-7. O processo de conversão é feito em duas

fases. (*i*) o documento original passa por um *parser* interno que o converte num documento Dublin Core (DC) "normalizado" com as 15 propriedades estipuladas [47]; (*ii*) é efectuado o mapeamento entre os elementos DC com os descritores MPEG-7. A ideia é permitir manter as anotações de componentes multimédia MPEG-1 (mais antigas) e compatibilizá-las com o formato MPEG-7, enriquecendo as anotações com descrições associadas às fontes multimédia, *e.g.*, aspectos espaciais, espácio-temporais e temporais para os quais o formato Dublin Core não se encontrava vocacionado. Na realidade trata-se de um conversor de formatos e não de uma ferramenta de anotação MPEG-7.

M-OntoMat-Annotizer

Esta ferramenta de anotação foi desenvolvida pelo aceMedia Consortium. O aceMedia foi um projecto de 4 anos coordenado pela empresa Motorola e que contou com a colaboração activa de 13 parceiros de diferentes áreas tecnológicas e académicas [49]. O objectivo do aceMedia foi o desenvolvimento de ferramentas e tecnologias de suporte à interacção dos utilizadores com os seus conteúdos multimédia, dando especial atenção à descrição automática [48]. Apesar dos esforços empreendidos, o projecto terminou sem dar origem a qualquer recomendação.

Anexo B. Publicação de Serviços no UDDI

Para ilustrar a publicação de serviços Web no serviço de registos UDDI apresenta-se a seguinte listagem que foi obtida a partir da janela de *Output* do NetBeans no separador *Apache Tomcat*. A listagem é constituída pelas mensagens trocadas entre o *Servlet* WSIG e o UDDI durante o registo de um agente produtor `uzina1` da camada intermédia.

```
30 Out 2012 20:34:56,400 --> Start wsigs's registration from agent:
( agent-identifier :name Uzina1@10.0.2.15:1099/JADE :addresses
(sequence http://10.0.2.15:7778/acc ))
30 Out 2012 20:34:56,456 --> Write WSDL for service: Uzina1
30 Out 2012 20:34:56,480 --> Create new wsig service: WSIGService
(name=Uzina1, agentOnto=NegPub, mapper=class
brokerage.NegPub.ontology.NegPubOntoMapProducer)
30 Out 2012 20:34:56,480 --> Register service Uzina1 into UDDI

Request message WSIG – > UDDI:
<uddiv2:get_authToken cred="" generic="2.0" userID="wsig" xmlns:uddiv2="urn:uddi-
org:api_v2"/>

Response message UDDI – > WSIG:
<ns1:authToken generic="2.0" operator="jUDDI.org" xmlns:ns1="urn:uddi-
org:api_v2">
<ns1:authInfo>authToken:424762A0-22D1-11E2-A4DB-B376B5E39613</ns1:authInfo>
</ns1:authToken>

Request message WSIG – > UDDI:
<uddiv2:save_tModel generic="2.0" xmlns:uddiv2="urn:uddi-org:api_v2">
<uddiv2:authInfo>authToken:424762A0-22D1-11E2-A4DB-B376B5E39613</uddiv2:authInfo>
<uddiv2:tModel tModelKey=""><uddiv2:name>WSIG's tModel for Uzina1</uddiv2:name>
<uddiv2:overviewDoc>
<uddiv2:overviewURL>http://localhost:8080/wsig/ws/Uzina1?WSDL</uddiv2:overviewURL
>
</uddiv2:overviewDoc>
</uddiv2:tModel>
</uddiv2:save_tModel>

Response message UDDI – > WSIG:
<ns1:tModelDetail generic="2.0" operator="jUDDI.org" xmlns:ns1="urn:uddi-
org:api_v2">
<ns1:tModel authorizedName="WSIG Publisher" operator="jUDDI.org"
tModelKey="uuid:42757780-22D1-11E2-A4DB-E19491F01B9B">
<ns1:name>WSIG's tModel for Uzina1</ns1:name>
<ns1:overviewDoc>
<ns1:overviewURL>http://localhost:8080/wsig/ws/Uzina1?WSDL</ns1:overviewURL>
</ns1:overviewDoc>
</ns1:tModel>
</ns1:tModelDetail>

Request message WSIG – > UDDI:
<uddiv2:get_authToken cred="" generic="2.0" userID="wsig" xmlns:uddiv2="urn:uddi-
org:api_v2"/>

Response message UDDI – > WSIG:
<ns1:authToken generic="2.0" operator="jUDDI.org" xmlns:ns1="urn:uddi-
org:api_v2">
```

```
<ns1:authInfo>authToken:429E3530-22D1-11E2-A4DB-93B3805D99A5</ns1:authInfo>
</ns1:authToken>

Request message WSIG – > UDDI:
<uddiv2:save_service generic="2.0" xmlns:uddiv2="urn:uddi-org:api_v2">
<uddiv2:authInfo>authToken:429E3530-22D1-11E2-A4DB-93B3805D99A5</uddiv2:authInfo>
<uddiv2:businessService businessKey="7A7B3E00-00C5-11E1-BE00-979858722BD2"
serviceKey="">
<uddiv2:name>WSIG's businessService for Uzina1</uddiv2:name>
<uddiv2:categoryBag>
<uddiv2:keyedReference keyName="fipaServiceName" keyValue="Uzina1" tModelKey=""/>
<uddiv2:keyedReference keyName="GetAdProfile" keyValue="GetAdProfile"
tModelKey=""/>
<uddiv2:keyedReference keyName="fipaServiceName" keyValue="Uzina1" tModelKey=""/>
<uddiv2:keyedReference keyName="SetAdResult" keyValue="SetAdResult"
tModelKey=""/>
<uddiv2:keyedReference keyName="fipaServiceName" keyValue="Uzina1" tModelKey=""/>
<uddiv2:keyedReference keyName="GetProduct" keyValue="GetProduct" tModelKey=""/>
<uddiv2:keyedReference keyName="fipaServiceName" keyValue="Uzina1" tModelKey=""/>
<uddiv2:keyedReference keyName="GetProfile" keyValue="GetProfile" tModelKey=""/>
</uddiv2:categoryBag>
</uddiv2:businessService>
</uddiv2:save_service>

Response message UDDI – > WSIG:
<ns1:serviceDetail generic="2.0" operator="jUDDI.org" xmlns:ns1="urn:uddi-
org:api_v2">
<ns1:businessService businessKey="7A7B3E00-00C5-11E1-BE00-979858722BD2"
serviceKey="42C174A0-22D1-11E2-A4DB-84E7A992F209">
<ns1:name>WSIG's businessService for Uzina1</ns1:name>
<ns1:categoryBag>
<ns1:keyedReference keyName="fipaServiceName" keyValue="Uzina1"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="GetAdProfile" keyValue="GetAdProfile"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="fipaServiceName" keyValue="Uzina1"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="SetAdResult" keyValue="SetAdResult"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="fipaServiceName" keyValue="Uzina1"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="GetProduct" keyValue="GetProduct"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="fipaServiceName" keyValue="Uzina1"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
<ns1:keyedReference keyName="GetProfile" keyValue="GetProfile"
tModelKey="UUID:A035A07C-F362-44dd-8F95-E2B134BF43B4"/>
</ns1:categoryBag>
</ns1:businessService>
</ns1:serviceDetail>

Request message WSIG – > UDDI:
<uddiv2:get_authToken cred="" generic="2.0" userID="wsig" xmlns:uddiv2="urn:uddi-
org:api_v2"/>

Response message UDDI – > WSIG:
<ns1:authToken generic="2.0" operator="jUDDI.org" xmlns:ns1="urn:uddi-
org:api_v2">
<ns1:authInfo>authToken:42D6F870-22D1-11E2-A4DB-F8B9527F471D</ns1:authInfo>
</ns1:authToken>

Request message WSIG – > UDDI:
<uddiv2:save_binding generic="2.0" xmlns:uddiv2="urn:uddi-org:api_v2">
<uddiv2:authInfo>authToken:42D6F870-22D1-11E2-A4DB-F8B9527F471D</uddiv2:authInfo>
<uddiv2:bindingTemplate bindingKey="" serviceKey="42C174A0-22D1-11E2-A4DB-
84E7A992F209">
```

```
<uddiv2:accessPoint
URLType="http">http://localhost:8080/wsig/ws</uddiv2:accessPoint>
<uddiv2:tModelInstanceDetails>
<uddiv2:tModelInstanceInfo tModelKey="uuid:42757780-22D1-11E2-A4DB-
E19491F01B9B"/>
</uddiv2:tModelInstanceDetails>
</uddiv2:bindingTemplate>
</uddiv2:save_binding>

Response message UDDI – > WSIG:
<ns1:bindingDetail generic="2.0" operator="jUDDI.org" xmlns:ns1="urn:uddi-
org:api_v2">
<ns1:bindingTemplate bindingKey="42E43EE0-22D1-11E2-A4DB-8787F8D4A2F8"
serviceKey="42C174A0-22D1-11E2-A4DB-84E7A992F209">
<ns1:accessPoint URLType="http">http://localhost:8080/wsig/ws</ns1:accessPoint>
<ns1:tModelInstanceDetails>
<ns1:tModelInstanceInfo tModelKey="uuid:42757780-22D1-11E2-A4DB-E19491F01B9B"/>
</ns1:tModelInstanceDetails>
</ns1:bindingTemplate>
</ns1:bindingDetail>

30 Out 2012 20:34:57,805 --> End wsigs's registration from agent:
( agent-identifier :name Uzina1@10.0.2.15:1099/JADE :addresses
(sequence http://10.0.2.15:7778/acc ))
```

Para registrar um serviço o Servlet WSIG troca um conjunto de chaves com o UDDI. Este processo é realizado em 3 etapas:

(i) O Servlet WSIG envia um pedido de armazenamento do modelo do serviço (`save_tModel`) que comporta o nome do modelo e o URL do ficheiro WSDL do serviço; o UDDI gera uma chave para o `tModel` que devolve ao Servlet WSIG;

(ii) O Servlet WSIG envia um pedido de armazenamento do serviço (`save_service`) com as operações do serviço, a chave da empresa (`BusinessKey`) e a chave a usar nessa operação; o UDDI gera e devolve uma chave do serviço (`serviceKey`) única para todas as invocações de operações do serviço;

(iii) O Servlet WSIG envia um pedido de armazenamento do ponto de acesso ao serviço (`save_binding`) que inclui a chave do serviço (`serviceKey`), a chave do modelo (`tModelKey`) com a indicação do URL do serviço (`accessPoint`); o UDDI gera a chave da ligação (`bindingKey`) que devolve juntamente com a confirmação das chaves e dos dados recebidos.

Cada fase é precedida pela obtenção da respectiva chave de autorização (`getAuthToken`) gerada pelo UDDI, obrigando ao cumprimento da sequência exacta de mensagens.