ISCTE IUL
Instituto Universitário de Lisboa

Departamento das Ciências e Tecnologias da Informação

ONTOLOGIAS O3F: CONVERSÃO DE UML E EXTRACÇÃO EM CO3L

Jairo Avelar Maria

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia Informática
Especialidade em Sistema de Informação e Gestão do Conhecimento

Orientador:

Doutor Luís Botelho, Professor Associado

INSTITUTO SUPERIOR DE CIÊNCIAS DO TRABALHO E DA EMPRESA

Agosto de 2010

ISCTE IUL
Instituto Universitário de Lisboa

Departamento das Ciências e Tecnologias da Informação

ONTOLOGIAS O3F: CONVERSÃO DE UML E EXTRACÇÃO EM CO3L

Jairo Avelar Maria

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia Informática
Especialidade em Sistema de Informação e Gestão do Conhecimento

Orientador:

Doutor Luís Botelho, Professor Associado

INSTITUTO SUPERIOR DE CIÊNCIAS DO TRABALHO E DA EMPRESA

Agosto de 2010

Resumo

Esta dissertação centra-se no modelo de representação de ontologias O3F e da correspondente linguagem CO3L, ambos desenvolvidos no grupo de Agentes e Inteligência Artificial do Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL. Pretende-se alcançar dois objectivos. O primeiro consiste em melhorar e expandir o O3F e o CO3L para aumentar a sua expressividade, especialmente no sentido de permitir a representação de modelos UML e ontologias OWL. O segundo consiste na criação de ferramentas computacionais para ontologias O3F, a principal das quais é um conversor automático de UML para O3F/CO3L. Pretende-se assim que as ontologias O3F possam ser especificadas em UML, tirando partido das suas ferramentas de desenvolvimento, e aproximar o O3F da grande comunidade de especialistas de modelação UML. Adicionalmente, foram também feitas contribuições para a construção de um servidor de ontologias O3F e desenvolveu-se um mecanismo capaz de extrair, na linguagem textual CO3L, as ontologias armazenadas nesse servidor.

Decidiu-se apostar no O3F porque, quando comparado com outras abordagens à representação de ontologias (OWL, Ontolingua e UML), revela várias vantagens e, além disso, tem sido desenvolvida no grupo de investigação em que esta dissertação se integra. Exemplos dessas vantagens são a capacidade de especificar acções ou de modelar segundo vários paradigmas (orientado a objectos, relacional/funcional, ou misturando ambos).

A avaliação deste trabalho conclui que as melhorias feitas ao modelo e à linguagem têm utilidade e cumpriram os seus objectivos, e que as ferramentas desenvolvidas efectuam processamentos correctos, têm utilidade e são práticas de usar.

Palavras chave: Ontologias, UML, O3F, CO3L, ferramentas para ontologias

Abstract

This dissertation is focused on the ontology representation model O3F and the corresponding language CO3L, both developed by the Agents and Artificial Intelligence group of the Department of Sciences and Information Technologies of ISCTE-IUL. It is intended to achieve two goals. The first one consists of improving and expanding O3F and CO3L in order to increase their expressiveness, especially to allow the representation of UML models and OWL ontologies. The second one consists of developing computer tools for O3F ontologies, the most important of which is an automatic translator from UML to O3F/CO3L. This translator is meant to allow the specification of O3F ontologies in UML, taking advantage of its development tools, and to bridge the gap between O3F and the large and important UML community. In addition, contributions were made to build an ontology server for O3F, and a mechanism capable of extracting, in the CO3L text language, the ontologies stored on the developed server.

It was decided to invest in O3F because, when compared to other formal representation languages (OWL, Ontolingua and UML), it shows several advantages and also because it has been developed in the research group where this dissertation was done. Some examples of O3F advantages are its capability to specify actions and is multi-paradigm modeling abilities (object oriented, relational/functional or mixing both).

The evaluation of this work concludes that the improvements made to the model and language achieved their goals, and that the developed tools are correct, useful and user-friendly.

Key words: Ontologies, UML, O3F, CO3L, Ontology tools

Agradecimentos

Ao professor Luís Botelho por todo o trabalho de orientação, por todos os debates sempre produtivos e todo o esforço e apoio dispendidos ao longo da dissertação. Acima de tudo agradeço pelo grande interesse e dedicação extrema que são de facto inspiradores e qualidades cada vez mais raras.

Ao Carlos Correia por todo o trabalho que desenvolvemos em comum. Pelas várias sessões de trabalho onde discutimos ideias e por toda a partilha de informação e dicas durante todo o trabalho da dissertação.

Ao professor Pedro Ramos por ter disponibilizado os vários diagrama UML que serviram para testar e avaliar o trabalho desenvolvido.

A todos os que participaram na realização dos inquéritos de avaliação deste trabalho que muito ajudaram no desenvolvimento do capítulo de avaliação.

À minha familia por todo o apoio e encorajamento os quais foram fundamentais para alcançar os meus objectivos. É também a eles que dedico este trabalho.

Glossário

ATL: *ATLAS Transformation Language*. Linguagem para a transformação de modelos que integra a tecnologia M2M.

DAML+OIL: *DARPA Agent Modeling Language + Ontology Inference Layer.* Linguagem para descrição de ontologias que precedeu o OWL. (DARPA - Defense Advanced Research Projects Agency).

FIPA: *Foundation for Intelligent Physical Agents.* Organização para o desenvolvimento e estabelecimento de normas para sistemas baseados em agentes.

Frame Ontology: Conjunto de termos que caracterizam as convenções usadas em sistemas de representação de conhecimento orientados a objectos.

KIF: *Knowledge Interchange Format*. Linguagem para a troca de conhecimento entre sistemas computacionais díspares.

M2M: *Model to Model.* Implementação do Eclipse do *standard* QVT do OMG

MDA: *Model Driven Architecture*. Técnica de desenho de software criada pelo OMG, onde são definidas algumas directrizes para a estruturação de especificações expressas sob a forma de modelos.

MOF: *Meta-Object Facility*. Uma especificação do OMG para a criação de meta modelos.

OCL: *Object Constraint Language*. Uma linguagem declarativa para a descrição de regras que podem ser aplicadas a modelos UML ou a qualquer outro modelo descrito segundo um meta modelo concordante com o MOF.

ODM: *Ontology Definition Metamodel*. Uma especificação do OMG para tornar os conceitos da norma MDA aplicáveis ao desenvolvimento de ontologias.

OKBC: *Open Knowledge Base Connectivity.* Um protocolo para acesso a conhecimento armazenado em sistemas de representação do conhecimento.

Ontolingua: Ao longo da dissertação o termo Ontolingua é usado para referir a linguagem e o sistema como o mesmo nome.

Ontolingua (Sistema): Consiste num sistema para criar, editar, modificar e usar ontologias de uma forma distribuída e colaborativa e também numa biblioteca de ontologias modulares e reutilizáveis. Usa o modelo de representação OKBC Knowledge Model.

Ontolingua (Linguagem): Linguagem baseada em KIF e com base na *Frame Ontology*.

O3F: *Object Oriented Ontology Framework*. Modelo para descrição de ontologias, compatível com diferentes paradigmas.

O3 Server: Servidor de gestão e manutenção de ontologias O3F para o qual contribui parte do trabalho desenvolvido na dissertação.

CO3L Edit: Um editor de ontologias, especializado na linguagem CO3L, com capacidade de armazenamento de ontologias quer em ficheiros locais do sistema de ficheiros, quer remotamente em base de dados, através de ligação ao *O3 Server*.

UML2O3F: Conversor automático de modelos UML em ontologias O3F desenvolvido no âmbito da dissertação.

OMG: *Object Managment Group*. Instituição de normalização de tecnologias de objectos entre as quais o UML e o MOF.

CO3L: *Compact Object Oriented Ontology Language.* Linguagem de descrição de ontologias de acordo com o modelo O3F.

OWL: *Web Ontology Language*. Linguagem de descrição de ontologias criada pelo W3C especialmente pensada para a Web. É a linguagem de descrição de ontologias mais disseminada hoje em dia.

QVT: *Query/View/Transformation*. Especificação do OMG para a transformação de modelos compatíveis com o MOF.

RDF: *Resource Description Framework*. Especificação do W3C para a troca de informação entre sistemas.

XMI: *XML Metadata Interchange*. Formato baseado em XML, definido pelo OMG, para a troca de informação de modelos, em particular modelos UML.

XPath: *XML* Path Language. Linguagem de interrogação para consulta de elementos de um ficheiro XML.

XSLT: *eXtensible Stylesheet Language Transformation*. Linguagem para a transformação de ficheiros XML noutros formatos de texto.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1 Introdução

Actualmente as ontologias têm, cada vez mais, um papel activo nas várias áreas das Tecnologias de Informação, em particular destaca-se o papel que representam na comunicação entre agentes e na emergente Web Semântica, uma Web onde os seus elementos são descritos de uma maneira que possam ser percebidos por computadores e não apenas por pessoas. Contudo, o trabalho em ontologias enfrenta alguns obstáculos à sua expansão e desenvolvimento. Os mais significativos são o pequeno número de profissionais com conhecimento nesta matéria, a inexistência de uma abordagem consensual à representação de ontologias, e a escassez e reduzida sofisticação de ferramentas computacionais para o desenvolvimento e processamento de ontologias, as quais requerem um nível elevado de conhecimento especializado (Falkovych, Sabou, & Stuckenschmidt, 2003) .

Apesar de não ser uma linguagem originalmente pensada para a representação de ontologias, são muitos os autores, como Cranefield (Cranefield, 2001) e Kogut e a sua equipa (Kogut *et al*., 2002), que advogam que o UML (*Unified Modeling Language*) permite representar conhecimento do mesmo tipo daquele que é representado em ontologias. Recorrendo em particular aos seus Diagramas de Classes e Diagramas de Objectos é possível descrever os aspectos estáticos de domínios, especificando as propriedades dos seus elementos e as relações entre eles.

Ao contrário do que se passa com as linguagens de representação de ontologias, o UML é amplamente disseminado e conta com um vasto número de profissionais que o estudam, usam e compreendem em profundidade (Kabilan & Johannesson, 2004). É uma linguagem madura e que é ensinada nas faculdades e usada nas indústrias (Kogut *et al*., 2002). O UML conta também com imensas ferramentas CASE (*Computer-Aided Software Engineering*) que o suportam e que apoiam o desenvolvimento dos seus modelos e diagramas (Kogut *et al*., 2002). Essas ferramentas são também desenvolvidas por empresas especializadas que oferecem suporte à sua utilização e manutenção. Pode ainda ser visto com uma vantagem o facto de o UML se exprimir

de uma forma gráfica o que é mais apelativo ao olho humano e mais propício a compreensão de quem lê os diagramas sem conhecer profundamente o UML.

Um dos objectivos da presente dissertação é a criação de uma ponte entre as abordagens à descrição de ontologias e o UML, com o fim de suprir alguns dos problemas, acima referidos, relativos ao trabalho em ontologias, através das vantagens do UML. Mais concretamente, a presente dissertação pretende contribuir para a evolução do modelo O3F (*Object Oriented Ontology Framework*) (Mota *et al*., 2003) e da linguagem CO3L (*Compact Object Oriented Ontology Language*) (Botelho & Ramos, 2003) através da melhoria e expansão do modelo e da linguagem e da construção de ferramentas computacionais para operar com estas tecnologias.

1.1 Assunto e Motivação

A presente dissertação tem dois objectivos fundamentais: melhorar o O3F e o CO3L e criar ferramentas para o desenvolvimento de ontologias O3F.

O primeiro objectivo consiste no melhoramento e expansão do próprio O3F e do CO3L. Esta motivação advém do facto de que apesar de ambos – o modelo e a linguagem – serem ainda pouco conhecidos, são também apostas importantes por duas razões. A primeira razão diz respeito às vantagens do modelo O3F e da linguagem CO3L em relação às linguagens de representação de ontologias mais usadas hoje em dia. Entre essas vantagens destacam-se a capacidade de representar acções e a capacidade de modelar ontologias segundo um paradigma orientado a objectos, segundo um paradigma não orientado a objectos (funcional e relacional estendido com acções), ou ainda misturando ambos. Estas e outras vantagens estão descritas em maior detalhe no Capítulo 3. Este objectivo foi partilhado e atingido em conjunto com a dissertação "Ontologias O3F: Um Servidor e um Editor CO3L" (Correia, 2010). A segunda razão para a aposta desta dissertação no modelo O3F e na linguagem CO3L prende-se com a génese, o desenvolvimento e a utilização desses modelo e linguagem. Ambos foram e continuam a ser desenvolvidos pelo grupo de Agentes e Inteligência Artificial do Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL; ambos são usados nas aulas das licenciaturas do departamento; ambos continuam a dar origem a enunciados de dissertações de mestrado e teses de doutoramento; e ambos foram objectos de um número significativo de publicações científicas deste

grupo de investigação. As mudanças realizadas no O3F pretendem melhorar alguns dos seus elementos os quais, ao longo do tempo, foram revelando falhas que os impediam de satisfazer completamente o propósito para o qual foram criados ou que, por outro lado, foram redesenhados para se enquadrarem melhor na lógica do modelo O3F. Para além disto foram ainda propostas extensões que permitem melhorar a compatibilidade do O3F com outras linguagens de modelação importantes como o OWL e o UML.

O segundo objectivo consiste em criar ferramentas para o desenvolvimento e partilha de ontologias, contribuindo também para atrair mais profissionais para esta área, em particular profissionais de modelação altamente qualificados por exemplo, os que trabalham com o UML. Este objectivo decompõe-se em vários outros subobjectivos. O mais importante destes é o desenvolvimento da ferramenta UML2O3F, um conversor de Diagramas de Classes e de Objectos UML em ontologias O3F (*Object Oriented Ontology Framework*) (Mota *et al*., 2003). O segundo destes subobjectivos é contribuir para o desenho e criação de um servidor de ontologias O3F (O3 Server) que possibilite a gestão e disponibilização pública de ontologias. Finalmente, o terceiro subobjectivo é a criação de um mecanismo de extracção de ontologias associado ao servidor O3F. Este mecanismo permite obter, na linguagem textual CO3L (*Compact Object Oriented Ontology Language*) (Botelho & Ramos, 2003), qualquer ontologia mantida no servidor de O3F.

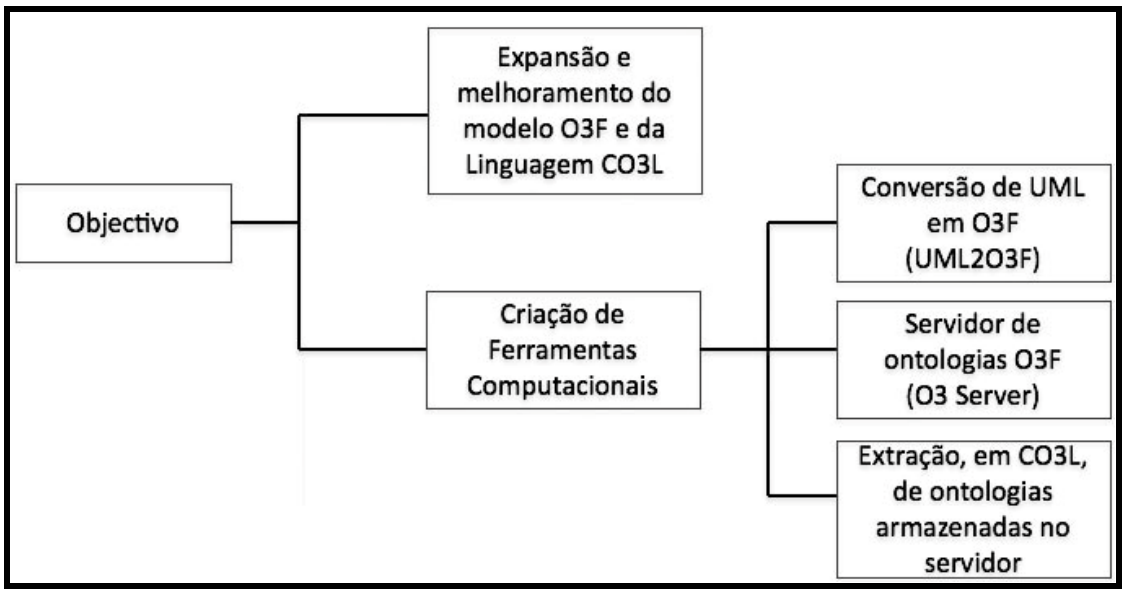


Figura 1 – Objectivos da dissertação

A motivação de criar uma ferramenta que transforme modelos UML em ontologias O3F é que essa conversão permite tirar partido das vantagens do UML para resolver os problemas referidos relativamente ao trabalho com ontologias - existirem ainda poucos profissionais e poucas ferramentas associados ao desenvolvimento e processamento de ontologias, para além de não existir uma abordagem consensual à sua representação. O conversor permitirá que pessoas com conhecimento em UML (um modelo largamente disseminado, com uma grande comunidade de utilizadores, e suportado por variadas ferramentas de desenvolvimento) possam usar os seus Diagramas de Classes e Diagramas de Objectos para especificar ontologias.

Para além disto, o conversor oferece também a possibilidade de que todo o conhecimento que existe expresso sob a forma de um modelo UML constituído por Diagramas de Classes e Diagramas de Objectos possa ser expresso sob a forma de uma ontologia.

O objectivo da construção de um servidor de ontologia O3F, o O3 Server, é motivado pela necessidade de facilitar o acesso público e partilha de ontologias, e pela existência de um projecto mais geral que pretende reunir várias ferramentas para o desenvolvimento de ontologias O3F e para a sua partilha em diversos formatos. Pretende-se que o servidor de ontologias O3F (O3 Server) seja capaz de armazenar as ontologias geradas pelo UML2O3F, bem como por outra qualquer ferramenta, e que as torne publicamente acessíveis em formatos adequados para cada cliente (e.g., UML, OWL). Este objectivo é também partilhando com a dissertação "Ontologias O3F: Um Servidor e um Editor CO3L" (Correia, 2010) pelo que a construção do servidor foi feita em parceria com o seu autor.

O mecanismo de extracção de ontologias está directamente relacionado com o servidor de ontologias O3F uma vez que faz parte da sua interface de acesso público. Não haveria qualquer utilidade em guardar ontologias sem a existência de um meio de lhes aceder posteriormente. A motivação para a criação deste mecanismo de extracção é portanto evidente: possibilitar que as ontologias armazenadas no servidor possam ser obtidas no formato CO3L por qualquer ferramenta e para os mais variados usos.

1.2 Contribuições

As contribuições deste trabalho estão relacionadas com os seus principais objectivos (Figura 1). No entanto, as contribuições mais significativas do ponto de vista do avanço do estado dos conhecimentos sobre a representação e processamento de ontologias dizem respeito à conversão de UML em ontologias O3F, e aos melhoramentos do O3F e da linguagem CO3L.

Os melhoramentos ao O3F e ao CO3L foram feitos em conjunto com outra dissertação (Correia, 2010) que também se centra neste modelo e linguagem. Destaca-se a introdução de novos conceitos como indivíduos, hierarquias e novas facetas. Todos estes melhoramentos são descritos com o devido pormenor no Capítulo 3

A conversão de UML para O3F é um problema inovador na medida em que nunca foi resolvido – não há outro conversor de UML para O3F. Adicionalmente, muitos autores realizaram já conversões de UML para linguagens de ontologias partilhando a motivação de usar o UML para resolver alguns problemas relativos ao trabalho com ontologias. Essas conversões podem ser classificadas em duas categorias. Uma primeira que agrupa as conversões feitas a partir de UML normalizado (*standard*) onde os elementos presentes num dado modelo UML são representados usando elementos equivalentes de uma linguagem de representação de ontologias. Uma segunda categoria que se destina a converter modelos construídos à custa de extensões à expressividade do UML. Estas extensões são feitas recorrendo a estereótipos e meta classes que pretendem reproduzir as construções de uma dada linguagem de representação de ontologias sendo a conversão feita a partir dos elementos que pertencem à versão estendida do UML.

A primeira categoria de conversões, embora preserve a expressividade original dos elementos do UML, resulta em conversões incompletas quando elementos dos diagramas UML não têm tradução para a linguagem de descrição de ontologias na qual se pretende converter o modelo. Por outro lado, a segunda abordagem consegue uma conversão completa mas sempre à custa de extensões à expressividade do UML o que deturpa o verdadeiro significado das suas construções. Neste sentido a presente dissertação mostra-se inovadora em dois aspectos. Apresenta uma conversão de

modelos UML para ontologias O3F, uma conversão ainda nunca implementada, e realiza essa conversão a partir de UML normalizado conseguindo a tradução de todos os elementos que possam figurar num Diagrama de Classes ou num Diagrama de Objectos para O3F, desde que expressos no ficheiro XMI (*XML Metadata Interchange*) que descreve o modelo UML.

Embora não possa ser considerada uma inovação científica, há uma diferença relativamente à tecnologia usada para fazer a conversão de diagramas UML (expressos em XMI) em ontologias O3F, a qual é explicada em detalhe no Capítulo 5

Para além da ferramenta UML2O3F, a dissertação contribuiu ainda para o desenho do servidor de ontologias O3 Server (feito em conjunto com o autor da dissertação "O3F: Um Servidor e um Editor CO3L" (Correia, 2010)), e desenvolveu um mecanismo de extracção das ontologias armazenadas no servidor, usando a linguagem textual CO3L, sendo este mecanismo um dos componentes da interface de acesso público do servidor.

Estes dois aspectos da contribuição da dissertação não constituem inovações científicas tão significativas ao estado dos conhecimentos sobre ferramentas para ontologias. No entanto, e embora não se use nenhuma tecnologia inovadora, o armazenamento e disponibilização pública de ontologias encontra-se entre as melhores práticas associadas à criação de ferramentas computacionais para ontologias. Houve mesmo casos de boas ferramentas de ontologias que foram descontinuadas ou cuja utilização decresceu muito porque não se basearam num servidor publicamente acessível (Farquhar, Fikes, & Rice, 1996).

É essencial que as ferramentas de ontologias se baseiem num servidor publicamente acessível de modo que as ontologias armazenadas possam ser usadas e partilhadas por terceiros. Além da partilha e disponibilização de ontologias, um servidor possibilita o seu desenvolvimento cooperativo (Farquhar, Fikes, & Rice, 1996), o que é especialmente relevante no caso de ontologias muito extensas e complexas. Por exemplo, uma ontologia de drogas terapêuticas (Gennari *et al*., 1995) é tão extensa e complexa que não pode ser criada por um único perito, teria de ser desenvolvida cooperativamente, recorrendo a um servidor de ontologias.

1.3 Abordagem Técnica

No que diz respeito à implementação da ferramenta UML2O3F, o presente trabalho propõe uma maneira diferente das usadas em implementações passadas. A maioria das conversões realizadas no passado recorre à tecnologia XSLT (*eXtensible Stylesheet Language Transformations*), que apesar de ser bastante poderosa na leitura ficheiros baseados em XML possui uma linguagem de transformação considerada por muitos como bastante incómoda de usar e pouco flexível em algumas conversões como por exemplo conversões que exijam muito processamento de texto. A técnica de conversão proposta usa XPath (o modulo de leitura de ficheiros da XSLT) para a leitura dos modelos UML, mas recorre à linguagem Java, a qual é mais poderosa e flexível, para uma conversão dos elementos lidos. Além do mais permite ainda comunicação com uma base de dados, o que é um requisito fundamental da ferramenta que se pretende construir. Com esta solução foi mantida a parte da tecnologia XSLT associada à leitura do XMI (XPath) e, simultaneamente, aumentou-se as possibilidades da abordagem.

Depois de lidos os modelo UML, a informação relativa aos seus elementos é armazenada numa estrutura de objectos Java com a capacidade de gerar dois resultados: o primeiro é a representação textual em CO3L; o segundo é uma estrutura de objectos Java que espelha o modelo da interface de objectos do servidor de ontologias O3F. Esta estrutura de objectos é produzida a partir de uma biblioteca de classes Java criada com a *Framework Hibernate* (www.hibernate.org), implementada no âmbito de outra dissertação (Correia, 2010). Estes objectos são depois inseridos na base de dados do O3 Server sendo este o passo final do processo de conversão.

Quanto ao conversor de O3F para CO3L, este foi implementado no próprio servidor sob a forma de um procedimento de SQL armazenado no sistema de gestão de base de dados (*stored procedure*) o que facilita a sua utilização por qualquer ferramenta O3F não ficando limitado ao UML2O3F.

A construção do servidor O3 Server, mais exactamente a sua base de dados (realizada em conjunto com o autor da dissertação "O3F: Um Servidor e um Editor CO3L" (Correia, 2010)) foi feita recorrendo a tecnologia SQL Server. O modelo relacional da base de dados foi construído manualmente aplicando, de forma ponderada, as regras

de transposição de diagramas de classes para modelos relacionais. A razão que levou a aplicação manual das regras foi a de gerar um modelo mais simples e optimizado e de obter mais controlo na sua geração.

1.4 Avaliação

Os produtos desta dissertação passíveis de avaliação rigorosa são o UML2O3F (ferramenta computacional de conversão de diagramas UML em ontologias O3F), o O3 Server (servidor de ontologias), e o mecanismo de extracção de ontologias do servidor. Os melhoramentos introduzidos no modelo O3F e consequentemente na linguagem CO3L não se podem avaliar de forma directa e rigorosa.

A ferramenta de conversão UML2O3F foi avaliada relativamente a dois aspectos: a correcção técnica das conversões efectuadas (avaliação objectiva), e a sua utilização como ferramenta de software (avaliação subjectiva).

As conversões efectuadas ou estão correctas ou não. A correcção da conversão não é uma questão de opinião. No entanto, sairia completamente fora do âmbito desta dissertação, proceder a uma demonstração formal da medida em que as conversões são tecnicamente correctas. Para contornar essa dificuldade, optamos por sujeitar a correcção da avaliação a um painel de especialistas que apreciou um conjunto de conversões bastante diversificado e classificou a qualidade da conversão numa escala de 1 (mau) a 4 (bom).

A avaliação objectiva da correcção das conversões produzidas apresenta resultados muito positivos. Praticamente todas as conversões (99%) foram avaliadas com notas 4 (67%) e 3 (32%). Apenas a conversão mais complexa foi avaliada com uma nota 2, por um único elemento do painel, e nenhuma conversão foi avaliada com nota inferior a 2.

A avaliação subjectiva da ferramenta de software consistiu de um inquérito visando a utilidade e a usabilidade da ferramenta, e reflectindo ainda a apreciação global subjectiva sobre a correcção da conversão. Numa escala de 1 (mau) a 4 (bom) foram obtidos os seguintes resultados, também francamente positivos. 98% das respostas relativas à utilidade da ferramenta e à correcção global das conversões foi de 3 e 4; e 97% das respostas relativas à usabilidade da ferramenta foi de 3 ou 4. O inquérito foi

respondido por 54 participantes, na sua maioria alunos do terceiro ano das licenciaturas do Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL.

Este inquérito subjectivo deixava ainda espaço para comentários e críticas. Muitos desses comentários sugeriram a junção das ferramentas UML2O3F e CO3L Edit (Correia, 2010) numa só. Espera-se que a implementação desta junção seja conseguida a curto prazo.

A avaliação do servidor de ontologias – O3 Server – e do mecanismo de extracção de ontologias do servidor foram avaliados em tandem. Foi verificado que as ontologias inseridas no servidor são 100% iguais (à parte da formatação do texto) às ontologias extraídas do servidor. Isto mostra que, para o conjunto de 9 ontologias experimentadas, as sequências de operações relativas ao armazenamento de ontologias no servidor e à sua posterior extracção do servidor funcionam correctamente. Esta avaliação foi levada a cabo por um pequeno programa, elaborado para o efeito, que compara a ontologia antes de ser inserida na base de dados e depois de ser extraída. Ignorando questões de indentação e espaçamento, o programa mostra uma correspondência de 100%.

Na medida em que as ferramentas computacionais criadas nesta dissertação e ainda as desenvolvidas na dissertação (Correia, 2010) as quais assentam no modelo O3F e na linguagem CO3L, funcionam correctamente, permitindo o armazenamento de descrições CO3L e UML no servidor e a extracção de ontologias no formato CO3L pode ser considerada evidência de que o modelo e a linguagem são suficientemente expressivos, pelo menos para representar qualquer diagrama UML de classes e de objectos. Isto não prova que o modelo seja bom, muito menos prova que as melhorias que nele introduzimos sejam adequadas, mas constitui uma espécie de boa esperança.

1.5 Organização da Dissertação

Para além do capítulo introdutório, a dissertação encontra-se divida em mais seis capítulos.

O Capítulo 2 contém a revisão da literatura relacionada com o trabalho desenvolvido. Dado que um dos objectivos da dissertação é o melhoramento do O3F, este capítulo

inclui uma revisão das principais linguagens de descrição de ontologias. É descrito o OWL (e a recente expansão OWL 2) e o Ontolingua. Inclui-se ainda uma descrição do UML dada a sua importância para a dissertação e por muitos considerarem que apresenta características que o tornam apto para a modelação de ontologias. Para além do UML é também referido o ODM (Ontology Definition Metamodel) que pretende tornar os conceitos do MDA (*Mode-Driven Architecture*) do OMG compatíveis com a especificação de ontologias. Por fim é descrito o O3F e o CO3L e feita a comparação com as restantes abordagens.

Neste capítulo são também estudadas as várias abordagens de conversão de UML em linguagens de representação de ontologias. As várias abordagens, bem como as diferentes tecnologias usadas para a sua implementação são também comparadas entre si discutindo as suas vantagens e desvantagens.

O Capítulo 3 é dedicado à descrição completa do O3F e CO3L. O capítulo começa por descrever todos os melhoramentos e expansões introduzidas pelo presente trabalho e em seguida descreve cada elemento do actual modelo O3F.

No Capítulo 4 são descritas todas as regras de conversão de UML para O3F indicando como cada elemento que possa figurar numa Diagrama de Classes ou num Diagrama de Objectos é transformado nos correspondentes elementos O3F.

O Capítulo 5 descreve a concepção e implementação da ferramenta UML2O3F, a qual automatiza as regras de conversão descritas no Capítulo 4 Este capítulo descreve também as contribuições feitas na criação do servidor de ontologias O3 Server enfatizando a implementação da base de dados e elaboração do respectivo modelo relacional, e o mecanismo de extracção de O3F para CO3L que constitui parte da interface de acesso público do O3 Server.

O Capítulo 6 dedica-se à avaliação da dissertação. São descritos os critérios de avaliação e forma como foi feita bem como todos os resultados obtidos.

Por fim, no Capítulo 7 são apresentadas as principais conclusões sobre todo o trabalho elaborado e apontadas direcções para trabalho futuro.

Capítulo 2 Estado da Arte

De todos os quatro objectivos que a presente dissertação pretende alcançar, presentes nas folhas da árvore da Figura 1, apenas dois são de natureza que justifique uma revisão de literatura. Tratam-se do melhoramento do modelo O3F e da linguagem CO3L, e da criação da ferramenta UML2O3F, a principal ferramenta desenvolvida na dissertação que permite a conversão de modelos UML constituídos por Diagramas de Classes e Diagramas de Objectos para O3F. Estes problemas envolvem essencialmente duas áreas do conhecimento: a representação de ontologias e a conversão entre linguagens de descrição de ontologias. O presente capítulo estuda tanto as principais linguagens de descrição de ontologias como as abordagens de conversão já existentes.

Na secção 2.1, é feita uma introdução ao tema das ontologias explicando o que são e quais as suas principais aplicações.

Nas secções seguintes, até à 2.7, são descritas as principais linguagens formais de representação de ontologias. São descritos o OWL (*Web Ontology Language*) incluindo a recente extensão OWL 2, e o Ontolingua por serem as linguagens de representações de ontologias mais importantes hoje em dia. Optou-se por não se fazer uma descrição detalhada do DAML+OIL (*DARPA Agent Modeling Language + Ontology Inference Layer*) e do RDF (*Resource Description Framework*) pois o OWL é vista como sucessor do primeiro e como um sobreconjunto do segundo. No entanto ambos são referidos aquando da descrição do OWL.

Por razões óbvias, e embora o UML não tenha sido originalmente pensado para representar ontologias, os Diagramas de Classes do UML serão também incluídos neste capítulo. Como complemento ao UML é apresentado o *Ontology Definition Metamodel* (ODM). Por fim é descrito o modelo O3F e a correspondente linguagem, o CO3L (*Compact Object Oriented Ontology Language*). É mostrado que o O3F e o CO3L representam um poderoso meio para descrição de ontologias oferecendo inúmeras vantagens em relação às abordagens anteriormente referidas. O modelo O3F

e a linguagem de especificação CO3L são usadas pelo grupo de investigação onde a presente dissertação tem sido conduzida e são também ensinadas na unidade curricular Tecnologias para Sistemas Inteligentes, no terceiro ano dos primeiros ciclos de Engenharia Informática, Engenharia de Telecomunicações e Informática, e Informática e Gestão de Empresas. A sua utilização em investigação e no ensino tem possibilitado o contínuo desenvolvimento e aperfeiçoamento da abordagem.

Na secção 2.8, serão analisadas as conversões já existentes de UML para linguagens de representação de ontologias, concluindo que ainda nenhuma consegue fazer o que a presente dissertação pretende - uma conversão completa de todos os elementos de um diagrama de classes (e também de objectos).

Na secção 2.9, são estudadas as principais tecnologias para realizar as conversões anteriormente descritas. Nesta secção conclui-se que a tecnologia XSLT *(eXtensible Stylesheet Language Transformation*), apesar de ser a mais usada e de apresentar inúmeras vantagens, carece no entanto de algumas funcionalidades como a possibilidade de armazenar a ontologia numa base de dados, e o processamento de diagramas de classes com vários pacotes (*packets*), não satisfazendo todos os requisitos necessários para o desenvolvimento do sistema que se pretende construir.

Por fim, na secção 2.10, é apresentada uma análise crítica das abordagens apresentadas ao longo de todo o capítulo, identificando a área de contribuição da dissertação.

2.1 Representação de Ontologias

Uma ontologia é uma especificação formal de uma conceptualização de um domínio (Gruber, 1993). Uma ontologia providencia um vocabulário que permite a representação e a comunicação do conhecimento relativo ao domínio da conceptualização (Gruber, 1993).

As ontologias tornam-se assim cruciais no desenvolvimento e interoperabilidade de sistemas abertos heterogéneos, como é o caso dos sistemas ou das sociedades de agentes inteligentes artificiais, pois definem um vocabulário partilhado que suporta a comunicação sem ambiguidades.

Para que o desenvolvimento de ontologias não seja feito de forma *ad hoc*, é necessário que exista também um modelo subjacente à sua representação.

No entanto, não existe um consenso relativo nem a um modelo nem a uma linguagem que melhor se adeqúe ao desenvolvimento de ontologias. As abordagens existentes mais importantes são descritas e analisadas no decorrer desta secção.

É necessário compreender que existem ontologias com diferentes propósitos, o que faz com que uma dada abordagem seja mais apropriada para uma dada ontologia e menos para outra.

As ontologias podem ser classificadas em várias categorias (e.g. ontologias genéricas, ontologias de domínio, ontologias de aplicação). As ontologias de domínio são o tipo de ontologias mais desenvolvidas, e podem ser encontradas em áreas como química (Gómez-Pérez, Fernández, & Juristo, 1996), modelação de empreendimentos - (TOVE, *Toronto Virtual Enterprise*), *design* - (YMIR: A *sharable ontology for the formal representation of engineering design knowledge*) (Alberts, 1994), (DORPA: Uma ontologia de Design que integra requisitos, artefactos e processos) (Varejão, 1999), medicina - (UMLS, *Unified Medical Language System*) (Humphreys & Lindberg, 1993), modelação de processos de software - (Falbo, 1998), biologia molecular e bioquímica - (GENSIM, *Genetic Simulator System*) (Karp, 1993) e ciência dos materiais - (*The Plinius Ontology of Ceramic Materials*) (Vet *et al*., 1994), entre outros.

As secções que se seguem descrevem as principais abordagens à representação de ontologias.

2.2 Web Ontology Language (OWL)

O OWL (*Web Ontology Language*) (W3C, 2004a) é a recomendação do W3C (*World Wide Web Consortium*) para uma linguagem de representação de ontologias na Web e nasce como uma revisão da linguagem DAML+OIL (W3C, 2001).

É construído sobre o RDF (*Resource Description Framework)* (W3C, 2004b) e RDF Schema (W3C, 2004c) tendo uma maior expressividade que este, tornando assim possível a representação de mais conhecimento (e.g. disjunção de classes, restrições de cardinalidade).

O aumento de expressividade que o OWL oferece requer alguma atenção, pois "quanto mais rica se torna uma linguagem, mais ineficiente se torna o seu apoio ao raciocínio" (Antoniou & Van Harmelen, 2003). De facto, algumas primitivas do OWL não são computáveis.

Tendo em conta este facto, o W3C decidiu definir o OWL em três variantes (muitas vezes também chamadas de "espécies") de modo a satisfazer os utilizadores consoante a sua necessidade de expressividade e apoio ao raciocínio. As três variantes são as seguintes:

- OWL Full. O OWL Full é a mais completa das três variantes uma vez que compreende todo o OWL, em que o uso de todas as suas primitivas é válido e sem restrições. A grande vantagem do OWL Full é que é completamente compatível com o RDF Schema podendo ser visto como um sobreconjunto deste. O facto de o OWL Full não impor restrições num ficheiro RDF ou RDF Schema faz com que estes, mesmo sem terem qualquer definição OWL, possam ser considerados também como ficheiros OWL Full. No entanto, o preço a pagar pelo uso da total expressividade que o OWL oferece é que cada utilizador tem à sua responsabilidade a resolução de qualquer problema proveniente da execução do raciocínio.
- OWL DL. O grande objectivo do OWL DL é de oferecer uma Linguagem de Descrição (*Description Logic* - DL) que recupere o apoio ao raciocínio e eficiência computacional perdidos no OWL Full. O OWL DL usa todas as construções do OWL Full mas coloca restrições nalgumas delas. Estas restrições são limitações ao uso do RDF Schema. Mais exactamente o OWL DL restringe a mistura das primitivas OWL com as RDF e requer uma disjunção de tipos, pelo que uma coisa apenas pode ser classificada como uma classe, propriedade ou indivíduo. São restrições como estas que garantem o apoio ao raciocínio, isto é, que as primitivas escritas em OWL DL sejam computáveis em tempo finito (decidibilidade da linguagem). As desvantagens destas restrições são que, para além de imporem restrições à linguagem, eliminam também a total compatibilidade com o RDF e RDF Schema.
- OWL Lite. No OWL Lite para além de serem aplicadas as restrições presentes no OWL DL que garantem a eficiência computacional e o apoio ao raciocínio, apenas

pode ser usado um subconjunto das construções do OWL Full. Isto faz com que o OWL Lite seja a versão do OWL mais simples. O seu grande objectivo é o de proporcionar uma linguagem mais simples para quem se inicia no mundo das ontologias, e de permitir que sejam criadas ferramentas que suportem o OWL sem que tenham de ser implementados os aspectos mais complicados da linguagem.

As ontologias OWL são muitas vezes representadas recorrendo a uma sintaxe baseada em XML, mas existem outras, como por exemplo a sintaxe abstracta que é usada no próprio documento de especificação da linguagem ou a GrOWL (Krivov, Williams, & Villa, 2007) uma linguagem gráfica para representação de ontologias OWL.

Os principais elementos OWL que podemos encontrar na definição de uma ontologia são:

- Classes: No OWL, as classes são definidas como conjuntos de indivíduos. São construídas através de descrições que especificam as condições que devem ser satisfeitas por um indivíduo para que este possa pertencer a uma determinada classe. O OWL permite também a especificação de hierarquias de classes.
- Propriedades: Em OWL as propriedades são relações binárias que ligam dois indivíduos entre si (*object properties*) ou que associam um indivíduo a um valor (*datatype properties*). As propriedades podem ser usadas para fazer declarações sobre todos os indivíduos de uma classe ou apenas sobre alguns em particular. Tal como acontece na definição de classes, o OWL permite a especificação de subpropriedades, relacionando várias propriedades hierarquicamente.
- Indivíduos: Os indivíduos representam objectos do domínio e são também muitas vezes conhecidos como instâncias. Há no entanto indivíduos que podem não ser instâncias de nenhum tipo de dados.

O OWL permite também, através de várias primitivas, fornecer informação sobre classes, propriedades e indivíduos. Alguns exemplos disto são a hierarquização de classes e propriedades referidas acima ou a indicação de que duas classes são iguais ou que uma propriedade é o inverso de outra. Esta capacidade pode ser vista como uma forma limitada de axiomas.

2.3 OWL 2

O W3C realizou recentemente uma revisão e extensão da linguagem OWL a qual originou a segunda versão do OWL, o OWL 2 (W3C, 2009). As principais funcionalidades que são introduzidas por esta revisão são as seguintes:

- Enriquecimento da descrição de propriedades (e.g., especificar a disjunção de propriedades).
- Enriquecimento do conceito de *datatype*, sendo agora possível criar novos tipos de dados de maneira explicita, bem como aplicar-lhes um maior leque de restrições.
- Capacidades de meta modelação no OWL DL (e.g., possibilidade de usar a mesma designação para uma classe e para uma instância).

Para além das três variantes da linguagem já definidas (OWL Lite, OWL DL e OWL Full), o OWL 2 define outras três variantes que impõem restrições específicas na linguagem, de maneira a satisfazerem determinados propósitos. Estas três novas variantes são:

- OWL 2 EL. Esta variante, com expressividade bastante reduzida, é adequada para ontologias de grande escala, permitindo o uso de algoritmos de raciocínio com complexidade polinomial. O acrónimo EL provém da inspiração desta variante na família de lógicas descritivas EL.
- OWL 2 QL. O OWL 2 QL (*Query Language*) impõe restrições que permitem a integração de RDBMS (*Relational Database Management System*), possibilitando que um motor de SQL possa ser usado como motor de raciocínio de uma ontologia. Deste modo a ontologia pode ser usada para realizar interrogações a uma base de dados através de um mecanismo de reescrita de *querys* (*query rewriting*).
- OWL 2 RL. Esta variante consiste num subconjunto sintáctico do OWL 2 que seja susceptível de ser implementado usando tecnologias baseadas em regras, o que justifica o acrónimo RL (*Rule Language*). O raciocínio em OWL 2 RL é escalável sem comprometer em demasiado o poder de expressividade da linguagem.

2.4 Ontolingua

O termo Ontolingua (Gruber, 1993) é usado para referir o sistema e a linguagem com esse nome. Enquanto sistema o Ontolingua é definido como num conjunto de ferramentas para criar e analisar ontologias, um processador sintáctico e lexicográfico (*parser*) de KIF (*Knowledge Interchange Format*) (Genesereth & Fikes, 1992) e um conjunto de conversores que possibilitem a transformação de ontologias no formato Ontolingua para outros formatos de representação de conhecimento.

Enquanto linguagem, o Ontolingua possui uma sintaxe e semântica baseadas na notação de uma versão estendida do cálculo de predicados de primeira ordem, o KIF. O KIF foi concebido para ser uma linguagem de publicação e comunicação do conhecimento, não sendo uma linguagem de representação pois não oferece mecanismos de inferência ou interrogação.

O Ontolingua foi criado para descrever ontologias num formato compatível com múltiplas linguagens de representação do conhecimento. Ontologias escritas em Ontolingua podem ser partilhadas por vários utilizadores e grupos de investigação utilizando os seus sistemas de representação preferidos, podendo ser traduzidas por exemplo para Loom (MacGregor, 1991), Epikit (Singh & Genesereth, 1991), e uma forma canónica de KIF. A partilha de ontologias ganha assim uma nova alma com o Ontolingua. Uma mesma ontologia poderá estar traduzida em diferentes linguagens respeitando a mesma conceptualização. Estas ontologias poderão ser publicadas na Web e acedidas via protocolo HTTP (*Hypertext Transfer Protocol*) ou em alternativa acedidas pelo protocolo OKBC (*Open Knowledge Base Connectivity*), sendo que o protocolo OKBC tem actualmente várias implementações em diferentes linguagens de programação.

O conjunto de expressões que o Ontolingua pode reconhecer e traduzir é definido com base numa Ontologia de Enquadramentos (*Frame Ontology*) usando relações de segunda ordem. A Ontologia de Enquadramentos especifica, de uma forma declarativa, as primitivas de representação que são muitas vezes apoiadas em representações centradas em objectos. Assim o Ontolingua tem a capacidade de definir classes, relações, funções, instâncias e axiomas que são utilizados como restrições.

As definições do Ontolingua são escritas em língua natural e em instruções KIF. As definições em língua natural são utilizadas para os comentários. Quando se proceder à tradução para um sistema de representação específico, os comentários serão colocados nos locais indicados, caso o sistema de representação alvo tenha essa possibilidade. As instruções KIF são utilizadas para restringir o significado da definição dos termos.

A sintaxe do KIF é baseada na da linguagem de programação Lisp (McCarthy, 1962). Os objectos são denotados por constantes (Lisp *atoms*) e são construídos com base em listas cujo primeiro elemento é um operador relacional. As variáveis podem ser quantificadas universal ou existencialmente.

Em Ontolingua, as classes, as relações e as funções são representadas de forma muito semelhante. As relações são definidas por um conjunto de túpulos, onde cada túpulo é uma sequência de objectos. As classes não são mais do que relações unárias (Gruber, 1993). Podem ser relacionadas através de herança, complemento, união e intersecção. As funções são representadas como um caso especial das relações onde o último objecto do túpulo é o resultado da invocação da função. Então, uma função de N argumentos representa-se como uma relação de N+1 argumentos.

2.5 Unified Modeling Language (UML)

O UML (*Unified Model Language*) (OMG 2009b) é uma linguagem de modelação gráfica orientada a objectos, criada na década de 90. O UML tem a sua génese na junção de várias técnicas de engenharia que provaram ter bastante sucesso na modelação de sistemas. Sintetiza as notações do método de Booch (Booch, 1991), a Técnica de Modelação de Objectos (*Object-Modeling Technique - OMT*) (Rumbaugh *et al*., 1991) e a Engenharia de Software Orientado por Objectos (*Object-Oriented Software Engineering - OOSE*) (Jacobson, 1992) A sua especificação normalizada (*standard)* foi criada e é mantida pelo *Object Management Group* (OMG) e encontra-se hoje em dia na sua versão 2.2.

Sendo uma linguagem gráfica, o UML permite a construção de modelos, utilizando vários tipos de diagramas para o efeito. Na versão actual existem 14 tipos de diagramas diferentes. Alguns exemplos são os Diagramas de Classes, os Diagramas de Componentes e os Diagramas de Sequência.

Os Diagramas de Classes são frequentemente escolhidos para a representação de ontologias. Estes diagramas destinam-se a representações estáticas, sendo bastante apropriados para a modelação de domínios. Nos Diagramas de Classes, os elementos com mais relevância são os seguintes:

- Classes. Estes são os elementos centrais do Diagrama de Classes. As classes representam conjuntos de objectos que partilham dos mesmos atributos e métodos. Os atributos são caracterizados pelo seu tipo, cardinalidade e visibilidade. Os métodos são caracterizados pela sua assinatura.
- Associações. São um dos tipos de relações que podem existir entre objectos de uma ou mais classes. Podem definir-se várias propriedades das associações, nomeadamente a multiplicidade dos argumentos relacionados, o papel desempenhado por cada um dos argumentos, a direccionalidade da associação, onde é possível indicar quais os argumentos que estão cientes da sua relação com os outros. É ainda possível indicar, se a associação se trata de uma agregação[1] ou composição.
- Generalizações. Exprimem relações hierárquicas entre classes. É indicada qual a classe mais genérica e qual ou quais as suas subclasses.

Recorrendo ao Diagrama de Objectos é possível representar instâncias das classes definidas num diagrama de classes.

Além da sua sintaxe gráfica, o UML conta ainda com dois componentes textuais:

O primeiro trata-se de uma linguagem declarativa, a *Object Constraint Language* (OCL) (OMG 2006b), cujo objectivo é a descrição de regras e restrições que podem ser aplicadas a vários elementos dos diagramas. Inicialmente foi definida apenas como uma extensão do UML, mas actualmente pode ser usada com qualquer modelo ou meta modelo que esteja de acordo com a MOF (*Meta-Object Facility*) (OMG, 2006a).

O segundo é um formato padrão em XML, denominado XMI (*XML Metadata Interchange*) (OMG, 2007), que se destina à troca de modelos entre ferramentas de modelação. O XMI não é mais do que a representação textual, em XML, de um

[1] É também comum encontrar a designação agregação partilhada(*shared*) para a agregação e agregação composta para a composição(*composite*)

modelo. À semelhança do que acontece com o OCL, também o XMI pode ser aplicado a qualquer modelo MOF.

São cada vez mais os autores que identificam características e vantagens no UML que justificam que este seja usado como linguagem de representação de ontologias. Algumas destas são a capacidade de representar conceitos que são também comuns nas linguagens de representação de ontologias (e.g. classes e relações) (Cranefield, 2001), ser amplamente disseminada, com um grande leque de ferramentas CASE (*Computer-aided software engineering*) que o suportam (Kogut *et al*., 2002), e possuir uma sintaxe gráfica mais propícia à compreensão humana (Kabilan & Johannesson, 2004)

O próprio OMG acabou por criar uma arquitectura que, entre outras funcionalidades, permite aumentar a capacidade do UML de representar conhecimento ontológico, o *Ontology Definition Metamodel* (ODM).

2.6 Ontology Definition Metamodel (ODM)

Devido à crescente importância das ontologias e ao papel que representam na Web Semântica, o OMG decidiu compatibilizar os conceitos presentes na *Model-Driven Architecture* (MDA) com a especificação de ontologias. Para tal, o OMG lançou um pedido de propostas para a definição de um meta modelo capaz de satisfazer as necessidades acima descritas. Como resultado, e depois de várias revisões das propostas recebidas, o OMG criou o *Ontology Definition Metamodel* (ODM) (OMG 2009b). O ODM é constituído por outros meta modelos (RDF Schema, OWL, Topic Maps, Common Logic e Description Logic). Para o presente trabalho, o meta modelo para OWL tem especial relevância visto ser a linguagem de representação de ontologias mais usada e para a qual são tentadas varias conversões a partir do UML.

Segundo o OMG (OMG, 2009), a expressividade do UML não é tão completa (no âmbito das ontologias) como a do OWL. Faltam por exemplo construções que permitam modelar a intersecção ou o complemento de conjuntos (e.g., classes). Além disto, o UML não está especialmente pensado para o propósito da inferência automática.

Todavia, o UML é amplamente disseminado e possui um vasto leque de aplicações que suporta o desenvolvimento dos seus modelos. Para aproveitar estes factos, o ODM faculta a definição de três perfis de UML (RDF Schema, OWL e Topic Maps) para que este possa ser usado no desenvolvimento de ontologias de acordo com os respectivos meta modelos presentes no ODM.

No Perfil de UML para OWL, por exemplo, são usadas as construções UML quando têm um significado equivalente em OWL e, quando tal não é possível, a semântica do UML é estendida através do uso de estereótipos que a aproximam do OWL.

Adicionalmente, o ODM define ainda a correspondência entre alguns dos meta modelos que o constituem e modelos UML mais antigos que não tenham sido elaborados com os perfis referidos anteriormente. Desta forma é também possível representar informação presente nestes modelos usando uma linguagem de ontologias como o OWL.

2.7 O3F/CO3L

O O3F (*Object Oriented Ontology Framework*) inicialmente definido em (Mota *et al*., 2003) é um modelo de representação de ontologias em que os conceitos do domínio são descritos através de objectos, classes, atributos, e métodos, mas também relações, funções e acções, entre outros tipos de entidade. A descrição textual de ontologias O3F faz-se na linguagem CO3L (*Compact Object Oriented Ontology Language*) inicialmente proposta em (Botelho & Ramos, 2003).

Ambos são projectos académicos desenvolvidos por investigadores do Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL (Instituto Universitário de Lisboa) e seus colaboradores, e têm vindo a ser melhorados ao longo dos tempos. Têm ainda uma disseminação muito reduzida, embora sejam usados nas aulas da unidade curricular Tecnologias para Sistemas Inteligentes do 3º ano dos primeiros ciclos de Engenharia Informática, Engenharia de Telecomunicações e Informática, e Informática e Gestão de Empresas.

As características da linguagem CO3L advêm do modelo subjacente consequentemente, falar de características da linguagem é falar implicitamente das características do modelo O3F e vice-versa.

A principal característica do O3F é a possibilidade que ele oferece de conceptualizar um dado domínio, quer em termos de objectos e todos os conceitos que lhe estão tipicamente associados (e.g., classes, instâncias, atributos, métodos), quer em termos de relações, funções e acções, e ainda misturando conceitos das duas naturezas. Pessoas com uma orientação mais virada para os objectos (como acontece com frequência nas comunidades da engenharia de software e dos sistemas de informação) podem recorrer ao O3F para as suas conceptualizações. Pessoas com pendor mais orientado para a visão relacional ou funcional do mundo (como frequentemente acontece com as pessoas da inteligência artificial simbólica) encontram igualmente no O3F os conceitos com os quais se sentem mais familiarizados, os quais podem ser usados para descrever os domínios desejados. Finalmente, quando a conceptualização de um domínio é mais fácil ou mais natural quando se misturam conceitos da orientação por objectos com outros conceitos como proposições, relações e funções, o O3F pode também ser usado para essas conceptualizações mistas. Apenas o Ontolingua possibilita algo semelhante a isto mas não de forma tão completa e directa como o O3F permite. No Ontolingua apenas se podem definir classes que apenas podem ser caracterizadas por atributos, mas não por métodos.

Além de possibilitar a conceptualização orientada para objectos a par ou em simultâneo com a conceptualização em termos de relações, funções e acções, o O3F procura captar a relação existente entre estas duas maneiras de pensar os domínios que se descrevem. Os casos mais emblemáticos desse relacionamento dos dois paradigmas são:

- A forte relação entre atributos e métodos, por um lado, e funções, relações e acções, por outro;
- O conceito de objectos com métodos relacionais;
- A conceptualização de hierarquias (um conceito fortemente enraizado nos modelos de objectos) de relações, de funções e de acções.

Em O3F salienta-se que a invocação do o atributo A do objecto x (x.A) corresponde apenas a uma notação diferente da aplicação da função A a x (A(x)); que a invocação do método funcional F com argumentos $<a_1, a_2, ..., a_n>$ do objecto x (de uma dada classe) ($x.F(a_1, a_2, ..., a_n)$ corresponde à aplicação da função F aos argumentos $<x, a_1,$

a_2, ..., a_n> (F(x, a_1, a_2, ..., a_n)); e de forma semelhante para os métodos relacionais e predicados e para os métodos de acção e acções.

O O3F permite a definição de objectos com métodos relacionais. Por exemplo, na classe *Pessoa*, pode definir-se o método relacional *filho* que relaciona as pessoa com os seus filhos. Se x e y forem membros da classe *Pessoa*, x.filho(y) significa que y é filho de x.

Em O3F podem estabelecer-se hierarquias de quaisquer conjuntos. Como os tipos de dados são conjuntos, podem definir-se hierarquias de tipos de dados, por exemplo hierarquias de números e hierarquias de classes. Além disso, como as funções e os predicados denotam conjuntos de túpulos, também se pode estabelecer hierarquias de funções e hierarquias de predicados. Por exemplo, o predicado P é um subpredicado do predicado Q se o conjunto de túpulos denotado por P estiver contido no conjunto de túpulos denotado por Q. Indirectamente, também se podem estabelecer hierarquias de acções porque as acções têm assinaturas funcionais (funções), ou assinaturas relacionais (predicados) as quais se podem organizar em hierarquias. Consequentemente, como atributos e métodos correspondem a funções, predicados e acções, os atributos e os métodos podem também arrumar-se de acordo com hierarquias.

Além destas características, o O3F pode caracterizar as entidades de uma ontologia através de facetas. Exemplos de facetas incluem, por exemplo *minimum-value*, *default-value*, *maximum-size*, *values-set*, *mandatoty*, e *distinct*, entre muitas outras. A grande variedade de facetas do O3F confere-lhe uma grande expressividade. A utilização de facetas permite expressar conceitos que, noutras linguagens só poderiam ser expressos através de axiomas. Deste modo, a expressividade ganha com a utilização de facetas não está associada à complexidade computacional das linguagens em que a mesma expressividade se conseguiria através de axiomas, geralmente usando uma linguagem inspirada na lógica simbólica.

Apesar da facilidade e das vantagens computacionais da utilização de facetas, está prevista a extensão da linguagem de representação de ontologias (O3F) para que venha a permitir a definição de axiomas capazes de captar conceitos que não se possam expressar à custa unicamente de facetas.

Como o modelo O3F dispõe do conceito de tipo básico de dados (*datatype*), por oposição aos tipos compostos (por exemplo, classes), a linguagem de especificação de ontologias (CO3L) permite a definição de novos tipos de dados simples conceptualmente mais próximos dos domínios que se pretende representar, sem obrigar à definição de classes (as quais teriam de ter pelo menos um atributo).

Finalmente, o modelo O3F permite captar relações arbitrárias de dependência entre os vários elementos da ontologia.

Analogamente ao que se passa com o UML e com a linguagem OWL, o O3F e a sua linguagem possibilitam a representação de indivíduos, os quais podem ser instâncias de tipos de dados da ontologia. Podem representar-se instâncias concretas, quer de classes e associações, quer de outros tipos de dados.

2.8 Conversões entre UML e Linguagens de Representação de Ontologias

Muitos autores estudaram já a possibilidade de converter diagramas de classes UML em representações usando linguagens de representação de ontologias. Alguns exemplos são Cranefield que propôs uma conversão para RDF Shema (Cranefield, 2001); Baclawski (Baclawski *et al*., 2002) e Falkovych (Falkovych 2002) propondo ambos conversões para DAML+OIL; e Leinhos (Leinhos, 2006), e Gašević e a sua equipa (Gašević *et al.*, 2004) que sugerem uma conversão para OWL.

Tendo em conta as várias propostas de conversão já realizadas pode constatar-se que estas em geral podem agrupar-se em duas classes relativamente ao objecto da conversão. Umas convertem UML normalizado (*standard*), e outras convertem UML estendido.

A conversão de UML normalizado pretende representar os elementos presentes nos seus modelos usando linguagens de representação de ontologias.

A primeira proposta de conversão visando o UML normalizado foi feita por Cranefield que, ao identificar conceitos em comum entre o UML e as linguagens de representação de ontologias (e.g. classes e relações), advoga que este pode ser também usado para a representação de ontologias. Cranefield propõe uma conversão de UML para RDF Schema (e também para classes Java) (Cranefield, 2001). A sua

conversão tem algumas limitações devido às diferenças entre as duas linguagens. A mais notória destas diferenças é o facto de, em UML, elementos como propriedades (atributos) terem de ser definidos no âmbito de classes enquanto que numa ontologia, estes podem existir de forma independente delas.

Leinhos (Leinhos, 2006) e Falkovych (Falkovych 2002) propuseram a conversão de diagramas de classes UML em ontologias OWL e DAML+OIL respectivamente. O DAML+OIL é a linguagem que precedeu e originou o OWL.

Por último há ainda que referir a transformação de UML em OWL proposta pelo OMG, no documento de especificação do seu *Ontology Definition Metamodel* (ODM). Guillaume Hillairet da Universidade de La Rochelle em França implementou uma conversão com base nesta proposta. Informação detalhada sobre esta conversão pode ser encontrada no site do autor (http://perso.univ-lr.fr/ghillair/projects.html).

Apenas Hillairet e Cranefield consideram os Diagramas de Objectos para a representação de instâncias em ontologias.

O facto de a conversão ser feita a partir de UML normalizado faz com que seja preservada a expressividade da linguagem, porém nenhuma destas conversões consegue representar todos os elementos do UML, na linguagem alvo de representação de ontologias.

Outros autores partilham a motivação de Cranefield de usar o UML como linguagem de modelação de ontologias, no entanto defendem que faltam muitos conceitos ao UML para que este possa desempenhar tal tarefa, como por exemplo definir atributos (propriedades) fora do âmbito de uma classe. Para resolver este problema são propostos perfis de UML que definem extensões à sua expressividade por meio de estereótipos. Estes perfis permitem que sejam desenhados diagramas de classes que consigam representar os mesmos conceitos presentes numa linguagem de ontologias facilitando também a sua posterior conversão para essa linguagem.

Com base nesta ideia, Baclawski e a sua equipa propuseram uma conversão de diagramas UML em ontologias DAML+OIL. Para tal definiram um perfil que aproxima o UML da DAML+OIL (Baclawski *et al*., 2002).

Outros trabalhos similares são os realizados por Gašević e a sua equipa (Gašević *et al*., 2004) e Leinhos (Leinhos, 2006). Ambos definem um perfil de UML que pretende aproximar o UML do OWL propondo a respectiva conversão. Há que distinguir no entanto a solução de Gašević e os seus colaboradores, pois o perfil por eles proposto baseia-se num meta modelo de definição de ontologias, também por eles desenhado, que está de acordo com a arquitectura MDA (*Model Driven Architecture*) do OMG e pode ser consultado com pormenor em (Djurić *et al*., 2005). De facto este meta modelo e o respectivo perfil de UML serviram como proposta para o *Ontology Definition Metamodel* (ODM) definido pelo OMG.

Enquanto que na conversão a partir de UML normalizado (*standard*), alguns autores consideraram o Diagrama de Objectos para a representação de instâncias, na conversão do UML estendida, opta-se por representar esta informação aplicando um estereótipo à classe *Class*.

Este segundo grupo de propostas de conversão tem a vantagem de aproximar o UML das linguagens de representação de ontologias permitindo a representação de conhecimento ontológico de uma maneira gráfica. Contudo tal é feito á custa de extensões que fazem com que a semântica original dos elementos do UML seja passada para segundo plano.

2.9 Tecnologias Usadas nas Conversões

Em seguida são descritas as principais tecnologias usadas para a implementação das conversões enunciado as suas principais vantagens e desvantagens. Serão descritas as tecnologias XSLT *(eXtensible Stylesheet Language Transformation*), a QVT (*Query/View/Transformation*) e ainda conversões com recurso ao software Protégé.

A maior parte dos autores que propõem uma conversão de UML para representações usando linguagens de descrição de ontologias implementa a sua solução com recurso à tecnologia XSLT. Exemplos de autores que recorrem a esta tecnologia são Cranefield na conversão para RDF Schema (Cranefield, 2001), Falkovych na conversão para DAML+OIL (Baclawski *et al*., 2002), e Gašević e a sua equipa (Gašević *et al*., 2004) e Leinhos (Leinhos, 2006) ambos na conversão para OWL.

No caso do XSLT os diagramas UML não são directamente convertidos em ontologias. Antes disso, os diagramas UML têm de ser representados em XMI, uma representação textual dos diagramas UML, usando a sintaxe XML. Esse passo é efectuado, não pelo sistema de conversão, mas sim pelas ferramentas de especificação de diagramas UML.

Esta tecnologia permite que sejam desenvolvidos guiões (*scripts*) que contêm regras e expressões regulares para identificar e referir os vários elementos presentes num ficheiro XML. Em seguida, usando algumas construções, esses elementos podem ser convertidos num qualquer outro formato de texto.

Para realizar a conversão é usado um processador próprio para o efeito (e.g. *Xalan-Java 2 XSLT processor*) que aplica um *script* XSLT (contendo as regras de conversão) ao ficheiro XMI. O resultado é um ficheiro de texto com a representação da informação presente no diagrama de classes usando uma linguagem de especificação de ontologias como o OWL.

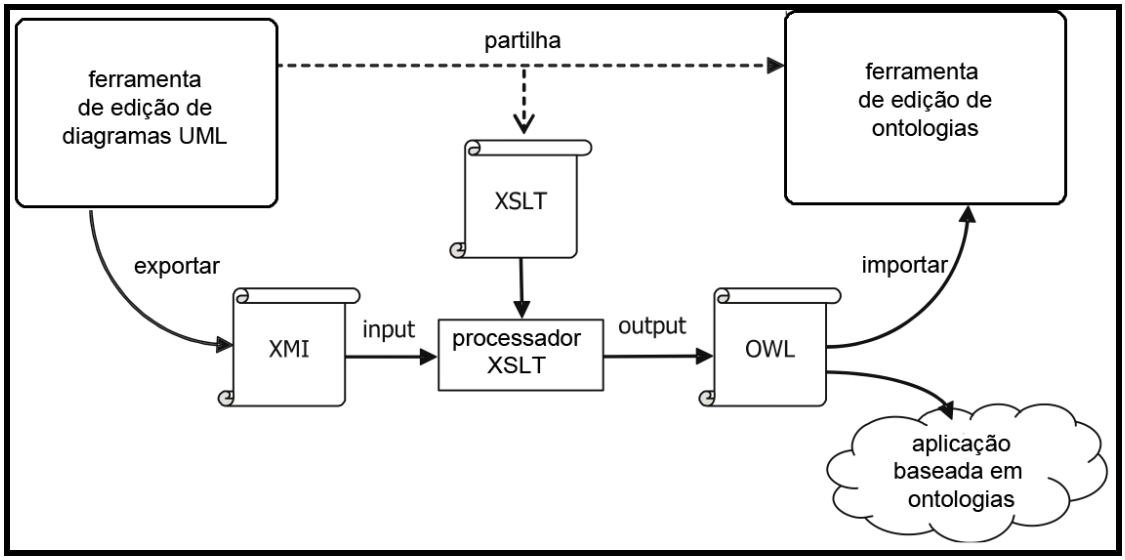


Figura 2 - Conversão usando XSLT (Gašević *et al*., 2004)

A vantagem de usar XSLT é que "não é necessário alterar a ferramenta UML. Apenas temos de aplicar o XSLT ao ficheiro XMI gerado pela ferramenta UML" (Gašević *et al*., 2004). No entanto, os ficheiros XMI gerados por diferentes ferramentas possuem algumas diferenças na sua estrutura o que dificulta o aproveitamento desta vantagem. Outro ponto a favor da XSLT é que foi especialmente pensada para trabalhar com XML, um formato amplamente aceite para representação de informação.

O XSLT é uma linguagem bastante sensível quanto ao ficheiro de *input* e um pouco incómoda de programar para conversões mais complexas (Falkovych, Sabou, & Stuckenschmidt, 2003). Além do mais, apresenta a limitação de não poderem ser convertidos modelos UML com vários pacotes.

O OMG, na especificação do ODM, propõe o uso da sua arquitectura para conversão de modelos, o QVT (*Query/View/Transform*) (OMG, 2008). O QVT define três linguagens para transformação de modelos - Relations, Core e Operational Mappings - organizadas numa arquitectura. As duas primeiras, Relations e Core, são duas linguagens declarativas com diferentes níveis de abstracção. QVT Relations é mais complexa e de mais alto nível. É baseada numa mistura de linguagem natural (em inglês) e lógica de predicados de primeira ordem. O QVT Core apresenta uma sintaxe mais simples de mais baixo nível e é baseada em OCL. É ainda possível converter transformações QVT Relations em QVT Core. O QVT Operational Mappings é uma linguagem imperativa que estende as duas primeiras. Ao contrário do que é feito com XSLT, as implementações QVT não se baseiam em ficheiros de texto. As transformações são feitas apenas entre modelos. Contudo, transformações que necessitem de recorrer a ficheiros podem ser feitas recorrendo a um módulo do QVT chamado *Black Box* que oferece compatibilidade com outras tecnologias.

Guillaume Hillairet realizou uma implementação QVT de acordo com o especificado no ODM. Para tal recorreu à linguagem ATL (*ATLAS Transformation Language*), um componente da arquitectura M2M (*Model to Model*), a implementação (parcial) de QVT da Eclipse (www.eclipse.org).

A conversão funciona em dois passos. Num primeiro passo, um diagrama de classes UML, desenvolvido com o Eclipse, é convertido para uma ontologia OWL. Esta conversão é feita entre representações computacionais de acordo com o meta modelo do UML e o meta modelo do OWL especificado no ODM. Em seguida, num segundo passo, a representação computacional é convertida num formato de texto. Esta representação textual usa a sintaxe baseada em XML tal como definido pelo W3C.

O Protégé (http://protege.stanford.edu/) é uma ferramenta para o desenho de ontologias que permite conversões entre vários formatos. Possui uma arquitectura que possibilita o fácil desenvolvimento de extensões, o que lhe confere suporte para várias linguagens. No que diz respeito ao UML, o Protégé possui um *plugin* para a

conversão de Diagramas de Classes UML para o formato Protégé que é compatível com o OKBC (*Open Knowledge Base Connectivity*) e, por conseguinte, a conversão também para os outros formatos suportados por este *software*. Esta conversão funciona em ambos os sentidos, pois o Protégé também tem a capacidade de exportar/converter ontologias no seu formato (ou em outros que esteja preparado para suportar) para UML. Knublauch, o autor original deste *plugin* para UML (e também do Protégé OWL) efectuou algumas experiências onde testa as capacidades de conversão entre UML, OWL e o formato Protégé (Knublauch, 2003).

A importação para o formato Protégé contém no entanto uma grande limitação, pois não consegue representar a informação referente a associações entre classes.

Durante a exportação para UML também existem perdas de informação referente aos elementos mais específicos do Protégé, como por exemplo as restrições PAL (*Protégé Axiom Language*).

De todas as tecnologias, o XSLT é a que oferece mais vantagens para o tipo de conversão que se pretende neste trabalho, já que é especialmente pensada para converter ficheiros no formato XML, como é o caso dos ficheiros XMI que descrevem um modelo UML. Porém esta tecnologia não oferece suporte para armazenamento das conversões numa base de dados e não permite converter diagramas UML com vários pacotes, o que faz com que não satisfaça todos os requisitos do sistema que se pretende desenvolver.

2.10 Resumo das Abordagens Estudadas

Existem várias linguagens para a descrição de ontologias onde estão sempre presentes conceitos como classes, relações e herança. Actualmente a mais disseminada e com maior relevância é o OWL, que foi recentemente melhorado e estendido através do OWL 2. O OWL foi especialmente pensado para a Web Semântica e possui várias sublinguagens que permitem escolher o equilíbrio mais adequado entre a expressividade e o apoio ao raciocínio, conforme seja necessário.

O Ontolingua é também uma alternativa interessante pois, por ser baseado em KIF e compatível com o protocolo OKBC, permite o desenvolvimento de ontologias num formato compatível com múltiplas linguagens de representação facilitando a partilha.

O UML começa também a ser considerado para a representação de conhecimento ontológico de uma maneira gráfica. Para reforçar esta ideia, o OMG desenvolveu também o ODM (*Ontology Definition Metamodel)* para tornar compatíveis os conceitos da MDA (*Model Driven Architecture*) com a especificação de ontologias.

Características	OWL	Ontolingua	UML[2]	O3F/CO3L
Classes	sim	sim	sim	sim
Atributos	sim	sim	sim	sim
Restrições de propriedades	sim	sim	sim	sim
Métodos de classes	não	não	sim	sim
Funções	só unárias	sim	não	sim
Predicados	só binários	sim	não	sim
Acções	não	não	não	sim
Hierarquias de funções e métodos funcionais	só funções	só funções	não	sim
Hierarquias de acções e métodos de acção	não	não	não	sim
Hierarquias de predicados e métodos relacionais	só predicados	só predicados	não	sim
Relações entre classificadores	binária	n-árias	n-árias	n-árias
Dependências	não	não	sim	sim
Axiomas formais	não	sim	sim	não
Instâncias	sim	sim	sim	sim

Tabela 1 - Resumo das linguagens abordadas

Vários autores propõem conversões entre o UML e linguagens de representação de ontologias. Existem abordagens que convertem a partir da norma UML (*standard*) e outras que estendem a expressividade do UML de modo a aproximá-lo das linguagens de representação de ontologias e facilitar a sua posterior conversão.

[2] Apenas considerando Diagramas de Classes e Diagramas de Objectos

Autor	Versão do UML	Método de conversão	Linguagem Objecto
Cranfield	UML *standard*	XSLT	RDF Schema e classes Java
Baclawski	UML com extensões propostas	-	DAML+OIL
Falkovych	UML *standard*	XSLT	DAML+OIL
Gašević *et al.*	UML com extensões (versões próprias do ODM e perfil de UML)	XSLT	OWL
Leinhos	UML *standard*	XSLT	OWL
	UML com algumas extensões propostas	XSLT	OWL
Hillairet	UML *standard*	Conversão QVT conforme ODM (usando ATL)	OWL
Protégé	UML s*tandard*	Programada	Formato Protégé (e outros suportados pela ferramenta)

Tabela 2 - Resumo das várias abordagens de conversão

A revisão da literatura permite concluir que vale a pena optar pelo modelo O3F para representar ontologias devido às várias vantagens que apresenta em relação às outras abordagens como a representação de associações, predicados e funções de qualquer aridade (vantagem em relação ao UML e OWL); a representação de métodos de classes e sua hierarquização (vantagem em relação a qualquer linguagens de representação de ontologias[3]); especificação de relações de dependência (vantagem em relação ao OWL e Ontolingua); representação de acções e métodos de acção (vantagem em relação a qualquer linguagem de representação). Além do mais, o O3F permite modelar/representar um domínio de acordo com uma visão orientada a objectos ou de acordo com uma visão mais funcional ou relacional do mundo ou ainda misturando ambas. Apenas o Ontolingua oferece uma funcionalidade semelhante a esta mas muito menos expressiva porque apenas permite classes com atributos, mas não com métodos. Além do mais, através das suas facetas predefinidas, o O3F permite a especificação de elementos típicos do paradigma orientado a objectos como classes abstractas ou elementos estáticos. Finalmente, o O3F e a sua linguagem CO3L são

[3] Nesta breve conclusão, o UML não foi considerada uma linguagem de representação de ontologias.

usados e têm sido desenvolvidas no grupo de trabalho de Agentes e Inteligência Artificial do ISCTE-IUL, onde esta dissertação foi feita.

A revisão da literatura mostra também que nenhuma das conversões feitas a partir da norma UML (*standard)* consegue converter a totalidade dos diagramas de classes e diagramas de objectos. Existe sempre informação perdida em relação a alguns elementos (e.g. métodos de classes).

Tendo em conta o objectivo enunciado, a presente revisão dos conhecimentos e das práticas actuais no domínio desta dissertação mostra que há espaço de contribuição inovadora em três aspectos: a conversão da totalidade dos elementos dos Diagramas de Classes e/ou Objectos do UML normalizado (*standard*) numa linguagem de representação de ontologias, o enriquecimento do modelo O3F e da linguagem de especificação correspondente (CO3L) que permitam a total conversão, e a disponibilização pública das ontologias que resultam da conversão.

O requisito de conversão da totalidade dos Diagramas de Classes e/ou Objectos do UML em O3F implica a extensão deste modelo relativamente à Representação de Hierarquias (que permitirá representar generalizações UML) e à Representação de Indivíduos (o que permitirá a representação de Instâncias UML).

A necessidade de disponibilizar publicamente as ontologias que resultam da conversão dos Diagramas de Classes e Diagramas de Objectos condiciona ainda a escolha da tecnologia de conversão que será usada. Além de não permitir o processamento de diagramas UML com vários pacotes (*packets*), a tecnologia XSLT (a mais usada para realizar conversões) não permite também o acesso à base de dados onde a ontologia resultado será armazenada para suportar o acesso público. Ao invés disso, considera-se o uso de XPath (a componente do XSLT que permite ler ficheiros) para ler os modelos UML e da linguagem Java para realizar a conversão e armazenamento no sistema de informação.

Capítulo 3 Object Oriented Ontology Framework

Este capítulo descreve com pormenor o *Object Oriented Ontology Framework* (O3F) (Mota *et al*, 2003). Sobre este modelo recai uma grande parte do trabalho desta dissertação já que é o modelo usado para a representação de ontologias e é para o seu formato que os Diagramas de Classes UML são convertidos. Na primeira secção deste capítulo são apresentadas as melhorias introduzias no O3F pela presente dissertação em cooperação com outra (Correia, 2010). As restantes partes do capítulo (desde a secção 3.2 até a secção 3.10) descrevem os vários elementos que constituem o O3F incluindo já as melhorias descrita na secção 3.1.

3.1 Melhorias e Expansões ao O3F

O O3F foi inicialmente definido em (Mota *et al*., 2003). Desde a sua criação que o modelo O3F tem sido alvo de constantes melhorias. Algumas dessas melhorias tiveram origem no presente trabalho em conjunto com outra dissertação (Correia, 2010) onde são propostos vários melhoramentos e funcionalidades redesenhando também o seu diagrama de classes o qual pode ser visto no Anexo A.

As principais melhorias introduzidas são as seguintes:

- Representação de Hierarquias – Originalmente, o modelo O3F apenas permitia a representação de generalizações. Agora é possível a representação de hierarquias. A representação de generalizações é feita à custa do conceito de hierarquia.
- Representação de Indivíduos – O modelo O3F foi aumentado com a possibilidade de representar indivíduos. Um indivíduo pode ser instância de qualquer tipo de dados da ontologia (e.g. Classes, Associações, Funções, Inteiros), mas pode também não se pertencer a nenhum deles.

- Reestruturação das Assinaturas de Acção - Em versões anteriores o modelo O3F representava as assinaturas de acção relacionais e funcionais (*RelationalActionInterface* e *FunctionalActionInterface*) recorrendo a associações entre a classe *ActionInterface* e as classes *Function* e *Predicate*. Agora estes conceitos passam a ser representados através de subclasses das classes *Predicate* e *Function*. Isto reforça o conceito de que uma assinatura de acção relacional é um predicado e que uma assinatura de acção funcional é uma função.
- Reestruturação da relação entre métodos e operadores - Antigamente existia apenas uma classe que representava a relação entre métodos e operadores, a classe *MethodOperator*. Na sua forma actual, o modelo recorre a três classes distintas para representar as relações entre métodos e operadores, *FMethodFunction*, *AMethodAction* e *RMethodPredicate*. Cada uma destas classes permite representar um paralelismo que pode ser traçado entre um método e um operador. A grande vantagem de haver uma classe específica para cada relação é que capta a restrição de que apenas se podem relacionar funções com métodos funcionais, predicados com métodos relacionais, e acções com métodos de acção.
- Reestruturação da representação dos tipos pré-definidos e das relações entre eles - Na versão anterior do modelo, os tipos pré-definidos não apareciam explicitamente representados no modelo; eram descritos apenas textualmente. Actualmente, o modelo O3F contempla agora, no seu diagrama de classes, todos os seus tipos pré-definidos (*built in*). Adicionalmente, os tipos *Classifier*, *Function*, *Predicate*, *FunctionalMethod* e *RelationalMethod* aparecem explicitamente representados como subtipos do tipo Set.
- Desambiguação de Facetas - Como funções, métodos funcionais e atributos representam conjuntos, a aplicação de algumas facetas a estes tipos gerava interpretações ambíguas. Por exemplo, aplicar a faceta *Minimum_value* a uma função não deixa claro se queremos caracterizar o conjunto denotado pela função ou o seu valor de retorno. Para resolver este problema acrescentou-se semântica às facetas *Minimum_value*, *Maximum_value*, *Minimum_size*, *Maximum_size* ficando agora definido que estas facetas caracterizam o contradomínio de uma função, método funcional ou de um atributo. Por outro

lado foram criadas as facetas *Smallest_instance*, *Largest_instance*, *Instance_maximum_size* e *Instance_minimum_size*, entre outras, que se aplicam a quaisquer conjuntos para caracterizar as suas instâncias.

- Novas facetas - Para além das facetas que foram acrescentadas para resolver a ambiguidade acima descrita, muitas outras foram criadas para completar muitos aspectos da linguagem. Alguns exemplos são as facetas *Navigability*, *Arg_direction*, *Whole*, *Part* entre outras. *Navigability* indica se os argumentos de uma associação são conhecidos pelos restantes. *Arg_direction* indica se um argumento de um método é usado como parâmetro de entrada, de saída, ou como ambos os sentidos. *Whole* especifica que o argumento de uma agregação ou composição representa o todo dessa mesma agregação ou composição e *Part* especifica os argumentos que constituem as partes desse todo.
- Criação de Facetas - Foram incluídos dois dispositivos relativos à definição de facetas novas *ValidFacetElement* e *ValidFacetType*. O predicado *ValidFacetElement*/2 usa-se para especificar os elementos a que uma faceta se pode aplicar. *ValidFacetType*/2 especifica os valores válidos de uma faceta.
- Dependências - O conceito de dependência, embora anteriormente representado no modelo O3F também foi melhorado e foram acrescentadas as asserções *Dependency*/1 e *DependencyArgument*/3 que em conjunto com a faceta *Dependence* (também introduzia nesta nova versão) permite à linguagem CO3L descrever relações de dependência.

3.2 Tipos

Um dos conceitos mais importantes do modelo O3F é o conceito de tipo de dados uma vez que este conceito é usado numa grande variedade de definições. Importa salientar que o O3F enfatiza a ideia de que um tipo é um conjunto. Embora não seja invulgar noutros contextos, esta realidade nem sempre é realçada. Por exemplo, o tipo *Integer* é o conjunto dos inteiros, o tipo *Char* é o conjunto dos caracteres e *Pessoa* (muito provavelmente, uma classe) é igualmente um conjunto.

O conceito *tipo de dados* é representado no modelo O3F através da classe *Type.* Esta classe generaliza outras duas, *Scalar* e *Collection* que representam os dois grandes grupos de tipos de dados.

A classe *Scalar* destina-se a representar os tipos de dados simples entre os quais números, caracteres, palavras, cadeias de caracteres (*strings)* e datas, cada um representado pela classe correspondente.

A classe *Collection* representa colecções de elementos do mesmo ou de tipos diferentes. *Collection* inclui sacos (*Bag*), conjuntos (*Set*), e sequências (*Sequence*).

Os tipos de dados O3F incluem ainda o conceito de classe que permite a representação de tipos compostos estruturados. As classes são representadas pela classe *Class* do diagrama de classes do modelo O3F. Uma classe é um conjunto de indivíduos com uma estrutura semelhante. Cada um destes indivíduos é representado, no modelo O3F, como um conjunto de pares nome (que neste caso se referem a atributos) e valor.

Talvez menos evidente seja o facto de que as associações, as quais relacionam classes e/ou outras associações, são igualmente conjuntos, ou seja, também são tipos. Consideremos, por exemplo a associação binária *CarroDaPessoa* cujos argumentos se chamam *carro* e *proprietário* (relaciona um carro com as pessoas que o detêm e relaciona uma pessoa com os seus carros). *CarroDaPessoa* é o conjunto dos pares {carro = *Carro*, proprietário = *Pessoa*} tal que *Pessoa* é proprietária de *Carro*. A classe *Association* do diagrama de classes representa as associações. Em geral, também podem ser definidos atributos numa associação. Uma associação é também um conjunto de indivíduos estruturados, cada um dos quais é um conjunto de pares nome e valor. Alguns desses pares representam argumentos da associação, outros representam atributos.

Predicados, funções, métodos relacionais e métodos funcionais denotam também conjuntos de indivíduos, cada um deles é um conjunto de pares nome e valor. Os predicados, funções, métodos relacionais e métodos funcionais são por isso também considerados tipos que representam indivíduos estruturados. Por exemplo, o predicado Distância, o qual relaciona duas localizações num dado espaço (*localização1* e *localização2*) e a distância (*distância*) entre elas, representa o conjunto dos conjuntos {localização1 = *Loc-1*, localização2 = *Loc-2*, distância = *Distância*} tal que *Distância* é o valor da distância entre as localizações *Loc-1* e *Loc-2*.

Estes conceitos serão descritos com mais pormenor ao longo deste capítulo.

3.3 Hierarquias

Ao ter a capacidade de representar tipos de dados é natural que o O3F permita também especificar hierarquias em que se organizam esses mesmos tipos. Isto significa que podemos estabelecer, por exemplo, hierarquias de classes e de tipos básicos (como é habitual), mas também hierarquias de funções, e de predicados, o que não é assim tão habitual. Embora não esteja explicitamente captado no diagrama de classes, as acções também se podem hierarquizar através do estabelecimento de uma hierarquia dos predicados ou das funções que constituem as suas assinaturas. Imaginemos que temos um predicado que relaciona uma pessoa o número do seu bilhete de identidade, e outro predicado que relaciona uma pessoa com o seu número de identificação fiscal. Ambos são casos particulares do predicado que relaciona uma pessoa com a sua identificação. Embora pouco habitual, os conceitos de hierarquia de funções, de predicados, de associações e de acções é uma ferramenta de modelação extremamente poderosa que permite representar mais informação sobre o domínio que é modelado.

No diagrama de classes do modelo O3F, as hierarquias são representadas pela classe *Hierarchy.* As relações entre os vários tipos que constituem uma hierarquia são representadas pelas classes *HierarchicRelation* e *SubHierarchicRelation* e as suas relações com a classe *Type*.

Além do grande poder de expressão que as hierarquias de tipos conferem ao modelo, esta possibilidade tem ainda a vantagem de aproximar o O3F do OWL, a linguagem mais divulgada actualmente para a representação de ontologias, uma norma de facto, suportado pelo W3C (*World Wide Web Consortium*), a qual permite também definir não só hierarquias de classes mas também hierarquias de propriedades (i.e., relações binárias que representam ora funções unárias, ora predicados binários).

3.4 Operadores

O O3F permite a especificação de três operadores: funções, predicados e acções. Estes operadores são representados no diagrama de classes do O3F pela classe *Operator* e pelas suas subclasses *Function, Predicate* e *Action*.

As funções permitem especificar uma aplicação de um conjunto de partida num conjunto de chegada, definindo conjuntos de conjuntos de pares nome valor, em que um deles representa o valor retornado pela função.

Os predicados podem relacionar um número indefinido de argumentos, os quais podem ter qualquer tipo de dados, enquanto que as associações podem apenas relacionar classes e/ou associações. Ao contrário das associações, os predicados não podem ter nem atributos nem métodos. Embora não sejam considerados operadores, o modelo O3F permite a representação de símbolos proposicionais, os quais são representados pela classe *ProposicionalSymbol.* Enquanto que os predicados têm de ter pelo menos um argumento, os símbolos proposicionais são formados por uma sequência de caracteres sem quaisquer argumentos.

Ao contrário das funções e dos predicados, as acções são um tipo de operador bastante mais invulgar pois para além do O3F não estão presentes em mais nenhuma abordagem à representação de ontologias. O aspecto mais importante de uma acção é o efeito que produz no mundo. Quando uma acção é executada, o mundo sofre uma alteração. Por exemplo, se a acção abrir a porta for executada, aplicada a uma porta particular, essa porta deixa de estar fechada e passa a estar aberta. O mundo altera-se. Funções e predicados e até mesmo associações não causam qualquer alteração no estado do mundo.

Tal como as associações, os predicados, e as funções, as acções têm igualmente argumentos e, por vezes, também devolvem valores. Diremos que as acções têm dois aspectos: um aspecto procedimental e um aspecto declarativo. O aspecto procedimental diz respeito aos efeitos da acção sobre o mundo, as alterações que a execução da acção causa no estado do mundo. O aspecto declarativo diz respeito aos seus argumentos, aos seus tipos de dados e, no caso de haver, aos valores de retorno.

Por agora o O3F não tem como representar o aspecto procedimental de uma acção, contudo esta funcionalidade está prevista através da futura modelação de axiomas no O3F. Ao aspecto declarativo de uma acção chamaremos assinatura da acção. As assinaturas de acção estão representadas, no diagrama de classes, pela classe *ActionInterface* e das suas particularizações *RelationalActionInterface* e *FunctionalActionInterface*, as quais são por sua vez particularizações de *Predicate* e de *Function*. Se uma acção não devolver nada, diz-se que tem uma assinatura relacional. Se, pelo contrário, a acção devolver um valor, diz-se que tem uma assinatura funcional. Para todos os efeitos, uma assinatura relacional é um predicado e uma assinatura funcional é uma função.

3.5 Classificadores

Os classificadores (*classifiers)* permitem representar elementos que partilhem das mesmas características, mais especificamente que partilhem dos mesmos atributos e métodos. O modelo O3F permite a representação de dois tipos de classificadores: classes e associações, os quais são representados pela classe *Classifier* do diagrama de classes do modelo O3F e pelas suas subclasses, *Class* e *Association*.

As classes permitem representar conjuntos de indivíduos semelhantes. Por exemplo a classe Pessoa, que possui os atributos “nome” e “idade”, permite representar todos os indivíduos que são instâncias desta classe e que portanto possuem esses mesmos atributos. A classe *Key* do modelo O3F permite indicar os conjuntos alternativos de atributos de uma classe que podem servir de identificadores inequívocos para as suas instâncias.

As associações possibilitam a representação de relações entre vários indivíduos, cada um deles pertencente a um classificador. Os vários argumentos de uma associação são representados pela classe *AssociationArgument* do modelo O3F. A classe *AssociationArgument* permite também representar o papel, o tipo e a multiplicidade de cada argumento da associação.

3.6 Métodos

Á semelhança do que acontece com os operadores, o O3F permite a especificação de três tipos de métodos: métodos funcionais, métodos relacionais e métodos de acção. Estes métodos estão representados no diagrama de classes pelas classes *FunctionalMethod, RelationMethod* e *ActionMethod* respectivamente, sendo todas particularizações da classe *Method*.

Os métodos funcionais não operam qualquer alteração no estado do mundo e, quando aplicados aos objectos da classe ou associação em que estão definidos (e eventualmente aos seus argumentos, se os houver), denotam um valor.

Os métodos relacionais também não operam alterações no estado do mundo e, quando aplicados aos objectos da classe ou associação em que se definem (e eventualmente aos seus argumentos, se os houver) denotam uma proposição verdadeira ou falsa.

Os métodos de acção, quando aplicados aos objectos da classe ou associação em que são definidos (e eventualmente aos seus argumentos, se os houver) efectuam alterações ao estado do mundo. Há métodos de acção que computam valores que retornam e há métodos de acção que não computam qualquer valor de retorno.

Existem alguns paralelismos que podem ser traçados entre métodos e operadores, mais exactamente entre funções e métodos funcionais, predicados e métodos relacionais, e entre acções e métodos de acção. São precisamente estes paralelismos que dão a liberdade ao utilizadores de fazer descrições de domínios segundo uma perspectiva mais funcional ou relacional, ou segundo uma perspectiva orientada a objectos, ou até mesmo misturando as duas.

Por exemplo, tomando o caso das funções e métodos funcionais, imaginemos o método funcional F com aridade 1, definido na classe C. Se x for um objectos da classe C e se α for o argumento de F numa dada invocação, a expressão $x.F(\alpha)$ em notação de objectos representa exactamente a mesma coisa que $F(x, \alpha)$ numa notação funcional. Ou seja, um método funcional com N argumentos, em notação de objectos, funciona como uma função de N+1 argumentos em notação funcional, em que o argumento extra é exactamente o objecto a que a função se aplicaria no modelo de

objectos. O mesmo paralelismo é aplicado entre os métodos relacionais e predicados, e entre métodos de acção e acções.

Para reforçar este conceito o O3F permite ainda que os métodos, para além de poderem ser definidos directamente nas suas classes ou associações, sem qualquer referência a funções, predicados ou acções, possam também ser definidos à custa dos respectivos operadores. Tal pode ser feito recorrendo à classe associativa *FMethodFunction* para fazer a correspondência entre um método funcional e uma função, à classe associativa *RMethodPredicate* para fazer corresponder um método relacional a um predicado, ou ainda à classe associativa *AMethodAction* para fazer a correspondência entre um método de acção e uma acção.

Tomando mais uma vez como exemplo o caso das funções, um método funcional de uma classe ou associação C poderia ser definido à custa de uma função seguindo os seguintes passos:

1 – Declaração da função F com N+1 argumentos que fará o papel de método funcional dos elementos de C.

2 – Declaração do método funcional FM, da classe ou da associação C, dizendo que FM se define à custa da relação de C com F. Esta especificação faz-se à custa da classe associativa *FMethodFunction*. O atributo *Self* de *FMethodFunction* permite especificar qual dos argumentos de F faz o papel do objecto a que FM é aplicado.

O mesmo processo pode ser feito de forma análoga para declarar métodos relacionais e métodos de acção à custa de predicados e acções respectivamente.

Para além das correspondências feitas entre métodos e operadores, pode também ser feita a simetria entre atributos de classificadores e funções, já que um atributo pode também ser definido como uma função unária, em que o único argumento representa o objecto do qual o atributo faz parte. Ou seja, se A for o atributo de uma classe e se x for um objecto dessa mesma classe *x.A*, na notação de objectos, é o mesmo que A(*x*), na notação funcional.

Tal como acontece com os métodos de classificadores, os atributos da classe ou associação C podem também ser definidos á custa de uma função. Esta especificação

faz-se através da associação *AttributeFunction* do diagrama de classes do modelo O3F através dos seguintes passos:

1 – Declaração da função unária que fará o papel de atributo dos elementos de C, por exemplo F/1.

2 – Declaração do atributo A, da classe ou da associação C, dizendo que A se define à custa da relação de C com F recorrendo à associação *AttributeFunction.*

3.7 Facetas

Um dispositivo extremamente flexível e fácil de usar no modelo O3F são as facetas. Uma faceta pode ser usada para caracterizar quase tudo quanto se possa escrever numa ontologia, desde classes e associações até aos argumentos e valores retornados por métodos, passando por atributos, entre outras coisas. Existe um conjunto pré-definido de facetas no O3F e, havendo necessidade, podem definir-se novas facetas. De entre as facetas pré-definidas, exemplificam-se *Mandatory, Default-value* e *Distinct*, entre muitas outras. *Mandatory* é uma faceta que especifica, por exemplo, se é obrigatório ou não que um dado atributo ou argumento tenha valor. *Default-value* pode ser usada para especificar o valor em caso de omissão (*by default*) de um atributo ou de um argumento. *Distinct* permite especificar se o valor do atributo de uma dada instância tem de ser diferente do valor do mesmo atributo de uma instância diferente.

As facetas estão representadas no diagrama de classes do modelo O3F pela classe *Facet*. A associação *FacetElement* entre a classe *Facet* e *ElementType* permite especificar os elementos a que uma faceta pode ser aplicada. A classe associativa *ElementFacet* representa as aplicações de facetas a determinados elementos.

3.8 Dependências

O modelo O3F permite captar relações de dependência entre os vários elementos da ontologia. Uma relação de dependência é uma relação sobretudo semântica, onde uma alteração da parte independente pode afectar a semântica da parte dependente. Uma relação de dependência pode também indicar que determinado elemento necessita de

outro para poder completar a sua especificação. A informação relativa a uma dependência e aos seus argumentos é captada pelas classes *Dependency* e *DependencyArgument* respectivamente.

A faceta *Dependence* permite completar a especificação de dependências indicado se um argumento de uma relação de dependência é dependente ou independente. Presentemente o modelo O3F não tem forma de especificar de que maneira um elemento depende de outro. Quando o O3F estiver munido de uma linguagem para especificação de axiomas será possível melhorar a representação de dependências e descriminar de que forma um elemento depende de outro.

3.9 Indivíduos

O O3F permite representar indivíduos e especificar se estes são instâncias de um ou mais tipos. É permitido indicar que um indivíduo é uma instância de qualquer tipo de dados. Em particular, pode declarar-se que um indivíduo é uma instância de uma classe ou de uma associação. Para isso, basta que esse indivíduo tenha os atributos obrigatórios dessa classe ou associação, mesmo que não tenha outros atributos da definição da classe ou da associação, e mesmo que tenha outros atributos que não correspondam à definição da classe ou da associação. Um indivíduo pode não ser instância de nenhum tipo, podendo ter uma constituição diferente das constituições dos tipos de dados estruturados, como por exemplo classes, associações e predicados da ontologia. O facto de instâncias de classes, associações, predicados e funções serem representados da mesma forma (i.e., conjuntos de pares nome/valor) é reflexo da unificação que o O3F estabelece entre a modelação orientada a objectos e a modelação relacional e funcional.

A classe *Individual* do diagrama de classes do modelo O3F é responsável pela representação de indivíduos. A associação entre esta classe e a classe *Type* representa os indivíduos que são instâncias dos vários tipos de uma ontologia.

3.10 Axiomas

Os axiomas conferem uma expressividade muito grande aos modelos e linguagens de representação de ontologias. Há relações, definições e restrições que se podem definir

através de axiomas. Imaginemos, por exemplo a classe Pessoa com o atributo *data_nascimento* e o método funcional *idade* que determina a idade da pessoa, numa determinada data, recebida como argumento. Pode usar-se um conjunto de axiomas para definir o valor da idade de uma pessoa numa data especificada. Usando a lógica de predicados de primeira ordem, um desses axiomas teria uma aparência semelhante à seguinte proposição:

$\forall x$ [$x \in$ Pessoa $\wedge$ x.data_nascimento.mes() < d.mes()] $\Rightarrow$ x.idade(d) = d.ano() - x.data_nascimento.ano()

Apesar do modelo O3F contemplar axiomas, como é visível no diagrama de classes O3F, a representação de axiomas em CO3L ainda não foi convenientemente estudada. Os axiomas serão criados em próximos trabalhos que envolvam o estudo e avanço do modelo O3F e da linguagem CO3L.

Capítulo 4 Conversão de Modelos UML em Ontologias O3F

Neste capítulo descreve-se a conversão de modelos UML constituídos por Diagramas de Classes e Diagramas de Objectos para ontologias O3F/CO3L. Cada secção será dedicada à conversão de um elemento ou uma família de elementos UML para O3F.

Ao contrário do que foi feito por muitos autores, como Baclawski e seus colaboradores (Baclawski *et al*., 2002), e Gašević e a sua equipa (Gašević *et al*., 2004), os quais usaram versões estendidas do UML com estereótipos e meta classes, a conversão descrita nesta dissertação tem por objecto a versão normalizada (*standard*) dos Diagramas de Classes e dos Diagramas de Objectos UML.

Pretende-se que toda a informação presente em diagramas de classes e diagramas de objectos seja representada através do modelo O3F e consequentemente descrita usando a linguagem CO3L. O processo de conversão de um diagrama UML tem dois resultados, um que corresponde ao preenchimento de uma base de dados com a ontologia O3F, e um segundo correspondente a um ou mais ficheiros contendo a especificação textual da mesma ontologia recorrendo à linguagem CO3L. Em seguida é descrito em que é que os vários elementos dos diagramas de classes e de objectos UML são convertidos. São mostrados exemplos por meio da linguagem CO3L.

4.1 Pacotes

Num Diagrama de Classes UML, um pacote pode conter a definição de todos os outros elementos da conceptualização e até mesmo de outros pacotes. Assim, estabeleceu-se que um pacote UML será convertido numa ontologia O3F. No caso de um pacote conter outros pacotes, os elementos dos pacotes internos farão também parte da ontologia correspondente ao pacote mais geral. Tudo funciona como se uma ontologia tivesse sido incluída noutra.

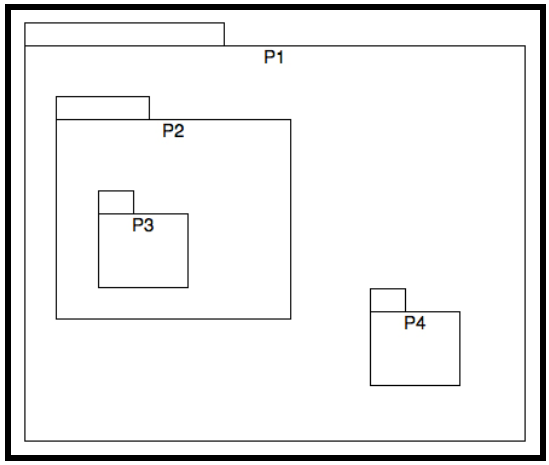


Figura 3 - Estrutura de pacotes UML

Por exemplo, os pacotes UML representados Figura 3 são convertidos em quatro ontologias. A ontologia correspondente ao pacote P1 com todos os seus elementos e os elementos de todos os pacotes internos (P2, P3 e P4), uma outra ontologia, correspondente ao pacote P2 com os elementos de P2 e P3, uma ontologia apenas com os elementos de P3 correspondente ao pacote P3, e outra com os elementos de P4 correspondente ao pacote P4.

4.2 Classes

As classes UML são convertidas em classes O3F tratando-se de uma conversão directa. A especificação de uma classe, na linguagem CO3L, faz-se através de uma asserção com o predicado *Class/1* (com aridade 1).

A conversão dos atributos e métodos das classes será descrita mais adiante.

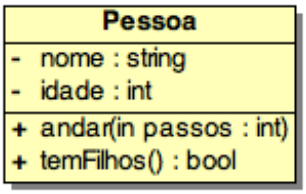
UML	O3F/CO3L
Abstract	Uso da faceta Materialization com o valor Abstract Exemplo: EntityFacet(Ser, Materialization, Abstract)
Active	Uso da faceta Process_control com o valor Active Exemplo: EntityFacet(Aluno, Process_control, Active)

Tabela 3 - Conversão das propriedades adicionais das classes

Em UML pode dar-se ainda o caso de uma classe ser categorizada como abstracta/concreta, e como activa/passiva. A Tabela 3 descreve como é feita a conversão de ambos os casos, através do uso das facetas *Materialization*, com valores *Abstract*/*Concrete*, e *Process_control* com valores *Active*/*Passive*

4.3 Atributos

Os atributos UML são convertidos em atributos O3F, uma vez que ambas as linguagens possuem o mesmo conceito. A linguagem CO3L permite descrever esta informação usando a asserção *Attribute/3* especificando a classe a que pertencem, o seu nome e o seu tipo de dados.




```
Class(Pessoa)
Attribute(Pessoa, Nome, String)
Attribute(Pessoa, Idade, Integer)
ActionMethod(Pessoa, andar)
Argument(Pessoa.andar, passos, Integer)
FunctionalMethod(Pessoa, temFilhos, Boolean)
```

Figura 4 – Conversão de uma classe e respectivas propriedades

O UML permite definir a multiplicidade dos atributos de classe. Quando um atributo é classificado com uma multiplicidade superior a um, a sua conversão para O3F resulta num atributo cujo tipo será uma colecção. Para isso recorre-se a *Collection_of* ou ao operador *Sequence_of* caso o atributo também tenha sido classificado como *ordered* em UML.

Caso em UML um atributo seja classificado com multiplicidade um e apenas um (não podendo ser zero e portanto indicado que se trata de um atributo obrigatório) é usada a faceta *Mandatory* com o valor *True* para representar este conceito em O3F e CO3L.

UML	O3F/CO3L
Static	Uso da faceta *Scope* com o valor *Classifier* Exemplo*:* EntityFacet(Empregado.nEmpregados,Scope,Classifier)
Read only	Uso da faceta *Access* com o valor *Read_only* Exemplo*:* EntityFacet(Pessoa.nome,Access,Read_only)
Derived	Uso da faceta *Dependence* com o valor *Derived* Exemplo*:* EntityFacet(Pessoa.idade,Dependence, Derived)
Derived union	Uso da faceta *Dependence* com o valor *Derived_union* Exemplo*:* EntityFacet(Pessoa.idade,Dependence,Derived_union)
Ordered	Uso do operador *Sequence_of* Exemplo: Sequence_of Pessoa
Unique	Uso da faceta *Distinct* com o valor *True* Exemplo*:* EntityFacet(Pessoa.id, Distinct,True)
Valor por defeito	Uso da faceta *Default_value* indicando qual o valor a ser assumido na ausência da especificação Exemplo*:* EntityFacet(Cão.raça, Default_value, rafeiro)

Tabela 4 - Conversão das propriedades adicionais dos atributos

Em UML um atributo pode ser ainda caracterizado com muitas outras propriedades. A Tabela 4 mostra como pode ser feita a conversão dessas propriedades

4.4 Métodos

Ao contrário do O3F, o UML não distingue os métodos que provocam alterações no mundo dos que não provocam. Em O3F os métodos que produzem alterações no mundo são classificados como métodos de acção (*action methods*) e podem ou não ter um valor de retorno.

Na presença de um método UML que não devolve nenhum valor este problema não se põe, pois sabe-se que se trata de um método que a única coisa que pode fazer é uma alteração no mundo, ou seja, um método de acção. Os métodos UML que não

devolvem nada são por isso convertidos em métodos de acção O3F. Quando se pretende converter um método UML que devolve algo como resultado da sua execução, é impossível saber se esse método apenas calcula um valor, isto é, se é meramente funcional, ou se para além disso realiza também alterações no mundo. Para uma conversão mais correcta de métodos UML que retornam algo, a ferramenta UML2O3F pergunta ao utilizador se o método em questão provoca ou não alterações no mundo. Se não provocar alterações no mundo, o método é simplesmente convertido para um método funcional O3F (*functional method*). Caso em que para além de devolver um valor, o método provoque também alterações, a conversão é feita para um método de acção O3F (*action method*) e é lhe aplicada a faceta *Return_type* que permite indicar que tipo de dados o método de acção devolve.

O CO3L especifica estes conceitos através das asserções *FunctionalMethod/3 e ActionMethod/2,* com as quais se pode indicar a classe a que pertencem os métodos, o seu nome e, no caso das funções, o tipo de dados que devolvem. Caso as acções também devolvam um tipo de dados usam-se as facetas *Retun_type* e *Return_role*, tal como referido. Os métodos podem ser ainda definidos como estáticos e/ou abstractos. A conversão de um método estático recorre à faceta *Scope* do O3F, com valor *Classifier*. A conversão de um método abstracto recorre à faceta *Materialization* do O3F com valor *Abstract*. Os métodos UML são convertidos em argumentos O3F. A linguagem CO3L usa asserções do predicado *Argument/3* para especificar a informação sobre argumentos de métodos ou de operadores, indicando o método a que pertencem, o seu nome e o seu tipo de dados.

UML	**O3F/CO3L**
Valor por defeito	Uso da faceta *Default_value* indicando qual o valor a ser assumido na ausência da especificação de um Exemplo*:* EntityFacet(número_mínimo, Default_value,0)
Direcção	Uso da faceta *Agr_direction* indicando qual o valor a ser assumido na ausência da especificação de um Exemplo*:* EntityFacet(Carro.cor, Arg_direction, In)

Tabela 5 - Conversão das propriedades adicionais dos argumentos de métodos

Sobre os argumentos de métodos podem ainda ser especificadas as propriedades presentes na Tabela 5 a qual mostra também a sua conversão.

4.5 Associações

À semelhança do que acontece com as classes, também uma associação em UML é convertida numa associação em O3F. No entanto, na maior parte das ferramentas UML é permitido que duas associações diferentes tenham o mesmo nome, podendo ser distinguidas graficamente. Tal não é permitido em CO3L. Para resolver este problema, quando várias associações UML partilham o mesmo nome, na conversão para O3F/CO3L é acrescentado um número de ordem ao seu nome permitindo assim a sua distinção.

No caso em que, num diagrama de classes, uma associação não tem nome, é gerado um automaticamente concatenando a palavra "UnnamedAssociation" com um número de ordem. A especificação de associações na linguagem CO3L é feita com asserções do predicado *Association/1*. As diferentes classes ou associações relacionadas por meio de uma associação são convertidas em argumentos de associação em O3F (*AssociationArgument*).

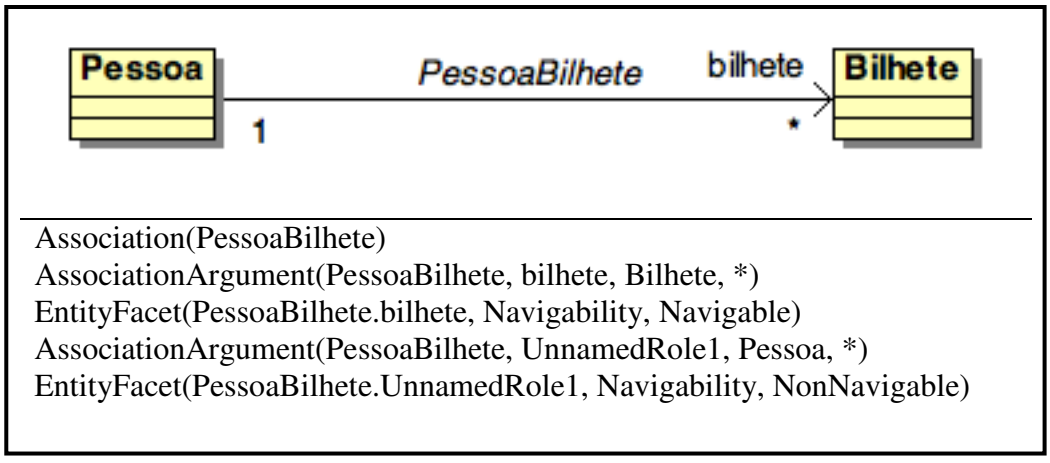


Figura 5 - Conversão de uma associação e respectivos membros

A linguagem CO3L permite especificar esta informação com a asserção *AssociationArgument/4* indicando a associação a que pertence o argumento especificado, o seu papel (*role*) na associação, o seu tipo de dados, e a multiplicidade. Deste modo toda a informação que está presente em UML é mantida em O3F.

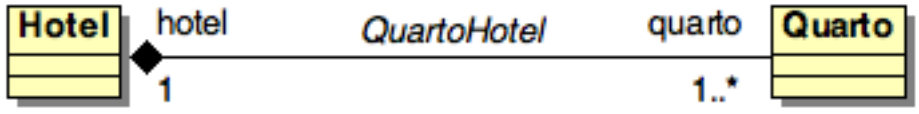
Quando o nome do *role* não é especificado em UML, é gerado um pela junção da palavra "UnnamedRole" com um número de ordem.

Em UML é ainda possível indicar se um membro de uma associação é navegável ou não, isto é, se os restantes membros da mesma associação estão cientes da sua existência. Este conceito é também muitas vezes utilizado para representar associações unidireccionais.

Para converter essa informação para O3F faz-se uso da faceta *Navigability* com o valor *Navigable* para indicar que o argumento é navegável, com o valor *Non_navigable* para indicar que o argumento não é navegável ou ainda com o valor *Non_specified* que significa que não é indicada informação sobre a navegabilidade do argumento.

4.6 Agregações e Composições

As agregações e composições em UML são convertidas para associações em O3F, cuja faceta *Association_type* tem o valor *Aggregation* ou *Composition*, conforme o caso.



Association(QuartoHotel)
EntityFacet(QuartoHotel, Association_type, Composition)
AssociationArgument(QuartoHotel, quarto, Quarto, *)
AssociationArgument(QuartoHotel, hotel, Hotel, 1)
EntityFacet(QuartoHotel, Whole, hotel)
EntityFacet(QuartoHotel, Part, quarto)

Figura 6 - Conversão de uma agregação e composição

Para especificar o argumento da agregação ou da composição que agrega ou é composto, e o argumento que compõe ou que é agregado, usam-se as facetas *Whole* e *Part.*

A Figura 6 ilustra como é feita esta conversão, para que seja mais facilmente compreendido o funcionamento deste processo.

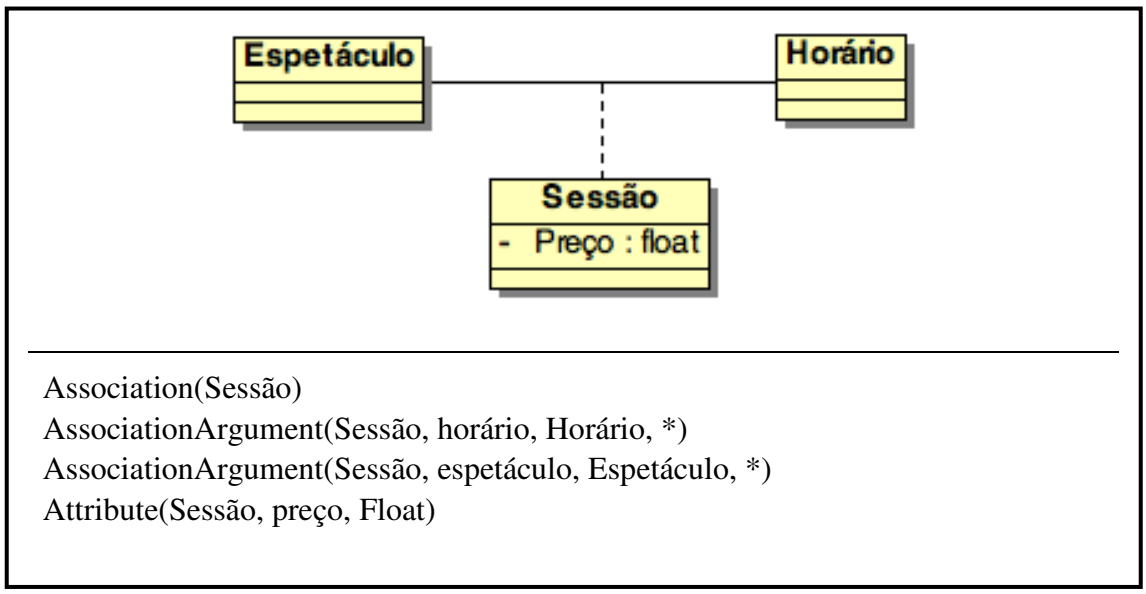


Figura 7 - Conversão de uma classe associativa

À semelhança do que acontece com as classes, por vezes há necessidade de caracterizar uma associação com atributos e métodos. Em UML, as associações com atributos e/ou métodos chama-se classes associativas. As classes associativas UML são convertidas em associações O3F com atributos e métodos.

4.7 Generalizações

A forma como as generalizações são representadas em UML e em O3F difere um pouco, pois em O3F as generalizações são representadas à custa do conceito de hierarquia. Por esta razão, a conversão de uma generalização UML requer alguma atenção.

A diferença entre uma generalização e uma hierarquia reside em dois aspectos:

- Uma hierarquia tem um ou mais níveis de generalização, enquanto que uma generalização tem apenas um nível;
- Todas as generalizações da mesma hierarquia reflectem o mesmo critério de organização/decomposição de um conjunto em subconjuntos.

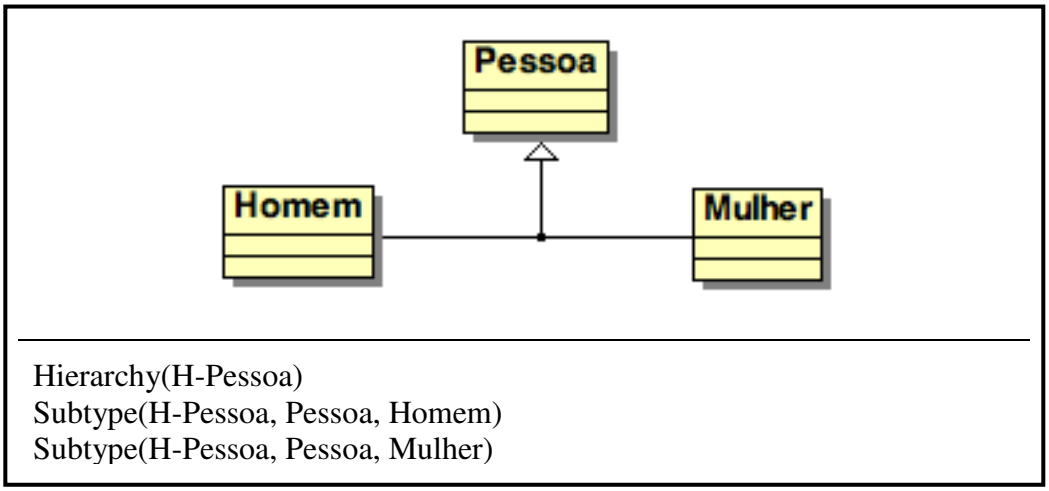


Figura 8 - Conversão de uma generalização

Infelizmente os ficheiros XMI não indicam se várias generalizações constituem ou não uma mesma hierarquia. Devido a isto optou-se por converter todas as generalizações que partilhem a mesma super classe para uma única hierarquia O3F. A linguagem CO3L representa hierarquias através da asserção *Hierarchy/1*. Em seguida, com a asserção *Subtype/3,* são indicadas as classes que pertencem a uma hierarquia e as respectivas particularizações.

4.8 Interfaces e Realizações

Uma interface, em UML, representa um padrão ou um modelo de uma classe. Em UML, podem criar-se classes de acordo com o padrão definido por uma interface – diz-se dessas classes que realizam a interface.

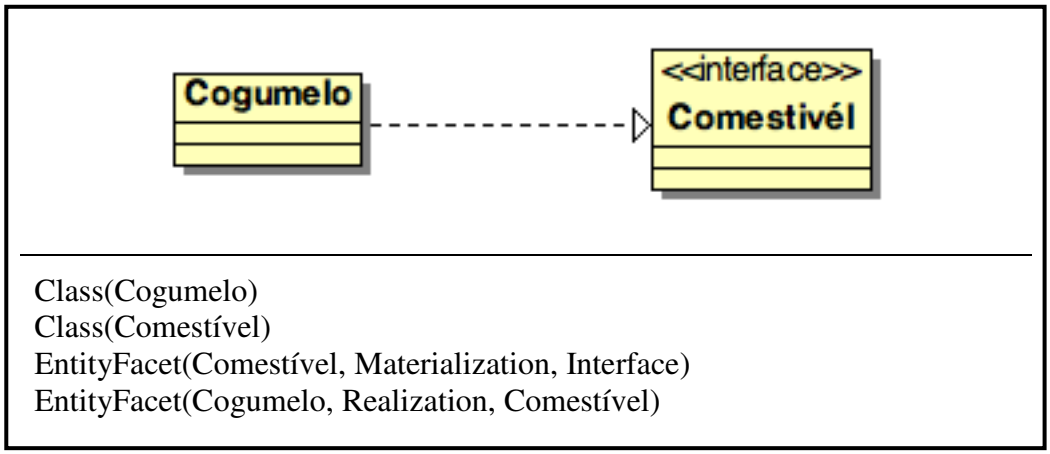


Figura 9 - Conversão de uma interface e uma realização

Tal como o UML também o O3F representa o conceito de interface. As interfaces UML são primeiramente convertidas para classes O3F, cuja faceta *Materialization* tem o valor *Interface* para indicar que se tratam de interfaces.

As classes UML que realizam uma dada interface são convertidas em classes O3F cuja faceta *Realization* especifica a interface realizada.

4.9 Dependências

As dependências UML são convertidas em dependências O3F. Para representar dependências usando a linguagem CO3L usa-se o predicado *Dependency*/1 onde é indicado o nome da dependência. As várias entidades que estão relacionadas por meio de uma relação de dependência UML são convertidas em argumentos da relação de dependência em O3F (representados pela classe *DependencyArgument*, no diagrama de classes do O3F). Para representar os argumentos de uma dependência em CO3L é usado o predicado DependencyArgument/3.

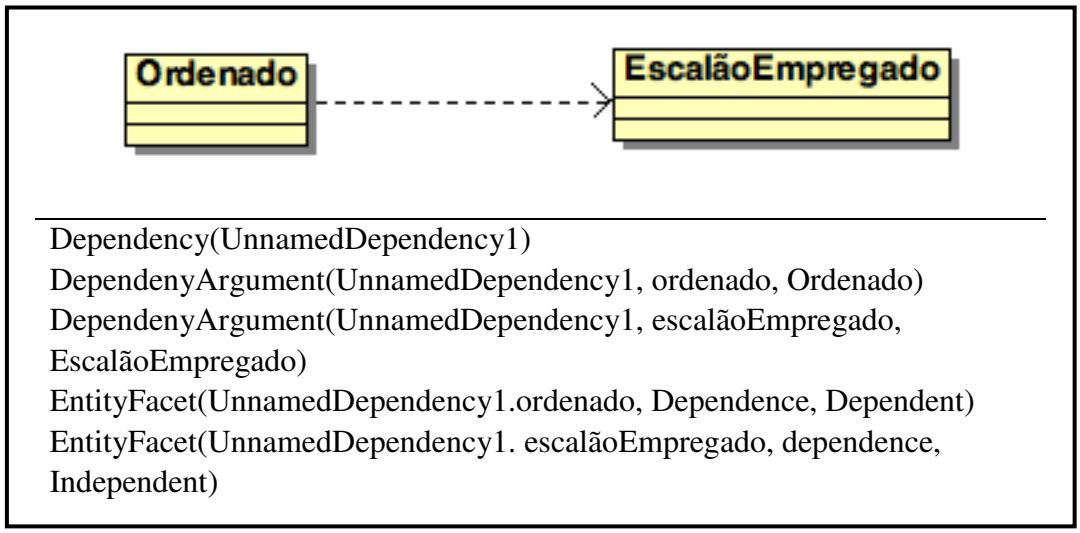


Figura 10 - Conversão de uma dependência

Os elementos dependentes, numa dada relação de dependência, têm a faceta *Dependence* com valor *Dependent*. Os elementos independentes têm a faceta *Dependence* com valor *Indepedent*, embora isso não seja obrigatório porque a faceta tem o valor *Independent* por omissão.

Caso o ficheiro XMI não contenha informação sobre o nome das dependências é gerado um, concatenando a palavra "UnnamedDependency" com um número de ordem.

4.10 Visibilidade

Em UML é possível definir a visibilidade de alguns elementos como classes, associações, métodos e atributos. Tal também é possível em O3F. A conversão desta propriedade é feita aplicando a faceta *Visibility* aos elementos cuja visibilidade queremos especificar, indicado o valor *Public*, *Private* ou *Protected*, exactamente com o mesmo sentido que em UML. A visibilidade *Ontology*, em O3F, representa o mesmo que a visibilidade *Package* em UML.

4.11 Objectos

O UML permite a representação de instâncias de classes por via dos seus Diagramas de Objectos. Tendo em conta que objecto e instância são conceitos comummente encontrados nas linguagens de descrição de ontologias, optou-se por fazer a sua conversão para O3F. Os objectos em UML são convertidos para objectos em O3F não havendo qualquer perda de informação.

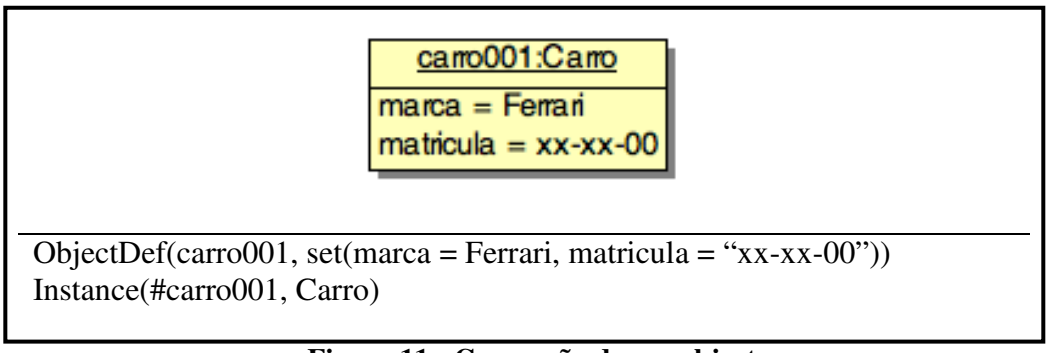


Figura 11 - Conversão de um objecto

Em O3F os objectos de uma classe são representados sob a forma de um conjunto de atributos com os respectivos valores. A linguagem CO3L permite definir um identificador para cada objecto através da asserção *ObjectDef/2*. Para indicar a classe a que o objecto pertence, é usada a asserção *Instance/2*

Capítulo 5 Arquitectura e Implementação

Todas as ferramentas desenvolvidas no âmbito desta dissertação enquadram-se num projecto de âmbito maior, um projecto que tem como objectivo o desenvolvimento de um conjunto de ferramentas para ontologias O3F. Actualmente, foram já desenvolvidas as seguintes ferramentas e/ou funcionalidades:

1. O3 Server: Servidor de Ontologias O3F com a possibilidade de armazenar e gerir ontologias de acordo com o modelo O3F
2. Conversão de ontologias armazenadas no Servidor em representações textuais na linguagem CO3L
3. UML2O3F: Ferramenta de conversão de UML em ontologias O3F
4. CO3L Edit: Editor especializado na linguagem CO3L

Prevê-se ainda a criação de outras ferramentas futuras:

5. Ferramenta de conversão de ontologias OWL em ontologias O3F
6. Ferramenta de conversão de ontologias O3F em ontologias OWL
7. Geração de ontologias JADE a partir de ontologias O3F

O UML2O3F, a principal ferramenta desenvolvida no âmbito da dissertação, permite a conversão automática de modelos UML constituídos por Diagramas de Classes e/ou Diagramas de Objectos em ontologias O3F de acordo com o especificado no Capítulo 4 O UML2O3F permite gerar directamente ficheiros de texto na linguagem CO3L, mas também permite armazenar o resultado da conversão na base de dados do O3 Server. O mesmo se passa com o CO3L Edit: as ontologias CO3L podem ser guardadas localmente em ficheiros do computador em que a ferramenta for usada, mas podem também ser armazenadas no O3 Server.

Para além do UML2O3F, o presente trabalho contribuiu também para a criação do servidor de ontologias O3F, o O3 Server. Esta contribuição traduz-se, mais

concretamente, no desenvolvimento da base de dados do servidor (feito em cooperação com outra dissertação (Correia, 2010)) e de um mecanismo de extracção, na linguagem CO3L, de ontologias armazenadas na base de dados do servidor. O O3 Server está actualmente alojado no computador venus.iscte.pt acessível através do porto 4432, um dos servidores da rede ISCTE-IUL. Todas as ferramentas e funcionalidades descritas estão totalmente integradas com o O3 Server.

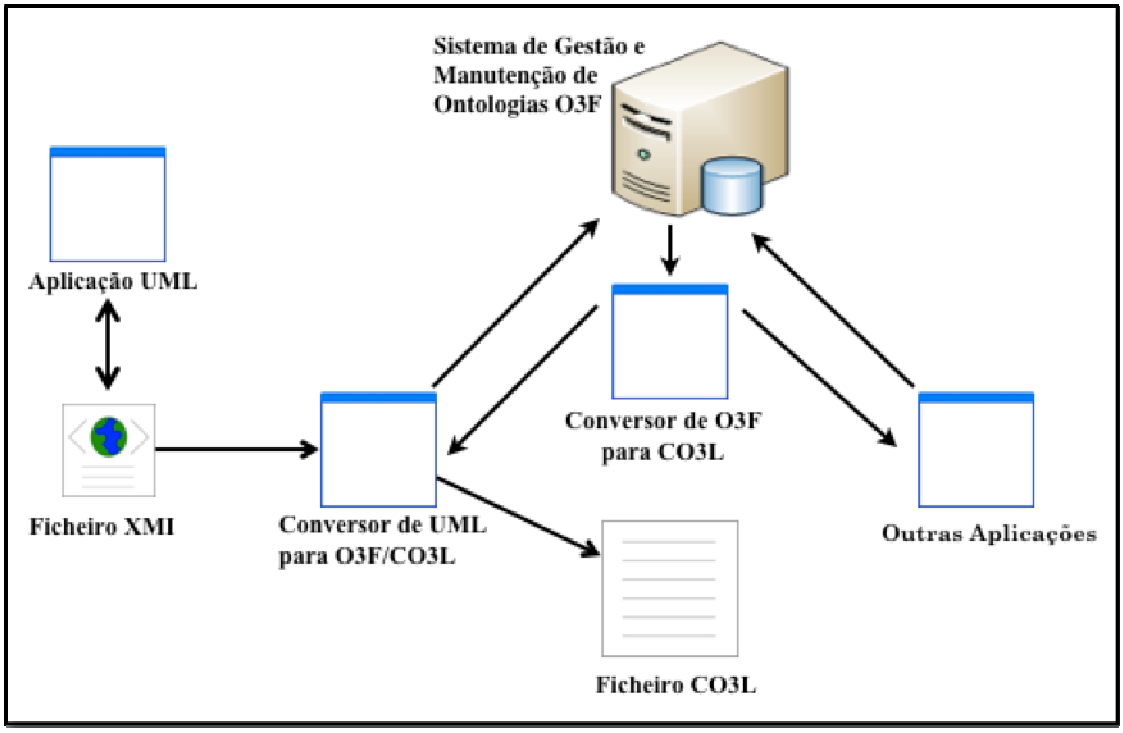


Figura 12 - Arquitectura geral do sistema

Este capítulo começa por fazer uma descrição do servidor (secção 5.1), visto este constituir o ponto central e comum a todas as ferramentas O3F. De seguida, na secção 5.2, é descrito o funcionamento do UML2O3F e a forma como interage com o servidor tal como representado na Figura 12. As tecnologias usadas para a implementação do sistema são também descritas bem como todas as decisões de implementação.

5.1 Servidor de Ontologias O3 Server

O servidor de ontologias O3 Server é o componente central e comum a todas as ferramentas O3F. É este servidor que permite o armazenamento das ontologias O3F

quer estas tenham sido originadas a partir de uma conversão feita com o UML2O3F ou escritas usando o CO3L Edit. O O3 Server é constituído por dois grandes componentes que serão descritos mais adiante: A base de dados que armazena as ontologias O3F e uma interface de acesso público que faculta acesso às ontologias armazenadas. O desenvolvimento deste servidor de ontologias implicou também os melhoramentos feito ao modelo O3F descritos no capítulo 3. Tudo isto foi feito em colaboração entre os elementos de uma equipa constituída pelo Professor Luís Botelho, orientador de ambas as dissertações "O3F: Um Servidor e um Editor CO3L(Correia, 2010), e esta mesma, e pelos autores das dissertações, visando os seguintes objectivos:

Garantir a solidez do modelo O3F e a possibilidade de representar qualquer diagrama de classes e de objectos UML;

Garantir um mecanismo eficiente de armazenamento de ontologias O3F numa base de dados relacional; e

Garantir uma interface flexível mas normalizada entre a base de dados do servidor e as suas aplicações cliente.

5.1.1 Base de Dados do servidor

O modelo relacional que representa a base de dados do O3 Server foi desenhado tendo por base o diagrama de classes que representa o modelo O3F, que pode ser visto no Anexo A. Optou-se por desenhar o correspondente modelo relacional de forma manual, isto é, sem recorrer a nenhum processo automático de aplicação de regras de tradução de modelos de objectos em modelos relacionais. O objectivo desta escolha – a definição do modelo relacional sem usar processos automáticos – deve-se à complexidade do diagrama de classes do modelo O3F aliado ao facto que muitas vezes a aplicação sistemática das regras de tradução gera modelos pouco eficientes. Assim, para obter mais controlo na forma como o modelo relacional é gerado, as regras de tradução foram aplicadas manualmente com sentido crítico e tendo em conta os restantes processos que dependem da base de dados. Foram também usados várias regras de optimização, em particular nas várias generalizações do modelo. Não existe por isso uma correspondência directa entre todas as classes do diagrama de classes do

modelo O3F e as tabelas do modelo relacional da base de dados. No Anexo B pode ser visto o esquema do modelo relacional da base de dados.

A tabela central do modelo relacional é a tabela *Ontology*. Esta tabela, para além de armazenar o nome e parâmetros de uma ontologia, relaciona-se com os seus vários elementos, como por exemplo tipos, facetas, acções, dependências, etc.

A Tabela *Type* é possivelmente a tabela mais complexa do modelo e a qual é responsável por armazenar mais informação. Ao contrário do que se passa no modelo de classes O3F, onde existe uma classe para cada tipo de dados, no modelo relacional, a tabela *Type* é responsável por representar todos os tipos de dados. Tal deve-se à aplicação das regras de transposição de diagramas de classes para diagramas relacionais e às respectivas optimizações. Os tipos são depois distinguidos através do campo *TypeOfType* que representa o tipo do tipo (i.e. meta tipo).

As funções, predicados, métodos funcionais e métodos relacionais para além de serem interpretados como métodos e operadores, também representam tipos, e estão por isso incluídos na tabela *Type.* Por esta razão, e para simplificar e optimizar o modelo relacional optou-se por não criar as tabelas *Operator* e *Method* que corresponderiam às classes com o mesmo nome do diagrama de classes do O3F. A tabela *TypeArgument* foi criada para representar os argumentos dos métodos e operadores. Os argumentos das associações são de natureza diferente dos argumentos dos métodos e operadores, por este motivo, tal como no diagrama de classes do modelo O3F, são representados numa tabela diferente, a tabela *AssociationArgument*.

A tabela *Action* representa as acções e a tabela *ClassifierAction* representa os métodos de acção. Os argumentos destes elementos são representados pelas tabelas *ActionArgument* e *ClassifierActionArgument* respectivamente. A razão pela quase se optou por criar tabelas especificas para as acções, métodos de acções e os seus argumentos ao invés de agregar tudo na tabela *Type* deve-se ao facto de tanto as acções como os métodos de acção não serem classificados como tipos, apenas as suas assinaturas representam predicados ou funções que por sua vez constituem tipos.

No modelo relacional, as tabelas *RMethodPredicate*, *FMethodFuncion*, *AMethodAction* tem a mesma função que as classes com o mesmo nome no diagrama de classes do O3F: representar a relação entre métodos e operadores. Para além

destes, o modelo relacional conta também com a tabela *AttributeFunction* que representa a relação entre funções e atributos, tal como a associação com o mesmo nome, no diagrama de classes do O3F.

Para armazenar a informação referente a hierarquias, o modelo relacional conta com três tabelas: *Hierarchy*, *HierachyType* e *HierachyTypeChild*. A tabela *Hierachy* é responsável por armazenar informação sobre a identificação de hierarquias. A relação entre a tabela *Hierarchy* e *HierachyType* permite saber quais os tipos de uma determinada hierarquia. Finalmente, para definir os diferentes subtipos de um tipo é usada a tabela *HierarchyTypeChild*.

A aplicação de facetas foi um aspecto particularmente interessante de modelar pois podem ser aplicadas a qualquer elemento de uma ontologia. No diagrama de classes do modelo O3F, este facto está representado pelas relações da classe *Facet* com a classe *Element* que generaliza todas as classes. Para resolver este problema, no modelo relacional, foi também criada a tabela *Element* e, sempre que se verifica o registo de algo na ontologia, seja uma classe, uma acção ou qualquer outra coisa, para além de ser registado na tabela a que diz respeito é também feito um registo na tabela *Element*. Deste modo, a tabela *Element* contém informação sobre todos os elementos que figuram na ontologia permitindo assim a aplicação de facetas a cada um deles. O registo das facetas aplicadas a um dado elemento é feito na tabela *EntityFacet* (relacionada com as tabelas *Element* e *Facet*). Adicionalmente, a tabela *AplicableFacetType* permite saber a que tipos se podem aplicar uma dada faceta.

A tabela *Element* é igualmente útil para a representação de dependências uma vez que, tal como as facetas, as dependências também podem existir entre quaisquer elementos. Para representar os vários argumentos de uma dependência é usada a tabela *DependencyArgument* tal como diagrama de classes do O3F.

A forma de armazenamento da informação relativa a objectos difere um pouco de como é feito no diagrama de classes do O3F. No modelo relacional a declaração do objecto e a definição dos seus atributos é feita em tabelas separadas. São usadas as tabelas *ObjectDefinition* para a declaração dos objectos e a tabela *ObjectDefinitionAttribute* para a definição dos seus atributos e correspondentes valores. O objectivo desta escolha deve-se a promover a capacidade de inferência que a base de dados deve disponibilizar com os sistemas que com ela comuniquem. O

armazenamento segundo esta forma facilita a avaliação de expressões como Object.atributo = 'valor' ou a consulta de, por exemplo, todos os objectos que partilhem um determinado atributo. A tabela *Instance* destina-se a registar os objectos que são instâncias de tipos correspondendo à associação *instanceOf* do diagrama de classes do O3F.

Por último há que referir as tabelas *BuiltIn*. Estas tabelas, como é o caso das tabelas *BuiltInFacet* ou *BuilInType*, destinam-se a armazenar os elementos pré-definidos pelo O3F e que devem estar disponíveis em todas as ontologia como por exemplo a faceta *Materialization* ou o tipo *Integer*.

A tecnologia usada para a implementação da base de dados foi o SQL Server 2008. Esta escolha deve-se ao facto de o SQL Server, ao contrário do seus concorrentes gratuitos, contar com poderosas ferramentas de desenvolvimento e administração. O ISCTE-IUL disponibiliza aos alunos, através do protocolo MSDNAA (*Microsoft Developer Network Academic Alliance*), essas mesmas ferramentas, o que torna esta opção menos dispendiosa e portanto mais vantajosa que as restantes tecnologias pagas. Foi também possível alojar o O3 Server num servidor do ISCTE-IUL com SQL Server.

5.1.2 Interface de Acesso ao Servidor

Para além da base de dados que permite armazenar as ontologias, o O3 Server conta ainda com uma interface de acesso público que permita aceder, criar e editar essas mesmas ontologias. Por outro lado a interface de acesso tem também o objectivo de normalizar a comunicação feita entre o servidor e os diferentes clientes que com ele comunicam. A interface de acesso é constituída por duas partes.

A primeira, foi desenvolvida no âmbito da presente dissertação (para além do UML2O3F) e trata-se de um ferramenta implementada no próprio servidor que permite a conversão de ontologias armazenadas na base de dados do servidor em ontologias em formato de texto, na linguagem CO3L, ou seja permite extrair ontologias mantidas no O3 Server. Esta ferramenta foi desenvolvida sob a forma de um procedimento armazenado (*stored procedure*) SQL. Este procedimento armazenado é constituído por um conjunto de *queries* SQL que interrogam a base de dados e devolvem o conteúdo de uma ontologia no formato CO3L. A grande

vantagem de usar um procedimento armazenado é que outras ferramentas que venham a ser desenvolvidas, assim como qualquer outra já existente com suporte para SQL Server, necessitam apenas de a invocar para que possam obter, na linguagem CO3L, uma ontologia mantida no servidor.

A segunda parte da interface de acesso público trata-se de uma biblioteca de classes Java criada com a *Framework Hibernate* (www.hibernate.org) implementada no âmbito da dissertação "O3F: Um Servidor e um Editor CO3L" (Correia, 2010). A *Framework Hibernate* permite estabelecer uma correspondência entre uma tabela de uma base de dados relacional e uma classe Java de modo a facilitar a comunicação entre uma aplicação e a base de dados. A biblioteca de classes Java criada com esta *Framework* contém uma classe para cada tabela do modelo relacional referido na secção anterior, pelo que consegue representar todos os elementos de uma ontologia O3F como por exemplo classes, métodos de acção e facetas. Cada ferramenta O3F deve usar esta interface para implementar a consulta, a inserção, a edição e a remoção de ontologias no servidor o que garante uma normalização do acesso às ontologias.

A interface *Hibernate* do O3 Server permitiria também extrair uma ontologia da base de dados, porém para que uma aplicação possa usar esta API é necessário que seja desenvolvida em Java. A principal vantagem de usar o procedimento armazenado, descrito anteriormente, é que elimina a dependência do Java. Isto é, qualquer aplicação cliente do servidor (por exemplo, outras ferramentas O3F) pode realizar a extracção de uma ontologia sem recurso a nenhuma biblioteca adicional não ficando limitada a nenhuma linguagem de programação. Além disto, nos casos em que se pretenda desenvolver uma aplicação que apenas necessita de obter uma ontologia, o procedimento armazenado permite fazê-lo sem que seja necessário compreender o funcionamento da interface *Hibernate* do O3 Server. Para proceder à extracção, uma aplicação necessita apenas de invocar o procedimento armazenado para obter uma ontologia O3F mantida no servidor. Na invocação do procedimento armazenado devem ser passados, como argumentos, o nome da ontologia e a respectiva versão, sendo estes os parâmetros que a definem inequivocamente. Como retorno é devolvida a ontologia em formato de texto, na linguagem CO3L.

5.2 UML2O3F

A aplicação UML2O3F é a principal ferramenta desenvolvida no âmbito do presente trabalho. Permite realizar conversões automáticas de modelos UML constituídos por Diagramas de Classes e/ou Objectos em ontologias O3F, as quais podem ser armazenadas no servidor de ontologias O3 Server, ou escritas em ficheiros de texto no formato CO3L.

Na sua maioria o UML2O3F foi desenvolvido em Java (contendo 73 classes num total de 11.573 linhas de código). Foi seguida a arquitectura MVC (*Model/View/Controller*) (http://java.sun.com/blueprints/patterns/MVC.html), o que garante que o sistema possa ser facilmente mantido e mais facilmente integrado com os novos componentes que venham a ser construídos em futuros projectos ligados ao O3F.

Esta secção descreve o funcionamento do UML2O3F e a sua interacção com o O3 Server. O funcionamento da aplicação assenta em dois grandes fluxos: a leitura de modelos UML (representados no formato XMI) e a conversão dos modelos lidos.

A relação entre estes dois fluxos difere um pouco da maioria dos conversores de UML para linguagens de representação de ontologias estudados no Capítulo 2. Muitos autores, como Cranefield (Cranefield, 2001), Leinhos (Leinhos, 2006) e Gašević e a sua equipa (Gašević *et al*., 2004) implementaram mecanismos de conversão entre UML e linguagens de representação de ontologias, recorrendo a *scripts* XSLT (*eXtensible Stylesheet Language Transformation*) onde são usadas expressões regulares que representam elementos UML e aplicadas regras de conversão para os transformar noutro formato de texto. Tal como explicado no Capítulo 2, o processo de leitura de modelos e a correspondente conversão estão fortemente ligados e são ambos feitos recorrendo à mesma tecnologia. O XSLT é de facto bastante eficaz na leitura de ficheiros XML, como é o caso do XMI. No entanto, apresenta algumas desvantagens no processo de conversão, pois não permite a conversão de modelos com múltiplos pacotes (Gašević *et al*., 2004), é pouco flexível em situações de conversão mais complexas (Falkovych, Sabou & Stuckenschmidt, 2003) como por exemplo conversões de elementos que exijam muito processamento de texto ou que dependam

do valor de vários outros elementos. Para além disto não oferece suporte para comunicação com bases de dados, o qual é também um dos requisitos desta aplicação.

Para resolver este problema, e proporcionar uma solução mais poderosa, o processo de leitura de diagramas e o processo de conversão para O3F e CO3L são feitos de forma separada e recorrendo a tecnologias diferentes. A implementação do processo de leitura de modelos UML (no formato XMI) é feita usando a tecnologia XPath, o componente da tecnologia XSLT responsável pela leitura de ficheiros baseados em XML. O processo de conversão dos modelos é implementado em Java possibilitando assim a conversão de diagramas com vários pacotes e o armazenamento na base de dados do O3 Server. Deste modo pretende-se continuar a usufruir do poder da tecnologia XSLT para a leitura de ficheiros XML e obter um processo de conversão mais poderoso e flexível através do uso de uma linguagem de programação orientada a objectos, como é o caso do Java.

5.2.1 Leitura de Modelos UML

A leitura de modelos UML é a fase inicial da conversão para O3F. Para que esta leitura possa ser realizada é indispensável que o modelo UML seja construído com uma ferramenta que tenha a capacidade de exportar os modelos no formato XMI 2.1 (*XML Metadata Interchange*). Este é o formato usado para a representação textual de modelos UML. É a partir dele que os modelos UML são lidos.

Tal como já foi referido, o processo de leitura de diagrama é feito recorrendo à tecnologia XPath. Esta tecnologia usa expressões regulares que permitem endereçar de uma forma flexível os diferentes elementos de um ficheiro XML. Graças a isto é possível referenciar e ler toda a informação relativa a um determinado elemento com um mínimo de esforço.

Por exemplo para obter uma lista de todos os elementos de um ficheiro XMI que representem classes UML poderá ser usada a seguinte expressão regular "//packagedElement[@*[name()='xmi:type']='uml:Class']".

A leitura de modelos UML, representados textualmente em ficheiros XMI, não é uma tarefa trivial e apresenta duas grandes dificuldades. A primeira consiste em que muitos dos elementos presentes nos Diagramas de Classes têm, na sua definição, referências para outros elementos do mesmo diagrama. Em particular, é possível a

existência de referências cruzadas, o que obriga a que a ordem com que os elementos são lidos tenha de ser feita de forma muito cuidada. Os vários elementos de um diagrama são lidos por uma dada ordem, determinada pela sua categoria. Por exemplo, primeiro são lidas todas as classes, depois todas as classes associativas e depois todos os atributos de todas as classes. Esta ordenação é necessária porque, por exemplo, um atributo pode ser de um tipo que é representado por uma classe. Se esse atributo for lido antes da classe, a referência para o seu tipo ainda não está definida e resulta num erro.

A segunda dificuldade deve-se ao facto de que diferentes ferramentas, perante o mesmo modelo UML, geram ficheiros XMI com estruturas diferentes tal como foi verificado por Castelli (Castelli, 2009). Devido a este facto, é impossível garantir que a aplicação consiga ler modelos gerados por todas as ferramentas de desenho UML existentes. Contudo foi feito um esforço para que a aplicação desenvolvida seja compatível com o maior número possível de ferramentas possível. Destaca-se a compatibilidade com as ferramentas *Enterprise Architect* (http://www.sparxsystems.com/), *Altova UModel* (www.altova.com/umodel.html) e BOUML (http://bouml.free.fr/index.html). A BOULM foi a ferramenta usada para a realização da maior parte dos testes durante o desenvolvimento. As razões que levaram à preferência desta ferramenta são a de ser uma aplicação gratuita, compatível com vários sistemas (Windows, Linux e Mac OS) e de oferecer uma ampla lista de funcionalidades para desenvolvimento UML. O sistema está preparado para poder espoletar a execução da ferramenta BOUML caso esta se encontre também instalada.

Depois de lidos, os modelos são representados num conjunto de objectos Java. Esses objectos são instâncias de uma estrutura de classes capaz de representar qualquer modelo constituído por Diagramas de Classes e de Objectos. Existe uma classe para cada elemento que possa figurar no modelo lido. Os vários objectos estão também relacionados entre si por uma relação pai filho feita nos dois sentidos. Por exemplo, uma classe UML sabe quais são os seus atributos e os seus atributos sabem a que classe pertencem.

5.2.2 Conversão de UML em O3F e CO3L

Depois de lido um modelo e de este se encontrar representado pela estrutura de objectos Java referida na secção anterior, realiza-se a conversão. Cada um destes objectos implementa uma versão dos métodos *toCO3L* e *toO3F* de acordo com o elemento que representa. É através destes métodos que as conversões para CO3L e O3F são conseguidas. A conversão gera dois resultados (correspondendo às setas que saem da aplicação UML2O3F na Figura 12):

- Ficheiro(s) CO3L com a ontologia correspondente aos diagramas UML processados.
- Ontologia correspondente ao diagrama UML, armazenada na base de dados do servidor O3F.

O primeiro resultado, a representação textual, na linguagem CO3L, é gerado seguindo uma técnica da conversão em cascata. Isto é, como todos os objectos se encontram devidamente relacionados, para iniciar a conversão é apenas necessário que a aplicação invoque o método *toCO3L* de um pacote (sendo este o elemento correspondente a uma ontologia). Este invoca por sua vez o método *toCO3L* dos elementos que contém, por exemplo classes, associações e até mesmo outros pacotes que reptem o processo recursivamente. Esta técnica faz com que cada elemento tenha apenas de conhecer as suas próprias regras de conversão, o que para além de ser uma boa prática, facilita a alteração ou melhoramento deste processo. Uma vez completa, a conversão é apresentada no ecrã principal da aplicação para que possa ser analisada pelo utilizador. Nesta altura, o utilizador pode mandar armazenar a ontologia num ficheiro local do computador em que a aplicação é usada. Estes ficheiros podem posteriormente ser usados por outras aplicações, como por exemplo o CO3L Edit.

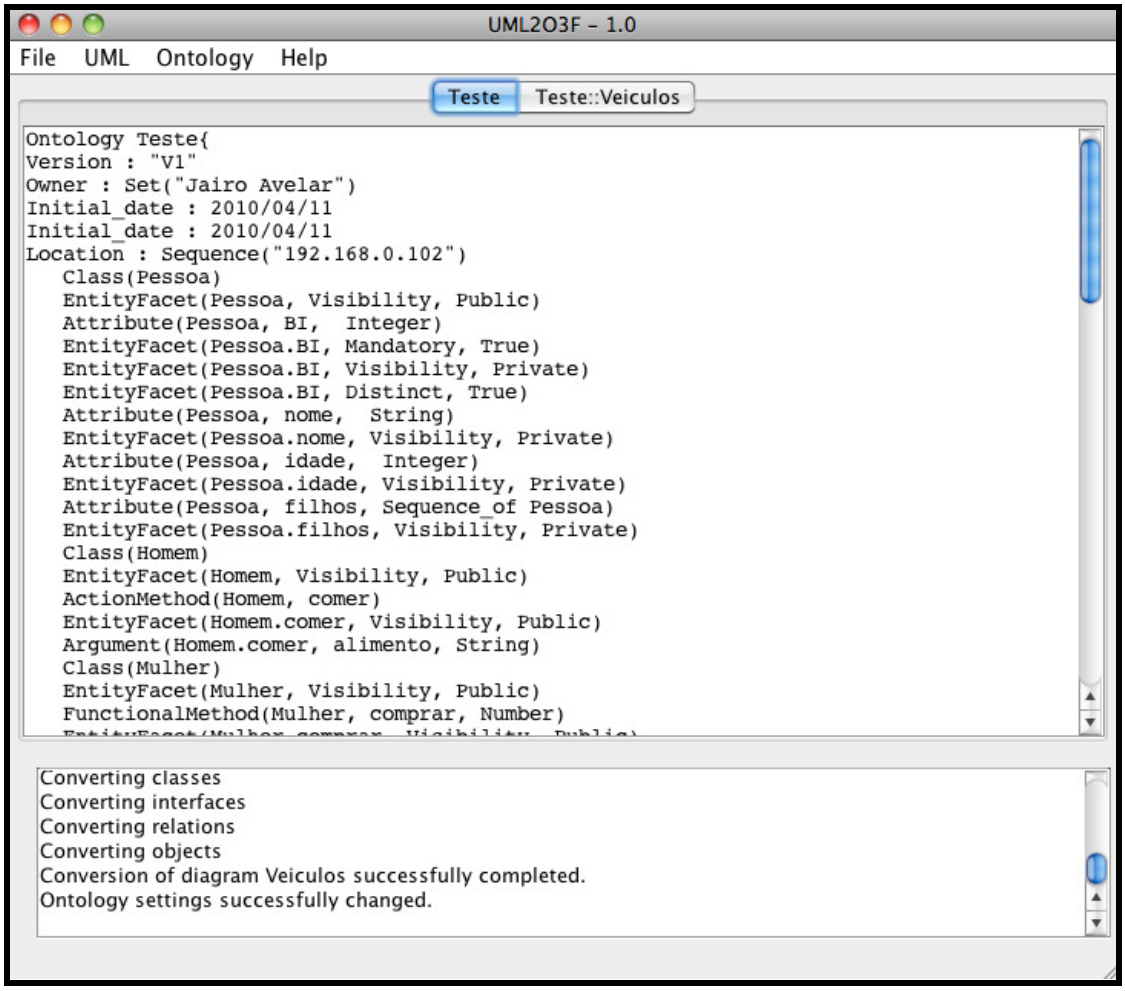


Figura 13 - Conversor de UML para O3F e CO3L

Depois de visualizar a ontologia no formato CO3L, é possível gerar o segundo resultado, i.e., o armazenamento da ontologia no servidor O3F. Este processo de conversão subdivide-se em duas fases distintas: uma primeira onde é realizada a conversão de facto e uma segunda onde a ontologia é inserida no servidor. Na primeira fase, é mais uma vez usada a técnica de conversão em cascata, mas desta vez invocando o método *toO3F* de cada objecto Java que representa um elemento UML. Ao contrário do método *toCO3L* que retorna texto, o método *toO3F* retorna, como resultado da conversão, objectos das classes Java da biblioteca *Hibernate* já referida, a qual constitui a interface de objectos com a base de dados do O3 Server. Cada chamada ao método *toO3F* pode retornar um ou mais destes objectos conforme seja necessário um ou mais elementos O3F para representar o elemento UML que se pretende converter. Por exemplo na conversão de uma classe privada UML seriam gerados pelo menos 2 objectos para representar o equivalente em O3F, um objecto

para representar a classe propriamente dita e um objecto para representar a faceta que indica que a classe é privada. Como já foi referido, esta biblioteca faz parte da interface de acesso ao servidor e deve ser usada sempre que for necessário inserir ontologias no O3 Server. No final da conversão, o conjunto de objectos retornados pelas chamadas aos vários métodos *toO3F* constitui uma representação do modelo UML, de acordo com o modelo O3F. Estes objectos também se encontram devidamente ligados pelo que o objecto que representa a ontologia sabe quais os objectos que representam os seus vários elementos, por exemplo classes, associações e métodos, e vice-versa. Para além de representarem os vários elementos de um ontologia O3F, os objectos das classes Java da biblioteca *Hibernate* contêm também uma implementação do método *save* que é responsável pela sua inserção na base de dados do O3 Server. A segunda parte da conversão inicia-se quando o método *save* do objecto que representa a ontologia é invocado. A inserção na base de dados também é feita em forma de cascata, pelo que a chamada ao primeiro método *save* desencadeia a chamada dos restantes, inserindo assim todos os elementos na base de dados do servidor.

Capítulo 6 Avaliação

O objectivo deste capítulo é a avaliação do trabalho realizado na dissertação. O trabalho associado à dissertação deu origem aos seguintes resultados:

- Melhoramento e extensão do modelo O3F e da linguagem CO3L de representação de ontologias;
- Contribuição para o desenho do modelo relacional usado no servidor O3 Server;
- Mecanismo de extracção, em formato textual, na linguagem CO3L, de ontologias armazenadas no O3 Server; e
- Conversão de Diagramas UML de Classes e de Objectos em ontologias O3F (armazenadas no servidor O3 Server) e CO3L (em ficheiros de texto).

A avaliação rigorosa de um modelo ou de uma linguagem de representação sai do âmbito desta dissertação pois passaria pela comparação sistemática das suas propriedades formais com as de outros modelos. Além disso, esta dissertação não criou nem o modelo nem a linguagem de representação. Foram apenas introduzidos os melhoramentos e as extensões descritas no Capítulo 3. Em vez da via formal, a avaliação que apresentamos desta faceta do trabalho é de carácter meramente argumentativo (secção 6.1).

O modelo relacional do servidor O3 Server foi desenhado de acordo com a aplicação das regras de conversão mas não de uma forma automática. Antes da sua aplicação as regras foram ponderadas tendo em conta a simplificação e optimização do modelo relacional. Caso em que as regras tivessem sido aplicadas de uma forma estrita, em princípio seria possível desenhar um procedimento de validação rigoroso capaz de avaliar se o modelo relacional corresponde ao diagrama de classes de que partiu. Mesmo que tivessem sido aplicadas sistematicamente as regras de conversão de modelos de objectos em modelos relacionais, a validação rigorosa do procedimento de aplicação dessas regras sairia fora do âmbito desta dissertação. A avaliação deste

aspecto do trabalho da dissertação (secção 6.2) limita-se a mostrar que foi possível armazenar um número significativo de ontologias com grande variedade de elementos, as quais puderam depois ser extraídas correctamente. A avaliação deste aspecto foi feita em tandem com a avaliação do mecanismo de extracção discutido seguidamente.

A avaliação do mecanismo, implementado no servidor, para extrair representações textuais na linguagem CO3L das ontologias armazenadas no O3 Server (secção 6.2) mostra que, para todos os exemplos de diagramas testados, as ontologias extraídas do servidor são exactamente iguais às ontologias originalmente armazenadas. Este procedimento não constitui uma prova rigorosa mas constituí forte evidência de que o mecanismo de extracção está correcto.

A ferramenta UML2O3F, de conversão de diagramas UML de classes e de objectos em ontologias O3F foi submetida a dois tipos de avaliação. Por um lado, fez-se uma avaliação subjectiva da usabilidade e utilidade da ferramenta, recorrendo a um inquérito (secção 6.3). Por outro lado, procedeu-se à avaliação tão objectiva quanto possível da correcção das conversões através de um painel de especialistas em UML e em O3F/CO3L (secção 6.4). A avaliação subjectiva mostrou que 98% dos 54 inquiridos acha a ferramenta útil ou muito útil; que 97% dos 54 inquiridos achou a ferramenta fácil ou muito fácil de usar; e ainda que 98% acha que a correcção das conversões se situa entre o Correcto e o Muito Correcto. A avaliação objectiva realizada pelo painel de especialistas conclui que as conversões são correctas.

Finalmente, na última secção deste capítulo, apresentamos uma análise crítica do procedimento de avaliação objectiva da ferramenta de conversão de UML para O3F onde argumentamos que esse processo teve diversas deficiências, e concluímos resumidamente que, não obstante as dificuldades, os resultados da avaliação foram muito positivos.

6.1 Modelo O3F e Linguagem CO3L

No decorrer do presente trabalho foram feitas várias extensões e melhoramentos no modelo O3F e na linguagem CO3L. Essas extensões e melhoramentos tinham o propósito de atingir dois objectivos:

O primeiro seria de tornar o O3F compatível com outras abordagens de descrições conceptuais, em particular com o UML, devido ao interesse de converter Diagramas de Classes (e também de Objectos) para ontologias O3F. O facto de vários diagramas de classes presentes em (Ramos, 2009) terem sido integralmente convertidos de forma correcta para ontologias O3F constitui evidência de que este objectivo foi alcançado com sucesso. Para além dos diagramas presentes em (Ramos, 2009) foram ainda convertidos com sucesso outros diagramas com elementos mais variados como por exemplo relações de dependência e objectos, o que reforça a convicção de compatibilidade entre o UML e o O3F.

Não se pretende que o O3F seja apenas mais um modelo que permite modelar as mesmas coisas que outras tecnologias de modelação conceptual. Por essa razão o segundo objectivo das extensões e melhoramentos feitos é de possibilitar ao O3F expressar conhecimento que outras abordagens não conseguem. Destaca-se a capacidade de representar acções que nenhuma outra abordagem consegue representar; a capacidade de representar métodos funcionais, relacionais e de acção algo que não pode ser representado nem no OWL nem no Ontolingua; ou ainda a possibilidade de representar predicados, métodos relacionais, e funções (para além das já referidas acções), uma vantagem em relação ao UML.

Apesar de não constituir nenhuma prova formal rigorosa, o facto de ambos estes objectivos terem sido atingidos são argumentos que sugerem que o modelo O3F resultante das melhorias e extensões introduzias nesta dissertação é um bom modelo e com muito boas capacidades de modelação tal como já foi mais profundamente analisado nos capítulos 2 e 3. Além do mais, o modelo O3F que contempla as modificações referidas consegue representar elementos que a antiga versão não conseguia como por exemplo, hierarquias, objectos, relações de dependência e associações unidireccionais.

6.2 Modelo relacional, armazenamento e extracção de ontologias do servidor

Não existe também nenhuma demonstração formal para o correcto funcionamento do mecanismo que extrai as ontologias presentes no O3 Server. O método usado para avaliar este mecanismo é bastante mais simples, porém constitui uma prova forte do

seu correcto funcionamento. Para avaliar o extractor elaborou-se um pequeno programa em *shell script* (baseado no comando *diff,* que devolve a diferença entre dois ficheiros de texto) para comparar a ontologia no formato CO3L antes de ser inserida no servidor e depois de ser extraída. Em todas as experiências realizadas verificou-se uma igualdade de 100% (ignorado questões de espaçamento e indentação).

O correcto funcionamento do extractor é também uma boa prova de que o modelo relacional da base de dados do O3 Server é correcto pois se as ontologias extraídas são exactamente iguais às ontologias antes de serem armazenadas, isto significa que o modelo relacional permite representar qualquer ontologia que se possa conceber em O3F ou escrever em CO3L.

6.3 Avaliação Subjectiva da Conversão de UML para O3F

A avaliação subjectiva foi realizada por via de um inquérito, sendo necessário que os participantes experimentassem a ferramenta UML2O3F. Cada participante podia desenhar o seu próprio diagrama de classes ou usar os vários disponibilizados para a experimentação da ferramenta.

O inquérito subjectivo era constituído por três perguntas onde, numa escala de 1 (mau) a 4 (bom), os inquiridos avaliaram a qualidade geral das conversões, a utilidade da ferramenta, e a sua usabilidade. No final era ainda deixado espaço para comentários e/ou sugestões.

O inquérito foi respondido, através de um sítio na Internet com toda a informação e elementos necessários (ferramenta UML2O3F, instruções, diagramas para teste, etc.), por um total de 54 participantes, na sua maioria alunos do ISCTE-IUL e também alguns docentes.

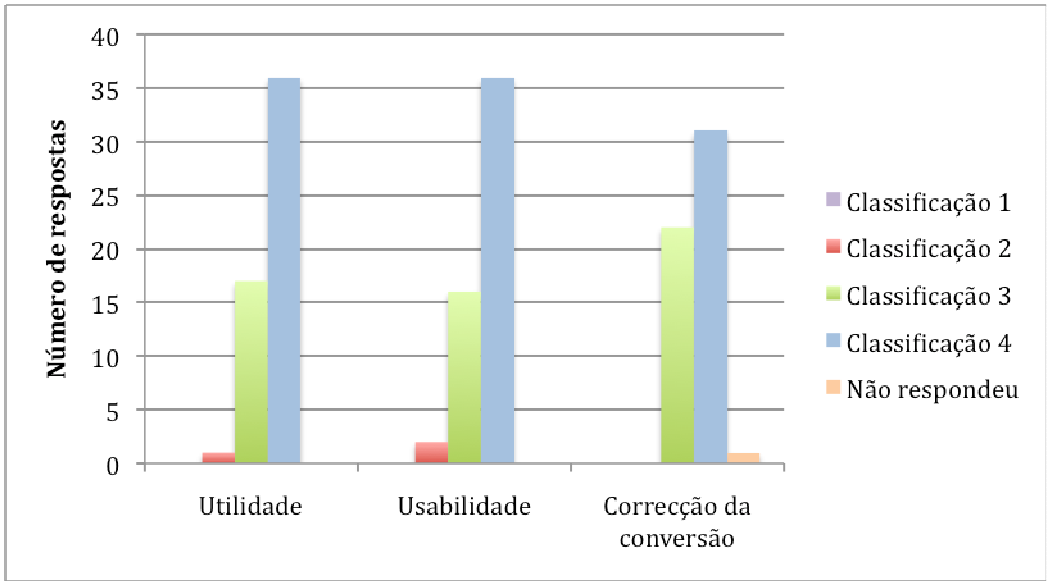


Figura 14 - Resultados do inquérito subjectivo

No que diz respeito a correcção global percebida da conversão, dos 54 participantes, 57% classificaram a correcção da conversão com o valor máximo. Dos restantes, 41% classificaram a correção global percebida com o valor 3. 2% não respondeu. Nenhum participante classificou a correcção geral da conversão com valor abaixo de 3.

67% dos participantes classificaram a utilidade e usabilidade da ferramenta UML2O3F com valor 4. Os restantes inquiridos classifcaram estes itens com valor 3 excepto 3% dos participantes que classificaram a usabilidade com valor 2 e 2% que classificaram a utilidade com o mesmo valor.

Em geral os resultados foram bastantes positivos em todos os campos (correcção da conversão, utilidade e usabilidade) o que demonstra que, para além de a ferramenta desempenhar bem a sua função de conversão, é considerada útil e bastante simples de utilizar.

O inquérito subjectivo pedia também aos participantes que contribuíssem com comentários e sugestões em relação à aplicação UML2O3F. Muitos dos inquiridos referiram que gostariam de ver uma integração do UML2O3F com o CO3L Edit (Correia, 2010). Em resposta a essas sugestões dos inquiridos, os autores de ambas as aplicações decidiram criar um IDE (*Integrated Development Environment*) que centralize todas as ferramentas úteis ao desenvolvimento de ontologias O3F.

Este IDE poderá também ser expandido no futuro como novas funcionalidades como por exemplo um conversor de O3F para UML, conversores entre OWL e O3F/CO3L, e um gerador de especificações de ontologias em JADE.

6.4 Avaliação Objectiva da Conversão de UML para O3F

A correcção de uma conversão entre dois modelos de representação formal não é uma questão de opinião. Ou a conversão é correcta ou é incorrecta. No entanto, a avaliação da correcção das conversões efectuadas pela ferramenta UML2O3F, além de pressupor a existência das semânticas rigorosas dos modelos alvo e objecto, envolveria a aplicação de técnicas de validação da teoria da computação, as quais saem do âmbito do próprio mestrado em Engenharia Informática, em particular do âmbito desta dissertação. Em vez da validação por via formal, optamos por recorrer a um painel de especialistas em UML e em O3F/CO3L. Os participantes desse painel de especialistas foram seleccionados com base na avaliação dos seus conhecimentos através das suas respostas a perguntas de controlo integradas no mesmo inquérito que serviu de avaliação da qualidade das conversões efectuadas pela ferramenta UML2O3F.

Oito dos nove diagramas objecto do inquérito foram extraídos de um conjunto de exercícios elaborados pelo Prof. Pedro Ramos (Ramos, 2009). Este inquérito estava disponível em três versões, cada uma das quais visando a conversão de três diagramas de classes e objectos, totalizando nove conversões. A organização do inquérito em três versões diferentes teve como único propósito diminuir o trabalho de cada participante, o qual respondeu apenas a uma única versão. Por outro lado, procurou-se que as quantidade e complexidade do trabalho de análise necessário a responder aos inquéritos fossem aproximadamente iguais.

Cada participante tinha de responder, numa escala de 1 (mau) a 4 (bom), em que medida as ontologias geradas representavam correcta e totalmente a informação presente nos diagramas de classes e objectos. Na avaliação objectiva não era necessário que os participantes experimentassem o UML2O3F, apenas era necessário avaliar as conversões disponíveis no inquérito.

Além das perguntas relativas à correcção da conversão, cada diagrama e a conversão correspondente foram associados a uma pergunta de controlo. Estas perguntas de controlo questionavam o participante sobre a informação presente nos diagramas UML e serviam o propósito de avaliar a sua prestação no inquérito e conhecimento sobre UML e CO3L.

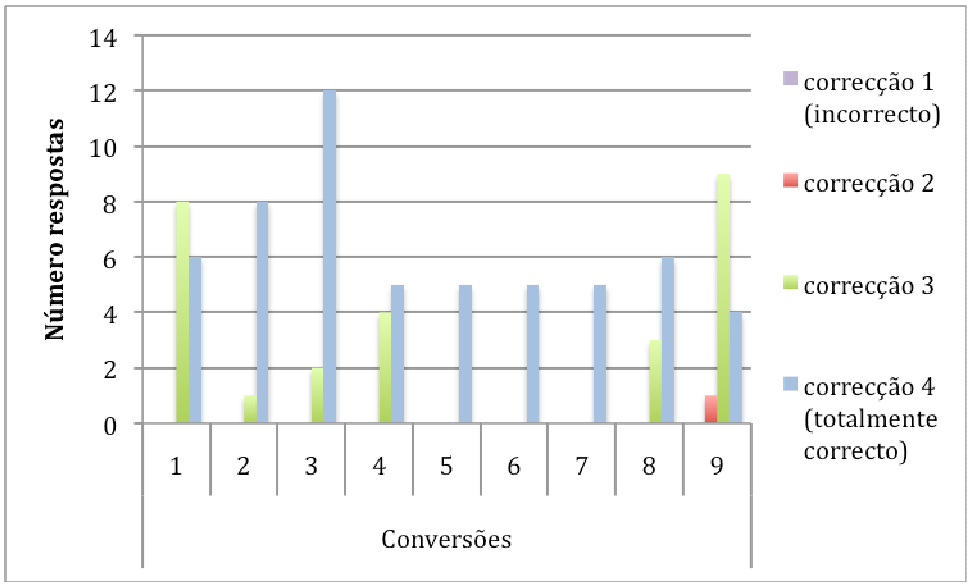


Figura 15 - Resultados das conversões presentes nos inquéritos objectivos

O grupo de inquiridos era constituído por um total de 40 alunos do terceiro ano das licenciaturas Engenharia Informática e Informática e Gestão de Empresas do Instituto Superior da Ciência do Trabalho e da Empresa (ISCTE) que, numa sessão organizada pelo Professor Luís Botelho, responderam aos inquéritos. Adicionalmente foram ainda consideradas mais 10 respostas, obtidas através de um sítio na Internet, de alunos que não compareceram na sessão presencial. No total 50 alunos participaram na avaliação objectiva. Tendo em conta as respostas às perguntas de controlo, foi seleccionado um painel de 28 especialistas formado exclusivamente por participantes que responderam correctamente a todas as perguntas de controlo. Todas as conversões foram avaliadas por um número mínimo de 5 participantes especialistas. 14 especialistas responderam à versão 1 do inquérito (diagramas e conversões 1, 3 e 9), 9 especialistas responderam à versão 2 (diagramas e conversões 2, 4, e 8), e por fim 5 especialistas responderam à versão 3 (diagramas e conversões 5, 6 e 7).

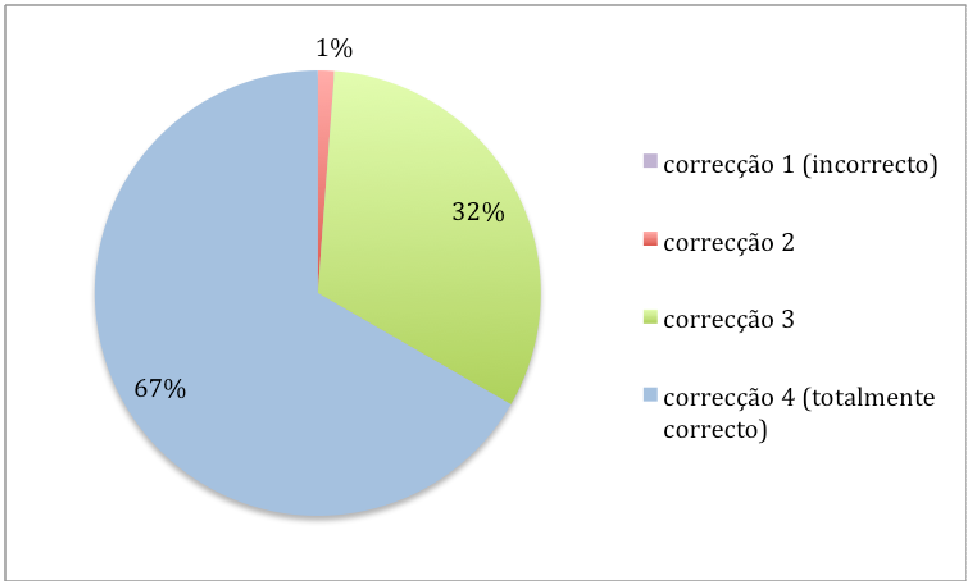


Figura 16 - Resultados gerais das conversões

Os resultados da avaliação das conversões presentes nas três versões do inquérito foram bastante positivos. A qualidade de todas as conversões excepto a número 9 foi classificada apenas com 3 e 4. Um dos participantes classificou a qualidade da conversão do diagrama 9 com 2. Todos os outros atribuíram uma qualidade de 3 ou 4. Talvez a causa para que a conversão 9 tenha recebido uma classificação de 2 esteja relacionada com as suas maiores extensão e complexidade. No entanto, esta hipótese é meramente especulativa.

Exceptuando as conversões 1 e 9, todas as outras apresentam um maior número de classificações 4 do que 3. As conversões 4, 5 e 6 foram as mais bem classificadas apresentando unicamente classificações de 4

Conclui-se ainda que 99% das conversões foram classificadas como boas, dado que a maior parte delas (67%) foram classificadas com a nota máxima (4) e 32% foram classificadas com a segunda melhor nota (3). Apenas 1% das conversões foram classificasas com qualidade 2 não tendo havido classificações mais baixas que esta.

6.5 Análise da Avaliação

Apesar dos resultados positivos, existe desconforto em relação ao método de avaliação objectiva da conversão de UML para O3F. O painel de especialistas deveria

ser constituído por pessoas cujos conhecimentos de UML e de O3F não deixasse qualquer dúvida. Seria também preciso formar equipas com possibilidade prática e motivação para fazer a avaliação. Infelizmente, não foi possível identificar essas pessoas de forma óbvia. Haveria no Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL reputados especialistas de UML. No entanto, os únicos especialistas de O3F são o orientador e o autor desta dissertação e ainda o Carlos Correia, autor da outra dissertação sobre O3F (Correia, 2010), sob a mesma orientação. Simples critérios de isenção ditam o afastamento destas pessoas do processo de avaliação.

O procedimento usado para a avaliação objectiva recorreu a perguntas de controlo feitas aos inquiridos para avaliar os seus conhecimentos sobre UML e O3F. Só integraram o painel de especialistas aqueles inquiridos que responderam correctamente a todas as perguntas de controlo. A principal razão do desconforto com este procedimento de constituição do painel de especialistas é que seria preciso mais do que apenas o uso de perguntas de controlo para determinar quem tem conhecimento suficiente de UML e de O3F para integrar o painel. Um tal processo, para além de poder afastar os eventuais participantes, envolveria recursos materiais e de tempo incompatíveis com a dissertação.

A segunda fonte de desconforto com o procedimento de constituição do painel é que as respostas às perguntas de controlo foram avaliadas pelo próprio autor da dissertação, o que resulta quase num paradoxo: O autor da dissertação está em posição de escolher os "especialistas", mas não está em posição de ser ele próprio a efectuar a avaliação.

A impossibilidade, quase sempre real, de recorrer a painéis de verdadeiros especialistas lança algumas dúvidas sobre a viabilidade de efectuar este tipo de avaliação em muitas das obras de engenharia desenvolvidas no âmbito de dissertações de mestrado.

Apesar de todos os obstáculos referidos a avaliação que foi possível realizar mostrou resultados positivos e encorajadores. Mesmo sendo uma avaliação difícil de realizar os resultados mostraram que muitos participantes perceberam e aprovaram as conversões de UML para O3F.

Os resultados de ambas as avaliações objectiva e subjectiva foram muito positivos. Na avaliação subjectiva, todos os que experimentaram a aplicação UML2O3F aprovaram a qualidade das conversões e consideram a aplicação útil e fácil de utilizar. Além de cumprir todos os requisitos e a especificação feita no Capítulo 4 , foi considerado pelos utilizadores que o UML2O3F realizava a leitura dos diagramas e as respectivas conversões quase instantaneamente, algo que foi difícil de conseguir dada a grande quantidade de relações e dependências que podem existir num Diagrama UML. A avaliação objectiva pretendia apenas avaliar a correcção de um grupo definido de conversões mas que também foram consideradas como correctas pela grande maioria dos participantes.

Capítulo 7 Conclusão

O trabalho realizado no âmbito desta dissertação, além de ter melhorado e estendido o modelo O3F e a sua linguagem de representação textual CO3L, desenvolveu uma ferramenta de conversão de UML para O3F e contribuiu para o desenho e implementação do servidor de ontologias O3 Server. Implementou também um mecanismo no O3 Server que permite a extracção, na linguagem CO3L, de qualquer ontologia mantida no servidor. A ferramenta de conversão está agora disponível a qualquer potencial utilizador, através da Internet, no endereço http://dcti.iscte.pt/O3F/ A disponibilização pública da ferramenta de conversão e do servidor, além de facilitar o desenvolvimento e partilha de ontologias, incluindo para um público oriundo de uma outra cultura de modelação – o mundo UML – introduz a fundamental potencialidade de contribuir decisivamente para o desenvolvimento do próprio O3F e para a expansão da sua comunidade de utilizadores, uma tecnologia totalmente portuguesa, criada e utilizada no Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL, Instituto Universitário de Lisboa.

Na secção 7.1 são revistos todos os objectivos propostos e atingidos resumindo o trabalho desenvolvido na presente dissertação. A secção 7.2 aponta que as principais linhas de desenvolvimento futuro do trabalho sobre O3F são melhoramentos da ferramenta de conversão, desenvolvimento de outros conversores, representação de axiomas e inferência.

7.1 Objectivos

Tal como referido ao longo de toda a dissertação, existem dois objectivos principais: O melhoramento e extensão do O3F e consequentemente do CO3L, e a criação de ferramentas O3F. A criação de ferramentas O3F subdivide-se em três objectivos distintos: A criação da ferramenta UML2O3F, que permite realizar a conversão de diagramas UML em ontologias O3F; contribuição para o desenho e a criação do O3 Server, um servidor cuja principal função é o armazenamento e disponibilização

públicas de ontologias; e por último a construção de um mecanismo de extracção, em CO3L, de ontologias O3F mantidas no O3 Server. Todos os quatro objectivos propostos foram atingidos e completados com sucesso.

No que diz respeito ao objectivo de melhorar o O3F e o CO3L, todas as pequenas falhas e aspectos do modelo e da linguagem com os quais não se estava completamente satisfeito foram corrigidas através do redesenho de alguns elementos ou através da criação de novos. Depois da introdução destas melhorias, o O3F e o CO3L ficaram mais ricos, com maior expressividade conseguindo agora representar elementos como instâncias, hierarquias e relações de dependência, para além dos elementos que já era capaz de descrever. Um dos princípios que norteou a introdução de melhorias e extensões no modelo e na sua linguagem foi a compatibilidade entre o O3F, e UML e o OWL. A secção 6.1 mostra que a compatibilidade com o UML foi conseguida com sucesso. A próxima secção propõe o desenho de ferramentas de conversão entre OWL e O3F como um dos principais objectivos do trabalho em O3F.

Além disto, na secção 6.2 conclui-se (embora de forma argumentativa) que o modelo relacional da base de dados do O3 Server consegue armazenar qualquer ontologia O3F e que o extractor desenvolvido consegue extrair qualquer ontologia presente nessa mesma base de dados e exibi-la, na linguagem CO3L.

Quanto às ferramentas O3F criadas, até à data não existia nenhum meio automático que possibilitasse a conversão de UML para O3F. A ferramenta UML2O3F veio colmatar essa lacuna, servindo de ponte entre o vasto mundo dos especialistas do UML e o mundo O3F. Tal pode ser visto como um facto importante na expansão e disseminação do O3F e do CO3L. Qualquer conhecimento expresso sob a forma de um Diagrama de Classes e/ou Objectos pode agora ser automaticamente expresso sob a forma de uma ontologia O3F. É também possível tirar partido de algumas ferramentas para construção de modelos UML, em especial a ferramenta BOUML (http://bouml.free.fr/), para criar ontologias O3F. Destaca-se a possibilidade de criar uma ontologia O3F de uma forma gráfica usando construções que são já conhecidas por qualquer utilizador de UML.

Graças à construção do O3 Server (realizada em conjunto com outra dissertação (Correia 2010)) , as ontologias O3F podem ser armazenadas e partilhadas. Isto é um ponto fundamental para que vários peritos possam contribuir para a construção de

ontologias mais complexas e é também um primeiro passo no que diz respeito a partilha do conhecimento expresso nas ontologias O3F.

A extracção em CO3L de ontologias O3F mantidas no O3 Server permite que qualquer ferramenta que tenha compatibilidade com a tecnologia SQL Server possa de uma maneira muito simples (em apenas um par de instruções) obter, na linguagem CO3L, qualquer ontologia O3F mantida no O3 Server, independentemente da forma como ela terá sido originalmente especificada (e.g., em CO3L ou em UML).

A avaliação descrita no Capítulo 6 permite ainda constatar a satisfação geral por parte dos especialistas em UML e O3F com a correcção das conversões do UML2O3F. Além da satisfação revelada com a correcção das conversões, a aplicação do inquérito mostra ainda a utilidade e usabilidade do conversor (secções 6.3 e 6.4).

7.2 Trabalho Futuro

Ainda que todos os objectivos propostos tenham sido alcançados com sucesso existem sempre melhorias que podem ser feitas. No caso da ferramenta UML2O3F, embora esta ferramenta consiga realizar com sucesso a conversão para O3F/CO3L de qualquer Diagrama de Classes e/ou Objectos UML, os diagramas a serem convertidos devem ser preferencialmente construídos com recurso à ferramenta BOUML (http://bouml.free.fr/). A palavra preferencialmente significa que esta ferramenta (BOUML) foi a mais usada para a construção dos modelos UML que serviram de teste ao mecanismo de conversão e é portanto a que tem uma maior taxa de compatibilidade com o conversor. No entanto foram também realizados testes com sucesso em outras ferramentas como por exemplo *Enterprise Architect* (http://www.sparxsystems.com/) e *Altova UModel* (www.altova.com/umodel.html). Escolheu-se o BOUML como ferramenta preferencial por ser uma ferramenta grátis, multi-plataforma, com um vasto número de funcionalidades e que é constantemente actualizada e melhorada pela sua equipa de desenvolvimento. A razão pela qual não se garante uma total compatibilidade com todas as ferramentas UML que tenham a capacidade de gerar ficheiros XMI 2.1 deve-se às diferenças existentes entre os ficheiros XMI gerados por uma ferramenta e os gerados por outras. Pretende-se como trabalho futuro que o UML2O3F ofereça compatibilidade com cada vez mais ferramentas.

Outro ponto de trabalho futuro, o qual está já em desenvolvimento, é a integração do UML2O3F com a ferramenta CO3L Edit (Correia, 2010) num único ambiente de desenvolvimento integrado, tal como sugerido por muitas das pessoas que testaram ambas as ferramentas.

Uma vez que o O3F permite representar muito mais informação ontológica do que os Diagramas de Classes e Objectos UML seria também interessante que o UML2O3F permitisse conversões a partir de UML estendido, muito à semelhança do que foi referido no Capítulo 2 acerca da conversão de UML estendido para OWL (Gašević et al., 2004) e (Leinhos, 2006). Deste modo seria possível representar com UML (recorrendo a construções estendidas) elementos que existem apenas no O3F como por exemplo facetas, acções e métodos relacionais. Estas construções seriam definidas num perfil de UML para O3F. Os modelos UML construídos com este perfil seriam depois convertidos para ontologias O3F/CO3L.

Por fim, seria também interessante realizar a conversão oposta à proporcionada pelo UML2O3F, ou seja, a conversão de O3F para UML. Esta conversão apresenta por um lado algumas dificuldades extra no que diz respeito à organização espacial dos diagramas gerados a partir de uma ontologia O3F. Por outro lado, apresenta também algumas facilidades que provêm do trabalho já realizado nesta dissertação. Uma das facilidades reside na leitura das ontologias O3F a serem convertidas uma vez que estão armazenadas no O3 Server. Outra facilidade é o facto de algumas regras de equivalências entre elementos UML e O3F terem sido já definidas nesta dissertação.

O mecanismo de extracção de ontologias, poderá também evoluir e para além de proporcionar a extracção completa de uma ontologia, permitir uma consulta individualizada dos elementos de uma ontologia como por exemplo consultar todas as classes de uma ontologia, ou todos os objectos de uma determinada classe, ou todas as associações que tenham uma dada classe como argumento, etc.

Além do trabalho futuro mais relacionado com as contribuições concretas desta dissertação, apontam-se ainda direcções de desenvolvimento do O3F em geral:

- Conversores OWL O3F e vice versa - Uma ou mais ferramentas que funcionem de modo semelhante à desenvolvida no âmbito desta dissertação. A conversão entre OWL e O3F tem especial interesse uma vez que o OWL é a

linguagem de representação de ontologias mais usada e disseminada hoje em dia.

- Geradores de ontologias JADE a partir do O3F – Embora não seja uma linguagem de representação de ontologias, o JADE (*Java Agent DEvelopment Framework*) (http://jade.tilab.com/) é uma plataforma que facilita a implementação e o funcionamento de sistemas multi-agente compatíveis com as especificações da FIPA (*Foundation for Intelligent Physical Agents*) (www.fipa.org). Por esta razão seria extremamente valiosa e útil uma ferramenta com a capacidade de transformar as ontologias O3F em conjuntos de comandos JADE que resultam na especificação de ontologias, as quais são usadas na interpretação das mensagens trocadas entre agentes.

- Representação de axiomas em CO3L – A representação de axiomas em CO3L é um assunto que já está a ser pensado há algum tempo. A existência de axiomas adicionaria expressividade à linguagem, permitindo por exemplo especificar definições e impor restrições aos elementos da ontologia e às relações entre eles.

- Inferência - A inferência é um dos pontos de desenvolvimento de trabalho futuro mais importantes. Ao possibilitarem a representação de conhecimento, as ontologias são extremamente importantes no que toca a capacidade de um sistema poder inferir conhecimento e de raciocinar automaticamente. A inferência estaria intimamente ligada à possibilidade de representação de axiomas, mas também a propriedades intrínsecas do modelo O3F, como por exemplo a herança, e as relações existentes entre operadores e métodos. Relacionado com este assunto está também uma possível integração do O3F com a linguagem de programação Prolog (Colmerauer & Roussel, 1996).

Tanto o trabalho desenvolvido nesta dissertação como o desenvolvido em (Correia, 2010) contribuíram para que o modelo O3F e a linguagem CO3L chegassem a um estado mais maduro e mais estável, o que pode ser visto como um incentivo para o desenvolvimento de todos os tópicos de trabalho futuro referidos ao logo desta secção.

Referências

Alberts, L. K. (1994). Ymir: A Sharable Ontology For The Formal Representation Of Engineering-Design Knowledge. In *Formal Design Methods For Cad, Ifip Transactions*, 3-32.

Antoniou, G., & Van Harmelen, F. (2003). Web Ontology Language: OWL. In *Handbook On Ontologies In Information Systems,* 67-92. Disponível *online* em http://www.cs.vu.nl/~frankh/postscript/OntoHandbook03OWL.pdf

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Letkowski, J., & Emery, P. (2002). Extending the Unified Modeling Language for Ontology Development. In *Software and Systems Modeling*, 142-156.

Booch, G. (1991). *Object-Oriented Analysis and Design with Applications* (1ª ed.). Addison-Wesley Professional.

Botelho, L., & Ramos, P. (2003). CO3L: Compact O3F language. *In Workshop On Ontologies In Agent Systems, Aamas 03 - Autonomous Agents And Multi Agent Systems*. Disponível *online* em http://iscte.pt/~luis/papers/RamosBotelho-CO3L.pdf

Castelli, C. (2009). XMI, a not so standard exchange format. PragmaDev. Disponível *online* em http://www.pragmadev.net/news/XMI_fm.pdf

Colmerauer, A., & Roussel, P. (1996). The birth of Prolog. In *History of programming languages,* 331-367. Addison-Wesley Professional.

Correia, C. (2010). *Ontologias O3F: Um Servidor e um Editor CO3L*. Dissertação de Mestrado. ISCTE-IUL (Instituto Universitário de Lisboa), Lisboa.

Cranefield, S. (2001). Networked Knowledge Representation and Exchange using UML and RDF. In *Journal of Digital Information*. Disponível *online* em http://jodi.ecs.soton.ac.uk/Articles/v01/i08/Cranefield/

Djurić, D., Gašević, D., Devedžić, V., & Damjanović, V. (2005). A UML Profile for OWL Ontologies. In *Model Driven Architecture*, 204-219. Disponível *online* em http://dx.doi.org/10.1007/11538097_14

Falbo, A. (1998). *Integração de Conhecimento em um Ambiente de Engenharia de Software*. Tese de Doutoramento. Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Falkovych, K. (2002). *Ontology Extraction from UML Diagrams.* Dissertação de Mestrado. Vrije Universiteit, Amsterdão.

Falkovych, K., Sabou, M., & Stuckenschmidt, H. (2003). UML for the Semantic Web: Transformation-Based Approaches. In *Knowl- edge Transformation for the Semantic Web*, 92-106

Farquhar, A., Fikes, R., & Rice, J. (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *International Journal Of Human-Computer Studies*,

Gašević, D., Djurić, D., Devedžić, V., & Damjanović, V. (2004). Converting UML to OWL ontologies. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 488-489. Nova Iorque

Genesereth, M., & Fikes, R. (1992). Knowledge Interchange Format: Version 3.0: Reference Manual. Logic Group Technical Report. Stanford University, California. Dispoível *online* em http://logic.stanford.edu/kif/Hypertext/kif-manual.html

Gennari, J. H., Oliver, D. E., Pratt, W., Rice, J., & Musen, M. A. (1995). A web-based architecture for a medical vocabulary server. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, 275-279.

Gómez-Pérez, A., Fernández, M., & Juristo, N. (1996). Towards a Method to Conceptualize Domain Ontologies. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, 33-4. Stanford.

Gruber, T. (1993). A translation approach to portable ontology specifications. In *Knowledge Acquisition*, 199-220.

Humphreys, L., & Lindberg, A. (1993). The UMLS Project: Making The Conceptual Connection Between Users And The Information They Need. In *Bulletin of the Medical Library Association*,, 170-177.

Jacobson, I. (1992). *Object Oriented Software Engineering: A Use Case Driven Approach (Revised.)*. Addison-Wesley Professional. Nova Iorque

Kabilan, V., & Johannesson, P. (2004). UML for Ontology Modelling and Interoperability. In *CAiSE Workshops (3)*, 349-354. Riga

Karp, P. (1993). A qualitative biochemistry and its application to the regulation of the tryptophan operon. In *Artificial intelligence and molecular biology*. American Association for Artificial Intelligence, 289-324. Disponível *online* em http://portal.acm.org/citation.cfm?id=166472

Knublauch, H. (2003). Case Study: Using Protege to Convert the Travel Ontology to UML and OWL. In *EON*. Florida

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., & Smith, J. (2002). UML For Ontology Development. In *The Knowledge Engineering Review,* 61-64.

Krivov, S., Williams, R., & Villa, F. (2007). GrOWL: A Tool For Visualization And Editing Of OWL Ontologies. In *Web Semantics: Science, Services and Agents on the World Wide Web*, 54-57.

Leinhos, S. (2006). *OWL Ontologieextraktion und -modellierung auf der Basis von UML Klassendiagrammen.* Dissertação de Mestrado. Universität der Bundeswehr München, Neubiberg.

MacGregor, R. (1991). Inside the LOOM description classifier. In *SIGART Bull*, 88-92.

McCarthy, J. (1962). *LISP 1.5 Programmer's Manual*. The MIT Press.

Mota, L., Botelho, L., Mendes, H., & Lopes, A. (2003). O3F: An Object Oriented Ontology Framework. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 639-646. Australia

OMG. (2006a). *MOF 2.0. Meta-Object Facility 2.0*. Disponível *online* em http://www.omg.org/spec/MOF/2.0/

OMG. (2006b). *OCL 2.0. Object Constraint Language*. Disponível *online* em http://www.omg.org/spec/OCL/2.0/

OMG. (2007). *XML Metadata Interchange. XML Metadata Interchange 2.1.1*. Disponível *online* em http://www.omg.org/technology/documents/formal/xmi.htm

OMG. (2008). *QVT 1.0. Query/View/Transformation*. Disponível *online* em http://www.omg.org/spec/QVT/1.0/

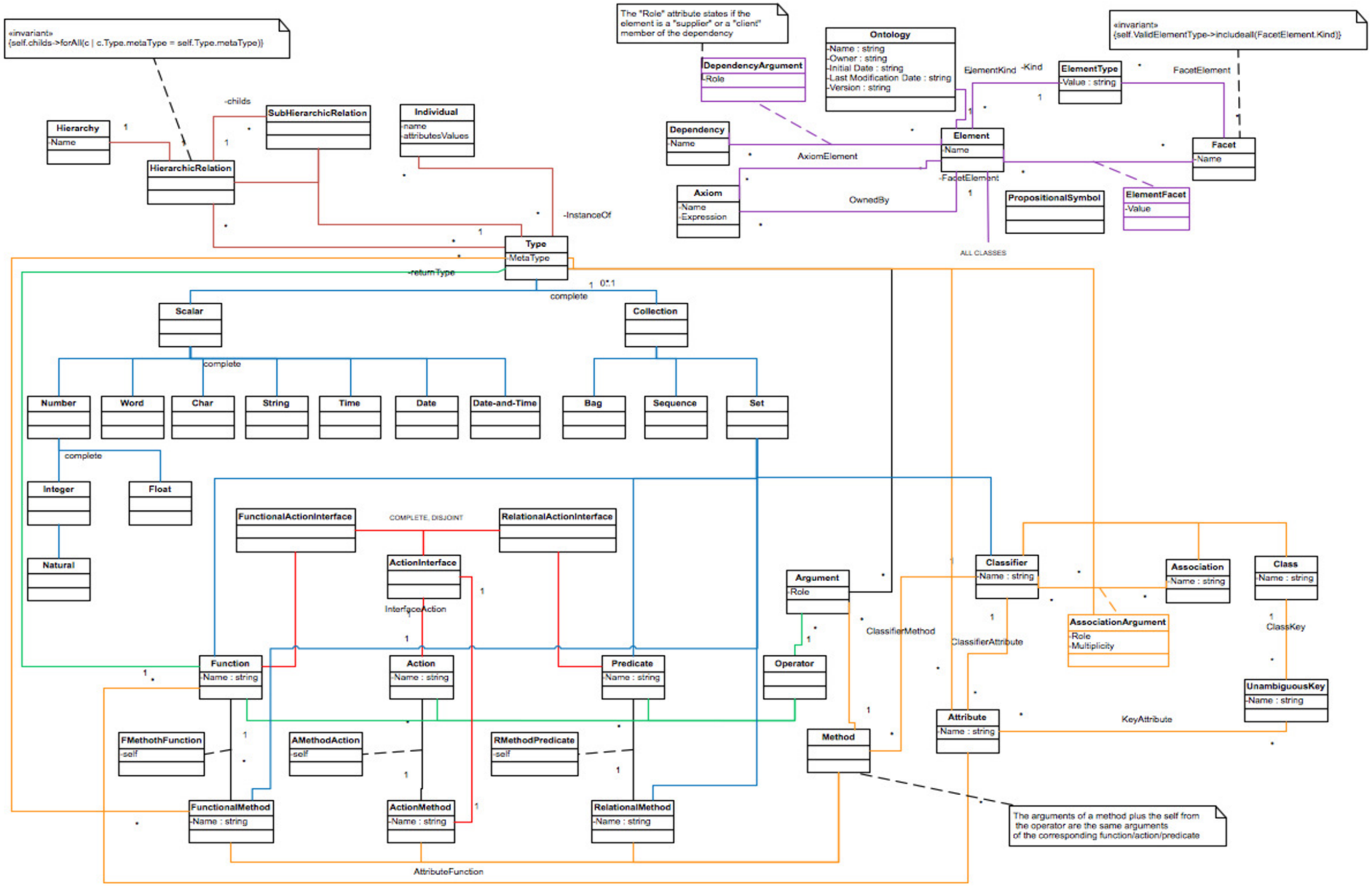
OMG. (2009). *ODM 1.0. Ontology Definition Metamodel*. Disponível *online* em from

http://www.omg.org/spec/ODM/1.0/
Ramos, P. (2009). *Colecção de Exercicios de Diagramas de Classes*. ISCTE - IUL(Instituto Universitário de Lisboa).
Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). Object Oriented Modeling And Design. In*, Book Distribution Center. Prentice Hall.* Nova Iorque.
Singh, N., & Genesereth, M. (1991). Epikit: A Library Of Subroutines Supporting Declarative Representations And Reasoning. In *SIGART Bull*, 143-151.
Varejão, F. M. (1999). *DORPA: Uma Ontologia De Design Que Integra Requisitos, Artefatos E Processos*. Tese de Doutoramento. Pontifícial Universidade Católica, Rio de Janeiro.
Vet, P., Van Der, P., Speel, P., & Mars, N. (1994). The Plinius Ontology Of Ceramic Materials. *Proceedings Of Ecai94's Workshop On Comparison Of Implemented Ontologies*, 8-12.
W3C. (2001) *DAML+OIL Reference Description*. Disponível *online* em http://www.w3.org/TR/daml+oil-reference
W3C. (2004a). *OWL Web Ontology Language Reference*. Disponível *online* em http://www.w3.org/TR/owl-ref/
W3C. (2004b). *RDF/XML Syntax Specification (Revised)*. Disponível *online* em http://www.w3.org/TR/REC-rdf-syntax/
W3C. (2004c). *RDF Vocabulary Description Language 1.0: RDF Schema*. Disponível *online* em http://www.w3.org/TR/rdf-schema/
W3C. (2009). *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. Disponível *online* em http://www.w3.org/TR/owl2-syntax/

Anexo A

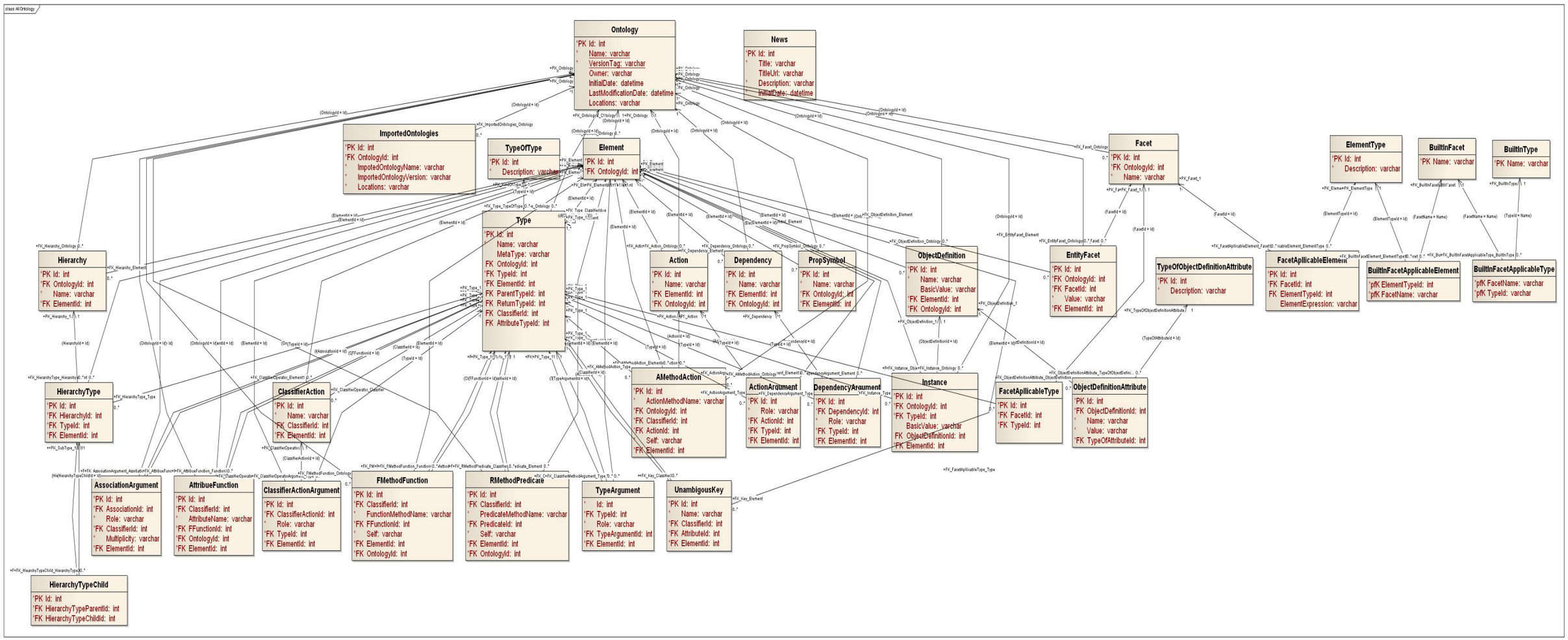
Diagrama de Classes do Modelo O3F

Static Structure



Anexo B

Modelo Relacional da Base de Dados do O3 Server


class All Ontology
Ontology
'PK Id: int
Name: varchar
VersionTag: varchar
Owner: varchar
InitialDate: datetime
LastModificationDate: datetime
Locations: varchar
News
'PK Id: int
Title: varchar
TitleUrl: varchar
Description: varchar
InitialDate: datetime
ImportedOntologies
'PK Id: int
'FK OntologyId: int
ImpotedOntologyName: varchar
ImportedOntologyVersion: varchar
Locations: varchar
TypeOfType
'PK Id: int
Description: varchar
Element
'PK Id: int
'FK OntologyId: int
Facet
'PK Id: int
'FK OntologyId: int
Name: varchar
ElementType
'PK Id: int
Description: varchar
BuiltInFacet
'PK Name: varchar
BuiltInType
'PK Name: varchar
Type
'PK Id: int
Name: varchar
MetaType: varchar
FK OntologyId: int
'FK TypeId: int
'FK ElementId: int
FK ParentTypeId: int
FK ReturnTypeId: int
FK ClassifierId: int
FK AttributeTypeId: int
Hierarchy
'PK Id: int
'FK OntologyId: int
Name: varchar
'FK ElementId: int
Action
'PK Id: int
Name: varchar
'FK ElementId: int
'FK OntologyId: int
Dependency
'PK Id: int
Name: varchar
'FK ElementId: int
'FK OntologyId: int
PropSymbol
'PK Id: int
Name: varchar
'FK OntologyId: int
'FK ElementId: int
ObjectDefinition
'PK Id: int
Name: varchar
BasicValue: varchar
'FK ElementId: int
'FK OntologyId: int
EntityFacet
'PK Id: int
'FK OntologyId: int
'FK FacetId: int
Value: varchar
'FK ElementId: int
TypeOfObjectDefinitionAttribute
'PK Id: int
Description: varchar
FacetApplicableElement
'PK Id: int
'FK FacetId: int
'FK ElementTypeId: int
ElementExpression: varchar
BuiltInFacetApplicableElement
'pfK ElementTypeId: int
'pfK FacetName: varchar
BuiltInFacetApplicableType
'pfK FacetName: varchar
'pfK TypeId: varchar
HierarchyType
'PK Id: int
'FK HierarchyId: int
'FK TypeId: int
'FK ElementId: int
ClassifierAction
'PK Id: int
Name: varchar
'FK ClassifierId: int
'FK ElementId: int
AMethodAction
'PK Id: int
ActionMethodName: varchar
'FK OntologyId: int
'FK ClassifierId: int
'FK ActionId: int
Self: varchar
'FK ElementId: int
ActionArgument
'PK Id: int
Role: varchar
'FK ActionId: int
'FK TypeId: int
'FK ElementId: int
DependencyArgument
'PK Id: int
'FK DependencyId: int
Role: varchar
'FK TypeId: int
'FK ElementId: int
Instance
'PK Id: int
'FK OntologyId: int
'FK TypeId: int
BasicValue: varchar
FK ObjectDefinitionId: int
'FK ElementId: int
FacetApplicableType
'PK Id: int
'FK FacetId: int
'FK TypeId: int
ObjectDefinitionAttribute
'PK Id: int
'FK ObjectDefinitionId: int
Name: varchar
Value: varchar
'FK TypeOfAttributeId: int
AssociationArgument
'PK Id: int
'FK AssociationId: int
Role: varchar
'FK ClassifierId: int
Multiplicity: varchar
'FK ElementId: int
AttribueFunction
'PK Id: int
'FK ClassifierId: int
AttributeName: varchar
'FK FFunctionId: int
'FK OntologyId: int
'FK ElementId: int
ClassifierActionArgument
'PK Id: int
'FK ClassifierActionId: int
Role: varchar
'FK TypeId: int
'FK ElementId: int
FMethodFunction
'PK Id: int
'FK ClassifierId: int
FunctionMethodName: varchar
'FK FFunctionId: int
Self: varchar
'FK ElementId: int
'FK OntologyId: int
RMethodPredicate
'PK Id: int
'FK ClassifierId: int
PredicateMethodName: varchar
'FK PredicateId: int
Self: varchar
'FK ElementId: int
'FK OntologyId: int
TypeArgument
Id: int
'FK TypeId: int
Role: varchar
'FK TypeArgumentId: int
'FK ElementId: int
UnambigousKey
'PK Id: int
Name: varchar
'FK ClassifierId: int
'FK AttributeId: int
'FK ElementId: int
HierarchyTypeChild
'PK Id: int
'FK HierarchyTypeParentId: int
'FK HierarchyTypeChildId: int