ISCTE IUL
Instituto Universitário de Lisboa

Departamento das Ciências e Tecnologias da Informação

ONTOLOGIAS O3F: UM SERVIDOR E UM EDITOR CO3L

Carlos Eduardo Fonseca Correia

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

Setembro de 2010

ISCTE IUL
Instituto Universitário de Lisboa

Departamento das Ciências e Tecnologias da Informação

ONTOLOGIAS O3F: UM SERVIDOR E UM EDITOR CO3L

Carlos Eduardo Fonseca Correia

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

Setembro de 2010

Resumo:

Esta dissertação debruça-se sobre o modelo O3F e a linguagem CO3L, sobre a construção de um servidor de ontologias O3F e ainda sobre a construção de um editor especializado na linguagem CO3L. Analisaram-se criticamente várias abordagens de representação de ontologias: Ontolingua, OWL, OWL2, UML e O3F/CO3L. Foram também analisadas ferramentas que dão suporte às anteriores linguagens, dando especial ênfase ao Protégé e ao Ontolingua Server.

Desta análise resultou que o modelo O3F é uma excelente opção para representar ontologias devido às várias vantagens que apresenta em relação às restantes abordagens. O modelo O3F é o único que permite modular um domínio de acordo com uma visão orientada a objectos ou com uma visão relacional do mundo, possibilitando ainda a integração de ambas.

A presente dissertação contribuiu com várias propostas de melhoria para modelo O3F e para linguagem CO3L, aumentando a sua expressividade e competitividade. Para possibilitar a real utilização deste modelo e linguagem foi criado o O3 Server, um servidor de ontologias que guarda e partilha as ontologias O3F a um nível global. A presente dissertação criou ainda o CO3L Edit, o primeiro editor especializado na linguagem CO3L que interage com o O3 Server para armazenar e extrair ontologias.

Os resultados produzidos na dissertação foram submetidos a avaliação que revelou evidência da correcção e abrangência das ferramentas computacionais, mostrando ainda a satisfação insofismável dos inquiridos com uma taxa de satisfação média de 92,5%.

Os sistemas computacionais estão publicamente acessíveis em http://dcti.iscte.pt/O3F/.

Palavras-chave: Representação de ontologias; Editor de ontologias; Servidor de ontologias

Abstract:

This dissertation focuses on the O3F model and the CO3L language, on building an O3F ontology server and on the construction of a specialized CO3L language editor. Some ontology representation approaches have been reviewed and analyzed: Ontolingua, OWL, OWL2, UML e O3F/CO3L. We also analyzed computational tools supporting those approaches, in particular and in more detail, Protégé and the Ontolingua Server.

This analysis showed that the O3F model is an excellent option for ontology representation due to several advantages that this model offers compared to other approaches. The O3F model is the only one that allows modeling a given domain, following an object-oriented paradigm or a functional and relational paradigm, allowing the integration of both.

This work has contributed with several proposals to improve the O3F model and the CO3L language, improving their expressiveness, thus making them more competitive. To enable the use of this model and language we have created O3 Server, a server that stores and shares ontologies globally. We have also developed CO3L Edit, the first CO3L language specialized editor that interacts with the O3 Server to store and extract ontologies.

The results produced in this dissertation were evaluated and revealed evidence of the correctness and completeness of software tools and showed the unmistakable satisfaction of users with an average satisfaction rate of 92.5%.

The computer systems are publicly accessible at http://dcti.iscte.pt/O3F/.

Key words: Ontology representation; Ontology editor; Ontology server

Glossário

ANTLR: *ANother Tool for Language Recognition***.** Uma *framework* de software aberto para construir interpretadores, compiladores, tradutores e carregadores.

CO3L Edit: Um editor de ontologias CO3L, cliente do O3 Server. O CO3L Edit foi desenvolvido no âmbito desta dissertação.

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

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

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

Hibernate: Projecto de software aberto de ORM para Java.

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

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 metamodelos.

O3 Server: Servidor de gestão e manutenção de ontologias O3F, parcialmente definido e implementado nesta dissertação.

O3F: *Object Oriented Ontology Framework*. Um modelo para descrição de ontologia segundo vários paradigmas.

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 metamodelo concordante com o MOF.

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

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

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

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

ORM: *Object-relational mapping*. É uma técnica de programação para converter dados entre bases de dados e linguagens de programação orientadas a objectos.

OWL: *Web Ontology Language.* Uma 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.

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

SWT: *The Standard Widget Toolkit.* Uma API de software aberto para construir interfaces gráficas em Java. Providencia um acesso eficiente e portável às interfaces gráficas do sistema operativo nativo onde a aplicação é executada.

UML2O3F: Conversor automático de diagramas de classes e objectos UML em ontologias O3F.

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

Agradecimentos

Ao Professor Luís Botelho pela excelente orientação científica e humana que deu no decorrer desta dissertação, pelo entusiasmo continuado e ainda pela excelente capacidade pedagógica e científica que detém.

Ao colega Jairo Avelar pela paciência e disponibilização contínua para discutir todos os aspectos comuns ao nosso trabalho.

À Rute Oliveira pelo apoio prestado no decorrer da dissertação e ainda pelo companheirismo e carinho demonstrados.

Aos docentes do ISCTE-IUL pelo excelente ensino que me proporcionaram.

Aos meus Pais pela motivação e apoio demonstrados ao longo de todos estes anos.

Índice

Lista de Figuras

Lista de Tabelas:

Capítulo 1 Introdução

A investigação em ontologias tem crescido e alargado horizontes na comunidade das ciências da computação. Originalmente, Ontologia era um ramo da Filosofia, mas actualmente as ontologias estão a ganhar um importante espaço nas mais diversas áreas de investigação como a representação do conhecimento, engenharia da linguagem, desenho de base de dados, modelação da informação, integração da informação, análise orientada a objectos, interrogação e extracção da informação, gestão do conhecimento e sistemas baseados em agentes (Guarino, 1998).

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

As ontologias não têm aplicação apenas em sistemas informáticos. A definição de ontologias torna-se útil sempre que seja necessário comunicar, educar e integrar conhecimento. É necessário que as pessoas e agentes consigam comunicar de uma forma precisa. De acordo com a definição mais bem aceite de ontologia, pelo menos no âmbito das ciências da computação, uma ontologia é uma representação formal de uma conceptualização de um domínio, definindo um vocabulário que permite a representação e a comunicação do conhecimento relativo ao domínio (Gruber, 1993).

A especificação de ontologias tem vantagens directas extremamente benéficas para o desenvolvimento de qualquer sistema ou actividade. Ao nível da comunicação, as ontologias podem ajudar as pessoas a raciocinar e a entender o domínio do conhecimento, actuam como uma referência para a obtenção do consenso numa comunidade sobre o vocabulário a ser usado nas suas interacções. A formalização do conhecimento elimina contradições e inconsistências, resultando idealmente numa especificação não ambígua. A formalização do conhecimento permitirá também a utilização de mecanismos de inferência para derivar novo conhecimento a partir do conhecimento previamente descrito. As ontologias podem ser reutilizadas ou estendidas por diferentes sistemas ou diferentes equipas.

Um dos principais problemas na utilização das ontologias está no facto de não existir um consenso relativamente a um modelo que melhor se adeqúe ao desenvolvimento de ontologias. É também necessário compreender que existem ontologias com diferentes propósitos, o que faz com que um dado modelo de representação de conhecimento seja mais apropriado para um dado tipo de ontologia e menos adequado para outro.

Para melhor rentabilizar a existência de ontologias, não basta representá-las formalmente em documentos locais, é necessário usar sistemas de informação capazes de suportar a gestão, o armazenamento de ontologias e a disponibilização de ontologias armazenadas a aplicações computacionais e mesmo a utilizadores humanos. Outras funcionalidades que podem estar disponíveis em sistemas de gestão de ontologias são a importação e exportação de ontologias, a avaliação das ontologias produzidas, indispensável para reduzir problemas ao nível da integração, o alinhamento de diferentes ontologias no mesmo domínio e a interrogação de ontologias idealmente associada a mecanismos de inferência.

1.1 Assunto e abordagem crítica

O assunto geral da presente dissertação é a representação e processamento de ontologias usando o modelo O3F (*Object Oriented Ontology Framework*) (Mota, Botelho, Mendes, & Lopes, 2003). Neste âmbito, a tese fará dois tipos de contribuição: a análise aprofundada e melhoria do modelo O3F e da linguagem CO3L (*Compact Object Oriented Ontology Language*) (Botelho & Ramos, 2003); e a construção de ferramentas computacionais que suportem este modelo e linguagem.

Para que seja possível melhorar o modelo O3F e a linguagem CO3L é necessário analisar as abordagens mais importantes para a representação de ontologias. Entre as abordagens analisadas salienta-se o Ontolingua, o OWL e o UML. Estas abordagens destacam-se porque o OWL é a recomendação do W3C para a linguagem de representação de ontologias na Web Semântica, o Ontolingua pela sua capacidade de descrever ontologias num formato compatível com múltiplas linguagens de representação do conhecimento e pela sua maturidade. O UML destaca-se pela sua utilização para modelação de domínios e sistemas e por ser amplamente disseminado, em especial na comunidade dos Sistemas de Informação. A análise crítica destas abordagens é descrita na secção 2.1.

No contexto do desenvolvimento de ferramentas que suportam modelos e linguagens para a descrição de ontologias foram estudados sete sistemas na secção 2.2, escolhendo-se o Ontolingua Server e o Protégé para uma análise mais detalhada.

O Ontolingua Server foi analisado em detalhe porque é o sistema mais importante e versátil para as ontologias em Ontolingua, sendo também o sistema mais antigo, com sucesso, que surgiu ligado ao processamento computacional de ontologias. O Ontolingua Server oferece um ambiente de desenvolvimento colaborativo e permite que as ontologias sejam acedidas globalmente por vários utilizadores através de um repositório central. No entanto, este sistema não tem sido actualizado ao longo dos anos e o repositório central não está assente sobre base de dados.

Uma ferramenta que tem uma actualização constante e que é suportada por uma grande comunidade é o Protégé. Esta ferramenta suporta o desenvolvimento de ontologias de enquadramentos (*frame-based*) e de ontologias OWL. Salienta-se que a globalidade dos sistemas não tem mecanismos que permitam publicar e disponibilizar globalmente uma ontologia. A preocupação primária que as tem norteado centra-se no desenvolvimento de ontologias. As ferramentas desenvolvidas nesta dissertação têm este aspecto em consideração e respondem com soluções que permitem o desenvolvimento de ontologias e têm também um bom mecanismo para armazenar e disponibilizar as ontologias publicamente.

1.2 Contribuições e Motivação

Esta dissertação contribui a dois níveis para avançar o estado do conhecimento e das práticas na comunidade de ontologias, na ciência da computação. A primeira contribuição, situada a um nível conceptual, tem por objectivo melhorar e estender um modelo e uma linguagem específicos de representação de ontologias. A um segundo nível, a dissertação contribui com o desenho e desenvolvimento de ferramentas computacionais para o processamento de ontologias. Foram desenvolvidas duas ferramentas: um editor de ontologias especializado numa linguagem de representação específica e um servidor de ontologias para o armazenamento, a gestão e a disponibilização pública de ontologias.

A primeira contribuição diz respeito à revisão e melhoria do modelo O3F e da linguagem que lhe está adjacente, o CO3L.

A escolha deste modelo e desta linguagem de representação de ontologias alicerça-se em duas razões. A primeira razão tem que ver com as potencialidades dessa abordagem. Da comparação detalhada das várias abordagens para a representação de ontologias (Capítulo 2) resultam óbvias vantagens do O3F e da linguagem CO3L em relação às outras, salientando-se a possibilidade de representar acções e métodos de acção e a possibilidade de optar por um paradigma centrado em objectos, por um paradigma não centrado em objectos (i.e., centrado em predicados, funções e acções), ou mesmo por ambos, caso em que podem ser estabelecidas relações formais entre as duas perspectivas. As vantagens do O3F face aos restantes modelos estão descritas na secção 2.1. Esta secção contém ainda os modelos que foram estudados no âmbito desta dissertação.

A segunda razão para a escolha do O3F e da sua linguagem CO3L diz respeito ao facto de terem sido inicialmente criadas e continuarem a ser desenvolvidas no grupo de Agentes e Inteligência Artificial do Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL, tendo sido objecto de várias publicações, sendo usados nas aulas do terceiro ano das licenciaturas do departamento e continuarem a motivar a realização de teses de mestrado e doutoramento do departamento.

O principal contributo desta dissertação para o O3F e para a sua linguagem foi a introdução do conceito de indivíduo, a introdução do conceito de hierarquia, a introdução de dependências e a definição de um maior número de facetas. A descrição do modelo O3F assim como as melhorias nele introduzidas estão descritas no Capítulo 2 .

A segunda contribuição diz respeito ao desenvolvimento de um servidor de ontologias O3F, chamado O3 Server. O O3 Server está actualmente assente na plataforma Microsoft SQL Server 2008. O O3 Server tem como principais funcionalidades o armazenamento e obtenção de ontologias segundo o modelo O3F. O modelo O3F foi transformado num modelo relacional optimizado, mantendo a abstracção do modelo e oferecendo bons níveis de desempenho relativamente ao armazenamento e extracção de ontologias. O O3 Server foi desenvolvido em cooperação com a tese de Jairo Avelar (Avelar, 2010).

Ainda no contexto do O3 Server, a presente dissertação desenvolveu uma API que recorre ao Hibernate 3. Esta API facilita a comunicação entre aplicações cliente e o

O3 Server. De salientar que esta API foi utilizada na ferramenta CO3L Edit (ferramenta desenvolvida no âmbito desta dissertação, descrita adiante) e na ferramenta UML2O3F (Avelar, 2010).

O O3 Server, por usar uma base de dados relacional para manter ontologias, não só é capaz de disponibilizar publicamente as ontologias armazenadas, mas é capaz de o fazer independentemente da linguagem em que a ontologia foi originalmente representada. Usando o servidor, é possível armazenar e extrair ontologias representadas em diversas linguagens. Actualmente existe um mecanismo que extrai ontologias mantidas no servidor e representa-as textualmente na linguagem CO3L (Avelar, 2010).

Espera-se que, num futuro próximo, diferentes ferramentas comuniquem também com o O3 Server. No futuro será possível criar mais serviços ligados ao O3 Server, por exemplo tradutores de ontologias O3F para diferentes formatos, mecanismos de alinhamento e de fusão e mecanismos de aprendizagem com base nas ontologias mantidas no servidor.

A terceira e última contribuição diz respeito ao desenvolvimento do CO3L Edit, uma ferramenta especializada na edição de ontologias na linguagem CO3L. Tendo em conta que o modelo O3F e a linguagem CO3L são projectos académicos relativamente recentes, ainda não existem ferramentas computacionais de suporte ao modelo ou à linguagem, excepto as desenvolvidas nesta dissertação e na dissertação de Jairo Avelar (Avelar, 2010). Assim o CO3L Edit é a primeira ferramenta que suporta a linguagem CO3L.

O CO3L Edit suporta, num ambiente gráfico intuitivo, funcionalidades como o desenvolvimento de ontologias CO3L, a validação de ontologias, a importação de ontologias CO3L que estejam em ficheiros locais ou em servidores remotos (utilizando protocolos HTTP e FTP) e permite ainda o armazenamento de ontologias em ficheiros locais. Recorrendo à integração com o O3 Server, o CO3L Edit permite armazenar e extrair ontologias no servidor. Tendo sido programado na linguagem JAVA, o CO3L Edit pode ser usado em diferentes sistemas operativos. A interface gráfica do CO3L Edit foi desenvolvida em SWT, o único *widget toolkit* capaz de recorrer às interfaces gráficas nativas do sistema operativo onde é executado, sendo ainda um dos mais rápidos. As tecnologias utilizadas na construção do CO3L Edit

assim como os detalhes das funcionalidades e detalhes de implementação estão descritos no Capítulo 1 .

Em resumo, esta dissertação produziu três contribuições:

1. Melhoria do modelo O3F e da linguagem CO3L
2. Construção do primeiro servidor de ontologias para o modelo O3F
3. Construção do primeiro editor especializado na linguagem CO3L

O presente trabalho visa colmatar a falta de ferramentas computacionais que suportam o modelo O3F e a linguagem CO3L. Deste modo será possível chegar a mais utilizadores e conseguir mostrar e disponibilizar as vantagens do modelo O3F e da linguagem CO3L.

1.3 Avaliação do trabalho realizado

A avaliação desta dissertação é fundamental para determinar o seu sucesso e a aceitação dos resultados produzidos pela comunidade científico-tecnológica em que se enquadram e pelo grupo alvo de utilizadores a que se destinam.

O grupo alvo de utilizadores deste trabalho contempla todos os utilizadores que tenham interesse no modelo O3F e na linguagem CO3L. Em última análise pode-se afirmar que todos os utilizadores interessados em especificar ontologias também beneficiam deste trabalho tendo em conta que o modelo O3F e a linguagem CO3L foram melhorados e foram criadas ferramentas que suportam o modelo e a linguagem. Os utilizadores irão beneficiar da expressividade do modelo e da linguagem, mas suportados computacionalmente.

A avaliação deste trabalho foi constituída por uma avaliação às ferramentas CO3L Edit e O3 Server. Esta avaliação foi realizada recorrendo a um questionário que foi dirigido a utilizadores externos ao projecto que tinham conhecimentos do modelo O3F e da linguagem CO3L assim como um contacto prévio com a ferramenta CO3L Edit e, através desta, um contacto com o O3 Server.

Esta avaliação permitiu apurar que os utilizadores ficaram muito satisfeitos com as soluções que lhes foram disponibilizadas. Cerca de 60 utilizadores responderam ao inquérito de uma forma bastante satisfatória conforme comprovam os resultados que constam na secção 6.5. Por exemplo, numa escala de 0 a 4, a média das 6 questões

que foram colocadas aos utilizadores é de 3,70 pontos. Esta média espelha uma clara satisfação dos utilizadores face ao CO3L Edit e ao O3 Server.

Para comprovar a conformidade do CO3L Edit e do O3 Server face aos requisitos iniciais, foi elaborado um conjunto de testes que visam demonstrar o correcto funcionamento das ferramentas. Estes testes não foram elaborados por utilizadores externos ao projecto por se tratar de uma tarefa repetitiva que carece de um ciclo de teste, falha e correcção. Os testes efectuados estão descritos em seguida:

- Testar todos os predicados do CO3L na ferramenta CO3L Edit e verificar que são reconhecidos e processados
- Testar todas as facetas *built-in* do CO3L na ferramenta CO3L Edit e verificar que são reconhecidas e processadas
- Processar todas as ontologias que são leccionadas na cadeira de Tecnologias de Sistemas Inteligentes e publicá-las no O3 Server

A avaliação do presente trabalho está descrita no Capítulo 6 .

Por fim, no Capítulo 6 estão descritas as conclusões finais do trabalho. Na secção 7.1 deste capítulo foram analisados os objectivos deste trabalho, concluindo-se que foram cumpridos na íntegra e que foram inclusivamente ultrapassados. O trabalho futuro que é possível desenvolver na sequência desta dissertação está descrito na secção 7.2. Existe ainda muito trabalho que é possível desenvolver para tornar o CO3L Edit e o O3 Server em ferramentas cada vez mais competitivas, passando pela criação de mais mecanismos que facilitem o desenvolvimento de ontologias e a rápida identificação e correcção de erros até à inclusão de mais serviços no O3 Server. Relativamente à melhoria do modelo O3F e da linguagem CO3L, deverão ser alvo de estudo para que seja possível descrever ontologias recorrendo a axiomas. Na secção 7.3 estão indicadas as principais dificuldades sentidas em todo o trabalho, tendo sido focadas as dificuldades tecnológicas e a dificuldade de gestão de âmbito do projecto.

Conclui-se que a presente dissertação contribuiu com o aumento de expressividade no modelo O3F e na linguagem CO3L e com ferramentas computacionais que suportam tanto o modelo como a linguagem. Deste modo será possível, pela primeira vez, concorrer com outros modelos já existentes e progredir no sentido de tornar o modelo

O3F num modelo a ter em consideração sempre que se pretenda modular um domínio através de ontologias.

Capítulo 2 Estado da Arte

A representação do conhecimento através de ontologias não é de todo novidade. Na realidade, já os antigos filósofos utilizaram ontologias para descrever domínios naturais, ou seja, as coisas naturais do mundo como os tipos de existência e as relações temporais. No entanto, ao longo dos tempos, a difusão das ontologias tem ganho bastante relevância em certos domínios científicos e tecnológicos, o que levou os cientistas e engenheiros a investigarem fortemente esta área.

A crescente necessidade de interligar sistemas e de conseguir que diferentes aplicações comuniquem de forma não ambígua está na origem da necessidade de bons modelos para especificar ontologias assim como de boas ferramentas que dêem suporte à criação, proliferação e manutenção das ontologias.

O recente desejo que a Internet passe a ser um contentor de informação que possa ser processada por computadores é também um bom impulsionador na investigação de ontologias (Kalinichenko, Missikoff, Schiappelli, & Skvortsov, 2003).

Neste sentido, o presente trabalho visa contribuir para a representação e utilização de ontologias, em particular para o aperfeiçoamento do modelo O3F e da respectiva linguagem de representação CO3L, e também com a criação do primeiro servidor de ontologias O3F e com o primeiro editor da linguagem CO3L. Torna-se assim necessário analisar as actuais tendências na representação de ontologias e perceber qual o actual estado das ferramentas que suportam o desenvolvimento e gestão de ontologias.

Na secção 2.1 serão analisados os principais modelos e linguagens utilizados para descrever ontologias e na secção 2.2 serão analisados os principais sistemas e ferramentas que suportam o desenvolvimento e gestão de ontologias.

2.1 Linguagens para Descrição de Ontologias

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

Imagine-se que um grupo de investigadores sobre biologia molecular necessita de partilhar os seus modelos. Os modelos partilhados necessitam de ser validados por outros investigadores que verificam a sua consistência. A interpretação do modelo não pode ser subjectiva, todos os investigadores têm que depreender o mesmo conhecimento que está descrito no modelo. Assim, torna-se necessário a existência de uma ontologia que permita a definição dos conceitos do domínio no qual os investigadores vão trabalhar.

Para que o desenvolvimento de ontologias não seja feito de forma *ad hoc*, é necessário que exista também um modelo subjacente à sua definição. No entanto, não existe um consenso relativo a um modelo que melhor se adeqúe ao desenvolvimento de ontologias. Os mais importantes modelos de representação de ontologias são descritos no decorrer deste capítulo, onde será feita a sua análise comparativa.

É necessário compreender que existem ontologias com diferentes propósitos, o que faz com que um dado modelo de representação de conhecimento seja mais apropriado para um dado tipo de ontologia e menos adequado para outro.

Segundo Guarino (Guarino, 1998), as ontologias podem ser classificadas nas seguintes categorias:

- Ontologias genéricas, ou ontologias de topo (*upper ontologies*): descrevem conceitos bastante gerais, tais como, espaço, tempo, matéria, objecto, evento, acção, entre outros; são independentes de um problema ou domínio particular;
- Ontologias de domínio: expressam conceptualizações de domínios particulares através da definição de um vocabulário relacionado com o domínio, tal como Medicina ou Automóveis;
- Ontologias de tarefas: expressam conceptualizações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descrevem o vocabulário relacionado com uma actividade ou tarefa genérica, por exemplo composição, sequência ou repetição;
- Ontologias de aplicação: descrevem conceitos dependentes do domínio e de tarefas em particular. Estes conceitos estão frequentemente relacionados com papéis desempenhados por entidades do domínio aquando da realização de uma dada actividade;

- Ontologias de representação: explicam as conceptualizações que fundamentam os formalismos de representação do conhecimento.

As ontologias de domínio são o tipo de ontologias mais desenvolvidas, e é também neste tipo de ontologias que o presente trabalho se centra.

Tendo em conta que uma ontologia é especificada através de uma linguagem formal (Guarino, 1998), torna-se necessário analisar as abordagens mais relevantes na representação de ontologias. A análise destas linguagens será feita nas secções seguintes e serve para perceber as suas potencialidades e fraquezas.

2.1.1 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 Semântica. O OWL nasce como uma revisão da linguagem DAML+OIL (W3C, 2001).

Esta linguagem foi construída sobre o RDF (*Resource Description Framework)* (W3C, 2004b) e RDF Schema (W3C, 2004b) para aumentar a expressividade existente, 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 & Harmelen, 2003). De facto, algumas possibilidades expressivas do OWL não são computáveis.

Tendo em conta este facto, o W3C decidiu definir três variantes do OWL:

- OWL-Full. O OWL-Full é a versão mais completa das três uma vez que compreende todo o OWL. A grande vantagem do OWL-Full é que é completamente compatível com o RDF Schema. O facto de o OWL-Full não impor restrições à utilização do RDF ou do RDF Schema faz com que estes possam ser considerados também como OWL-Full. No entanto, ao utilizar o OWL-Full, cada utilizador será responsável pela resolução de problemas provenientes da execução do raciocínio.

- OWL-DL. O grande objectivo do OWL-DL é 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 algumas restrições às construções OWL e ao uso do RDF Schema. Por exemplo, o OWL-DL restringe a mistura das primitivas do OWL com as primitivas do RDF e não permite que um indivíduo possa ser ao mesmo tempo considerado uma classe e uma propriedade, não podendo também uma propriedade ser ao mesmo tempo uma classe ou um indivíduo. São este tipo de restrições que garantem o apoio computacional ao raciocínio com as descrições OWL-DL. No entanto o OWL-DL reduz a expressividade da linguagem e elimina a total compatibilidade com o RDF e com o RDF Schema.
- OWL-Lite. O OWL-Lite, a versão mais simples do OWL, herda as restrições presentes no OWL-DL e utiliza apenas um subconjunto das construções do OWL-Full. O intuito subjacente ao OWL-Lite é o de facilitar a implementação de ferramentas de desenvolvimento e utilização de ontologias.

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

- Classes. No OWL, enfatiza-se que as classes são 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 e subclasses.
- 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 sub-propriedades, relacionando várias propriedades hierarquicamente.
- Indivíduos. Os indivíduos representam objectos do domínio. Estes objectos podem ser instâncias de uma ou mais classes.

O OWL permite, através de várias primitivas, fornecer informação sobre classes, propriedades e indivíduos. A hierarquização de classes e propriedades, a indicação de que duas classes são iguais ou que uma propriedade é o inverso de outra são alguns exemplos.

As ontologias OWL são usualmente representadas em sintaxe baseada no XML, no entanto existem outras. Existe a sintaxe abstracta que é usada no próprio documento de especificação da linguagem e existe o GrOWL (Krivov, Williams, & Villa, 2007), uma linguagem gráfica para representação de ontologias OWL.

2.1.2 OWL 2

O W3C realizou recentemente uma revisão e extensão da linguagem OWL à qual chamou 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 explícita, 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, o OWL 2 define outras três variantes que impõem restrições específicas na linguagem, de modo 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 QL (*Query Language*) impõe restrições que permitem a integração de RDBMS (*Relational Database Management System*), possibilitando que uma ontologia seja interrogada numa base de dados usando a linguagem SQL.

- 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 demasia o poder de expressividade da linguagem.

2.1.3 Ontolingua

O Ontolingua (Gruber, 1993) 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 (Genesereth & Singh, 1991), e uma forma canónica de KIF (*Knowledge Interchange Format*) (Genesereth & Fikes, 1992). 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 ou em alternativa acedidas pelo protocolo OKBC (*Open Knowledge Base Connectivity*) (Chaudhri, Farquhar, Fikes, Karp, & Rice, 1998), sendo que o protocolo OKBC tem actualmente várias implementações em diferentes linguagens de programação.

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

O conjunto de expressões que o Ontolingua pode reconhecer e traduzir é definido com base numa Ontologia de enquadramentos (*Frame Ontology*) (Stanford, 1994) 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. Já as instruções KIF são utilizadas para restringir o significado da definição dos termos.

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, intersecção e partição. 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.

A sintaxe do KIF é baseada na da linguagem de programação Lisp (Mccarthy, 1978).

2.1.4 UML

O UML (*Unified Model Language*) 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), da Técnica de Modelação de Objectos (*Object-Modeling Technique* - *OMT*) (Rumbaugh, Blaha, Premerlani, Eddy, & Lorensen, 1991) e da Engenharia de Software Orientado por Objectos (*Object-Oriented Software Engineering* - *OOSE*) (Jacobson, 1992). A sua especificação normalizada foi criada e é mantida pelo *Object Management Group* (OMG) e está na versão 2.2 (OMG, 2009).

Sendo uma linguagem gráfica, o UML permite a construção de modelos de sistemas 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 os 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 duas ou mais classes. Podem definir-se várias propriedades das associações, nomeadamente a multiplicidade dos argumentos da associação, o papel desempenhado por cada um e a direccionalidade da associação. É ainda possível indicar se a associação retrata uma agregação[1] ou uma composição.
- Generalizações. Exprimem relações hierárquicas entre classes. Indica qual a classe mais genérica e 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 tem dois componentes textuais:

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

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

Vários autores identificam características e vantagens no UML para que seja utilizado como uma linguagem de representação de ontologias. Algumas dessas características são a capacidade de representar conceitos comuns aos das linguagens de representação de ontologias (e.g. classes e relações) (Cranefield, 2001), ser amplamente disseminada, ter um grande leque de ferramentas CASE que o suportam

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

(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 para representar conhecimento ontológico, o *Ontology Definition Metamodel* (ODM).

2.1.5 O3F/CO3L

O O3F (*Object Oriented Ontology Framework*) inicialmente definido em (Mota, Botelho, Mendes, & Lopes, 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 IUL-ISCTE (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 que lhe está subjacente, consequentemente, falar de características da linguagem é falar das características do modelo O3F e vice-versa.

A principal característica do O3F reside na possibilidade de conceptualizar um domínio recorrendo ao paradigma da orientação por objectos, à modulação do domínio através de relações, funções e acções, e ainda a uma modelação mista recorrendo a ambos os tipos de primitivas. Salienta-se que, para além do O3F, apenas o Ontolingua oferece funcionalidades semelhantes a esta. No entanto, ao nível do paradigma da orientação por objectos o Ontolingua é menos expressivo porque apenas permite definir classes com atributos, mas não com métodos.

Além de possibilitar a conceptualização orientada a objectos em simultâneo com a conceptualização em termos de relações, funções e acções, o O3F tem ainda a

vantagem de captar a relação existente entre estes dois paradigmas. Os casos mais emblemáticos desse relacionamento 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 de relações, de funções e de acções;
- E a interpretação de classes, relações e funções como conjuntos de conjuntos de pares nome/valor.

Em O3F, enfatiza-se o paralelismo entre atributo de um objecto e função, no sentido em que a expressão x.A é apenas notação diferente para A(x). Também se realça o paralelismo entre um método funcional e uma função, exactamente no mesmo sentido - a expressão $x.F(a_1, a_2, ..., a_n)$ é uma notação alternativa da expressão $F(x, a_1, a_2, ..., a_n)$. Paralelismos semelhantes são estabelecidos entre os métodos relacionais e predicados, e entre métodos de acção e acções.

Sendo enfatizado que os tipos de dados são conjuntos, em O3F podem estabelecer-se hierarquias entre quaisquer tipos de dados, por exemplo, hierarquias de números e hierarquias de classes. Além disso, como as funções e os predicados denotam conjuntos, também se podem estabelecer hierarquias de funções e hierarquias de predicados. Por exemplo, o predicado P é um sub-predicado 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 interfaces funcionais (funções), ou interfaces relacionais[2] (predicados), as quais se podem organizar em hierarquias. Consequentemente, como atributos e métodos correspondem a funções, predicados ou acções, os atributos e os métodos podem também ser hierarquizados.

O O3F permite ainda caracterizar as entidades de uma ontologia através de facetas. A utilização de facetas permite expressar conceitos que, noutras linguagens, só poderiam ser expressos através de axiomas.

[2] "Interface de uma acção" é a terminologia O3F para "assinatura da acção".

Apesar da facilidade e das vantagens computacionais da utilização de facetas, está prevista a extensão do O3F para que venha a permitir a definição de axiomas capazes de expressar conceitos que não podem ser definidos através de facetas.

O O3F permite também a definição de novas facetas e de novos tipos de dados.

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

Analogamente ao UML e ao OWL, o O3F e a sua linguagem possibilitam a representação de indivíduos. Os indivíduos podem ser instâncias de classes, associações e outros tipos de dados. Mas também é possível representar indivíduos que não pertencem a nenhum dos conjuntos da ontologia.

2.1.6 Conclusão

Existem várias linguagens para a descrição de ontologias onde estão sempre presentes conceitos como classes, relações e herança. Actualmente a linguagem 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 sub-linguagens que permitem escolher o equilíbrio mais adequado entre a expressividade e o apoio ao raciocínio.

O Ontolingua é também uma alternativa interessante pois permite o desenvolvimento de ontologias num formato compatível com múltiplas linguagens de representação e facilita a sua partilha.

O UML começa também a ser considerado para a representação de ontologias de um modo gráfico. Para reforçar esta ideia, o OMG desenvolveu também o ODM (*Ontology Definition Metamodel)* para tornar compatíveis os conceitos do MDA (*Model Driven Architecture*) com a especificação de ontologias.

Características	OWL	Ontolingua	UML	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 *classifiers*	binária	n-árias	n-árias	n-árias
Dependências	não	não	Sim	sim
Axiomas formais	não	sim	Sim	não
Indivíduos	sim	sim	Sim	sim

Tabela 1 - Resumo das linguagens abordadas

A revisão da literatura permite concluir que o modelo O3F é uma excelente opção para representar ontologias devido às várias vantagens que apresenta em relação às restantes abordagens.

Através da Tabela 1 é possível depreender as seguintes vantagens:

- Representação de associações, predicados e funções de qualquer aridade (vantagem em relação ao UML e OWL);
- Representação de métodos de classes e sua hierarquização (vantagem em relação a quaisquer linguagens de representação de ontologias, excepto o UML);
- Representação de acções e métodos de acção (vantagem em relação a qualquer linguagem de representação).

Além destas vantagens, o O3F é o único que permite modular um domínio de acordo com uma visão orientada a objectos ou de acordo com uma visão mais relacional ou funcional do mundo, possibilitando ainda a integração de ambas. Finalmente, o O3F e a sua linguagem CO3L são usados e têm sido desenvolvidos no grupo de trabalho de Agentes e Inteligência Artificial do ISCTE-IUL, onde esta dissertação tem decorrido.

2.2 Suporte às ontologias

As ontologias deverão ser representadas em linguagens formais (Guarino, 1998) que eliminem ou reduzam ao mínimo a ambiguidade e a imprecisão. No entanto, a linguagem formal por si só não oferece os requisitos necessários para que uma ontologia possa ser criada, utilizada e disponibilizada a aplicações computacionais, em particular a agentes inteligentes artificiais.

A criação de ontologias é uma tarefa complexa e morosa (Kirasić & Basch, 2009), pelo que a necessidade de boas aplicações que suportem quer o seu desenvolvimento quer a sua utilização é crítica. O suporte às ontologias a nível aplicacional não se restringe a ferramentas que permitem ao utilizador criar e guardar ontologias. Existe um conjunto de ferramentas que podem ser agrupadas do seguinte modo (Gómez-Pérez, 2002):

- Ferramentas de desenvolvimento. Este grupo inclui ferramentas que permitem a criação de ontologias. Usualmente também contêm funcionalidades como a edição de ontologias, procura de ontologias e importação e exportação de ontologias em diferentes formatos e linguagens.
- Ferramentas de avaliação. Estas ferramentas avaliam o conteúdo da ontologia para reduzirem problemas ao nível da integração de ontologias em outros sistemas de informação.
- Ferramentas de fusão e alinhamento. Neste grupo estão ferramentas que permitem a fusão e alinhamento de diferentes ontologias no mesmo domínio.
- Ferramentas de interrogação e inferência. Estas ferramentas têm a capacidade de interrogar as ontologias armazenadas e processá-las efectuando inferência.
- Ferramentas de aprendizagem. Estas ferramentas conseguem de uma forma semi-automática derivar ontologias através do processamento de textos em língua natural, bases de dados e fontes de informação semi-estruturadas.

O presente trabalho, no que à obra de engenharia diz respeito, vai centrar-se nas ferramentas de desenvolvimento e de armazenamento.

Segundo Stojanovic Fzi (Fzi, Stojanovic, & Motik, 2002), as ontologias tendem a ser modificadas ao longo do tempo para reflectir o mundo real, mudanças nos requisitos, corrigir erros no desenho inicial e incorporar novas funcionalidades. Todas estas

actividades são efectuadas nos editores de ontologias. Segundo o mesmo autor, a necessidade de alterar ontologias é mais significativa actualmente do que no passado, porque é necessário acompanhar a constante mudança que acompanha os sistemas, as pessoas e o mundo. Assim, os editores deverão ter como principais características serem escaláveis, disponíveis, rápidos e fiáveis para que o processo da mudança se dê de uma forma simples e eficaz.

Outra necessidade importante no suporte às ontologias é a disponibilização, reutilização e consulta de ontologias. Para que seja possível a diferentes grupos aceder a uma mesma ontologia e partilhar assim um vocabulário, é necessário que exista um mecanismo que permita o acesso à mesma fonte de informação.

Tendo em conta que o presente trabalho tem como parte fulcral a construção de um editor e de um servidor de ontologias para o modelo O3F e para a sua linguagem CO3L, é necessário perceber o que existe em termos de sistemas que dêem suporte às abordagens descritas anteriormente, como o OWL, o Ontolingua e o UML.

No decorrer desta secção a ferramenta Protégé será analisada em detalhe assim como o servidor Ontolingua Server. O Protégé é utilizado principalmente para descrever ontologias em OWL e está descrito na secção 2.2.1. Já o Ontolingua Server contém um conjunto de aplicações Web e é utilizado em ontologias Ontolingua, sendo descrito na secção 2.2.2. Relativamente a ferramentas UML, não existe nenhum editor maduro que permita descrever ontologias de acordo com o ODM, assim sendo o presente trabalho não analisa concretamente nenhum editor UML. Salienta-se apenas o *plugin* UML2OWL para o Eclipse, criado por Guillaume Hillairet (Hillairet, 2007), que permite a criação de diagramas UML com base no ODM. Este *plugin* tem ainda a possibilidade de exportar o resultado da modulação para OWL.

2.2.1 Protégé

O projecto Protégé[3] é um editor de ontologias inicialmente desenvolvido na Universidade de Stanford e tem evoluído bastante desde que foi construído por Mark Musen em 1987 (Musen, 1989). A ferramenta original era uma pequena aplicação com o objectivo de construir ferramentas de aquisição de conhecimento para alguns programas específicos de planeamento médico. No entanto, o Protégé evoluiu para

[3] http://protege.stanford.edu/

uma plataforma durável e extensível que tem como objectivo o desenvolvimento e investigação de sistemas baseados no conhecimento. Actualmente o Protégé conta com a versão 4.0.1, é gratuito e de software aberto (o*pen source*).

2.2.1.1 Arquitectura

O Protégé percorreu um longo caminho desde a sua primeira versão Opal (nome antigo) em que apenas era compatível com máquinas Xerox LISP (Gennari, et al., 2003).

Actualmente o Protégé está implementado em Java, recorrendo ao Swing para disponibilizar uma interface gráfica ao utilizador. O facto de a sua implementação ter recorrido a tecnologia Java permite que o editor possa ser instalado localmente em qualquer máquina, independentemente do sistema operativo.

A arquitectura deste editor permite que sejam adicionadas extensões de uma forma facilitada. As extensões estão normalmente divididas em três tipos (Kirasić & Basch, 2009):

- *Tab Plug-ins* – São extensões que visam criar novas secções na aplicação (por via de tabs na interface) para providenciarem novas funcionalidades que não estão presentes na versão original. Estas extensões são as mais usuais.
- *Slot widgets* – São extensões utilizadas para ver e editar os valores de *slots* sem recorrer à interface original.
- *Backends* – Extensões que permitem adicionar a capacidade de exportar e importar ontologias em diferentes formatos. Existem também extensões que permitem que as ontologias sejam guardadas e acedidas em bases de dados.

O armazenamento das ontologias é conseguido através de ficheiros de texto CLISP do sistema operativo ou em base de dados, recorrendo ao mecanismo *jdbc back-end* que guarda toda a informação numa única tabela.

A arquitectura do Protégé está assim circunscrita ao seu editor e às suas extensões[4].

2.2.1.2 Funcionalidades

O editor de ontologias Protégé tem actualmente a capacidade de modelar ontologias de duas formas:

[4] http://protege.stanford.edu/download/plugins.html

- Protégé-Frames[5]
- Protégé-OWL

O editor Protégé-Frames permite aos utilizadores construírem e povoarem uma ontologia de enquadramentos (*frame-based*), de acordo com o OKBC Knowledge Model.

O editor Protégé-OWL permite construir ontologias para a Web Semântica, em particular em OWL.

Uma das funcionalidades interessantes deste editor está no modo como conduz o utilizador na criação das instâncias. Quando o utilizador pretende criar uma instância de um dado tipo, os ecrãs são gerados tendo em conta o tipo que se vai instanciar (Gennari, et al., 2003). Por exemplo, se na especificação de uma dada classe o campo "Casado" tiver o tipo de dados booleano, a especificação da instância terá um controlo de interface que permite escolher entre verdadeiro ou falso.

As ontologias desenvolvidas em Protégé podem ser exportadas para uma variedade de formatos incluindo RDF(S), OWL e XML Schema.

2.2.1.3 Conclusão

A aplicação Protégé mostra-se versátil no desenvolvimento de ontologias. No entanto não existe uma preocupação visível na publicação e partilha de ontologias. Todas as suas funcionalidades apostam no desenvolvimento e edição de ontologias.

Em conclusão, a aplicação Protégé é dos editores de ontologias mais utilizados, possuindo uma comunidade com mais de 126 mil utilizadores. É suportado por meios académicos, governos e corporações.

2.2.2 Ontolingua Server

Em Fevereiro de 1995 o KSL[6] (*Knowledge System, AI Laboratory*) lançou o Ontolingua Server (Farquhar, Fikes, & Rice, 1996). O Ontolingua Server veio revolucionar o modo como a escrita de ontologias era realizada e partilhada, através da Internet. Este servidor possui um conjunto de aplicações onde se destaca o Ontolingua Editor, que permite a criação e edição de ontologias.

[5] Não está disponível na última versão Protégé 4.0.1
[6] http://www-ksl.stanford.edu/

2.2.2.1 Arquitectura

O Ontolingua Server é composto por várias aplicações Web:

- Webster (para obter definição de termos);
- OKBC Server (para aceder a ontologias Ontolingua através do protocolo OKBC);
- Chimaera (para analisar, integrar e unir ontologias);
- Ontolingua Editor.

O Webster, o Chimaera e o Ontolingua Editor funcionam em ambiente Web, o servidor interage com o cliente através do protocolo HTTP e as interfaces gráficas com o utilizador são asseguradas recorrendo a formulários HTML, tornando a aplicação acessível a todos os utilizadores que detenham uma ligação à Internet. O OKBC Server não tem uma interface para pessoas; aceita apenas conexões remotas. Este mecanismo permite que diferentes aplicações (aplicações externas ao Ontolingua Server) possam comunicar com o servidor, independentemente da linguagem em que foram implementadas.

O armazenamento das ontologias é conseguido através de ficheiros de texto LISP que são guardados directamente no sistema (Kirasić & Basch, 2009), não existindo armazenamento em bases de dados.

A arquitectura do Ontolingua Server foi concebida para possibilitar uma grande heterogeneidade na ligação com o exterior. Será teoricamente possível aceder ao Ontolingua Editor através de qualquer browser e será também possível desenvolver uma aplicação externa que comunique com o OKBC Server para consulta e edição de ontologias.

2.2.2.2 Funcionalidades

Esta aplicação disponibiliza três modos de interacção com o servidor (Farquhar, Fikes, & Rice, 1996):

- Criar, editar e gerir ontologias;
- Editar e interrogar ontologias;
- Traduzir ontologias.

No primeiro modo, os utilizadores poderão utilizar o browser para aceder ao editor e construir ou manter ontologias no servidor. Tendo em conta que as ontologias ficam guardadas num repositório central, é possível descentralizar no tempo e no espaço o desenvolvimento das ontologias. O editor oferece um conjunto de funcionalidades que permitem o trabalho colaborativo. Tem-se como exemplo a funcionalidade que possibilita a múltiplos utilizadores trabalharem sobre a mesma ontologia. O editor permite também enviar notificações, comparar versões e aceder ao histórico de alterações de uma ontologia.

O segundo modo permite a interrogação de ontologias e a sua modificação através de aplicações remotas. Este modo de interacção utiliza uma API que estende o Generic Frame Protocol (Karp, Myers, & Gruber, 1995) com uma interface para o ambiente Web.

Por último, existe o modo de tradução que permite aos utilizadores traduzirem a sua ontologia para outros formatos.

2.2.2.3 Conclusão

Apesar das suas vantagens, esta ferramenta não tem sido actualizada ao longo dos anos, continuando a ser usada a mesma tecnologia e interface com que foi inicialmente criada, em 1995, as quais estão já bastante desactualizadas. O modo como as ontologias são guardadas, em ficheiros LISP, dificulta a evolução do sistema.

Note-se ainda que a ligação do Ontolingua Server com aplicações remotas não é conseguida em pleno. As aplicações externas não terão a possibilidade de criar ontologias, o que limita a utilização do sistema.

Por último, o Ontolingua Server não consegue disponibilizar mecanismos de tradução fiéis, falhando assim um dos seus principais propósitos. A fraca qualidade das traduções realizadas dificulta a utilização das ontologias traduzidas por ferramentas e sistemas de inferência (Silva, et al., 2002).

O Ontolingua Editor revolucionou os editores de ontologias na medida em que oferece um ambiente de desenvolvimento colaborativo e permite que as ontologias sejam acedidas por vários utilizadores através de um repositório central.

O Ontolingua Server encontra-se disponível em http://www-ksl-svc.stanford.edu: 5915.

2.2.3 Conclusão

Nas secções anteriores foi analisado em detalhe o Ontolingua Server, juntamente com o seu Ontolingua Editor, que foi a primeira ferramenta de desenvolvimento de ontologias a ser criada com sucesso, e também foi analisado o Protégé que é a ferramenta de ontologias mais utilizada actualmente. Para além delas, existem outras ferramentas menos disseminadas que foram analisadas em (Kirasić & Basch, 2009). O resultado desta análise está sintetizado na Tabela 2.

.

Ferramenta	Referência	Instituto	Ano	Interface	Mecanismo armazenamento	KR	Formatos de importação	Formatos de exportação
Ontolingua Server	(Farquhar, Fikes, & Rice, The Ontolingua Server: a Tool for Collaborative Ontology Construction, 1996)	Stanford University	1990	HTML	Ficheiros	Ontolingua	KIF, CORBA IDL, CML	KIF, LOOM, CLIPS, CML, Epikit, Prolog
OntoSaurus	(Swartout, Ramesh, Knight, & and Russ, 1997)	University of Southern Califórnia	1990	HTML	Ficheiros	LOOM	PowerLoom, Stella, CORBAs IDL	LOOM, PowerLoom8, Stella, Ontolingua, KIF, CORBAs IDL, C++
WebOnto	(Domingue, 1998)	Open University (United Kingdom)	1997	Java applets	Ficheiros	OCML	-	Ontolingua, RDF(S)
OilEd	(Bechhofer, Horrocks, Goble, & Stevens, 2001)	University of Manchester	2001	Java, Swing, standalone application	Ficheiros	OIL DAML+OIL	RDF(S), OIL, DAML+OIL, SHIQ	DAML+OIL, RDF(S), OWL, SHIQ, DIG
Protégé-2000	(Noy, Fergerson, & Musen, 2000)	Stanford University	2000	Java, Swing, standalone application	Ficheiros, Base de dados (*Plug-in*)	Frame-Based, OWL	RDF(S), XML, XML Schema, and XMI	RDF(S), XML, XML Schema, and XMI
WebODE	(Arpírez, Corcho, Fernández-López, & Gómez-Pérez, 2003)	Universidad Politécnica de Madrid	1999	HTML	Base de dados	WebODE	XML, XCARIN, RDF(S), DAML+OIL, OWL	XML, RDF(S), OIL, DAML+OIL, OWL, CARIN, FLogic, Jess, Prolog, Java
OntoEdit Free	(Sure, Erdmann, Angele, Staab, Studer, & Wenke, 2002)	University of Karlsruhe	-	standalone application	Ficheiros	OXML 2.0	RDF(S), OXML, DAML+OIL, FLogic	RDF(S), OXML, DAML+OIL, FLogic

Tabela 2 – Síntese das ferramentas de desenvolvimento de ontologias

Através da Tabela 2 podemos observar que existe um equilíbrio entre ferramentas que funcionam na Web e ferramentas que funcionam localmente, em cada computador. Segundo Kirasić e Basch (Kirasić & Basch, 2009), as ferramentas que não recorrem à Web conseguem ter mecanismos de edição mais robustos e diversificados.

O armazenamento das ontologias é outro ponto que se pode observar na síntese produzida. Apenas duas em sete ferramentas armazenam as ontologias em bases de dados. Apesar de muito mais usado, o armazenamento em ficheiros traz claras desvantagens:

- Necessidade de recorrer permanentemente à análise sintáctica do ficheiro;
- Problemas de performance na consulta e interrogação de ontologias;
- Incapacidade de desacoplar o modelo e a linguagem.

As ferramentas mais recentes como o Protégé e o OntoEdit (versão PRO) têm a possibilidade de armazenar as ontologias em bases de dados. Problemas como o acesso a partes específicas da ontologia, interrogação de ontologias e a capacidade de desacoplar o modelo de uma linguagem são alguns exemplos de funcionalidades que se ganham quando se armazenam as ontologias em bases de dados. A preocupação de facilitar a integração das ontologias com RDBMS (*Relational Database Management System*) tornou-se óbvia na revisão e extensão da linguagem OWL, originando propositadamente a variante OWL 2 QL. A utilização de bases de dados para guardar ontologias é extremamente importante para a escalabilidade dos sistemas (Perez-Urbina, Horrocks, & Motik, 2009).

Para concluir a análise da Tabela 2, pode observar-se a preocupação das diferentes ferramentas a nível de interoperabilidade, onde é visível o número de diferentes formatos que estas ferramentas exportam e importam.

Capítulo 3 O3F e CO3L

Este capítulo descreve o modelo O3F e recorre à linguagem CO3L para ilustrar exemplos de utilização.

O presente trabalho tem dois momentos bastante distintos, ambos dependentes do modelo O3F. Um primeiro momento onde a análise do modelo O3F e da linguagem CO3L originou várias propostas de alteração e melhoria. A análise e as propostas de melhoria foram realizadas em conjunto com a dissertação de Jairo Avelar (Avelar, 2010) e estão resumidas na secção 3.2. O segundo momento diz respeito ao desenvolvimento de um servidor de ontologias O3F e de um editor de ontologias especializado na linguagem CO3L. A descrição relativa ao servidor de ontologias está no Capítulo 3 , ao passo que a descrição do editor de ontologias está no Capítulo 5 .

Tendo em conta que as características da linguagem CO3L advêm do modelo que lhe está subjacente, falar de características da linguagem é falar das características do modelo O3F e vice-versa. O presente capítulo descreve o modelo O3F, apresentando alguns exemplos recorrendo à linguagem CO3L.

A secção 3.1 mostra que a existência de um modelo, para além de uma linguagem concreta, facilita quer a definição de várias linguagens e ferramentas de especificação e processamento de ontologias, quer a conversão de ontologias especificadas numa linguagem em ontologias especificadas noutra linguagem. Na secção 3.2 descrevem-se as alterações feitas ao modelo e à linguagem. As restantes secções deste capítulo descrevem o modelo, já depois de alterado.

3.1 Importância do modelo

Antes de prosseguir para a descrição do modelo propriamente dito, é importante perceber o porquê de existir um modelo. Um modelo permite a abstracção de qualquer linguagem de representação ou de qualquer ferramenta computacional específica. Permite definir os termos, conceitos e regras subjacentes às ontologias que venham a ser especificadas de acordo com o modelo. A existência de um modelo facilita não só o desenvolvimento de ferramentas de suporte à especificação e processamento de ontologias mas facilita também a interoperabilidade com outras abordagens.

Na modulação de ontologias, o modelo permite a abstracção de uma dada linguagem e descrever os conceitos e relações que existem na modulação de um domínio. A linguagem existe para suportar a especificação de ontologias de acordo com o modelo e para possibilitar na prática a representação de um domínio. O mesmo modelo poderá ter várias linguagens. O modelo O3F tem actualmente a linguagem CO3L, no entanto, poderá no futuro ter representações recorrendo a diferentes linguagens como o XML.

3.2 Melhorias Introduzidas

O presente trabalho contribuiu em muito para o avanço do modelo O3F e da linguagem CO3L. Nesta secção estão listadas as melhorias e extensões mais importantes que foram adicionas ao modelo O3F e à linguagem CO3L.

- Representação de Hierarquias – O modelo O3F apenas permitia a representação de Generalizações. Agora é possível representar hierarquias. A representação de Generalizações é feita recorrendo ao conceito de hierarquia.
- Representação de indivíduos – O modelo O3F foi estendido para que seja possível representar indivíduos. Um indivíduo pode ser uma instância de qualquer tipo de dados, ou pode ser independente dos tipos de dados existentes.
- Reestruturação das Interfaces de Acção – Em versões anteriores, o modelo O3F representava as interfaces de acção relacionais e funcionais recorrendo a associações entre a classe *ActionInterface* e as classes *Function* e *Predicate*. Estes conceitos passam a ser representados através de subclasses das classes *Predicate* e *Function* reforçando o conceito de que uma interface de acção relacional é na prática um predicado e que uma interface de acção funcional é uma função, mas que existem predicados e funções, para além das interfaces de acção.
- Reestruturação da relação entre métodos e operadores – Inicialmente o modelo representava a classe *MethodOperator* para relacionar métodos e operadores. Actualmente o modelo recorre a três classes distintas - *FMethodFunction* (*Functional Method Function*), *AMethodAction* (*Action Method Action*) e *RMethodPredicate* (*Relational Method Predicate*) - para representar as relações entre métodos e operadores. A grande vantagem de haver uma classe específica para cada relação reside na possibilidade de restringir o

relacionamento de funções com métodos funcionais, predicados com métodos relacionais, e acções com métodos de acção.

- Expansão do número de facetas pré-definidas – Foram adicionadas novas facetas: *Navigability*, *Arg_direction*, *Whole*, *Part*, *Dependence*, *Smallest_instance*, *Largest_instance*, *Instance_maximum_size*, *Instance_minimum_size*, entre outras. A faceta *Navigability* indica que elementos de uma associação são conhecidos pelos restantes. *Arg_direction* indica a direccionalidade de um argumento, podendo ser um argumento de entrada, saída, ou de ambos os sentidos. *Whole* especifica que o argumento de uma associação representa o todo, enquanto *Part* especifica os argumentos que constituem as partes desse todo. A faceta *Dependence* indica se uma entidade da ontologia é dependente de outras ou se é independente. As restantes facetas enumeradas anteriormente como *Smallest_instance*, foram criadas para resolver um problema de ambiguidade que existia no modelo. Por exemplo, a aplicação da faceta *Minimum_value* a uma função não exprime de forma clara se pretendemos caracterizar o conjunto denotado pela função ou o seu valor de retorno. Para resolver este problema acrescentou-se semântica às facetas M*inimum_value*, M*aximum_value*, M*inimum_size*, M*aximum_size*, e *Default_value*, ficando definido que estas facetas caracterizam o contradomínio de uma função, método funcional ou atributo. As facetas S*mallest_instance*, L*argest_instance*, I*nstance_maximum_size* e I*nstance_minimum_size*, aplicam-se a quaisquer conjuntos, incluindo os conjuntos denotados por funções, métodos funcionais e atributos.
- Melhoria da definição de novas facetas - Foram incluídos dois novos predicados relativos à definição de novas facetas, *ValidFacetElement* e *ValidFacetType*. O predicado *ValidFacetElement*/2 é utilizado para especificar os elementos a que uma faceta se pode aplicar, ao passo que *ValidFacetType*/2 especifica os valores válidos de uma faceta.
- Representação de dependências - O conceito de dependência, embora anteriormente representado no modelo O3F, foi melhorado e foram acrescentados à linguagem CO3L os predicados *Dependency*/1 e *DependencyArgument*/3, o que permite descrever relações de dependência.

A descrição do modelo realizada nas secções restantes deste capítulo já contempla todas as melhorias introduzidas no modelo e na linguagem.

3.3 Tipos

O tipo de dados é um dos conceitos mais importantes do modelo O3F, uma vez que é utilizado intensivamente nas demais definições e conceitos. Um tipo, representado pela classe *Type* na Figura 14, denota 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. Este facto é realçado no modelo O3F e é responsável pela melhor capacidade de representar um domínio através de várias potencialidades que esta definição viabiliza, por exemplo a definição de hierarquias de qualquer tipo de dados.

O modelo O3F, como praticamente todos os esquemas de representação existentes disponibiliza o tipo de dados escalar (*Scalar*) que abrange todos os tipos de dados simples entre os quais números, caracteres, palavras, *strings* e datas.

O O3F dispõe também de vários tipos de dados compostos, os quais podem ser usados na definição de outros tipos compostos. O tipo de dados *Collection* e os seus casos particulares representam uma boa variedade de colecções de objectos do mesmo tipo ou de tipos diferentes. *Collection* inclui sacos (B*ag*), conjuntos (S*et*), e sequências (S*equence*) de elementos de tipos diferentes ou do mesmo tipo de dados.

Os tipos de dados O3F incluem ainda o conceito de classe que permite a representação de tipos de dados estruturados. As classes são representadas pela classe *Class* no diagrama de classes do modelo O3F. Uma classe define um conjunto de indivíduos, os quais são por sua vez conjuntos de pares nome e valor.

O conceito de Associação, captado no O3F pela classe *Association* que relaciona classes e associações, é também um tipo de dados. Por exemplo, a associação binária CarroDaPessoa, definida no Exemplo CO3L 1, é o conjunto dos pares <Carro, Pessoa> tal que Pessoa é dona do Carro.

```
Class(Pessoa)
Class(Carro)
Association(CarroDaPessoa)
AssociationArgument(CarroDaPessoa, Proprietário, Pessoa, 1)
AssociationArgument(CarroDaPessoa, CarroDoProprietário, Carro,0..*)


Descrição:
Existe uma associação entre a classe Pessoa e Carro. Uma pessoa pode ter
zero ou mais carros. Um carro pertence exactamente a uma pessoa.
```

Exemplo CO3L 1 – Definição de classes e associações

Talvez menos evidente seja o facto de os predicados, funções, métodos relacionais e métodos funcionais denotarem conjuntos, sendo por isso considerados tipos. Por exemplo, o predicado DistânciaLocalizações, definido no Exemplo CO3L 2, representa o conjunto dos túpulos <Localização, Localização, Distância> tal que Distância é o valor da distância entre as duas localizações.

```
Class(Localização)
Class(Distância)
Predicate(DistânciaLocalizações)
Argument(Distância, Loc_1, Localização)
Argument(Distância, Loc_2, Localização)
Argument(Distância, DistânciaEntreLocalizações, Distância)


Descrição:
Relaciona duas localizações num dado espaço e a distância entre elas.
```

Exemplo CO3L 2 - Definição de Predicado

Para ilustrar a expressividade da linguagem CO3L descreve-se no Exemplo CO3L 3 a definição da classe Pessoa. Ainda neste exemplo, e no contexto dos tipos, mostram-se as potencialidades da definição de tipos novos à custa de tipos já existentes. A definição de novos tipos traz inúmeras vantagens, entre as quais:

- Possibilidade de melhorar a semântica de uma representação;
- Possibilidade de aplicar restrições ao novo tipo de dados através de facetas;
- Possibilidade de definir novos tipos que não existam no modelo.

Este género de potencialidade é extremamente importante no modelo O3F. Permite que o modelo seja extensível.

```
Class(Pessoa)
Datatype(Descricao,String)
Datype(IdadeHumana,Idade)
Attribute(Pessoa, Nome, Descricao)
Attribute(Pessoa, Idade, IdadeHumana)
Attribute(Pessoa, Sexo, Char)

Descrição:
Uma pessoa tem nome, idade e sexo.
```

Exemplo CO3L 3 - Definição de Pessoa

Ao longo deste capítulo será possível analisar mais características que testemunham a flexibilidade do modelo.

3.4 Hierarquias

A definição de hierarquias é um dos exemplos em que o modelo O3F tem claras vantagens relativamente a outros modelos. Esta vantagem é conseguida através da expressividade que deriva da forma como os tipos de dados são encarados.

A definição de hierarquias permite especificar o modo como se organiza um grupo de tipos de dados. Usualmente temos ao nosso dispor mecanismos que permitem hierarquizar tipos de dados como classes e tipos de dados básicos. No entanto, o modelo O3F tem a capacidade de hierarquizar classes, tipos básicos e consegue também hierarquizar funções, predicados e todos os tipos de dados que existem no modelo. Por exemplo, pode-se dizer que o predicado identificacao/2, o qual relaciona uma pessoa e um número de identificação, tem vários sub-predicados, por exemplo numero_bilhete_de_identidade/2, numero_de_contribuinte/2, e numer_da_seguranca_social/2, entre outros.

Além dos tipos de dados, é também possível hierarquizar acções através do estabelecimento de uma hierarquia de interfaces de acção, representadas no diagrama de classes do modelo O3F através das classes *FunctionalActionInterface* e *RelationalActionInterface*. O conceito de hierarquia de funções, predicados, associações e de acções constitui uma ferramenta de modelação poderosa que permite representar mais informação sobre o domínio que é modelado.

```
Hierarchy(ClassificaçãoSeresVivos)
Subtype(ClassificaçãoSeresVivos, Eucariota, Animal)
Subtype(ClassificaçãoSeresVivos, Eucariotas, Planta)
Subtype(ClassificaçãoSeresVivos, Eucariotas, Fungo)
Hierarchy(Domésticos_Selvagens)
Subtype(Domésticos_Selvagens, Animal, Doméstico)
Subtype(Domésticos_Selvagens, Animal, Selvagem)
```

Descrição:
Define a hierarquia dos Seres Vivos.

Exemplo CO3L 4 - Definição de Hierarquias

O Exemplo CO3L 4 ilustra a definição da hierarquia apresentada na Imagem 1.

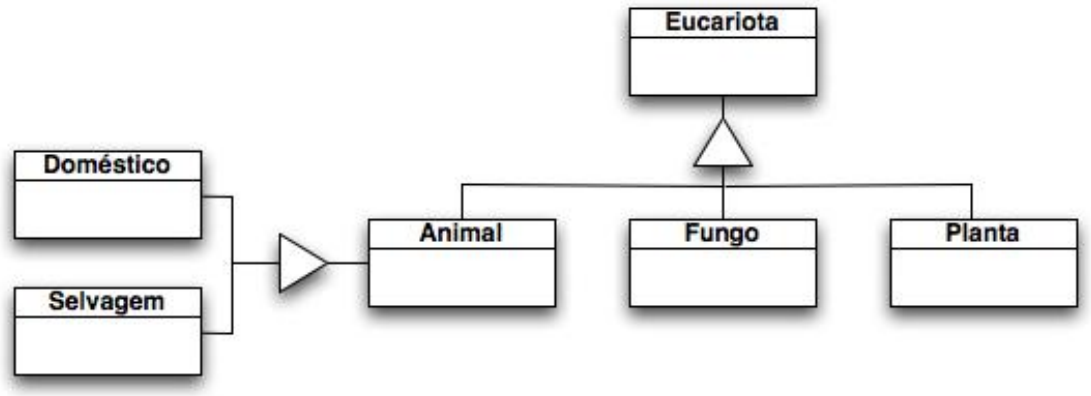

Imagem 1 – Hierarquias

Relativamente ao 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*.

3.5 Operadores

Os operadores em O3F permitem recorrer à visão relacional ou funcional do mundo. O O3F disponibiliza três tipos de operadores: predicados, funções e acções. Estes operadores estão representados no diagrama de classes do O3F pelas classes *Predicate, Function* e *Action*.

Os predicados estabelecem relações n-árias entre quaisquer tipos de dados. Comparando as associações com os predicados, conclui-se que as associações conseguem apenas relacionar classes e/ou associações mas têm a capacidade de conter atributos e métodos, capacidade que não existe nos predicados.

As funções denotam conjuntos de túpulos formados por elementos do domínio e pelos elementos correspondentes do contradomínio. Por exemplo, a função CapitalDoPaís definida no Exemplo CO3L 5, que aplicada a um país define a sua capital, representa o conjunto de pares <País, Capital> tal que o segundo elemento do par é a capital do primeiro elemento.

```
Class(Capital)
Class(País)
Function(CapitalDoPaís,Capital)
Argument(CapitalDoPaís,umPaís,País)
Descrição:
Função que devolve a capital de um país recebido como argumento.
```

Exemplo CO3L 5 - Definição de Função

Por último surge o operador mais complexo, o operador acção. Os operadores de acção, quando são executados com sucesso, produzem alterações no mundo. A descrição de um operador de acção tem dois componentes: um componente procedimental que é responsável pelas alterações que a execução da acção causa no mundo, e um componente declarativo que descreve o domínio dos objectos a que a acção se aplica e, se for caso disso, o seu contradomínio (i.e., o conjunto de valores que podem ser retornados pela acção). Actualmente, o O3F apenas descreve o componente declarativo dos operadores de acção, não existe mecanismo para definir o aspecto procedimental.

A definição declarativa do operador permite indicar os argumentos e os valores de retorno do operador. Em O3F, a descrição declarativa dos operadores de acção é chamada interface da acção. Caso em que exista valor de retorno, dizemos que a acção tem uma interface funcional, caso contrário, tem uma interface relacional. Estas interfaces são representadas pelas classes *RelationalActionInterface* e *FunctionalActionInterface*, sendo na prática particularizações das classes *Predicate* e de *Function* no diagrama de classes.

Embora não sejam considerados operadores, o modelo O3F permite representar símbolos proposicionais que são formados por uma sequência de caracteres sem quaisquer argumentos. Os símbolos proposicionais são representados pela classe *ProposicionalSymbol* captada no diagrama de classes do O3F.

3.6 Classifiers

Os *classifiers* permitem descrever elementos que partilham características e comportamentos idênticos. É nos *classifiers* que encontramos a primeira ponte relativa à representação de um domínio através do paradigma da orientação por objectos. Os *classifiers* podem ser classes ou associações, representadas pelas classes *Class* e *Association* do diagrama de classes do O3F. Tanto as classes como as associações podem ter atributos e métodos. As associações captam relações n-árias entre *classifiers*.

Um aspecto muito interessante na definição das classes reside na capacidade de definir mecanismos de identificação que permitirão a identificação unívoca das instâncias, assim como a definição das regras de pertença. Este aspecto é captado pela classe *Key* no diagrama de classes O3F.

O Exemplo CO3L 6 ilustra a representação simples de uma mota através da definição de uma classe e respectivos atributos e métodos. Este exemplo será enriquecido na secção 3.8.

```
Class(Roda)
Class(Matrícula)

Association(RodaMota)
AssociationArgument(RodaMota, roda, Roda, 2)
AssociationArgument(RodaMota, mota, Mota, 0..1)


Class(Mota)
Attribute(Mota, matrícula, Matrícula)
Attribute(Mota, númeroChassi, Integer)
ActionMethod(Mota,acelerar)
Key(Mota, identificaçãoMota, matrícula)
Key(Mota, identificaçãoMota, númeroChassi)
```

Descrição:
Uma mota tem duas rodas, tem uma matrícula e um número de chassi. Uma instância de mota é identificada univocamente pela sua matrícula e pelo seu número de chassi. Para que um indivíduo possa ser instância da classe Mota terá que pelo menos ter o atributo númeroChassi e matrícula com os mesmos tipos de dados.

Exemplo CO3L 6 - Definição de Mota

No exemplo anterior, evidencia-se que os atributos podem ter qualquer tipo de dados. Por exemplo, o atributo númeroChassi tem o tipo de dados básico *Integer*, enquanto o atributo matrícula tem o tipo de dados Matrícula, que é uma classe.

3.7 Métodos

Os métodos são a contraparte dos operadores da vertente relativa ao paradigma da orientação a objectos. O O3F disponibiliza métodos funcionais, métodos relacionais e métodos de acção.

A descrição de cada método é em tudo idêntica à descrição realizada na secção dos Operadores. Existe um claro mapeamento entre métodos funcionais e funções, métodos relacionais e predicados, e ainda entre métodos de acção e acções. No entanto é necessário perceber que os métodos são aplicados aos objectos dos *classifiers* e aos seus argumentos, caso existam. Por exemplo, o método funcional Salário da classe Trabalhador, que recebe o argumento HorasTrabalho, pode ser invocado em notação de objectos como x.Salário(160), em que x é uma instância da classe Trabalhador. Esta expressão de invocação do método é apenas a notação de objectos para a expressão Salário(x, 160). Ou seja, um método funcional com N argumentos em notação de objectos funciona como uma função de N+1 argumentos em notação funcional, sendo que o argumento adicional é o objecto ao qual o método funcional se aplicaria.

O modelo O3F, para além de possibilitar a conceptualização orientada por objectos em simultâneo com a conceptualização funcional, capta ainda a relação que existe entre estas duas abordagens. É possível definir atributos e métodos recorrendo a operadores. Por exemplo, se existir um operador funcional (i.e., uma função) que representa a conceptualização desejada (por exemplo, f), será possível dizer que uma dada classe (por exemplo C) possui um método funcional que se define à custa da função f. Este mecanismo é representado no diagrama de classes do O3F pela classe associativa *FMethodFunction*, que faz a correspondência entre um método funcional e uma função, pela classe associativa *RMethodPredicate*, que faz a correspondência entre o método relacional e um predicado, e ainda pela classe associativa *AMethodAction*, que faz a correspondência entre um método de acção e uma acção. O Exemplo CO3L 7 ilustra a utilização deste mecanismo. Ao recorrer a um operador para definir um método, está implícito que os argumentos a utilizar para o método serão exactamente os mesmos que foram definidos para o operador. No entanto existe a necessidade de especificar qual dos argumentos do operador fará o papel do objecto ao qual se aplica o método. Esta especificação é conseguida através do atributo *Self*

que consta em todas as classes associativas anteriormente descritas. Na linguagem CO3L, como é visível no Exemplo CO3L 7, a especificação do Self é conseguida através do último argumento do predicado *FMethodFunction*.

```
Class(Trabalhador)
FMethodFunction(Trabalhador,CalculaSálarioTotal,SalárioTotal,trabalhador)

Function(SalárioTotal,Float)
Argument(SalárioTotal,trabalhador,Trabalhador)
Argument(SalárioTotal,numeroHorasTrabalho,Number)

Descrição:
Definição do método funcional CalculaSalárioToral da classe Trabalhador recorrendo à
função SalárioTotal.
```

Exemplo CO3L 7 – Utilização *FMethodFunction*

Os atributos dos *classifiers* também podem ser definidos através de um mecanismo similar. É possível definir um atributo recorrendo a uma função. Este mecanismo é captado no diagrama de classes através da associação *AttributeFunction* que relaciona uma função com um atributo. O Exemplo CO3L 8 ilustra a utilização deste mecanismo.

```
Function(marcaCarro, string)
Argument(marcaCarro, carro, Carro)

Class(Carro)
AttributeFunction(Carro, marca, marcaCarro)

Descrição:
Definição do atributo marca da classe Carro através da função marcaCarro
```

Exemplo CO3L 8 – Utilização AttributeFunction

Os métodos funcionais, relacionais e de acção estão representados no diagrama de classes pelas classes *FunctionalMethod, RelationalMethod* e *ActionMethod* respectivamente.

3.8 Facetas

As facetas são um mecanismo muito poderoso, flexível e simples que permite adicionar mais características a quase todos os elementos de uma ontologia. Por pré-definição, o modelo O3F tem várias facetas que podem ser aplicadas aos elementos de uma ontologia. Além das facetas disponíveis no modelo, é possível criar novas facetas e indicar quais os seus valores possíveis e os elementos a que elas se

podem aplicar. No Exemplo CO3L 9, é possível ver a aplicação de facetas pré-definidas que visam tornar o Exemplo CO3L 6 mais completo.

```
Association(RodaMota)
AssociationArgument(RodaMota, roda, Roda, 2)
AssociationArgument(RodaMota, mota, Mota, 0..1)

Class(Mota)
Attribute(Mota, matrícula, Matrícula)
Attribute(Mota, númeroChassi, Integer)

EntityFactet(RodaMota, Whole, Mota)
EntityFactet(RodaMota, Part, Roda)
EntityFactet(RodaMota, Association_type, Aggregation)

Descrição:
Uma mota tem duas rodas, tem uma matrícula e um número de chassi. Através da aplicação das facetas Whole, Part e Association_Type foi possível indicar que as rodas são independentes da mota porque o tipo de associação é uma agregação. É possível saber que na associação RodaMota, a mota é o todo e as rodas são a parte.
```

Exemplo CO3L 9 - Definição de Mota com facetas pré-definidas

No Exemplo CO3L 10 é possível ver a criação de novas facetas. Neste exemplo será possível analisar a flexibilidade das facetas assim como o seu poder de abstracção. É possível indicar que uma faceta só pode ser aplicada a um elemento da ontologia como por exemplo: classe pessoa, atributo nome, função salário, ou outro qualquer elemento que já exista na ontologia. Outra forma de restringir a aplicação de facetas é recorrendo aos meta-elementos do modelo O3F.

```
Facet(Descrição)
ValidFacetElement(Descrição, Element)
ValidFacetValue(Descrição, String)

Class(Pessoa)
EntityFacet(Pessoa, Descrição, "Aqui descrevemos a classe pessoa...")

Descrição:
Foi criada a faceta Descrição que pode ser aplicada a qualquer elemento da ontologia. O valor possível para a faceta será o tipo de dados String dada a sua expressividade para descrever algo textualmente. Esta faceta poderia ser utilizada, por exemplo, para gerar automaticamente comentários nas traduções de CO3L para outros formatos.
```

Exemplo CO3L 10 – Definição de uma nova faceta

As facetas são captadas no diagrama de classes do modelo O3F pela classe *Facet.*

3.9 Dependências

Em O3F é possível especificar as relações de dependência que existem entre os vários elementos da ontologia. Por exemplo, é possível indicar que existe uma dependência entre associações ou que existe uma dependência entre métodos como ilustra o Exemplo CO3L 11. Para uma especificação mais rica pode-se utilizar a faceta *Dependence* para indicar se um argumento da dependência é dependente ou independente.

```
Class(ContaBancária)
ActionMethod(ContaBancária, debitaMontante)
Argument(ContaBancária.debitaMontante, montante, Number)

Class(Permissões)
FunctionalMethod(Permissões, validaAcesso, Boolean)
Argument(Permissões.validaAcesso, utilizador, Utilizador)

Dependency(PermissãoMovimento)
DependencyArgument(PermissãoMovimento, movimento, ContaBancária)
DependencyArgument(PermissãoMovimento, permissão, Permissões)

EntityFacet(ContaBancária, Dependence, Dependent)
EntityFacet(Permissões, Dependence, Independent)


Descrição:
Existe uma dependência entre a classe ContaBancária e a classe Permissões. Os métodos da
classe ContaBancária chamam métodos da classe Permissões para efectuar validações de
segurança. Esta ligação confere uma dependência da classe ContaBancária face à classe
Permissões.
```

Exemplo CO3L 11 – Utilização de dependências

Actualmente, as relações de dependência e a faceta Dependence dizem muito pouco relativamente à natureza de uma dependência. A descrição de dependências terá uma maior expressividade com a inclusão de axiomas no modelo e na linguagem. Com a inclusão de axiomas será possível detalhar de que forma um elemento depende de outro.

As dependências e os seus argumentos são captados no diagrama de classes do modelo O3F pelas classes *Dependency* e *DependencyArgument*, respectivamente.

3.10 Indivíduos

O O3F representa indivíduos através de conjuntos de pares de nome e valor. Um indivíduo pode ser instância de um ou mais tipos de dados. No entanto, não é obrigatório que um indivíduo seja uma instância. Para que um indivíduo possa ser

instância de classes ou associações, deverá conter os atributos obrigatórios definidos pelo respectivo *Classifier*. Para ilustrar a expressividade e facilidade da definição de indivíduos e instâncias, apresentam-se os seguintes exemplos:

ObjectDef(id001, set(bi=8265101, nome = "Manuel", morada = "Zona"))

Descrição:
Definição de um indivíduo. Esta especificação não diz que o indivíduo seja uma instância, ainda que possam existir tipos de dados estruturados (e.g., classes, associações ou funções) com os mesmos atributos que foram utilizados na descrição do indivíduo.

Exemplo CO3L 12 – Definição de um indivíduo

No Exemplo CO3L 12 é definido um indivíduo. Para que o indivíduo possa ser considerado também uma instância, é necessário que se use o predicado *Instance*. Para que a definição seja válida, é necessário que o indivíduo respeite as restrições de pertença ao tipo de dados definido no predicado *Instance*. O Exemplo CO3L 13 ilustra um exemplo da definição de um indivíduo e de uma instância.

ObjectDef(id001, set(bi=8265101, nome = "Manuel", morada="Zona"))
Instance(#id001,Pessoa)

Class(Pessoa)
Attribute(Pessoa, nome, String)
Attribute(Pessoa, bi, Integer)
Attribute(Pessoa, pai, Pessoa)
Attribute(Pessoa, mãe, Pessoa)
EntityFacet(Pessoa.bi, Mandatory, True)

Descrição:
Definição de um indivíduo que é instância da classe Pessoa. Como o indivíduo tem o atributo bi, respeita a regra de pertença da classe pessoa que indica que para um indivíduo ser uma instância da classe Pessoa deve ter o atributo bi.

Exemplo CO3L 13 – Definição de um indivíduo e de uma instância

Os indivíduos são captados pela classe *Individual* no diagrama de classes O3F e as instâncias são captadas pela associação entre a classe *Individual* e a classe *Type.*

3.11 Axiomas

Os axiomas conferem uma expressividade muito grande aos modelos e linguagens de representação de ontologias. Há relações, definições e restrições que se podem representar através de axiomas. Imagine-se, por exemplo a classe Pessoa com os atributos género, pai e mãe, entre outros. Podem usar-se dois axiomas para dizer, por exemplo, que o género do pai de uma pessoa é masculino e que o género da mãe de

uma pessoa é feminino. Usando a lógica de predicados de primeira ordem, poderíamos expressar estes axiomas como se segue.

$\forall x$ Instance(x, Pessoa) $\Rightarrow$ x.pai.género = masculino

$\forall x$ Instance(x, Pessoa) $\Rightarrow$ x.mãe.género = feminino

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 Análise, Concepção e Implementação do O3 Server

Ao longo deste capítulo é possível analisar o O3 Server, o primeiro servidor de ontologias O3F. Os objectivos do projecto relativamente ao O3 Server são descritos na secção 4.1, descrevendo-se posteriormente os casos de uso na secção 4.2. A arquitectura da ferramenta é detalhada na secção 4.3. As secções 4.4 e 4.5 descrevem a modulação que foi realizada para o O3 Server e para a API O3 Server, respectivamente. Por fim, a secção 4.6 descreve alguns aspectos importantes na implementação do O3 Server e da API O3 Server e termina com a conclusão do capítulo na secção 4.7.

4.1 Objectivos

Uma das contribuições mais importantes da presente dissertação é a análise, modelação e construção do primeiro servidor de ontologias capaz de suportar o O3F. Este servidor chama-se O3 Server. Salienta-se que a construção do O3 Server é realizada em parceria com a dissertação de Jairo Avelar (Avelar, 2010).

O servidor de ontologias deverá suportar a publicação, manutenção e pesquisa de ontologias em base de dados. As ontologias publicadas deverão estar acessíveis a qualquer aplicação cliente.

Para facilitar a integração de futuros clientes Java com o O3 Server, foi criada uma API que possibilita executar as operações elementares como inserir, apagar, editar e pesquisar ontologias. Esta API deverá também suportar qualquer outro serviço que venha futuramente a ser criado no servidor de ontologias. A API de acesso ao O3 Server foi desenvolvida exclusivamente no contexto desta dissertação.

4.2 Casos de utilização

Analisar os casos de utilização de um sistema permite desprender o sistema de uma implementação e ter a percepção global das necessidades e funcionalidades que devem vigorar no sistema. A Figura 1 evidencia que existem três grandes funcionalidades ligadas ao O3 Server: gestão de ontologias; gestão do servidor; consumo de serviços.

A gestão de ontologias engloba todos os mecanismos que possibilitam que sistemas exteriores publiquem, consultem, alterem e removam ontologias do O3 Server. A disponibilização de serviços está instanciada na Figura 1 com o único exemplo que está previsto para o O3 Server, serviço este que devolve uma ontologia textual na linguagem CO3L. No futuro poderão ser adicionados mais serviços, por exemplo, serviços que traduzam as ontologias publicadas no O3 Server para linguagens como OWL e UML. Por último existe o caso de utilização relacionado com a própria gestão do O3 Server, funcionalidades como backup e recuperação do servidor, atribuição de permissões e criação de utilizadores fazem parte deste caso de utilização.

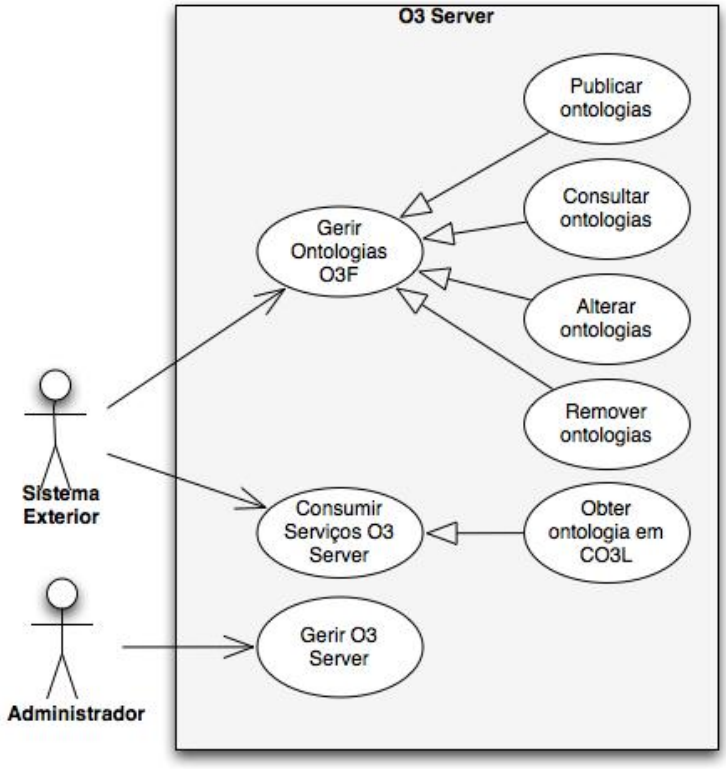

Figura 1 – Casos de utilização do O3 Server

A Figura 1 ilustra ainda dois actores principais: Sistema exterior e administrador. O sistema exterior está relacionado com qualquer sistema computacional que pretenda comunicar com o O3 Server. A presente dissertação desenvolveu um cliente que comunica com o O3 Server, o qual é descrito no Capítulo 5 . O administrador é um utilizador humano que interage directamente com o O3 Server para o gerir.

4.3 Arquitectura

O O3 Server é um servidor, publicamente acessível, desenhado de acordo com uma arquitectura cliente-servidor. O servidor tem como principais responsabilidades armazenar as ontologias e disponibilizar as ontologias. O modo como o servidor armazena as ontologias é independente da linguagem CO3L e de qualquer linguagem que venha a ser criada para o modelo O3F.

O O3 Server é constituído fundamentalmente por dois componentes: uma base de dados onde as ontologias são armazenadas, de onde podem ser extraídas, consultadas, alteradas e removidas; e uma interface que pode ser usada por aplicações cliente para solicitar operações ao O3 Server.

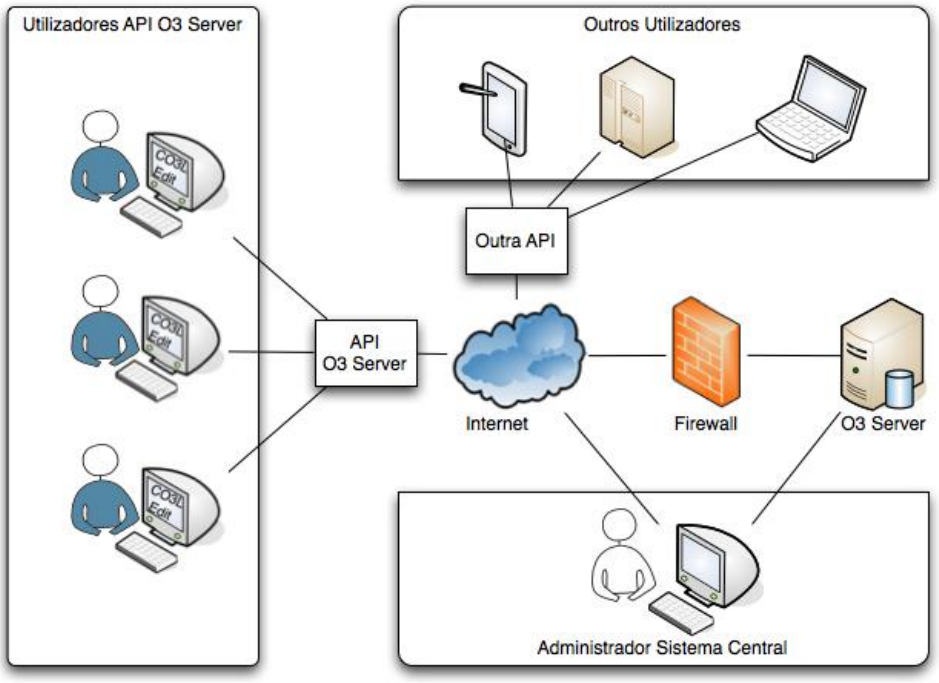

Figura 2 – Arquitectura do O3 Server

Apesar da presente dissertação desenvolver uma API para auxiliar a integração de clientes com o O3Server, é possível comunicar com o O3 Server sem recorrer a esta API. Tendo em conta que o O3 Server é um servidor de base de dados, é possível a qualquer cliente utilizar um *driver* JDBC para comunicar com o servidor e executar as acções pretendidas. No entanto a API O3 Server fornece de imediato uma abstracção relativamente às acções mais importantes, por exemplo inserir, consultar, eliminar e alterar.

Com uma arquitectura cliente-servidor é possível separar efectivamente o modelo da linguagem. O facto de o modelo ficar isolado permitirá a criação de vários serviços que não ficarão acoplados a nenhuma linguagem. Por exemplo, será possível criar mecanismos de inferência, tradutores e pesquisadores recorrendo exclusivamente à abstracção do modelo O3F.

4.4 Modulação do O3 Server

Tendo como base principal o diagrama de classes do modelo O3F e as características da linguagem CO3L, o modelo relacional que consta no Anexo B resultou de uma cuidada modelação que foi realizada manualmente seguindo as regras de transposição e respectivas optimizações que existem para transformar um modelo de classes num modelo relacional. A transposição podia ter sido realizada semi-automaticamente recorrendo a ferramentas como o EA[7] ou o PowerDesigner[8], no entanto os custos associados à aquisição destas ferramentas, a curva de aprendizagem para trabalhar com elas e o tempo dispendido para procurar a melhor ferramenta torna a solução de transposição manual mais viável. Mais ainda, a transposição manual permite gerar o modelo relacional de acordo com um objectivo e contexto específico. A aplicação das regras de transposição sem qualquer contexto origina muitas vezes modelos relacionais ineficientes (Bennet, McRobb, & Farmer, 1999).

O desenho do modelo relacional teve como um dos principais objectivos a performance. Esta preocupação reflecte-se em algumas das opções tomadas durante o seu desenho. As chaves primárias criadas são todas numéricas para tornar a pesquisa mais rápida. As chaves candidatas que existiam eram na sua maioria chaves com tipo de dados textual e muitas vezes chaves compostas, o que iria certamente degradar a performance. A parte do diagrama do modelo relacional que está representada na Figura 3 mostra também a aplicação de uma das regras de transposição (optimização de generalizações) que visa diminuir o número de tabelas resultantes.

[7] www.sparxsystems.com.au
[8] http://www.sybase.com/products/modelingdevelopment/powerdesigner

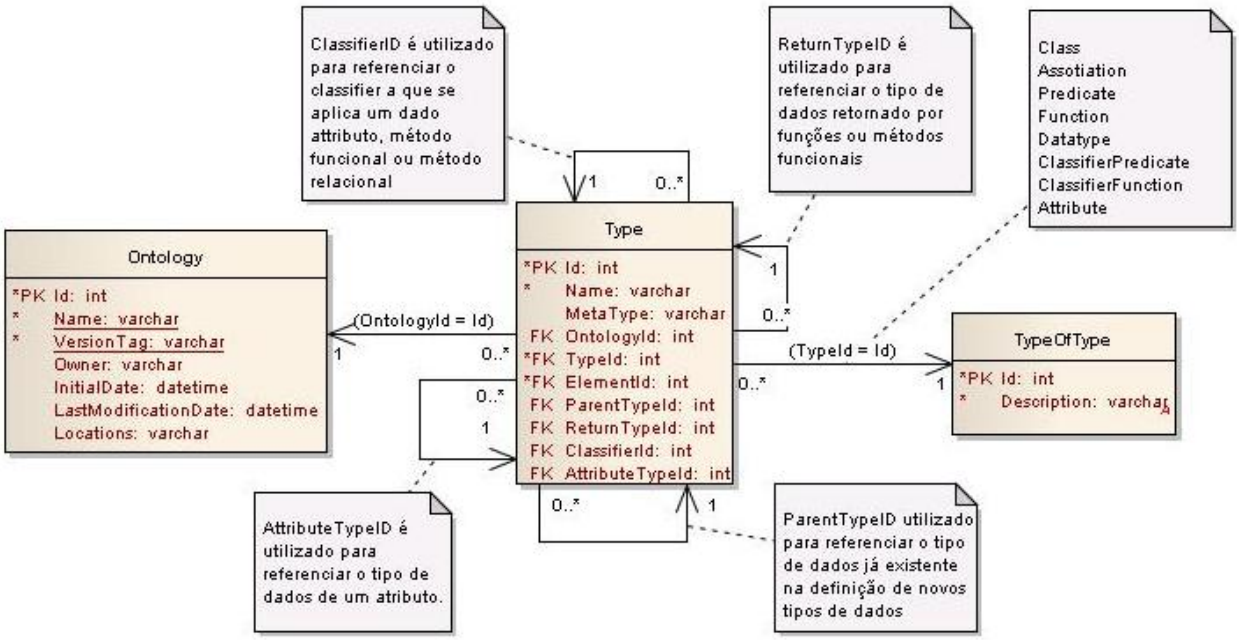

Figura 3 – Modelo Relacional da tabela Type

A Tabela *Type* é a tabela mais complexa do modelo. No modelo de classes O3F existe uma classe para cada tipo de dados, no entanto no modelo relacional existe apenas a tabela *Type* que aglomera todos os tipos de dados existentes, por exemplo funções, predicados, métodos funcionais e métodos relacionais. A aplicação desta optimização traz grandes ganhos na simplificação do modelo relacional já que a representação dos tipos de dados perfaz cerca de 40% de todo o diagrama de classes O3F. Com esta optimização o número de *Joins* necessários para interrogar a base de dados será muito menor. Estas optimizações simplificam e optimizam a interligação com *frameworks* de ORM (*Object Relational Mapping*), uma vez que diminuem a complexidade do mapeamento e aumentam a rapidez das operações de consulta e inserção dos dados (Minter & Linwood, 2005).

A optimização dos tipos de dados no diagrama relacional face ao diagrama de classes do O3F foi a principal transformação e optimização que foi realizada. As restantes classes do diagrama de classes tiveram, na sua maioria, uma transposição directa para o modelo relacional como ilustra a Figura 4.

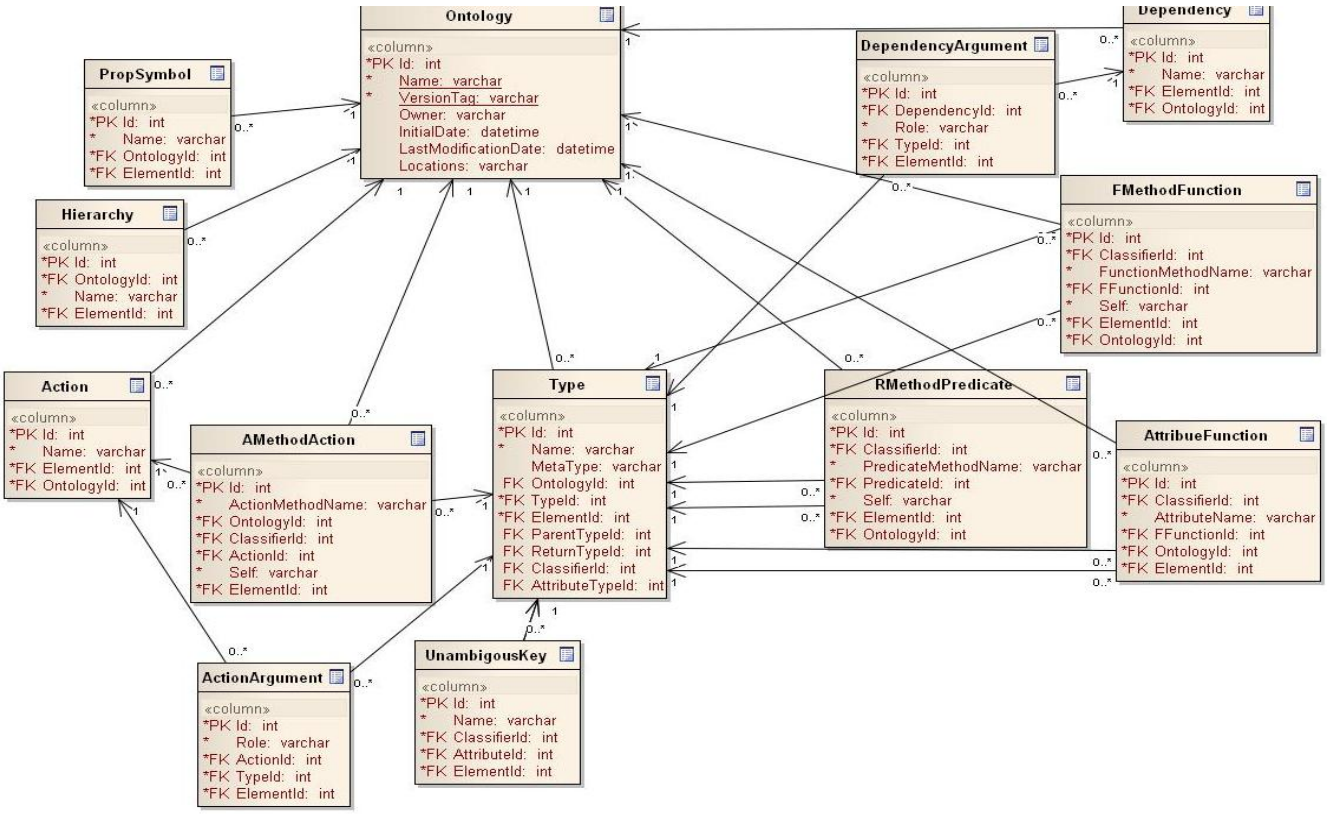

Figura 4 – Parte do diagrama relacional

A tabela Ontology representa precisamente o mesmo que a classe Ontology no modelo de classes O3F. Serve para identificar univocamente cada ontologia e todos os seus elementos.

De igual modo, as tabelas *RMethodPredicate, FMethodFuncion* e *AMethodAction* têm a mesma função que as classes com o mesmo nome no diagrama de classes O3F. Estas tabelas guardam as relações entre métodos e operadores. Na mesma óptica, a tabela *AttributeFunction* guarda a relação entre funções e atributos.

As acções e os métodos de acção, não sendo tipos de dados, são representados pelas tabelas *Action* e *ClassifierAction,* respectivamente. Já os argumentos destes elementos são representados pelas tabelas *ActionArgument* e *ClassifierActionArgument*.

A tabela *Element* foi criada para espelhar a classe *Element* no diagrama de classes O3F. Esta tabela é referenciada por todas as tabelas existentes no modelo relacional que guardam informação relativa a elementos de uma ontologia. Esta tabela torna possível a aplicação de facetas a qualquer elemento da ontologia e possibilita também a utilização de qualquer elemento na definição de dependências.

As facetas são assim guardadas na tabela *Facet* e o registo das facetas aplicadas a um elemento é feito na tabela *EntityFacet*. Para que seja possível saber a que tipos de dados se pode aplicar uma faceta, foi criada a tabela *ApplicableFacetType*. Relativamente às dependências, estas são registadas na tabela Dependency e os seus argumentos são registados na tabela *DependencyArgument*, espelhando directamente o modelo de classes O3F.

Como tem sido demonstrado, o modelo relacional espelha muitas características do modelo de classes O3F, no entanto a modulação resultante dos objectos (i.e., indivíduos) difere da do diagrama de classes. No modelo relacional os objectos são registados na tabela *ObjectDefinition* e os atributos dos objectos são registados na tabela *ObjectDefinitionAttribute*. Finalmente, a tabela *Instance* serve para registar os objectos que são instâncias de um dado tipo de dados. Esta modulação é mais rica face ao modelo de classes O3F pois permite pesquisas mais complexas, por exemplo, possibilita saber quais os objectos que partilham propriedades semelhantes ou quais as instâncias que têm um atributo com um dado valor.

Relativamente às regras de transposição, a regra mais frequentemente utilizada foi a regra da associação de "um para muitos", dado que uma ontologia pode ter vários elementos. No entanto foram também utilizadas outras regras. Foi utilizada a regra da associação de "um para um", no caso da tabela *Element*, que se relaciona com outras tabelas. Foi ainda utilizada a regra da transposição de classes associativas, por exemplo no caso das facetas. No caso das optimizações foi utilizada a regra de optimização de generalizações (Bennet, McRobb, & Farmer, 1999).

4.5 Desenho da API O3 Server

A API O3 Server foi desenvolvida para fornecer uma camada de acesso ao O3 Server. Deste modo novas ferramentas que venham a ser desenvolvidas em Java não precisam de implementar uma camada de acesso ao O3 Server, podem simplesmente utilizar esta API.

Esta API, por ter sido elaborada com uma *framework* de ORM, oferece as seguintes vantagens:

- Abstracção da tecnologia de base de dados utilizada;
- Conciliação da performance com a abstracção;

- Abstracção da necessidade de gerir as ligações à base de dados;
- Abstracção da necessidade de escrever comandos rotineiros em SQL;
- Agilidade na construção da camada de acesso aos dados.

De facto as *frameworks* ORM permitiram agilizar e melhorar a ligação às bases de dados (Minter & Linwood, 2005). Através do sistema de mapeamento, a construção da camada de acesso à base de dados torna-se ágil e dá origem a uma estrutura robusta. Cada tabela no modelo relacional corresponde a uma classe em Java e a um ficheiro XML. Apesar do presente trabalho ter realizado um mapeamento directo, é possível criar classes Java que não têm uma correspondência directa na base de dados. Por exemplo, é possível criar um conjunto de classes Java que representam fielmente o modelo de classes O3F e posteriormente fazer o mapeamento para o modelo relacional existente. Salienta-se que o mapeamento directo é mais fácil e rápido de implementar porque é possível recorrer a ferramentas[9] que geram as classes Java necessárias e os ficheiros XML de uma forma semi-automática.

Internamente a API O3 Server implementa ainda um conjunto de classes que são responsáveis por gerir o código que interroga a base de dados. Esta técnica é um padrão de desenho e tem como nome *Data Access Object* (Minter & Linwood, 2005).

[9] http://www.hibernate.org/subprojects/tools.html

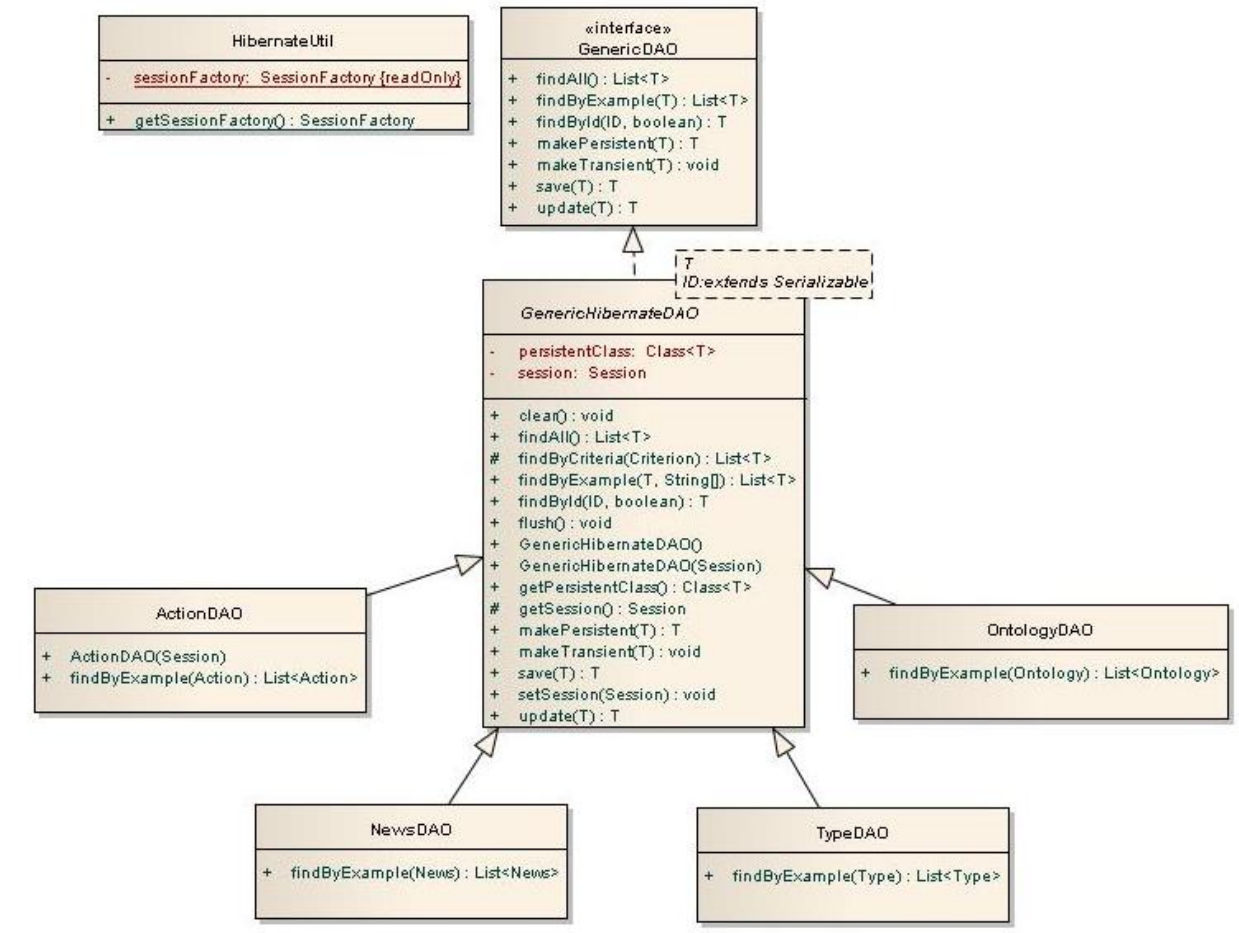

Figura 5 – Imagem DAO da API O3 Server

A Figura 5 ilustra a estrutura dos *Data Access Object* (DAO). Para efeitos de simplicidade apenas são referenciadas as entidades News, Type Ontology e Action, mas todas as entidades presentes no modelo relacional têm o seu respectivo DAO. Como é visível, é extremamente fácil estender esta API caso o modelo relacional seja alterado. Basta estender a classe GenericHibernateDAO que providencia implementação genérica para inserir, consultar, alterar e remover registos. Resumindo a estrutura da API, para cada tabela vigente no modelo relacional existem três ficheiros. Por exemplo, para a tabela PropSymbol existirão três ficheiros: PropSymbol.java; PropSymbol.hbm.xml; PropSymbolDAO.java.

O único problema da utilização de *frameworks* ORM está na curva de aprendizagem que é exigida. Estas *frameworks* têm muitos detalhes que é necessário perceber profundamente para assegurar o correcto funcionamento de todo o sistema.

4.6 Implementação

O O3 Server assenta sobre uma plataforma de base de dados, tendo sido utilizado o Microsoft SQL Server 2008. Este RDBMS guardará todas as ontologias que forem publicadas pelos mais diversos clientes, será responsável por disponibilizar todas as ontologias e deverá estar sempre acessível. O Microsoft SQL Server 2008 é uma solução robusta. Nos *benchmarks* realizados pelo Transaction Processing Performance Council (TPC), o Microsoft SQL Server 2008 aparece regularmente em primeiro lugar, face ao Oracle, ao MySQL e ao IBM DB2 (Council, 2010). Salienta-se ainda que o ISCTE-IUL, o instituto onde se realizou o presente trabalho e onde O3 Server está hospedado, dispõe do Microsoft SQL Server 2008.

A base de dados do O3 Server foi implementada recorrendo ao Microsoft SQL Server Management Studio. Esta ferramenta permitiu criar a estrutura de todas as tabelas assim como definir a ligação das tabelas através das respectivas chaves. Numa óptica de gestão de base de dados, esta ferramenta permite gerir e auditar os acessos à base de dados, criar perfis, criar utilizadores e providencia funcionalidades para realizar a manutenção da base de dados. O Microsoft SQL Server Management Studio assegura assim todo o trabalho relativo à gestão das bases de dados.

Para criar uma API que forneça uma integração flexível, transparente e robusta com o O3 Server, recorreu-se à *framework* Hibernate 3. O Hibernate 3 é a solução mais flexível e poderosa que existe no mercado para armazenar e obter objectos Java de uma base de dados, sendo também o ORM mais fácil de aprender (Minter & Linwood, 2005). Com o Hibernate, as aplicações não ficam dependentes de nenhuma tecnologia de base de dados. Por exemplo, no futuro o O3 Server poderá estar assente sobre tecnologias Oracle ou MySql e os clientes irão funcionar correctamente, necessitando apenas de mudar um parâmetro, o que confere uma enorme flexibilidade aos clientes e ao O3 Server. O cliente e o servidor estão assim fracamente acoplados.

Conforme foi mencionado na secção 4.5, a API O3 Server tem, na sua constituição, três ficheiros distintos para cada tabela existente no modelo relacional. Por exemplo, a tabela PropSymbol corresponde a uma classe PropSymbol.java, a uma classe PropSymbolDao.java e a um ficheiro PropSymbol.hbm.xml.

```
<hibernate-mapping>
   <class name="modelO3fGenerated.PropSymbol" table="PropSymbol" schema="dbo"
catalog="O3FModel">
      <id name="id" type="int">
         <column name="Id" />
         <generator class="increment"/>
      </id>
      <many-to-one cascade="all" name="element" class="modelO3fGenerated.Element"
fetch="select">
         <column name="ElementId" not-null="true" />
      </many-to-one>
      <many-to-one name="ontology" class="modelO3fGenerated.Ontology" fetch="select">
         <column name="OntologyId" not-null="true" />
      </many-to-one>
      <property name="name" type="string">
         <column name="Name" length="50" not-null="true" />
      </property>
   </class>
</hibernate-mapping>
```

Ficheiro Hibernate - Mapeamento classe PropSymbol

O ficheiro XML define o mapeamento que existe entre a classe Java PropSymbol e a tabela PropSymbol.

```
public class PropSymbol  implements java.io.Serializable {

    private int id;
    private Element element;
    private Ontology ontology;
    private String name;

   public PropSymbol() {
   }
   public PropSymbol(int id, Element element, Ontology ontology, String name) {
      this.id = id;
      this.element = element;
      this.ontology = ontology;
      this.name = name;
   }
  public int getId() {
       return this.id;
   }
   public void setId(int id) {
       this.id = id;
   }
//…outros métodos de Get e Set para as restantes variáveis…
}
```

Ficheiro Java - Classe PropSymbol

Como é visível, a classe PropSymbol tem apenas o código necessário para representar o símbolo proposicional, não sendo responsável por interagir com a base de dados. A classe responsável por interagir com a base de dados será a classe PropSymbolDao.java que herda os métodos básicos de acesso ao O3 Server através da classe abstracta GenericHibernateDao.java.

Relativamente ao protocolo de comunicação, o Microsoft SQL Server 2008 implementa o protocolo aplicacional Tabular Data Stream[10] (TDS). Tendo em conta que a plataforma Microsoft SQL Server é extremamente utilizada por diferentes organizações e por diferentes linguagens, existem vários *drivers* que implementam o protocolo TDS nas mais diversas linguagens. No entanto a utilização directa dos *drivers* de acesso à base de dados carece de um trabalho inicial árduo e de difícil manutenção, pelo que é preferível recorrer a técnicas mais ágeis como a utilização de *frameworks* ORM.

4.7 Conclusão

O O3 Server foi desenvolvido para armazenar e disponibilizar ontologias O3F. Oferece um nível de abstracção elevado, uma vez que não guarda as ontologias sobre a forma de texto, dependentes de uma linguagem. O O3 Server está assente sobre a plataforma MS SQL Server 2008 e contém uma base de dados que se baseia no diagrama de classes do modelo O3F, cujo modelo relacional está no Anexo B.

A presente dissertação desenvolveu também uma API para aceder ao O3 Server. Deste modo todas as ferramentas desenvolvidas em Java podem comunicar com o O3 Server através desta API sem desenvolver qualquer código. Actualmente já existem duas ferramentas que utilizam esta API com sucesso: CO3L Edit (ferramenta desenvolvida no contexto da presente dissertação) e UML2O3F (Avelar, 2010).

[10] http://msdn.microsoft.com/en-us/library/cc448435.aspx

Capítulo 5 Análise, Concepção e Implementação do CO3L Edit

Ao longo deste capítulo será possível analisar o CO3L Edit, o primeiro cliente a utilizar todas as funcionalidades disponibilizadas pelo O3 Server e a primeira ferramenta a suportar o desenvolvimento de ontologias na linguagem CO3L. Os objectivos da aplicação estão descritos na secção 5.1. As secções 5.2 e 5.3 descrevem os casos de uso e a arquitectura da aplicação, respectivamente. A secção 5.4 descreve a modulação, os algoritmos e as heurísticas utilizadas no CO3L Edit e a secção 5.5 foca-se na modulação e na descrição de algoritmos utilizados no processamento da linguagem CO3L. Finalmente, a secção 5.6 descreve aspectos ligados à própria implementação da aplicação e a secção 5.7 termina este capítulo com uma breve conclusão.

5.1 Objectivos

O objectivo subjacente ao CO3L Edit é a construção de uma aplicação gráfica que permita o desenvolvimento de ontologias CO3L e posterior publicação das ontologias no servidor de ontologias O3 Server.

Os objectivos do CO3L Edit estão ainda divididos em vários sub-objectivos mais específicos, como a criação de mecanismos gráficos capazes de auxiliar o utilizador a escrever ontologias CO3L, a validação sintáctica da especificação de ontologias e a visualização em CO3L de ontologias armazenadas no O3 Server. O armazenamento das ontologias assim como todos os mecanismos que necessitam de interagir com o O3 Server serão realizados através da API O3 Server.

5.2 Casos de utilização

Para que a arquitectura e respectiva implementação da aplicação sejam derivadas de uma forma transparente fez-se a análise de casos de utilização, apresentado na Figura 6.

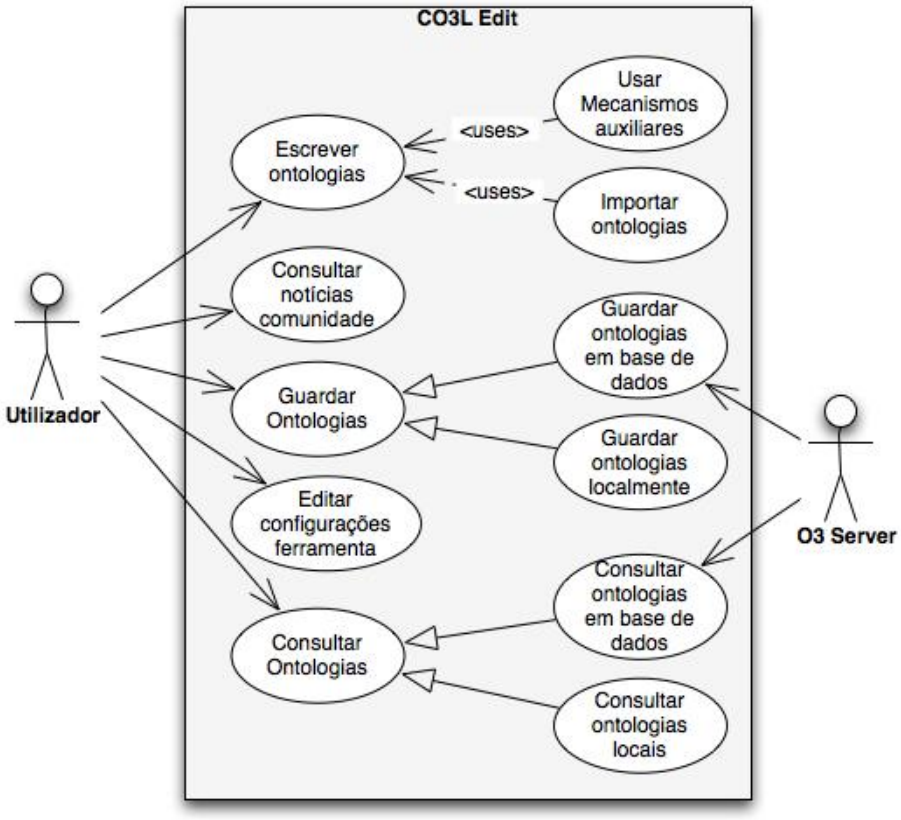

Figura 6 – Casos de utilização do CO3L Edit

Através da especificação de casos de uso é possível perceber que um dos requisitos impostos ao desenvolvimento do CO3L Edit é a possibilidade de armazenar ontologias em ficheiros locais ou alternativamente em base de dados, no O3 Server. Outro requisito é a parametrização da aplicação para indicar o endereço e credenciais de acesso ao O3 Server e parametrizar aspectos relativos à própria interface gráfica. Relativamente ao desenvolvimento de ontologias, um requisito importante é o de facilitar a visualização de ontologias e auxiliar o utilizador a detectar e corrigir os erros que advêm de uma incorrecta especificação.

5.3 Arquitectura

A arquitectura do CO3L Edit foi desenhada de forma bastante modular. O CO3L Edit é constituído por um conjunto de módulos que comunicam entre si. Este tipo de arquitectura traz vantagens na extensibilidade da ferramenta, pois permite substituir cada um dos módulos por um módulo com interfaces idênticas mas com comportamentos que podem ser distintos.

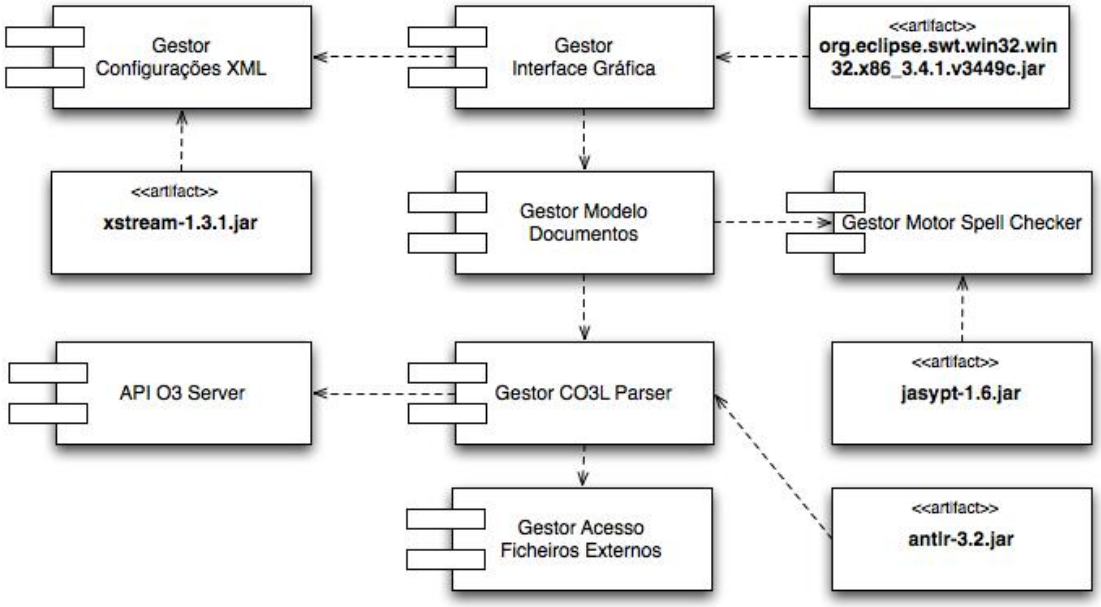

Figura 7 – Diagrama de Componentes do CO3L Edit

Como é visível na Figura 7, não existe um grande emaranhamento nas relações entre os módulos da aplicação, deste modo a manutenção e evolução ficam facilitadas.

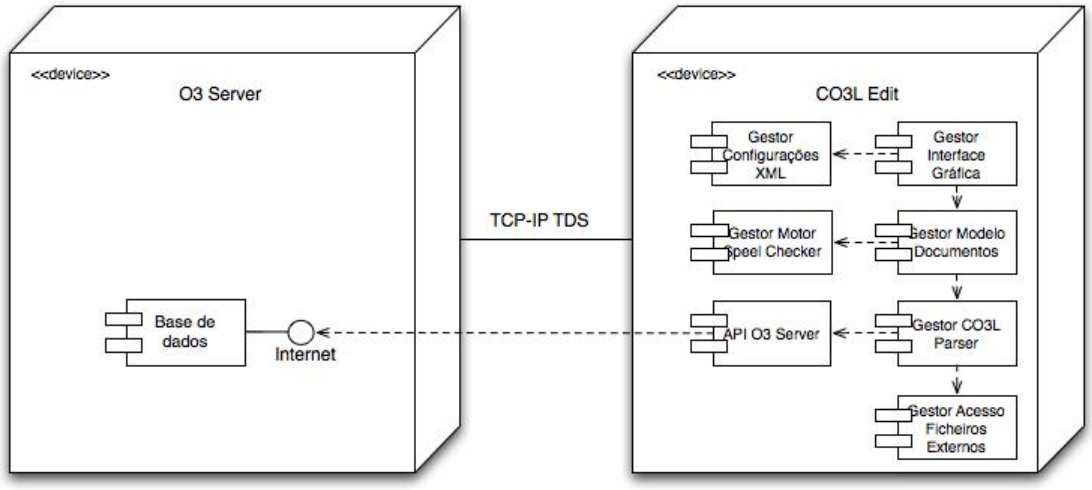

Figura 8 – Diagrama Físico do O3 Server e do CO3L Edit

Através da Figura 8 é possível perceber a ligação que existe entre o CO3L Edit e o O3 Server. A componente API O3 Server assegura a comunicação entre as duas ferramentas, possibilitando que toda a lógica e regras do modelo O3F fiquem num local bastante isolado.

Ainda que não faça parte dos planos iniciais e da arquitectura do CO3L Edit, enquanto ferramenta, é interessante analisar que a funcionalidade de importação de ontologias recorrendo aos protocolos HTTP e FTP pode dar origem a uma arquitectura descentralizada. Ao invés de guardar as ontologias num repositório central, as ontologias podem ser alojadas em ficheiros de texto CO3L em diferentes servidores. Deste modo as ontologias poderiam ser consultadas e partilhadas a uma escala global e descentralizada. No entanto a funcionalidade de importação de ontologias pretende apenas contemplar a noção de reutilização de ontologias por motivos de escalabilidade.

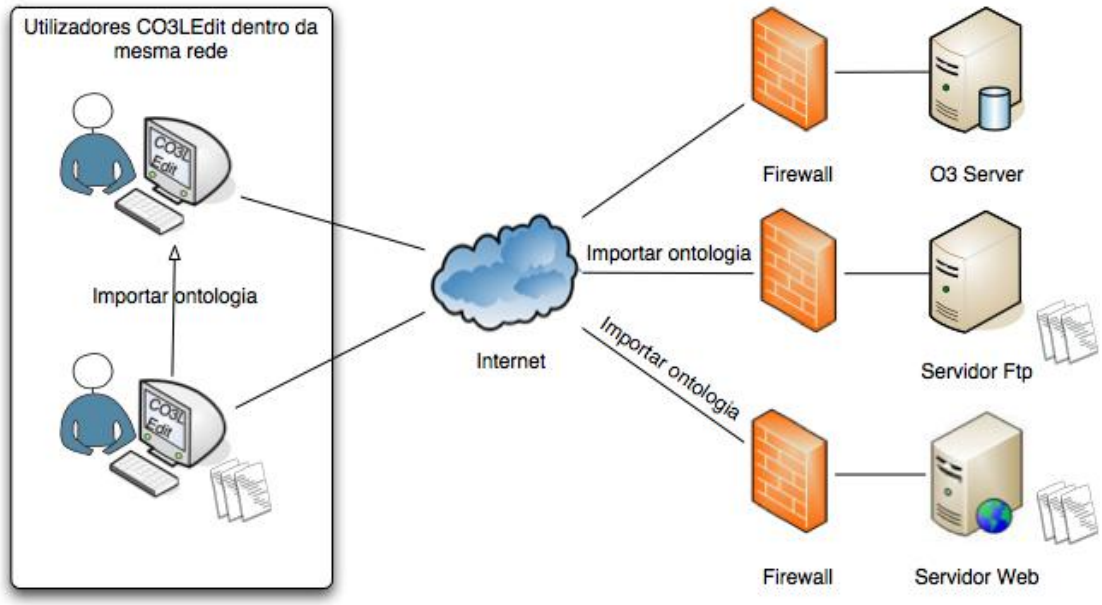

Figure 9 - Possível arquitectura descentralizada do CO3L Edit

Em forma de conclusão, a arquitectura do CO3L Edit mostra-se modular, organizada e aberta a extensões.

5.4 Funcionalidades, desenho e heurísticas do CO3L Edit

O CO3L Edit assume-se actualmente como a aplicação nuclear para a linguagem CO3L. A Tabela 3 resume as funcionalidades mais importantes do CO3L Edit que são visíveis pelos utilizadores.

Nome Funcionalidade	Descrição Funcionalidade
Gestão de projectos e ontologias locais	O utilizador tem a possibilidade de criar projectos e ontologias que são estruturadas de acordo com a sua metodologia de trabalho. Nesta macro funcionalidade estão incluídas as seguintes funcionalidades: • Criar e apagar projectos locais; • Criar e apagar ontologias locais; • Copiar e colar ontologias; • Navegar pelos projectos locais; • Abrir ontologias locais para edição ou consulta.
Desenvolvimento de ontologias CO3L	O utilizador tem a capacidade de desenvolver ontologias recorrendo à linguagem CO3L. Nesta macro funcionalidade estão incluídas as seguintes funcionalidades: • Possibilidade de escrever ontologias CO3L; • Análise sintáctica das ontologias CO3L em tempo real; • Funcionalidades auxiliares à escrita de ontologias como destaque de erros, destaque da sintaxe e corrector ortográfico. As funcionalidades de destaque de erros e de destaque da sintaxe são realizadas em tempo real; • Possibilidade de importar ontologias remotas recorrendo aos protocolos FTP e HTTP; • Possibilidade de importar ontologias locais; • Capacidade de guardar ontologias localmente, em ficheiros do sistema operativo hospedeiro; • Capacidade de publicar ontologias no O3 Server; • Capacidade de copiar e colar partes de ontologias recorrendo ao *clipboard;* • Mecanismo de *Undo* e *Redo.*

Nome Funcionalidade	**Descrição Funcionalidade**
Consulta de ontologias publicadas no servidor O3F	Nesta macro funcionalidade o utilizador terá a possibilidade de consultar ontologias que estão publicadas no O3 Server. O utilizador terá disponíveis as seguintes funcionalidades: • Listagem de todas as ontologias publicadas no O3 Server, indicando o nome e versão; • Capacidade de seleccionar e visualizar uma ontologia específica; • Funcionalidade visual de destaque da sintaxe*;* • Capacidade de copiar e colar partes da ontologia recorrendo ao *clipboard.*
Importação de ficheiros XMI	O utilizador terá a capacidade de importar qualquer ficheiro XMI. Após seleccionar o ficheiro XMI o CO3L Edit disponibilizará de imediato uma janela de texto com a tradução do conteúdo do ficheiro XMI na linguagem CO3L. O utilizador terá à sua disponibilidade as mesmas funcionalidades descritas na macro funcionalidade "Desenvolvimento de ontologias CO3L". O mecanismo de tradução de XMI para CO3L foi elaborado por (Avelar, 2010). A aplicação CO3L Edit disponibiliza uma integração com este tradutor para disponibilizar ao utilizador uma ferramenta central que congrega todas as funcionalidades interessantes para o desenvolvimento de ontologias CO3L.

Parametrização	O utilizador terá a possibilidade de gerir alguns aspectos da aplicação CO3L Edit. Enquanto algumas parametrizações são implícitas e transparentes para o utilizador, outras são explícitas. As parametrizações implícitas dizem respeito às manipulações que o utilizador faz com o *layout* da aplicação. O CO3L Edit irá guardar em disco as alterações e ajustamentos que o utilizador realiza no *layout*. Deste modo, as alterações não se perdem quando o utilizador fechar a aplicação. As parametrizações explícitas são realizadas recorrendo a um ecrã específico do CO3L Edit onde é possível: • Parametrizar a pasta onde serão guardados todos os projectos e ontologias locais; • Parametrizar o nome da máquina onde está o O3 Server; • Parametrizar as credenciais de acesso ao O3 Server; • Parametrizar as cores a serem utilizadas no mecanismo de destaque da sintaxe. As parametrizações indicadas anteriormente são armazenadas em ficheiros XML.

Nome Funcionalidade	Descrição Funcionalidade
Consulta de notícias	Para que os utilizadores da ferramenta CO3L Edit recebam notícias importantes relativas ao CO3L Edit, ao O3 Server, à linguagem CO3L e notícias relativas ao modelo O3F, a ferramenta CO3L Edit disponibiliza um conjunto de notícias que provêm do O3 Server. Estas notícias são constituídas por um título, uma descrição e uma data. O título é implementado no CO3L Edit na forma de um link, o utilizador poderá clicar no título e será redireccionado para uma página Web onde obtém detalhe sobre a notícia.

Tabela 3 - Funcionalidades da aplicação CO3LEdit

A Figura 10 ilustra a interface gráfica da aplicação CO3L Edit assim como algumas funcionalidades.

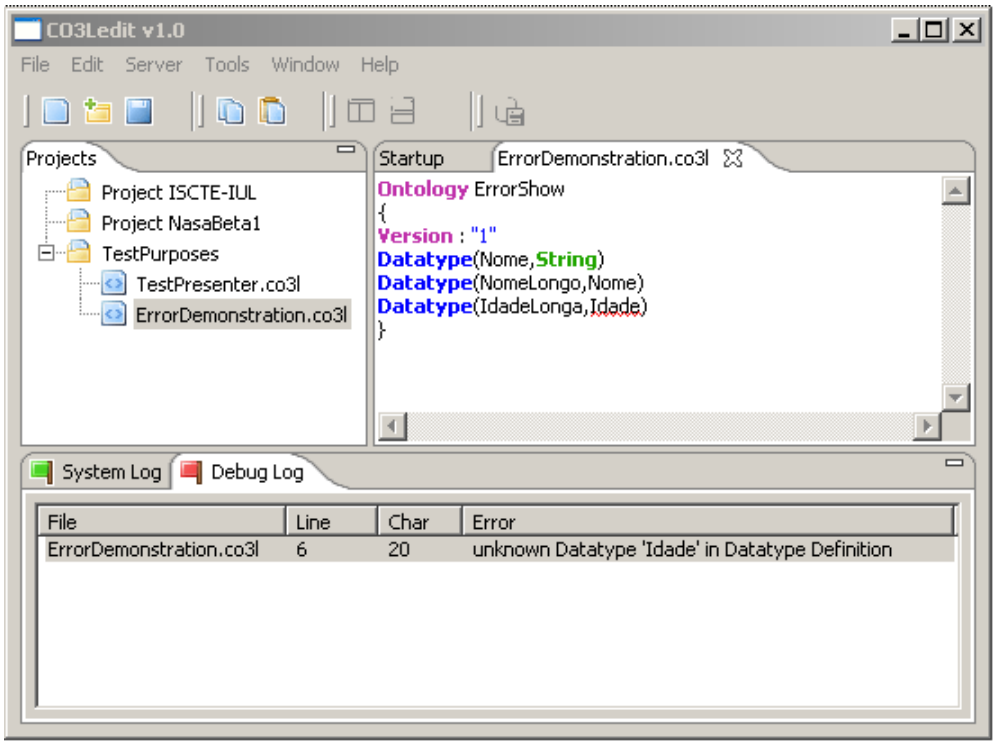

Figura 10 – Aplicação CO3Ledit

O CO3L Edit foi desenvolvido com base em requisitos que enfatizam a escalabilidade, a extensibilidade, a usabilidade e a performance. Tendo em conta que a aplicação CO3L Edit valida as ontologias em tempo real e ainda altera a interface gráfica do controlo que detém o texto, de forma a identificar as palavras-chave específicas da linguagem, foi necessário criar heurísticas que proporcionem uma boa performance, mesmo com tarefas consumidoras de tempo e recursos como é o caso dos processadores de linguagens.

Ao invés de processar a ontologia sempre que o utilizador adiciona mais informação, a ontologia é apenas processada quando o CO3L Edit detecta que o utilizador parou momentaneamente de adicionar informação. Deste modo é possível obter uma boa performance e validar as ontologias à medida que o utilizador as escreve, não sendo necessário esperar até que a especificação da ontologia esteja completa. Por exemplo, caso em que o utilizador esteja a descrever um predicado e o faça sem pausas tipográficas significativas, o CO3L Edit não irá processar a ontologia para efectuar validações. Esta e outras heurísticas estão descritas na Figura 11.

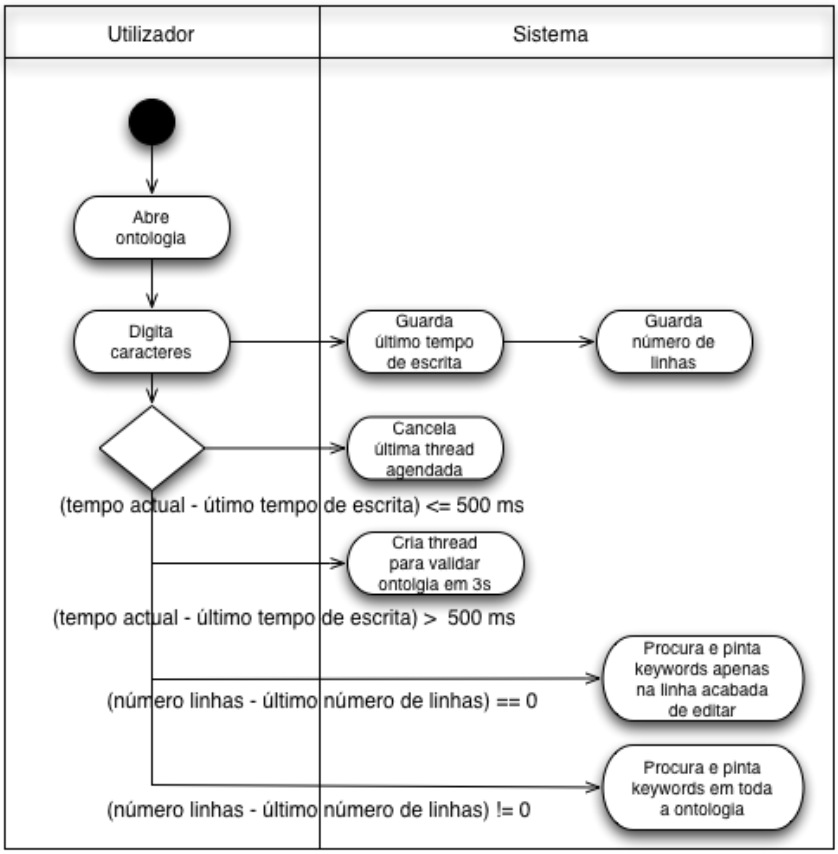

Figura 11 – Heurísticas no processamento da linguagem CO3L

Ao nível da usabilidade, o CO3L Edit também possui várias funcionalidades que permitem aos utilizadores desenvolver ontologias comodamente. Por exemplo, para além do usual menu de topo, o CO3L Edit tem outro menu onde é possível aceder às operações mais utilizadas como publicar uma ontologia no O3 Server, criar uma ontologia, criar um projecto, entre outras. A posição do grupo das operações poderá ser alterada ao gosto do utilizador. A própria interface gráfica do CO3L Edit poderá ser ajustada ao gosto do utilizador. Por exemplo, é possível esconder todos os

componentes do CO3L Edit (i.e. componente de gestão de projectos, componente de *log*) à excepção da área onde a ontologia é desenvolvida. Deste modo, o utilizador terá um espaço mais confortável para ler e escrever ontologias. Salienta-se ainda que todos os componentes podem ser redimensionados. A Figura 12 mostra um exemplo onde o CO3L Edit tem o *layout* alterado. Este imagem pode ser comparada com a Figura 10 para analisar as diferenças.

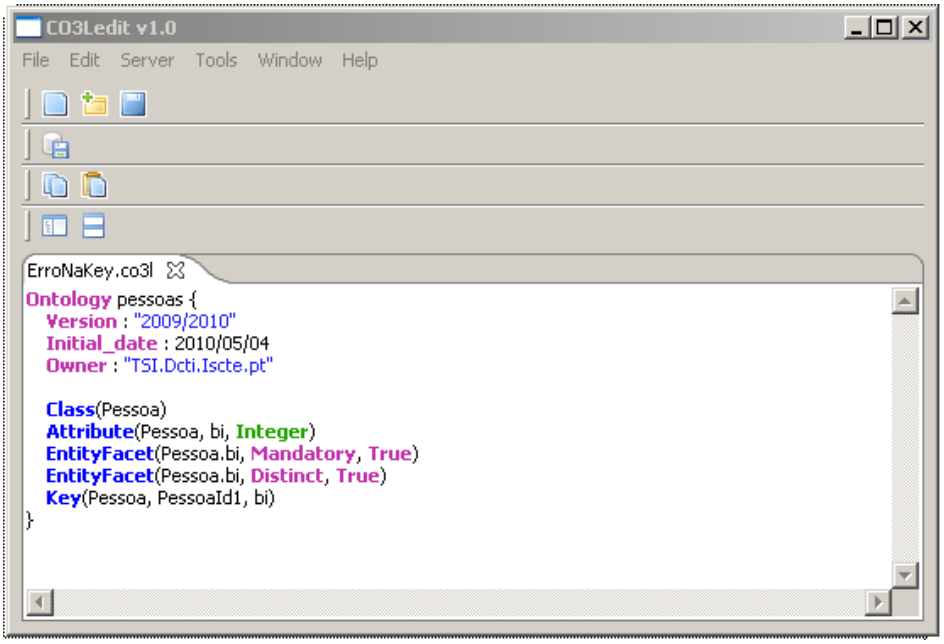

Figura 12 – CO3L Edit com *layout* alterado

Apesar da ferramenta CO3L Edit ser actualmente uma ferramenta especializada na linguagem CO3L, a estrutura que lhe está adjacente foi desenvolvida para permitir a evolução e extensão das funcionalidades existentes e das linguagens que a ferramenta suporta. Podem usar-se actualmente dois tipos de ficheiros no CO3L Edit. O primeiro tipo de ficheiros, com extensão co3l, diz respeito a ficheiros que guardam ontologias descritas em CO3L. O segundo, com extensão txt, representa qualquer tipo de informação que esteja guardada sobre a forma de texto. A possibilidade de ler o segundo tipo de ficheiros foi implementada para ilustrar a capacidade de extensão da ferramenta. Para que tivesse sido possível atingir tal extensibilidade, o CO3L Edit foi desenvolvido recorrendo a vários padrões de desenho como o *Strategy*, o *Factory Method*, o *Singleton*, o *Composite*, o *Command*, e o *Observer*. A Figura 13 ilustra a aplicação do padrão de desenho *Strategy* na modulação das classes que representam os documentos na ferramenta CO3L Edit.

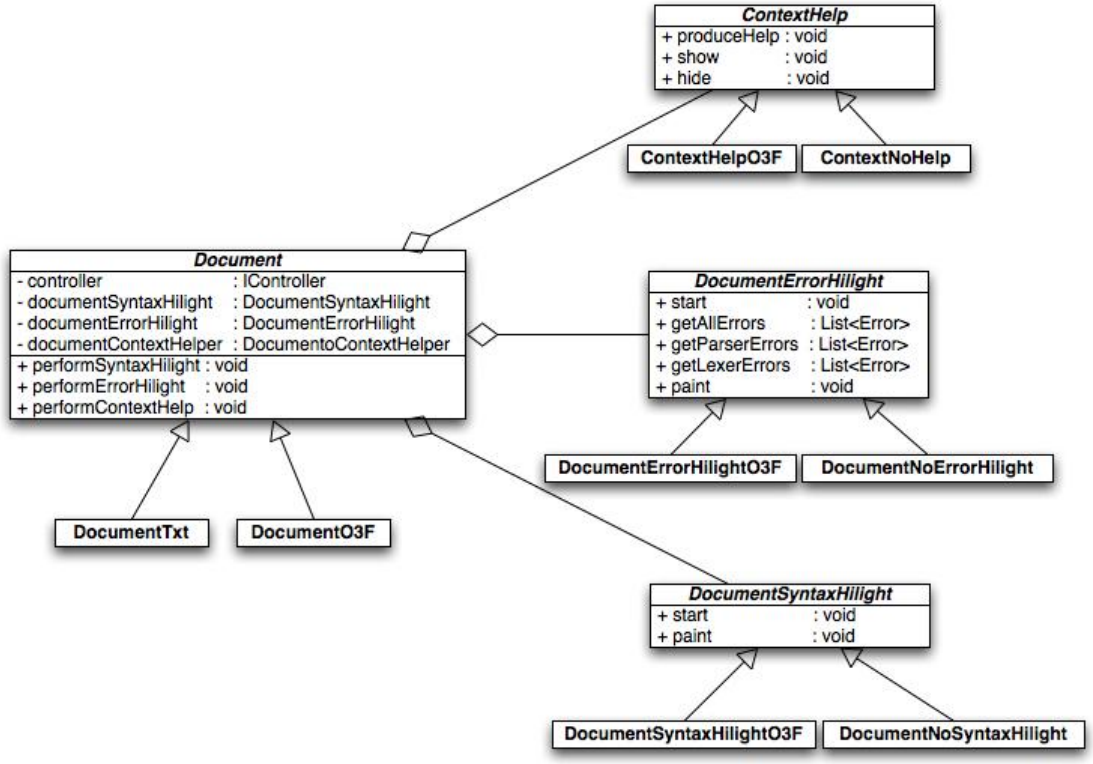

Figura 13 – Aplicação do padrão de desenho *Strategy*

O *strategy pattern* permite ao CO3L Edit ter vários tipos de documentos com os quais a aplicação sabe lidar e possibilita a implementação de outros tipos de documentos no futuro. Este tipo de técnica e perspectiva é vigente ao longo de toda a estrutura do CO3L Edit.

5.5 Processamento da linguagem CO3L

O processamento da linguagem CO3L é um dos pilares mais importantes e complexos deste trabalho. As funcionalidades relativas ao desenvolvimento de ontologias CO3L, na ferramenta CO3L Edit, resultam da capacidade de processar a linguagem CO3L e de conseguir interligar este processamento com os restantes processos da ferramenta. O estudo da teoria do processamento de linguagens assim como a aprendizagem da f*ramework* ANTLR foram também elementos chave em toda a aprendizagem e desenvolvimento do presente trabalho. Esta secção explica qual a estratégia seguida na concepção do processador de linguagem.

O CO3L Edit converte cada frase válida da linguagem CO3L num conjunto interligado de objectos JAVA que representam o modelo relacional O3F, em memória. Quando todas as frases CO3L da especificação da ontologia tiverem sido

validadas e convertidas em objectos JAVA, o conjunto de objectos gerado pode ser enviado para o O3 Server através da API O3 Server, para serem armazenados. Caso em que o processamento do texto encontre símbolos ou frases inválidas, serão gerados erros que são coleccionados até que todo o processamento esteja terminado. No fim do processamento são retornados todos os erros, indicando, para cada um, a posição (linha, coluna) e a mensagem de erro correspondente.

O processamento do CO3L tem certas complexidades que reflectem a natureza da linguagem. O facto de a linguagem permitir que o utilizador escreva os elementos da ontologia por qualquer ordem aliado a uma regra semântica que verifica que todos os elementos referenciados existem na ontologia, obriga a que se use um mecanismo de processamento de linguagens chamado "dupla passagem".

O processamento do CO3L assegura ainda as seguintes validações de semântica, que estavam inicialmente fora do âmbito da presente dissertação:

- Validar se os elementos referenciados existem;
- Validar se os elementos referenciados correspondem ao tipo certo num determinado contexto (exceptuando-se a validação dos atributos das instâncias, dos argumentos das hierarquias e ainda a validação dos argumentos das facetas, quando aplicadas a um elemento);
- Validar se as referências existentes nas instâncias correspondem a uma instância ou parte de instância existente;
- Validar que não existem colisões de nomes nos elementos;
- Verificar que pelo menos uma ontologia indicada no comando *Import* existe.

No contexto das validações de semântica foi necessário criar mensagens de erro personalizadas que estendem as mensagens de erro do ANTLR (Parr, 2007), a ferramenta usada para gerar o processador da linguagem (secção 5.6).

Para além da análise sintáctica e semântica, o processamento da linguagem é responsável por:

- Criar as representações JAVA das entidades do modelo O3F encontradas na especificação da ontologia;
- Realizar a inclusão recursiva das ontologias importadas;

- Enviar as representações JAVA das entidades para o O3 Server;
- Criar erros personalizado dependendo da regra semântica que foi avaliada;
- Coleccionar os erros sintácticos e semânticos provenientes da análise das frases da especificação.

Ainda no contexto do processamento da linguagem CO3L, falta falar sobre a funcionalidade de destaque da sintaxe que também é conseguida através de técnicas de processamento de linguagens. Esta funcionalidade é conseguida através de um segundo processador da linguagem, por motivos de performance. Este é responsável por procurar palavras reservadas. Sempre que uma palavra reservada da linguagem CO3L é encontrada, essa palavra é guardada numa lista. Quando toda a ontologia for processada, a lista de palavras reservadas é devolvida, com informação de contexto respectiva (localização, estilo). Posteriormente, um código que está implementado ao nível do módulo da interface gráfica será encarregue de modificar a configuração, disposição e aparência da área de texto para identificar visualmente as palavras reservadas encontradas.

Note-se que o processamento da linguagem CO3L em todos os seus aspectos é executado sempre que o utilizador altera a ontologia, não sendo necessário esperar que a especificação da ontologia esteja completa.

Devido à sua complexidade e importância, o módulo responsável pelo processamento da linguagem CO3L assume-se como um dos principais módulos da ferramenta CO3L Edit, permitindo o envio de *feedback* claro e contextualizado para o utilizador relativamente à validade de uma ontologia.

5.6 Implementação

Esta secção descreve sucintamente as ferramentas computacionais mais importantes usadas na implementação do CO3L Edit. A aplicação CO3L Edit foi desenvolvida com a linguagem de programação Java. Esta linguagem tem a peculiaridade de permitir a execução das aplicações em diferentes sistemas operativos, o que confere uma grande portabilidade. A escolha da linguagem condicionará de imediato as ferramentas, *frameworks* e tecnologias que podem ser utilizadas para construir o CO3L Edit.

A camada de interface gráfica foi implementada recorrendo ao SWT (Standard Widget Toolkit). O SWT tem de facto várias vantagens face ao SWING e ao AWT (Abstract Window Toolkit), que são as alternativas para a construção de interfaces em Java. O SWT é a única das soluções que consegue apresentar componentes de interface iguais aos componentes utilizados pelo sistema operativo no qual a aplicação está a ser executada. Os componentes do SWT conseguem, na sua maioria, ser mais rápidos do que os componentes do SWING ou do AWT (Guojie, 2005).

O CO3L Edit necessita de um processador sintáctico e lexicográfico da linguagem (*parser*) que permita a validação das especificações CO3L. O processador sintáctico e lexicográfico foi gerado pela ferramenta ANTLR (ANother Tool for Language Recognition). Esta ferramenta tem várias vantagens face a ferramentas concorrentes como o JLex[11], o CUP[12] e o JavaCC[13]. O ANTLR consegue gerar o lexer e o parser através de um único ficheiro, consegue gerar *abstract syntax trees* automaticamente e o output da geração de código é mais fácil de perceber (Gutiérrez, Díez, Castanedo, Gayo, & Cueva, 2000).

As tecnologias utilizadas na aplicação CO3L Edit estão listadas na Tabela 4.

Nome	**Descrição**
ANTLR 3.2 (www.antlr.org)	O ANTLR providencia uma *framework* de software aberto para construir interpretadores, compiladores, tradutores e carregadores.
SWT (www.eclipse.org/swt)	SWT é um widget toolkit de software aberto para Java desenhado para providenciar um acesso eficiente e portável às interfaces gráficas do sistema operativo nativo onde a aplicação é executada.
XStream (xstream.codehaus.org)	XStream é uma biblioteca de software aberto que serializa objectos para XML e desserializa XML para objectos Java.
Jazzy (jazzy.sourceforge.net)	Jazzy é um conjunto de APIs de software aberto que permitem adicionar capacidades de spell checking a aplicações Java.
Jtds 1.2.5 (jtds.sourceforge.net)	Jtds é uma implementação de software aberto do JDBC 3.0; oferece o mais rápido driver de acesso a base de dados Microsoft SQL Server.

Tabela 4 – Tecnologias utilizadas na aplicação CO3L Edit

A Tabela 4 demonstra também que todos os componentes utilizados no CO3L Edit são de software aberto (*open source*). Apesar do CO3L Edit utilizar *frameworks* que

[11] http://www.cs.princeton.edu/~appel/modern/java/JLex/
[12] http://www.cs.princeton.edu/~appel/modern/java/CUP/
[13] https://javacc.dev.java.net/

permitem abstrair de problemas muito concretos, agilizando assim o desenvolvimento da aplicação, o esforço no desenvolvimento da aplicação foi muito significativo.

Descrição	Volume
Número total de linhas de código	40.871
Número total de linhas de código proveniente de geradores de código ou código de terceiros	11.085
Número total de classes Java	226
Número total de ficheiros	360

Tabela 5 - Volume de desenvolvimento do CO3L Edit

A Tabela 5 resume algumas das métricas resultantes do desenvolvimento da ferramenta CO3L Edit. Salienta-se o facto de que foram desenvolvidas de raiz cerca de trinta mil linhas de código.

5.7 Conclusão

O presente trabalho desenvolveu o CO3L Edit, uma ferramenta especializada na linguagem CO3L. O CO3L Edit integra funcionalidades que permitem o desenvolvimento de ontologias CO3L, o armazenamento e a consulta de ontologias. O CO3L Edit consegue assim proporcionar ao utilizador um ambiente integrado onde é possível encontrar todas as funcionalidades necessárias para trabalhar com ontologias CO3L.

Apesar do CO3L Edit ser hoje uma ferramenta especializada na linguagem CO3L e comunicar com o O3 Server, a estrutura que lhe está adjacente foi construída para permitir que o CO3L Edit seja aberto a diferentes linguagens, funcionalidades e servidores. O CO3L Edit foi ainda construído recorrendo a módulos que podem ser reutilizados por outras aplicações e que podem ser substituídos por outros com a mesma interface, como é o caso do processador de linguagem CO3L. Foram usados padrões de desenho conhecidos com o objectivo de facilitar a manutenção da ferramenta e reduzir a possibilidade de erros.

O CO3L Edit assume-se como a principal aplicação para desenvolver ontologias CO3L e consegue pela primeira vez, desde que a linguagem CO3L foi criada, disponibilizar mecanismos que facilitam e ajudam o utilizador a desenvolver ontologias O3F e usufruir da inteira expressividade da linguagem CO3L e do modelo O3F.

Capítulo 6 Avaliação

A avaliação do trabalho realizado no âmbito desta dissertação é fundamental para determinar o seu sucesso e a aceitação dos resultados produzidos pela comunidade científico-tecnológica em que se enquadram e pelo grupo alvo de utilizadores a que se destinam.

A dissertação produziu três resultados, os quais são sujeitos a avaliação:

- Melhoramento do modelo O3F e da sua linguagem CO3L;
- Construção do servidor de ontologias O3F (O3 Server);
- Construção do editor para a linguagem CO3L (CO3L Edit).

A avaliação das melhorias efectuadas ao modelo e à linguagem é feita apenas de forma argumentativa na secção 6.1, onde se mostra o aumento de compatibilidade com o UML e a melhoria da expressividade do O3F.

As duas ferramentas computacionais concebidas estão sujeitas a dois tipos de avaliação, uma avaliação objectiva que se prende com a correcção e a abrangência dos procedimentos por elas efectuados, e uma avaliação subjectiva que diz respeito à sua usabilidade e utilidade. A validação formal da correcção das ferramentas passaria por determinar se as validações sintácticas e semânticas efectuadas pelo CO3L Edit estão correctas de acordo com a sintaxe e a semântica da linguagem, e se a base de dados do O3 Server e seus mecanismos de armazenamento e extracção correspondem ao modelo O3F.

A avaliação formal da correcção das validações feitas pelo editor recorreria a aspectos da teoria da computação totalmente fora do âmbito desta dissertação. A avaliação da abrangência (i.e., da completude) dos mecanismos de validação do editor poderia também ser efectuada formalmente, recorrendo à semântica do modelo. Essa abordagem está também fora do âmbito desta dissertação. No entanto, a secção 6.2 mostra empiricamente que o editor efectua a validação sintáctica de todos os elementos da linguagem e efectua algumas validações de semântica.

A correcção do modelo de dados e dos mecanismos usados no servidor para armazenar e extrair ontologias também estaria, em princípio, sujeito a validação

formal, a qual recorreria a conhecimentos que não adquirimos e que extravasam largamente os objectivos desta dissertação.

A prova da completude do modelo relacional e dos mecanismos de armazenamento e extracção de ontologias do servidor O3 Server, embora pudesse em princípio ser feita por via formal, sairia também do âmbito do trabalho e formação. A secção 5.1.2 de (Avelar, 2010) mostra empiricamente que, para todos os casos ensaiados, a ontologia extraída do O3 Server é igual à ontologia original, previamente armazenada. Não sendo uma demonstração, esta verificação constitui evidência da correcção e completude do servidor O3 Server.

A avaliação subjectiva da usabilidade e utilidade das ferramentas foi feita em conjunto para ambas as ferramentas porque o servidor não pode ser utilizado isoladamente por pessoas. A sua utilização é feita por aplicações cliente, neste caso, pelo editor que após as validações efectuadas envia a ontologia para o servidor que a armazena. A avaliação subjectiva recorreu a um inquérito aplicado a 51 alunos do terceiro ano das licenciaturas de Engenharia Informática e de Informática e Gestão de Empresas do ISCTE-IUL, a 4 docentes do Departamento de Ciências e Tecnologias da Informação do ISCTE-IUL e a 3 analistas e engenheiros de sistemas da Novabase. O inquérito mostrou que a média das respostas às seis perguntas incluídas se situa entre 3.26 e 3.9, numa escala de 1 a 4, o que é francamente positivo.

O procedimento de avaliação subjectiva utilizado está descrito na secção 6.3. A secção 6.4 identifica o tipo de utilizadores que foram alvo do inquérito e a secção 6.5 termina com a apresentação dos resultados obtidos.

6.1 Melhorias introduzidas no O3F e no CO3L

As melhorias que se introduziram no modelo O3F e na linguagem CO3L tinham como objectivo aproximar o O3F de outras abordagens à representação de ontologias, em particular o UML. Deste modo será possível representar em O3F ontologias originalmente escritas usando outras abordagens, por exemplo UML e OWL. De igual modo, será possível converter parcialmente ontologias O3F em representações UML e OWL, por exemplo. Um exemplo vem da dissertação de Jairo Avelar (Avelar, 2010), a qual deu origem a um conversor de diagramas UML de classes e objectos em ontologias O3F. Essa dissertação, no seu capítulo de avaliação, mostra objectivamente

que para todos os casos ensaiados em que se traduziram diagramas UML para o modelo O3F, a tradução foi completa e correcta. As melhorias introduzidas no modelo O3F e na linguagem CO3L permitiram assim a importação completa de qualquer diagrama UML de classes ou de objectos.

Outro objectivo ligado às melhorias introduzidas era o de aumentar a capacidade do O3F expressar conhecimento que outras abordagens não conseguem. Por exemplo, o OWL e o Ontolingua não conseguem expressar directamente métodos funcionais, métodos relacionais e métodos de acção. O UML por sua vez não consegue representar predicados ou métodos relacionais. Nenhum deles permite a representação de acções. O UML também não permite distinguir entre métodos funcionais e métodos de acção. O O3F para além de conseguir representar o conhecimento atrás mencionado, consegue também representar a relação entre métodos e operadores, e entre atributos e funções, entre outras possibilidades, em especial as que resultam da aplicação de facetas.

Ainda que não se possa concluir através de uma prova formal e rigorosa que as melhorias introduzidas originaram um modelo melhor, o facto de atingir os objectivos, aumentando a compatibilidade do O3F com o UML e o facto de o O3F possuir características únicas face a outros modelos faz com que se possa depreender que o modelo O3F beneficiou das melhorias introduzidas por esta dissertação.

6.2 Validações sintácticas e semânticas do CO3L Edit

Para comprovar o bom funcionamento da ferramenta e assegurar que o CO3L Edit é compatível com a nova especificação da linguagem CO3L, elaborou-se um conjunto de testes que incidem sobre todos os elementos que constituem a linguagem CO3L. Os testes consistiram em verificar que, para cada elemento do CO3L, existe uma correspondência no CO3L Edit. Esta correspondência deve originar uma correcta identificação do elemento sempre que a sua especificação esteja sintacticamente correcta, deve originar um erro sempre que o elemento não esteja sintacticamente bem especificado e deve ainda assinalar de uma forma visual o reconhecimento do elemento.

As especificações CO3L recorrem, na sua maioria, a predicados que servem para descrever os elementos de uma ontologia. No entanto existem outros elementos que

também constituem a gramática da linguagem CO3L, como o comando que indica as ontologias a importar e ainda os argumentos da ontologia, que indicam entre outras características o autor da ontologia e a data em que foi criada. Por exemplo, o predicado *Class* foi testado no CO3L Edit escrevendo-se inúmeras combinações correctas e incorrectas de especificações com o predicado, para averiguar se o output do CO3L Edit era o mais correcto.

Teste Predicado	**Resultado**

Ontology Teste{
Class(Pessoa)
}
	Sucesso

Ontology Teste{
Class()
}
	Erro: missing WORD at ‘)’ in Class Definition

Ontology Teste{
Class(Pessoa))
}
	Erro: extraneous input at ‘)’ expecting ‘}’ in Ontology Structure

Ontology Teste{
Class(Pessoa
}
	Erro: missing ‘)’ at ‘}’ in Class Definition

Tabela 6 – Testes à sintaxe do predicado Class

A Tabela 6 ilustra um pequeno exemplo de alguns testes individuais que foram feitos ao predicado *Class*.

Ainda no contexto da validação gramática da linguagem CO3L, foram elaborados também testes de semântica. O conjunto de testes de semântica foi:

- Validar se os elementos referenciados existem;
- Validar se os elementos referenciados correspondem ao tipo certo num determinado contexto (exceptuando-se a validação dos atributos das instâncias, dos argumentos das hierarquias e ainda a validação dos argumentos das facetas, quando aplicadas a um elemento);
- Validar se as referências existentes nas instâncias correspondem a uma instância ou parte de instância existente;
- Validar que não existem colisões de nomes nos elementos;
- Verificar que pelo menos uma ontologia indicada no comando *Import* existe.

O conjunto de validações de semântica atrás mencionadas não correspondem ao total de validações de semântica associadas à linguagem CO3L, no entanto o principal objectivo da presente dissertação era implementar a camada sintáctica.

Para além de todos os testes referidos, foi ainda realizado o seguinte conjunto de testes:

- Testar todas as facetas *built-in* do CO3L na ferramenta CO3L Edit e verificar que são reconhecidas e processadas;
- Testar todos os tipos de dados *built-in* do CO3L na ferramenta CO3L Edit e verificar que são reconhecidos e processados;
- Testar todas as ontologias CO3L que são utilizadas no âmbito da cadeira de Tecnologias de Sistemas Inteligentes;
- Publicar mais de 50 ontologias no O3 Server com diferentes conteúdos;
- Consultar no CO3L Edit as 50 ontologias publicadas no O3 Server.

Todos os testes descritos foram executados com sucesso. Salienta-se que todas as funcionalidades do CO3L Edit e do O3 Server foram devidamente testadas recorrendo a testes unitários e testes de integração. Apesar dos testes efectuados não garantirem uma prova formal e irrefutável em como as validações sintácticas e semânticas do CO3L Edit estão totalmente correctas, o conjunto vasto de testes realizados e o seu sucesso constitui evidência significativa de que o CO3L Edit implementou toda a linguagem CO3L, a qual consegue validar sintacticamente, para além de algumas validações semânticas.

6.3 Procedimento de avaliação subjectiva

A avaliação subjectiva das duas ferramentas foi realizada sob a forma de um questionário que foi preenchido por utilizadores que tiveram contacto directo com a aplicação CO3L Edit e, indirectamente, com o O3 Server.

O questionário foi constituído por 6 afirmações. Para cada afirmação, o utilizador teve que seleccionar, numa escala de 1 a 4, qual a medida da sua concordância com a afirmação proferida. A escala era constituída por: 1 - Discordo totalmente, 2 - Discordo parcialmente, 3 - Concordo parcialmente e 4 - Concordo totalmente.

N.º	Afirmação	Objectivo
1	A identificação de palavras-chave (*keywords*) efectuada pelo CO3L Edit facilita a leitura e a escrita de ontologias.	Avaliar o mecanismo de destaque da sintaxe relativamente à sua mais-valia na leitura e escrita de ontologias.
2	Os mecanismos de destaque de erros, destaque da sintaxe e mensagens de erros existentes no CO3L Edit permitem a identificação simples e intuitiva de erros presentes nas ontologias descritas.	Avaliar os mecanismos existentes no CO3L Edit que permitem identificar os erros em ontologias CO3L.
3	O CO3L Edit providencia um ambiente integrado que facilita o desenvolvimento, a publicação (armazenamento no O3 Server) e a extracção (do O3 Server) eficazes e simples de ontologias.	Avaliar a integração do editor com o servidor.
4	O CO3L Edit executa todas as operações com um tempo de resposta bastante rápido. As operações a avaliar incluem o arranque da ferramenta, a velocidade de resposta da interface gráfica, a publicação (armazenamento) de ontologias, o desenvolvimento e validação de ontologias e a extracção de ontologias.	Avaliar a performance no CO3L Edit e do O3 Server.
5	O CO3L Edit é adaptável às necessidades e gostos pessoais de cada utilizador e dispõe de uma interface fácil de usar que proporciona uma boa experiência na sua utilização.	Avaliar a usabilidade do CO3L Edit.
6	Em geral, a ferramenta CO3L Edit e O3 Server são úteis e fáceis de usar.	Avaliar a utilidade geral do CO3L Edit e do O3 Server.

Tabela 7 – Questões e objectivos do questionário

O questionário pretendia ainda perceber quais as características do sistema que os utilizadores gostariam de ver melhoradas e que outras funcionalidades deveriam estar presentes no sistema. Para este efeito existia a possibilidade de indicar as melhorias pretendidas e comentários gerais à ferramenta sob a forma de um texto livre.

Os utilizadores não estiveram limitados temporalmente nem espacialmente para responder ao inquérito.

6.4 Participantes no inquérito

Todos os participantes no inquérito têm um conhecimento mínimo que assegura a sua capacidade para construírem ontologias recorrendo à linguagem CO3L. Cerca de 90% deles são alunos ou professores do instituto onde o presente trabalho se desenvolveu e têm um conhecimento médio ou elevado sobre a linguagem CO3L e sobre o modelo O3F. Os participantes foram na sua maioria alunos que tinham terminado

recentemente a disciplina de Tecnologias de Sistemas Inteligentes, disciplina que dota o aluno de um bom conhecimento relativo à linguagem CO3L e ao modelo O3F. No entanto existiram participantes com experiências diferentes que também responderam ao inquérito como é o caso de gestores de projecto e consultores da Novabase. Estes últimos puderam não só olhar de um modo restrito para as capacidades do CO3L Edit enquanto sistema especializado na linguagem CO3L, mas como sistema informático genérico que responde a uma necessidade, e avaliá-lo comparando-o com muitas aplicações que já desenvolveram ou usaram.

Aspectos como a usabilidade e a integração de diferentes funcionalidades só podem ser testadas verdadeiramente quando diferentes utilizadores utilizam a ferramenta e emitem a sua opinião. São os utilizadores que irão utilizar a ferramenta e foi para eles que a ferramenta foi criada.

6.5 Resultados da avaliação subjectiva

A avaliação do presente trabalho baseia-se nas 58 respostas obtidas ao inquérito descrito. Com estas respostas foi possível perceber, com um grau de confiança elevado, a quase unanimidade dos utilizadores face à ferramenta CO3L Edit.

Considerando a Tabela 7, onde estão descritos os objectivos e as questões colocadas aos utilizadores, obtiveram-se os seguintes resultados para cada questão:

N.º Questão	Média	Desvio Padrão
1	3,90	0,40
2	3,85	0,41
3	3,6	0,53
4	3,26	0,64
5	3,64	0,49
6	3,70	0,5

Tabela 8 – Resultados da avaliação

Tendo em conta que o valor máximo é 4 e o valor mínimo é 1, os resultados que constam na Tabela 8 são francamente positivos.

Os resultados relativos às duas primeiras perguntas são os que mais se destacam. Na primeira questão relativa à utilidade da identificação e coloração automática das palavras-chave da linguagem CO3L, 100% dos utilizadores concordaram totalmente ou parcialmente que este mecanismo é útil na leitura e escrita de ontologias, sendo que 91% destes concordam totalmente (nota máxima). Resultado muito similar teve a resposta relativa aos mecanismos de detecção e indicação dos erros. Ainda que positiva, mas que obteve um resultado mais baixo face aos outros resultados, foi a resposta à questão 4, onde se pretendia apurar junto dos utilizadores qual a sua opinião relativamente à performance geral da aplicação.

Todos os utilizadores que manifestaram um descontentamento relativamente à performance do CO3L Edit justificaram que a aplicação tinha apenas dois pontos muito circunscritos de problemas de performance. O primeiro ponto dizia respeito à velocidade com que a aplicação disponibiliza a interface e o segundo ponto dizia respeito à interacção com o O3 Server (publicar e consultar ontologias). Na realidade os dois pontos dizem respeito a ligações ao O3 Server. Quando o CO3L Edit é iniciado, a última actividade que é executada antes de disponibilizar a interface é contactar o servidor para obter as últimas notícias. No entanto é preciso perceber que o O3 Server é acedido remotamente através da Internet. A velocidade de comunicação estará dependente da velocidade de ligação do utilizador à Internet e ainda do congestionamento que possa existir do lado do próprio O3 Server. Ainda relativamente à performance, apenas 10% dos utilizadores indicaram que não estavam satisfeitos com a performance (indicando a opção 2, discordo parcialmente). Os restantes 90% indicaram que estavam satisfeitos com a performance. Salienta-se ainda que o tempo médio de abertura da aplicação CO3L Edit ronda os 3 segundos.

Para além dos resultados descritos na Tabela 8, os utilizadores fizeram vários comentários relativos a melhorias e aspectos que foram do seu agrado. Alguns dos comentários que foram feitos ao CO3L Edit estão mencionados de seguida:

"Penso que a ferramenta está bastante simples e intuitiva de usar."

"Em termos de usabilidade, a ferramenta é bastante personalizável, sendo assim possível o utilizador adaptar a interface conforme deseja. As cores nas palavras-chave de definição de ontologias permitem também prevenir o erro, na medida em que auxiliam o utilizador na escrita da ontologia."

"O interface está bastante *user-friendly*, semelhante a outros IDEs populares (...). É simples e há um fácil acesso às funcionalidades de edição de ontologias e ao servidor."

"É realmente uma ferramenta interessante que facilita um utilizador com alguns conhecimentos em CO3L a elaborar ontologias."

"Acho incrivelmente útil esta ferramenta para explorar as ontologias e até para praticar em aulas do âmbito desta solução."

"Com estes mecanismos torna-se mais fácil identificar erros, sendo mais rápido corrigi-los, tal como a opção de *debug* que é apresentada, na minha opinião é uma mais-valia este projecto."

"O programa é fácil de utilizar (é intuitivo e os atalhos para cada acção são práticos) e as mensagens de erro são bastante úteis."

Através do questionário foi também possível analisar as novas funcionalidades e melhorias pretendidas pelos utilizadores. A maioria das melhorias foi incluída na lista de trabalho futuro que está descrita no Capítulo 6 . Outras melhorias foram corrigidas e estão agora incluídas na versão actual da ferramenta, destacando-se os ajustes às teclas de atalho para sair da aplicação e para copiar e colar texto das ontologias. Deste modo a avaliação não só permitiu determinar o grau de sucesso da dissertação mas permitiu também melhorar o trabalho realizado.

Face aos resultados provenientes da avaliação dos utilizadores e do conjunto de testes que foram realizados à ferramenta, conclui-se que a avaliação deste trabalho é francamente positiva e teve uma grande aceitação por partes dos utilizadores.

Capítulo 7 Conclusões e Trabalho Futuro

Além de ter contribuído para melhorar o modelo O3F e a sua linguagem CO3L, esta dissertação envolveu o desenvolvimento de duas ferramentas computacionais para ontologias – um servidor de ontologias O3F e um editor de CO3L – ambos disponíveis ao público, através da Internet, em http://dcti.iscte.pt/O3F/. O servidor (O3 Server) foi desenvolvido em cooperação com a tese de Jairo Avelar (Avelar, 2010), a qual deu ainda origem a um conversor de diagramas de classes e objectos UML em ontologias O3F. A disponibilização *on-line* e a divulgação destes sistemas não só contribui para o desenvolvimento, a partilha e o processamento de ontologias, como sobretudo constitui uma forte aposta no desenvolvimento do O3F, da sua linguagem CO3L, e das suas ferramentas.

Este último capítulo apresenta conclusões sobre o trabalho realizado e propõe linhas de futura investigação e desenvolvimento do O3F e suas ferramentas

A secção 7.1 pretende rever os objectivos da presente dissertação e analisar os objectivos atingidos. Na secção 7.2 são apresentadas várias indicações de possíveis melhorias e trabalhos adicionais que podem ser realizados para melhorar a ferramenta CO3L Edit, o O3 Server e ainda o modelo O3F e a linguagem CO3L. O presente capítulo conclui na secção 7.3 onde são apresentadas os principais desafios do presente trabalho, as principais dificuldades e as considerações finais de todo o trabalho desenvolvido.

7.1 Objectivos

Os objectivos propostos para a dissertação são o estudo e melhoria do modelo O3F e linguagem CO3L, construção de um editor de desenvolvimento de ontologias especializado na linguagem CO3L, e construção de um servidor capaz de guardar e disponibilizar as ontologias desenvolvidas.

A melhoria do modelo O3F e da linguagem CO3L não tinha à partida nenhum objectivo mensurável para que seja possível afirmar matematicamente que o objectivo foi de facto cumprido. No entanto foram propostas e aceites um conjunto significativo de melhorias, desde a reestruturação de diversos aspectos do modelo O3F (i.e., reestruturação das interfaces de acção, reestruturação da relação entre métodos e

operadores, e redefinição da declaração de novas facetas) até à inserção de novos predicados e entidades que aumentam a expressividade do modelo (i.e., representação de hierarquias, representação de indivíduos), passando ainda pelo aumento do número de opções que já existiam no modelo O3F e linguagem CO3L (i.e., expansão do número de facetas pré-definidas). Tendo em conta que as melhorias introduzidas aumentaram a expressividade do modelo O3F e da linguagem CO3L aumentando ainda a sua compatibilidade com o UML e o OWL, não é imprudente afirmar que os objectivos foram cumpridos.

O objectivo principal relativamente à construção de um editor de ontologias para a linguagem CO3L estava ligado à necessidade de validar a linguagem CO3L sintacticamente em tempo real e conseguir exportar o resultado do desenvolvimento para uma base de dados. No entanto o editor foi evoluindo para uma ferramenta muito mais complexa e prestativa. O CO3L Edit resultou numa ferramenta que suporta o desenvolvimento de ontologias em CO3L recorrendo a uma interface gráfica rápida e amigável, que consegue gerir os projectos relativos ao desenvolvimento de ontologias e ajudar o utilizador no desenvolvimento das ontologias através de mecanismos como o destaque da sintaxe, o destaque de erros e ainda o corrector ortográfico. Para além de todas estas funcionalidades que auxiliam o desenvolvimento de ontologias, o CO3L Edit possibilita a parametrização de muitos aspectos do editor, disponibiliza notícias provenientes do O3 Server e tem a capacidade de publicar e consultar ontologias através do O3 Server. Ao nível do processamento da linguagem CO3L, existiu ainda o esforço de implementar algumas validações de semântica. Conclui-se assim que o objectivo da construção de um editor especializado na linguagem CO3L foi cumprido. Mais ainda, pode afirmar-se que o objectivo inicial foi ultrapassado em larga escala.

O O3 Server foi também implementado na sua totalidade, atingindo todos os objectivos estabelecidos para este trabalho. O O3 Server tem a capacidade de guardar ontologias, disponibilizar as ontologias na linguagem CO3L e oferece ainda a capacidade de pesquisa vigente em qualquer RDBMS através da linguagem SQL ou T-SQL.

Apesar de ambiciosos, os três objectivos foram cumpridos com sucesso e culminaram com a aceitação de várias melhorias introduzidas no modelo O3F e na linguagem

CO3L, assim como a entrega de um sistema prático, estável e pronto a utilizar. Note-se que o O3 Server e o CO3L Edit não representam somente uma prova de conceito ou um protótipo. O O3 Server e o CO3L Edit estão aptos a ser utilizados por todos os interessados. O sistema, como um todo, foi testado por mais de 50 utilizadores externos ao projecto.

7.2 Trabalho futuro

Apesar do presente trabalho ter os seus objectivos cumpridos, existe muito trabalho que é possível realizar para melhorar o modelo O3F, a linguagem CO3L, a ferramenta CO3L Edit e o O3 Server.

Relativamente ao modelo O3F e à linguagem CO3L é necessário estudar convenientemente o melhor modo de incluir axiomas na linguagem. O modelo O3F já contempla axiomas, como é visível no diagrama de classes O3F, no entanto a representação de axiomas em CO3L ainda não foi convenientemente estudada. Os axiomas irão conferir uma expressividade muito grande ao modelo O3F e à linguagem CO3L e são por isso um trabalho futuro importante a ser considerado.

O CO3L Edit, apesar de ter implementadas várias funcionalidades que não estavam previstas nos requisitos iniciais do projecto, continua a ter muito espaço para evoluir. As seguintes melhorias foram identificadas como as mais importantes para que o CO3L Edit se afirme como uma ferramenta competitiva face a ferramentas já existentes para outros modelos e linguagens de ontologias:

- Inclusão do mecanismo de auto complemento às expressões da linguagem (“*autocompletion”)*. Deste modo será possível auxiliar directamente o utilizador a escrever expressões da linguagem e não apenas a reconhecê-las.
- Melhorar o processador de linguagem CO3L para implementar todas as validações de semântica necessárias. Actualmente os valores que são colocados ao nível do predicado *FacetEntity* não são validados de acordo com as restrições descritas nos predicados V*alidFacetElement* e *ValidFacetType*. De igual modo, os valores que são colocados ao nível dos indivíduos para indicar os seus atributos devem ser validados de acordo com os tipos de dados de que os indivíduos são instâncias.

- Implementação de uma camada visual que permita a construção de ontologias recorrendo principalmente a componentes gráficos. Deste modo o CO3L Edit poderá ser utilizado para modular ontologias sem recorrer directamente aos predicados da linguagem CO3L, o que facilitará a utilização da ferramenta por utilizadores menos experientes.
- Implementação de uma secção que permita inferir conhecimento a partir do conhecimento explicitamente representado nas ontologias mantidas no O3 Server.

Foram ainda identificadas outras melhorias no decorrer da avaliação do sistema, destacaram-se as seguintes:

- Ao nível da interface gráfica do editor textual do CO3L Edit, disponibilizar um mecanismo que permita mostrar o número da linha para todas as linhas.
- Criação de templates para acelerar o desenvolvimento de ontologias. Por exemplo, quando é criada uma nova ontologia, o editor deverá preencher automaticamente os campos obrigatórios sob a forma de um template.
- Ao nível da avaliação sintáctica e semântica das ontologias, para além de mostrar os erros, deverão ser emitidas mensagens que ajudem directamente na resolução do erro.

Por último o O3 Server poderá evoluir para algo mais do que um RDBMS. O O3 Server poderá estar assente sobre uma camada de serviços Web e utilizar o RDBMS apenas como uma camada para armazenar dados. Por exemplo, a camada de serviços Web será responsável por interrogar e inserir dados. Ao criar a nova camada de serviços Web, a performance sairá degradada mas o O3 Server ficará fortalecido uma vez que poderá implementar uma série de funcionalidades não dependentes de um RDBMS.

7.3 Considerações finais

O presente projecto conclui todos os seus objectivos com sucesso. Como demonstra a avaliação do trabalho, o projecto foi avaliado com distinção por mais de 50 utilizadores com conhecimentos tecnológicos avançados e com conhecimentos do modelo O3F e da linguagem CO3L.

Para atingir todos os objectivos e conseguir produzir um sistema que suscitasse tanto interesse e agrado por parte dos utilizadores foi necessário aprender uma variedade muito significativa de tecnologias e *frameworks* que não são leccionadas no ISCTE-IUL, instituto onde o presente trabalho se desenvolveu. Uma das maiores dificuldades tecnológicas foi sentida ao nível da disciplina relacionada com o processamento de linguagens. Foi necessário investir muito tempo na investigação e aprendizagem de matérias relacionadas com este tópico.

A segunda maior dificuldade foi encontrada ao nível da gestão do âmbito do projecto. Tendo em conta que o modelo O3F e a linguagem CO3L não estiveram estáticas ao longo do desenvolvimento do CO3L Edit e do O3 Server, existiram fases em que a mudança do modelo e da linguagem comprometeram funcionalidades que já estavam implementadas e que tiverem que ser refeitas. Em todas estas situações, a decisão foi de implementar as novas funcionalidades para que as ferramentas resultantes estivessem de acordo com as novas especificações do modelo e da linguagem, caso contrário o CO3L Edit seria o único editor para CO3L e já estaria desactualizado face à nova especificação da linguagem. Devido a estas mudanças inesperadas e à respectiva acomodação no desenvolvimento, outras funcionalidades não foram totalmente desenvolvidas como é o caso das validações de semântica no processador da linguagem CO3L.

O presente trabalho teve como principal contribuição o desenvolvimento de ferramentas que suportam o modelo O3F e a linguagem CO3L. Com esta contribuição será possível concorrer com outros modelos já existentes e progredir no sentido de tornar o modelo O3F num modelo a ter em consideração sempre que se pretenda modular um domínio através de ontologias.

Bibliografia

Antoniou, G., & Harmelen, F. (2003). Web Ontology Language: OWL. In *Handbook On Ontologies In Information Systems* (pp. 67-92).

Arpírez, J. C., Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2003). WEBODE in a Nutshell. *AI Magazine (AAAI)) , 24*.

Avelar, J. (2010). *Ontologias O3F: Conversão de UML e Extracção em CO3L*. Tese de Mestrado, ISCTE-IUL, DCTI.

Bechhofer, S., Horrocks, I., Goble, C., & Stevens, R. (2001). OilEd: A Reason-able Ontology Editor for the Semantic Web. *2174/2001*, pp. 396-408. Springer Berlin / Heidelberg.

Bennet, S., McRobb, S., & Farmer, R. (1999). *Object Oriented Systems Analysis and Design using UML*. (McGraw-Hill, Ed.)

Booch, G. (1991). *Object-Oriented Analysis and Design with Applications.* 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* .

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., & Rice, J. P. (1998). OKBC: A Programmatic Foundation for Knowledge Base Interoperability. *AAAI-98 Proceedings.*

Council, T. P. (15 de 06 de 2010). *TPC-H - Top Ten Price/Performance Results*. Obtido em 01 de 07 de 2010, de http://www.tpc.org/tpch/results/tpch_price_perf_results.asp

Cranefield, S. (2001). Networked Knowledge Representation and Exchange using UML and RDF. *Journal of Digital Information* .

Crespo, R. G. (1998). *Processadores de linguagens.* Lisboa: IST Press.

Domingue, J. (1998). Discussing, Browsing, and Editing Ontologies on the Web. *International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98).* Banff.

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

Farquhar, A., Fikes, R., & Rice, J. (1997). Tools For Assembling Modular Ontologies in Ontolingua. *In Proc. of AAAI 97* (pp. 436-441). AAAI Press.

Fzi, S. M., Stojanovic, L., & Motik, B. (2002). Ontology Evolution within Ontology Editors. (pp. 53-62). Madrid: EKAW' 02/EON. Workshop.

Gasevic, D. V. (2005). Bridging Mda And Owl Ontologies. *4* (2).

Genesereth, M., & Fikes, R. E. (1992). *Knowledge Interchange Format Version 3.0 Reference Manual.*

Genesereth, M., & Singh, N. (1991). Epikit: a library of subroutines supporting declarative representations and reasoning. *2* (3), 143 - 151.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., et al. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *58* (1).

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *5*, 199-220.

Guarino, N. (1998). Formal Ontology and Information Systems. Amsterdam: IOS Press.

Gómez-Pérez, A. (2002). Methodologies, tools and languages for building ontologies. *Data & Knowledge Engineering , 46* (1), 41-64.

Guojie, J. L. (2005). *Professional Java Native Interfaces with SWT/JFace.* Wiley.

Gutiérrez, D. B., Díez, C. L., Castanedo, R. I., Gayo, J. E., & Cueva, J. M. (2000). Improving the quality of compiler construction with object-oriented techniques. *ACM SIGPLAN Notices.* New York: ACM.

Hillairet, G. (01 de 02 de 2007). *ATL Use Case - ODM Implementation (Bridging UML and OWL).* Obtido em 11 de 04 de 2010, de Eclipse: http://www.eclipse.org/m2m/atl/usecases/ODMImplementation/

Jacobson, I. (1992). *Object Oriented Software Engineering: A Use Case Driven Approach.* (A.-W. Professional, Ed.)

Kabilan, V., & Johannesson, P. (2004). CAiSE Workshops (3)., (pp. 349-354).

Kalinichenko, L., Missikoff, M., Schiappelli, F., & Skvortsov, N. (2003). (Proceedings of the 5th Russian Conference on Digital Libraries RCDL2003)

Karp, P. D., Myers, K. L., & Gruber, T. (1995). The generic frame protocol. *International Joint Conference On Artificial Intelligence* (pp. 768-774). Montreal: Morgan Kaufmann Publishers Inc.

Kirasić, D., & Basch, D. (2009). Ontology Tools. In D. Kirasić, & D. Basch, *Ontology-Based Design Pattern Recognition* (Vol. Volume 5177/2009, pp. 384-393). Springer Berlin / Heidelberg.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., et al. (2002). UML for ontology development. *Knowl. Eng. Rev. , 17* (1), 61-64.

Krivov, S., Williams, R., & Villa, F. (2007). GrOWL: A tool for visualization and editing of OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web , 5* (2), 54-57.

MacGregor, R. (1991). Inside the loom clasifier. *3*, 70-76.

Mccarthy, J. (1978). *History of Lisp.* Academic Press.

Minter, D., & Linwood, J. (2005). *Pro Hibernate 3.* Apress.

Mota, L., Botelho, L., Mendes, H., & Lopes, A. (2003). O3F: an object oriented ontology framework. (pp. 639-646). Australia: ACM.

Musen, M. A. (1989). An editor for the conceptual models of interactive knowledge-acquisition tools. *Man-Machine Studies* .

Noy, N. F., Fergerson, R. W., & Musen, a. (2000). The Knowledge Model of PROTEGE-2000: Combining Interoperability and Flexibility. *Twelfth International Conference in Knowledge Engineering and Knowledge Management (EKAW- 00).* Juan-Les-Pins.

ODM. (01 de 05 de 2009). *Ontology Definition Metamodel*. Obtido de http://www.omg.org/spec/ODM/1.0/

OMG. (01 de 01 de 2006b). *Meta-Object Facility 2.0*. Obtido de http://www.omg.org/spec/MOF/2.0/

OMG. (01 de 05 de 2006a). *Object Constraint Language 2.0*. Obtido de http://www.omg.org/spec/OCL/2.0/

OMG. (01 de 02 de 2009). *Unified Modeling Language 2.2*. Obtido em 10 de 01 de 2010, de http://www.omg.org/spec/UML/2.2/

OMG. (1 de 12 de 2007). *XML Metadata Interchange 2.1.1*. Obtido de http://www.omg.org/technology/documents/formal/xmi.htm

Parr, T. (2007). *The Definitive ANTLR Reference Building Domain-Specific Languages.* The Pragmatic Programmers.

Perez-Urbina, H., Horrocks, I., & Motik, B. (2009). Practical Aspects of Query Rewriting for OWL 2. *OWL: Experiences and Directions 2009.*

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). Object oriented modeling and design. *PRENTICE HALL, BOOK DISTRIBUTION CENTER, 110 BROOKHILL DRIVE, WEST NYACK, NY 10995-9901(USA* .

Silva, F. S., Vasconcelos, W. W., Robertson, D. S., Brilhante, V., Melo, A. C., Finger, M., et al. (2002). On the insufeciency of ontologies: problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems* , 147-167.

Stanford, K. (31 de 07 de 1994). *Ontolingua Theory Frame-Ontology*. Obtido em 18 de 08 de 2010, de http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/frame-ontology/index.html

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., & Wenke, D. (2002). ONTOEDIT: Collaborative Ontolgy Engineering for the Semantic Web. *In Proceedings of the First International Semantic Web Conference (ISWC'02).* Berlin: Springer-Verlag.

Swartout, B., Ramesh, P., Knight, K., & and Russ, T. (1997). Toward Distributed Use of Large-Scale Ontologies. *Spring Symposium on Ontological Engineering.* Stanford.

W3C. (3 de 2001). *DAML+OIL Reference Description*. Obtido de http://www.w3.org/TR/daml+oil-reference

W3C. (27 de 10 de 2009). *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. Obtido em 05 de 01 de 2010, de http://www.w3.org/TR/owl2-syntax/

W3C. (2004a). *OWL Web Ontology Language Reference*. Obtido em 01 de 01 de 2010, de http://www.w3.org/TR/owl-ref/

W3C. (3 de 2004b). *RDF Vocabulary Description Language 1.0: RDF Schema*. Obtido de http://www.w3.org/TR/rdf-schema/

W3C. (3 de 2004). *RDF/XML Syntax Specification (Revised).* Obtido de http://www.w3.org/TR/rdf-syntax-grammar/

Anexo A Diagrama de classes O3F

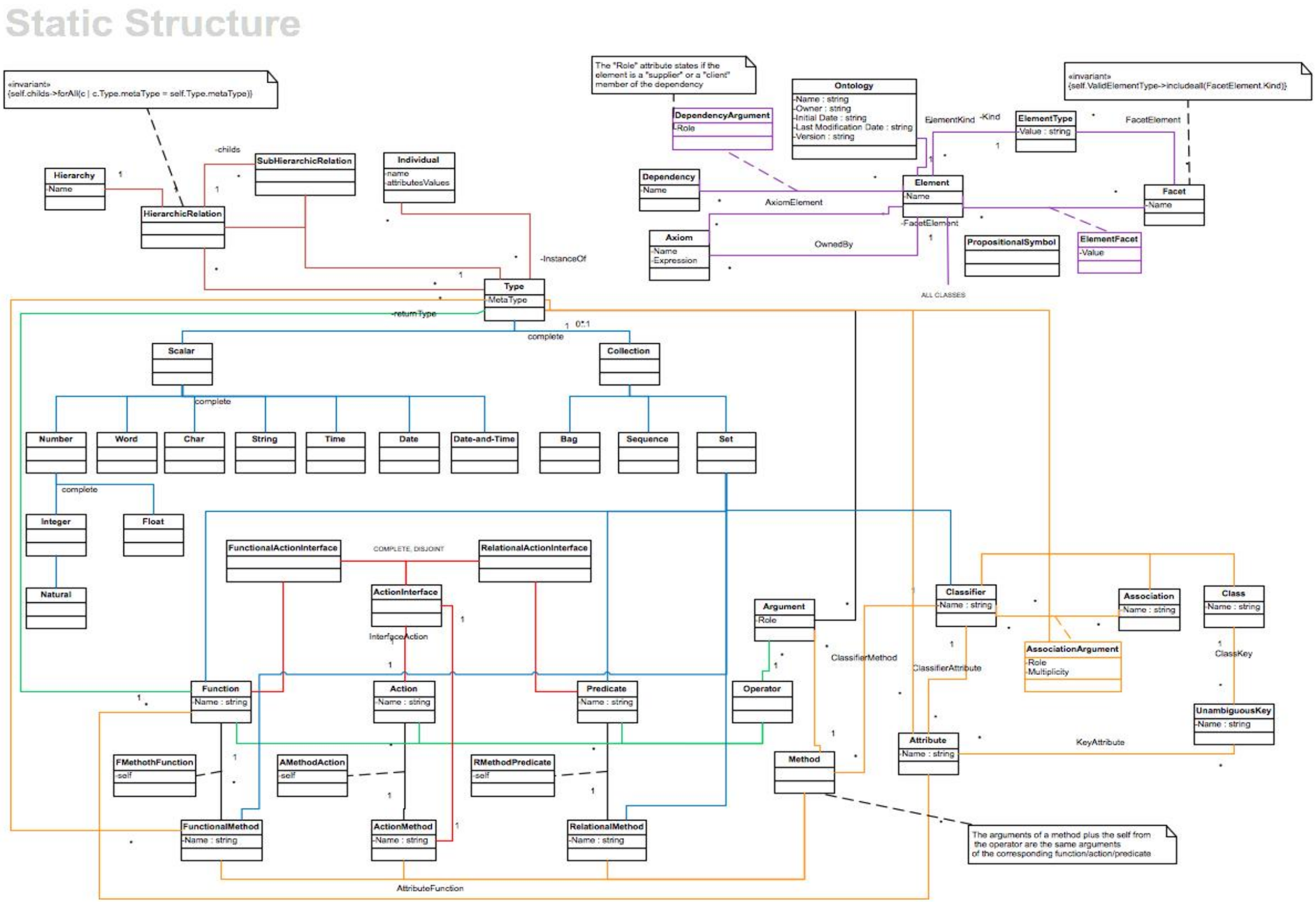

Figura 14 – Diagrama de classes O3F

Anexo B Modelo relacional O3F

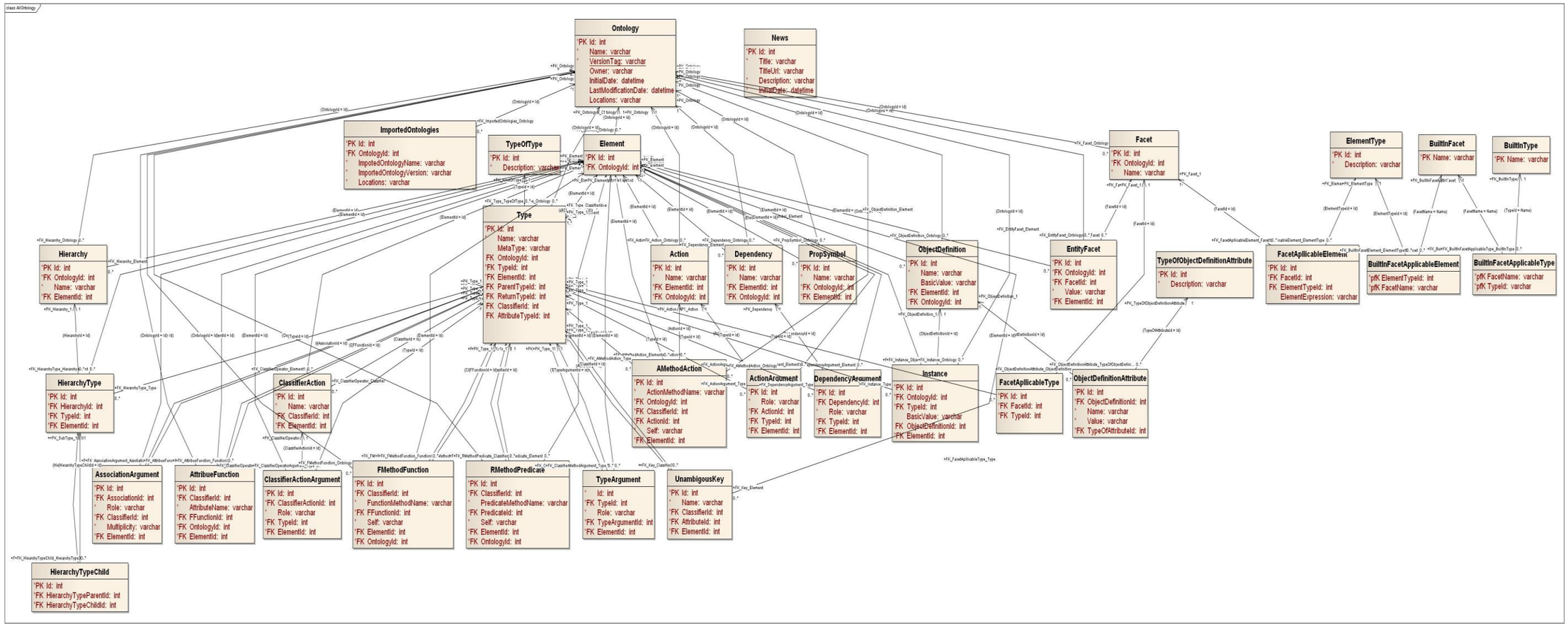

Figura 15 – Diagrama do modelo relacional