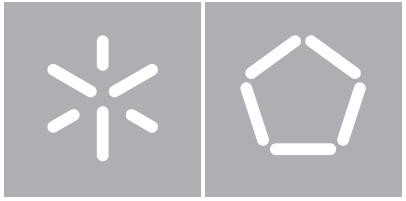
Universidade do Minho
Escola de Engenharia

Fábio Alves Rocha

Preservação de Bases de Dados com SIA

Outubro de 2014



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Fábio Alves Rocha

Preservação de Bases de Dados com SIA

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor Doutor José Carlos Ramalho

Outubro de 2014

AGRADECIMENTOS

Cada um que passa na nossa vida passa sozinho, pois cada pessoa é única, e nenhuma substitui outra. Cada um que passa na nossa vida passa sozinho, mas não vai só, nem nos deixa sós. Leva um pouco de nós mesmos, deixa um pouco de si mesmo. Há os que levam muito; mas não há os que não levam nada. Há os que deixam muito; mas não há os que não deixam nada. Esta é a maior responsabilidade de nossa vida e a prova evidente que duas almas não se encontram ao acaso.

Saint-Exupéry

Seguramente este espaço limitado não me permite agradecer, como devia, a todas as pessoas que me ajudaram, direta ou indiretamente, a cumprir os meus objetivos e a realizar esta etapa da minha formação académica. Desta forma, deixo apenas algumas palavras, poucas, contudo com um profundo sentimento de reconhecido agradecimento. Especialmente a Deus, por me ter capacitado, e por ter colocado pessoas tão especiais ao meu lado.

Aos meus pais e aos meus avós, o meu infinito agradecimento e o meu muito obrigado por todo o amor e dedicação que me deram.

À minha namorada, Natacha, que foi e sempre será fundamental na caminhada da minha vida, o meu muito obrigado por todos os momentos que vivemos juntos. O teu total apoio e encorajamento foram determinantes para a conclusão desta etapa.

À minha família, pela presença nos momentos importantes, pelo incentivo e apoio, o meu muito obrigado.

Aos amigos de sempre, obrigado pela amizade! Um agradecimento especial à Teresa e a Silvana pela paciência, ajuda e amizade ao longo desta dissertação. A nível académico agradeço aos meus amigos de curso (licenciatura e mestrado), obrigado pelos momentos divididos e por tornarem mais leve aqueles tantos dias de estudo. Foi bom poder contar convosco!

Ao Professor José Carlos Ramalho, um agradecimento especial, não só pelos conhecimentos transmitidos, mas também por toda a disponibilidade, dedicação e paciência dedicada a este trabalho. Finalmente o agradecimentos a todos os Professores. O vosso conhecimento e experiência foram determinantes no decorrer deste percurso, permitindo construir as bases da minha carreira profissional.

ABSTRACT

Nowadays, with the expansion of information technologies, much of human's knowledge has been recorded in digital media, which brings us to the use of intermediaries for reading the information: hardware and software. Due to these intermediaries are in constant evolution and the danger of their discontinuities, this information might be lost, not because of the disappearance of the digital object, but for the impossibility of new equipment and applications to interpret it. Those facts created a new problem in the digital world: digital preservation.

This project studies the problems of digital preservation, focused on a single class of digital objects: relational databases. Relational databases are of the utmost importance, particularly for organizations because all the essential information to their activities is stored on them. For this reason it is fundamental not to compromise their longevity, integrity and authenticity.

The present work aims at the development of a ontologies exploration system, which receives files in SIARD format, turns them into ontology (OWL) and adds them to the repository.

On a last stage, a Web browser that allows you to explore the information of ontologies stored was created, where it is possible to question the ontologies via SPARQL and keep those same questions for a later use.

The ontology repository was developed according to the OAIS standard, and is based on a Web application with multiple interfaces, providing, information ingestion, his administration, preservation and dissemination.

RESUMO

Atualmente, com a expansão das tecnologias de informação, muito do conhecimento humano passou a estar registado em suportes digitais, o que nos remete à utilização de intermediários para a leitura dessa informação: hardware e software. Devido a esses intermediários estarem em constante evolução e havendo o perigo da sua descontinuidade, essa informação poderá ser perdida, não pelo desaparecimento do objeto digital, mas por ficar ilegível para os novos equipamentos e aplicações. Devido a estes problemas emergiu uma nova problemática no universo digital, a preservação digital.

Este projeto estuda a problemática da preservação digital, mas foca-se numa única classe de objetos digitais: as bases de dados relacionais. As bases de dados relacionais são de extrema importância, particularmente para as organizações, pois é nas suas bases de dados que se encontra informação essencial às suas atividades. Por essa razão, é fundamental não comprometer a sua longevidade, integridade e autenticidade.

O presente trabalho visa o desenvolvimento de um sistema de exploração de ontologias, que recebe ficheiros em formato SIARD, transformando-os em ontologias (OWL) e acrescentando-os ao repositório.

Como última fase, foi criado um navegador Web que permite explorar a informação das ontologias armazenadas, onde é possível questionar as ontologias através de SPARQL e guardar essas mesmas interrogações para posterior uso.

O repositório das ontologias foi desenvolvido segundo a norma OAIS, e tem por base uma aplicação Web com várias interfaces, para assim proporcionar a ingestão da informação e a sua administração, preservação e disseminação.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE BLOCOS DE CÓDIGO

LISTA DE ACRÓNIMOS

ARELDA Archiving of Electronic Data and Records

ASCII American Standard Code for Information Interchange

BLOB Binary Large Object

CAMiLEON Creative Archiving at Michigan and Leeds

CLOB Character Large Object

DAML DARPA Agent Markup Language

DANS Data Archiving and Networked Services

DBML DataBase Markup Language

DeCS Descritores em Ciência da Saúde

DELOS Network of Excellence on Digital Libraries

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

InterPARES International Research on Permanent Authentic Records in Electronic Systems

ISO International Organization for Standardization

KIF Knowledge Interchange Format

OAIS Open Archival Information System

OCML Operational Conceptual Modelling Language

OKBC Open Knowledge Base Connectivity

OPF Ontological Programming Framework

OWL Web Ontology Language

PHP PHP: Hypertext Preprocessor

PLANETS Preservation and Long-term access via Networked Services

RDF Resource Description Framework

RODA Repositório de Objectos Digitais Autênticos

SFA Swiss Federal Archives

SHOE Simple HTML Ontology Extensions

SIARD Software Independent Archiving of Relational Databases

SPARQL Simple Protocol and RDF Query Language

SQL Structured Query Language

UTF Unicode Transformation Format

W3C World Wide Web Consortium

XML Extensible Markup Language

XOL XML-Based Ontology Exchange Language

1

INTRODUÇÃO

A preservação da informação é fundamental para a nossa evolução. O que seríamos sem toda a informação que nos foi legada pelos nossos antepassados, sem podermos ter conhecimento da nossa história e da nossa cultura através dos seus artefactos?

Hoje em dia, grande parte da informação produzida é realizada através de ferramentas digitais, o que acarreta consigo um problema de longevidade. Isso acontece devido à constante evolução do universo tecnológico, tanto a nível de hardware como a nível de software, existindo assim um problema de preservação digital.

"A preservação digital é a atividade responsável por garantir que a comunicação entre um emissor e um receptor é possível, não só através do espaço, como também através do tempo".Ferreira (2006)

Esta dissertação centra-se na preservação digital, mas apenas numa das classes de objetos digitais, as base de dados relacionais. As bases de dados distinguem-se das outras classes de objetos digitais por possuírem, além da estrutura interna, schemas e restrições de integridade, que se tornam fundamentais para a interpretação da informação. As bases de dados relacionais são de extrema importância para as organizações, pois nelas se encontra informação que é essencial para as suas atividades.

Para lidar com o problema da preservação digital das bases de dados relacionais foi criado o formato SIARD.

O objetivo deste trabalho foi a criação de um repositório de ontologias, com foco principal em ontologias que tenham sido geradas a partir de bases de dados relacionais. Este repositório tem

a particularidade de receber as bases de dados no formato SIARD e as converter em ontologias, sendo ainda possível fazer pesquisas SPARQL nas mesmas.

Esta tese está dividida em oito capítulos. Capítulo 2 explica as tecnologias e ferramentas usadas neste trabalho. Capítulo 3 está relacionado com a preservação digital, onde será abordado o estado da arte da mesma, assim como estratégias e requisitos para a preservação digital. Capítulo 4 introduz o formato SIARD, o seu conceito, a sua estrutura e o SIARD SUITE, no capítulo 5 encontra o estado da arte acerca das ontologias. Capítulo 6 é explicado o repositório, a arquitetura e o layout do mesmo. A conversão SIARD para ontologia é descriminada no capítulo 7 e, por fim, no último capítulo temos a conclusão, com os trabalhos futuros e bibliografia usada.

2

TECNOLOGIAS ENVOLVIDAS

Neste capítulo é apresentada a análise e descrição das tecnologias e ferramentas envolvidas no desenvolvimento deste trabalho. Para desenvolver a aplicação web foi utilizada a linguagem de programação PHP, assim como a linguagem Javascript foi utilizada, mas com o objetivo principal de tornar as páginas mais dinâmicas. O MySQL foi a escolha para gestão da nossa base de dados do repositório. Outras linguagens, ferramentas e formatos foram utilizadas, mas as suas descrições encontram-se ao longo da dissertação.

2.1 MYSQL

O MySQL é um sistema de gestão de bases de dados relacionais, que utiliza a linguagem SQL para a manipulação dos dados.

As suas vantagens passam pela sua fácil utilização, rapidez, fiabilidade, assim como o fato de ser gratuito e compatível com várias plataformas e sistemas operativos.

MySQL é desenvolvido, distribuído e suportado pela Oracle Corporation, sendo o sistema de gestão de bases de dados mais utilizado com a linguagem PHP.

Sendo utilizado tanto em pequenos como em grandes aplicações, no link que se segue http://www.mysql tem uma visão geral das empresas que utilizam o MySQL.

Pode ser efetuado o seu download a partir da página http://dev.mysql.com/downloads/.

2.2 LINGUAGEM DE PROGRAMAÇÃO – PHP

PHP é uma linguagem de programação que permite criar páginas Web dinâmicas e interativas de uma forma rápida e simples, sendo compatível com quase todos os servidores usados hoje em

dia, sendo ainda executada em múltiplas plataformas e suportando uma vasta gama de bases de dados.

A linguagem PHP é amplamente utilizada, livre e alternativamente eficiente aos seus concorrentes não gratuitos. O código PHP é executado no lado do servidor e enviado para o cliente apenas HTML, tendo assim a vantagem de não expor o código-fonte ao utilizador, sendo uma mais valia a nível de segurança e confidencialidade.

A linguagem PHP, criada por por Rasmus Lerdorf, em 1995, é utilizada em aplicações como o MediaWiki, Facebook, Drupal, Joomla, WordPress, Magento e o Oscommerce. O seu site oficial é PHP.net, no site encontra-se disponível a linguagem para download, sendo que as instruções de instalação para a linguagem PHP estão em: http://php.net/manual/en/install.php.

2.3 JAVASCRIPT

JavaScript é uma linguagem de programação client-side, ou seja, foi criada para que os scripts pudessem ser executados do lado do cliente e interagissem com o utilizador sem a necessidade destes serem executados pelo servidor. É uma linguagem de programação que traz melhorias para a linguagem HTML, sendo altamente dependente do navegador, que chama a página web, onde o script está incorporado, porém, por outro lado, não requer nenhum compilador.

Hoje em dia já é bastante utilizada do lado do servidor através de plataformas como o node.js.

Com o JavaScript é possível criar páginas Web dinâmicas, além de podermos proporcionar uma maior interatividade aos utilizadores.
É atualmente a principal linguagem para programação client-side em navegadores web, tendo sido desenvolvida pela Netscape, em 1995.

O JavaScript é uma linguagem orientada a objetos, ou seja, ela trata todos os elementos da página como objetos distintos, facilitando a tarefa do programador.

Em resumo, o JavaScript é uma linguagem que deve ser dominada por quem deseja criar páginas Web dinâmicas e interativas.

3

PRESERVAÇÃO DIGITAL

O problema da preservação digital acentuou-se devido à expansão tecnológica da informação nas últimas décadas. O uso das tecnologias da informação no mundo dos negócios, nas empresas, ou até mesmo na população em geral, faz com que a informação esteja agora em suporte digital, correndo o risco de no futuro essa informação ficar impercetível e inacessível, devido ao constante avanço informático a nível de Software e Hardware. Hodge (2000)

A preservação digital tem por base um conjunto de atividades ou processos responsáveis por garantir o acesso contínuo e a longo-prazo aos objetos digitais e restante património cultural existente em formatos digitais, sem que seja comprometida a sua autenticidade.

Existem diferentes estratégias a utilizar na preservação dos formatos mais convencionais, tais como documentos ou imagens. No entanto, as bases de dados distinguem-se dos formatos mais convencionais por terem, além da estrutura interna, schemas e restrições de integridade, que se tornam assim mais complexa a questão.

Há ainda um número significativo de projetos e soluções no domínio da preservação digital, em que, através dessas soluções, os utilizadores podem executar as ações de preservação desejadas. Um desses projetos é o projeto PLANETS, co-financiado pela União Europeia, projeto esse que deu origem à OPF. Em termos de preservação, o objetivo deste projeto é a preservação do património científico e de dados culturais, armazenados em formatos digitais, bem como bases de dados. O projeto possui vários serviços a fim de abordar os vários tipos e especificidades dos diferentes objetos digitais, sendo que o SIARD foi adotado pelo PLANETS para a preservação de base de dados.

Descrição de alguns dos projetos mais relevantes na preservação digital:

- O projeto RODA visa implementar uma preservação a longo prazo de conteúdos digitais. Ramalho et al. (2008).
- SCAPE é um projeto co-financiado pela União Europeia, focado na preservação digital a longo prazo de coleções de objetos digitais, coleções essas de grande escala e heterogéneas. Visa desenvolver serviços escaláveis para planeamento e ações de preservação em uma plataforma open source. King et al. (2012).
- DANS é um Instituto Holandês que promove o arquivamento e o acesso aos dados de investigação. Pretende fornecer uma plataforma para partilha de dados de pesquisa. DANS (2012).
- Projeto PLANETS é um projeto co-financiado pela União Europeia com o objetivo de construir ferramentas e serviços a fim de garantir o acesso a longo prazo de material digital cultural e científico. Farquhar (2007).
- Rosetta ExLibris fornece um sistema altamente escalável, seguro e de utilização fácil para a preservação digital por parte das bibliotecas e outras instituições, que pretendem ter as suas informações acessíveis agora e no futuro. ExLibris (2012).
- DELOS é uma rede de excelência sobre bibliotecas digitais, parcialmente financiado pela Comissão Europeia no quadro do programa IST (Information Society Technologies). Os principais objetivos da rede DELOS são pesquisa, cujos resultados estão no domínio público, através de acordos de cooperação com as partes interessadas. DELOS (2009). É desenvolvido ainda um modelo de referência para as Bibliotecas Digitais, o DelosDLMS, que servirá como uma implementação concreta do modelo de referência e vai abranger muitos dos componentes de software desenvolvidos pelos parceiros da rede DELOS.

Neste capítulo começamos por explicar o objeto digital. Em seguida, é apresentada uma visão geral das diferentes perspetivas e estratégias de preservação digital e são ainda expostos os projetos que se destacam por estarem relacionados com a preservação das bases de dados relacionais.

3.1 OBJETO DIGITAL

Um objeto digital pode ser definido como todo e qualquer objeto de informação que possa ser representado através de uma sequência de dígitos binários. Thibodeau (2002)

Os objetos digitais podem ser divididos em dois grupos, os objetos digitalizados (criados originalmente num outro suporte e posteriormente digitalizados) e os nado- digitais (documentos criados em formato digital, já falados anteriormente como aplicações de software e bases de dados).

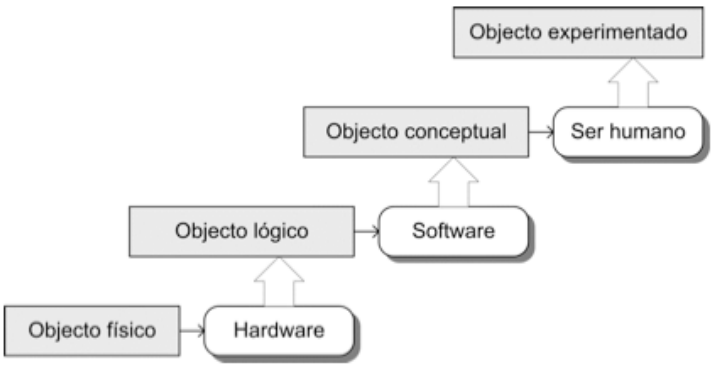


Figura 1: Níveis de abstração presentes num objeto digital. Ferreira (2006).

Esses objetos digitais são armazenados em suporte físico (por exemplo, discos rígidos, CD, DVD), que requerem dispositivos específicos para os ler. A informação armazenada nestes meios é caracterizada por um conjunto de símbolos, que são organizados e controlados por regras diferentes, dependendo do meio físico usado. Estas estruturas de dados são o nível lógico de abstração presente num objeto digital, que definem o formato do objeto, dependendo do software usado para os interpretar. Essa interpretação do objeto lógico irá corresponder ao aparecimento do objetoobjeto conceptual (aquele que o ser humano é capaz de entender) e então poder experimentar (objeto experimentado). Ferreira (2006).

A figura 2, mostra os níveis de abstração de um objeto digital, quando este é uma base de dados. Este esquema faz todo o sentido, visto que o nosso objeto de estudo são as bases de dados digitais.

Preservação digital é um processo ou um conjunto de processos que deve seguir um plano de atividades concreto, com a alocação de recursos adequados e utilização de tecnologias e práticas que garantam o acesso a um objeto digital, numa perspetiva de longo prazo.

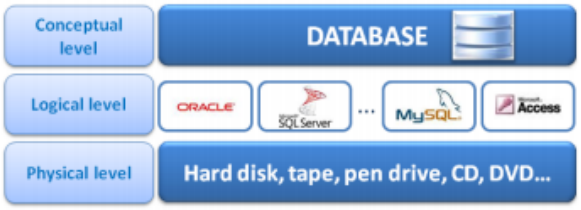


Figura 2: Níveis de abstração presentes num objeto digital, quando este é uma base de dados. Aldeias (2011).

Para ser capaz de preservar um objeto digital, os níveis de abstração acima descritos devem ser acessíveis e interpretáveis, caso contrário deixa de ser possível a utilização desse objeto.

3.2 REQUISITOS PARA PRESERVAÇÃO A LONGO PRAZO

Para uma preservação a longo prazo ser bem sucedida, no contexto das bases de dados, existem requisitos que se deve ter em consideração:

- Integridade - deve garantir que os dados armazenados na base de dados são corretos e consistentes, permanecendo intactos, sem alterações ou corrupções. Por exemplo, se houver uma referência a um objeto específico ou entidade, o objeto deve existir na base de dados e os seus dados devem ser precisos. Desai (1990).
- Autenticidade - A autenticidade dos dados deve ser vista como um conceito-chave na preservação, o que significa que os dados preservados não devem ser adulterados ou corrompidos. Gilliland-Swetland and Eppard (2000).
- Inteligibilidade - a inteligibilidade de uma base de dados é definida pela capacidade de perceber e interpretar os formatos de dados e os relacionamentos entre as tabelas e o que eles representam na realidade. Rahman et al. (2010).
- Acessibilidade - a acessibilidade para uma base de dados é garantir a possibilidade de utilizar os seus dados em formatos abertos que não exijam software específico, garantindo acesso aos dados em uma perspetiva de preservação a longo prazo. Rahman et al. (2010).

3.3 ESTRATÉGIAS PARA A PRESERVAÇÃO DIGITAL

Muitas estratégias foram propostas ao longo dos anos, assim como muita investigação tem sido elaborada. Projetos como CAMiLEON [Wheatley (2001)], InterPARES [Gilliland-Swetland and Eppard (2000)] ou FEDORA [Lagoze et al. (2005)] contribuíram para o estudo dos requisitos e das estratégias para a preservação dos objetos digitais e a sua autenticidade. Para os objetos digitais mais complexos, como as bases de dados, existem projetos como o RODA [Ramalho et al. (2007)], Chronos [Brandl and Keller-Marxer (2007)] e o SIARD [Innovation and Preservation (2008)].

Existem diferentes opiniões / preocupações acerca do que deve ser preservado, existindo quem defenda que deve ser garantida a autenticidade, e que a única maneira de o fazer é através da ”Preservação de tecnologia”, para assim garantir o objeto na sua forma original. Outros defendem que basta preservar o objeto conceptual, que no fundo é o que interage com o mundo real.

Existem outros tipos de abordagem, verificando-se que esta problemática não é nada consensual.

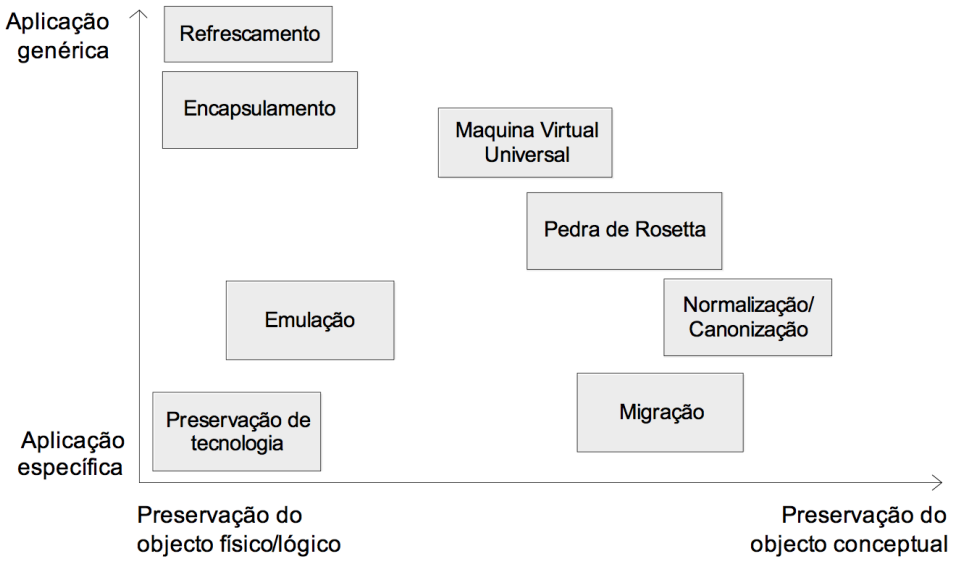


Figura 3: Classificação das diferentes estratégias de preservação digital. Ferreira (2006).

3.3.1 *Preservação de tecnologia*

A Preservação de Tecnologia é uma das estratégias para a preservação digital. Esta estratégia consiste, fundamentalmente, na conservação e manutenção de todo o hardware e software necessário à correta apresentação dos objetos digitais. Lee et al. (2002).

A presente técnica atribui especial atenção à preservação do objeto digital na sua forma original, criando para isso museus de tecnologia. Os defensores desta estratégia justificam que esta é a única forma suficientemente eficaz para assegurar que os objetos digitais são experimentados de forma fidedigna. Os principais problemas neste tipo de estratégia são o custo, a gestão do espaço físico e manutenção, ficando o acesso a estes objetos restrito a alguns locais físicos, contudo é possível imaginar que para alguns tipos de artefactos digitais seria interessante a sua preservação segundo esta abordagem.

3.3.2 *Refrescamento*

O refrescamento é outra técnica usada na preservação digital. Consiste na passagem da informação que está num determinado suporte físico, que pode no futuro ficar obsoleto, para um formato mais atual, para assim não incorrermos no risco de se perder a informação que contem para sempre. Como exemplo desta prática temos as cópias de informação de suportes que estão a ficar obsoletas, como por exemplo as disquetes, para suportes mais atuais, como os CDs ou DVD's.

Com esta técnica podemos pelo menos garantir que a informação está acessível do ponto de vista do hardware. Claro que é necessário combinar esta técnica com outras estratégias para haver uma preservação digital bem sucedida. Besser (2001).

3.3.3 *Emulação*

Uma estratégia já muito utilizada é a emulação, que consiste essencialmente na utilização de um software, designado por emulador, capaz de reproduzir o comportamento de outro software. Uma das desvantagens é que este emulador no futuro também poderá sofrer de obsolescência. É

importante relevar,que criar um emulador nunca será uma solução final para a preservação dos objetos digitais.

Um emulador é um software que tenta recriar as condições tecnológicas para que uma determinada aplicação possa correr sobre ele, sendo assim possível recriar o ambiente original. Esta técnica assume uma relevância maior quando falamos em aplicações de software com aspetos dinâmicos e interativos. É assim capaz de atingir altos níveis de preservação no que diz respeito às propriedades e características do objeto digital original. Lee et al. (2002).

Os emuladores de consolas de jogos são um exemplo real da utilização desta técnica. Através desta técnica é assim possível executar jogos antigos, mesmo na ausência física da consola, jogos esses que foram originalmente desenvolvidos para correr sobre uma determinada consola.

3.3.4 *Migração*

A migração é uma das estratégias mais usadas atualmente na preservação digital e com mais provas dadas. A migração centra-se sobretudo na preservação do seu conteúdo intelectual, ou seja, na preservação do objeto concetual ao contrário das estratégias já apresentadas que tentavam manter o objeto no seu formato original.

Esta estratégia tem como objetivo manter a informação sempre num estado atual e interpretável pelas tecnologias atuais, para que assim o utilizador comum seja capaz de interpretar os objetos digitais. Lee et al. (2002).

Existem ainda algumas desvantagens na migração, assim como a probabilidade de algumas das propriedades do objeto digital não serem bem transferidas para o novo formato de destino. Isto acontece devido às incompatibilidades existentes entre os formatos de origem e destino, ou à utilização de conversores incapazes de realizar a migração corretamente. Rauber and Aschenbrenner (2001).

Esta estratégia não resolve a questão definitivamente, pois será uma questão de tempo até que uma nova migração tenha de ser feita porque o formato de destino também se encontra sob ameaça de se tornar obsoleto.

Existem diversas variantes de migração: migração a-pedido, conversão para formatos concorrentes, atualização de versões, normalização e migração distribuída.

Migração a-Pedido

A Migração a-Pedido é uma das variantes da Migração, tendo sido criada esta estratégia para combater o fenómeno de degradação que acontecia em migrações sucessivas.

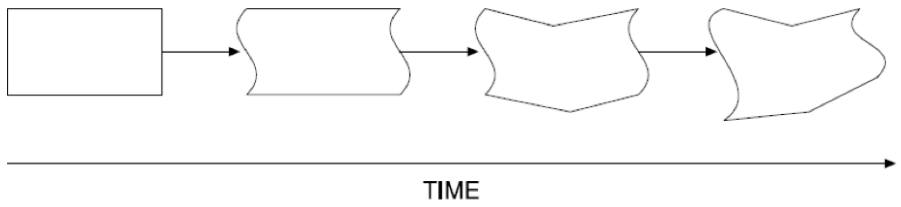


Figura 4: Degradação do objeto digital ao longo de sucessivas migrações. Ferreira (2006).

Como se pode ver pela figura 4, com as sucessivas migrações os objetos digitais degradam-se e para tal não acontecer neste tipo de migração, ao invés de as conversões serem aplicadas ao objeto mais atual, estas são sempre aplicadas ao objeto original como se pode ver na figura 5. Assim, caso uma dada conversão resultar em um objeto substancialmente diferente do original, o problema poderá ser resolvido recorrendo a um conversor de melhor qualidade ou a um formato de destino mais adequado numa futura conversão.

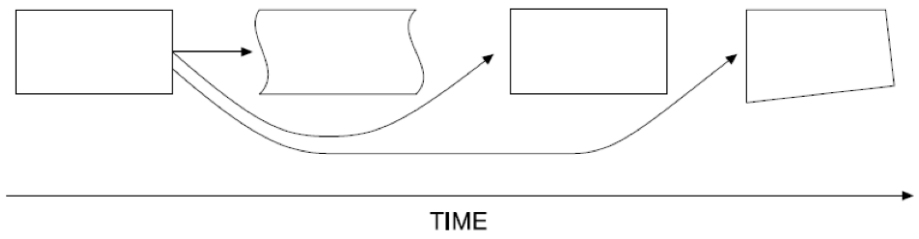


Figura 5: Migração a-pedido. Ferreira (2006).

Um migração bem sucedida depende da qualidade dos conversores e da capacidade que o formato de destino tem para conter o conjunto de propriedades do formato inicial.

Migração Distribuída

A variante mais recente da migração é a Migração Distribuída, sendo que esta técnica consiste num conjunto de serviços de conversão on-line que proporcionam assim uma conversão de diferentes formatos através de uma aplicação-cliente, os chamados serviços Web (Figura 6).

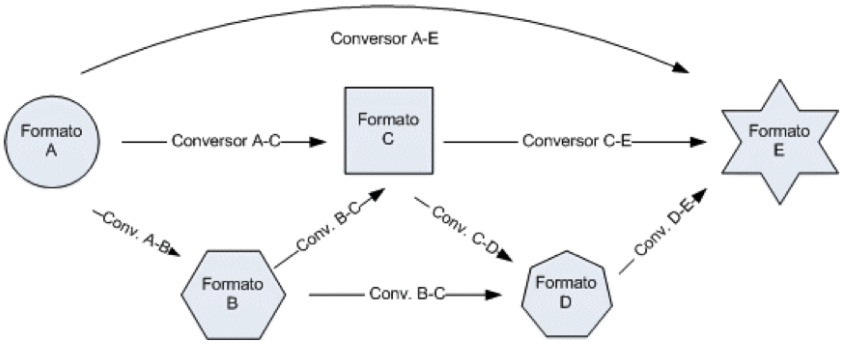


Figura 6: Migração distribuída baseada em Serviços Web. Ferreira (2006).

Algumas vantagens desta estratégia comparada com as mais convencionais:

- Esta variante é compatível, por exemplo com a migração a-pedido;
- Se algum conversor falhar ou ficar obsoleto existe sempre outros conversores que possam usar, existe assim múltiplos caminhos;
- Podemos assim criar uma rede global de conversores que pode contribuir muito para o sucesso da preservação.

A migração distribuída poderá não ser adequada a todos os contextos, verificando-se assim algumas das suas possíveis desvantagens:

- É necessário uma largura de banda muito elevada;
- Segurança dos dados;

- Tempo de transferência pode ser muito elevado.

3.3.5 *Normalização*

A normalização pretende encontrar formatos que sejam amplamente utilizados e com normas internacionais abertas. Com isto pretende simplificar o processo de preservação através da redução do número de formatos distintos que se encontram em um repositório de objetos digitais Thibodeau (2002). Havendo menos formatos, uma estratégia de preservação poderá ser aplicada a mais objetos digitais, o que poderá levar a uma redução generalizada dos custos de preservação que são muito elevados.

Vejamos um exemplo concreto. Na representação de imagens existem formados como JPEG, PNG e GIF, se durante o processo de ingestão de um repositório, todas as imagens digitais forem convertidas para um único formato, futuras intervenções ao nível da sua preservação poderão ser realizadas de forma mais simples e mais económica, tomando em atenção que a escolha do formato é muito importante. Ramalho et al. (2007)

3.3.6 *Encapsulamento*

Para aqueles objetos em que a migração não faz sentido ou é demasiado dispendiosa, porque os objetos não estão a ser necessários durante anos, deve ser usado o encapsulamento. Esta estratégia consiste em preservar, juntamente com o objeto digital, toda a informação (meta-informação) necessária e suficiente para assim no futuro ser desenvolvidos conversores, visualizadores ou emuladores. A informação deve ser uma descrição formal e detalhada do formato do objeto preservado. Ferreira (2006).

3.4 TRABALHOS RELACIONADOS

Esta secção resume o estado da arte na preservação digital, descrevendo os projetos desenvolvidos nesta temática, projetos estes em que as bases de dados relacionais são o objeto digital a preservar.

3.4.1 *RODA*

O Repositório de Objetos Digitais Autênticos foi desenvolvido pela Direção Geral de Arquivos (DGARQ), em parceria com a Universidade do Minho.

Este projeto teve como propósito o desenvolvimento de um arquivo digital com capacidade de integrar, gerir e disseminar os objetos digitais produzidos na Administração Publica, permitindo assim dar resposta ao desafio da preservação digital na vertente da gestão continuada de objetos digitais.

O RODA foi construído tendo como referência o modelo OAIS para assim assegurar a preservação e autenticidade das informações arquivadas. A base do repositório assenta na plataforma FEDORA, que oferece uma arquitetura projetada para servir como base para a implementação de repositórios digitais numa grande variedade de aplicações, tais como gestão de bibliotecas, sistemas de produção de multimédia, repositórios de arquivo, repositórios institucionais, bibliotecas digitais para educação.

Foram considerados para este projeto três classes de objetos digitais: texto estruturado (documentos Word, PDF, OpenOffice...), imagens (jpeg, tiff, png, gif...) e bases de dados relacionais (Access, Oracle, SQL Server...).

Para garantir a preservação a longo prazo das bases de dados, o RODA utiliza DBML. As base de dados são guardadas num único arquivo XML, que contém a estrutura e os dados da mesma.

3.4.2 *Chronos*

O projeto Chronos Archiving foi desenvolvido pelo departamento de ciências de computação da Universidade de Ciências Aplicadas em Landshut, Alemanha, em cooperação com a empresa CSP.

Este projeto foi financiado pelo governo da Baviera (Alemanha) entre os anos 2004 e 2006, sendo o principal objetivo a preservação das bases de dados a longo prazo. Este utiliza formatos abertos, independentes do sistema original para facilitar a recuperação dos dados arquivados.

A versão comercial Chronos foi inaugurada em 2007, como resultado do trabalho realizado no projeto Chronos Archiving. A sua interface para a gestão das bases de dados relacionais, foi implementada com todos os componentes de um sistema de arquivo OAIS (ISO 14721).

A fim de garantir o acesso futuro e a interpretação dos dados do arquivos, o Chronos extrai os dados das bases de dados e cria arquivos em formato de texto (ASCII / UTF com os dados primários e XML com os metadados). Assim sendo, podemos aceder aos dados arquivados sem qualquer programa e até mesmo sem o software Chronos. Pode-se também usar um navegador web para aceder aos dados arquivados e fazer pesquisas normais de texto e consultas SQL.

3.4.3 *SIARD*

O Software Independent Archiving Of Relacional Databases (SIARD) foi desenvolvido pelos Arquivos Federais Suíços, para lidar com o problema da preservação de base de dados relacionais e foi conceptualizado e desenvolvido como parte do projeto ARELDA. O SIARD foi apresentado em 2004 e desde então está a ser suportado pelo projeto PLANETS.

O software SIARD SUITE e a versão completa do formato SIARD foram apresentados em 2008. O SIARD SUITE converte as bases de dados: Oracle, Microsoft Acess, Microsoft SQL Server e MySQL para o formato SIARD (com a extensão de ficheiro .siard).

Para assegurar que o armazenamento de dados seja acessível no futuro, o SIARD SUITE usa SQL 1999, UNICODE e XML 1.0. O formato livre SIARD é uma das soluções mais aceites a nível mundial na preservação a longo prazo das bases de dados relacionais. Este formato vai ser devidamente analisado no capítulo 4.

4

SIARD

Ao arquivar os seus dados, as organizações / instituições garantem o acesso futuro aos mesmos e previnem a sua perda. Quase 85% dos registos arquivados estão inativos, registos esses que se encontram em bases de dados muito complexas e dispendiosas quanto à sua manutenção. Arquivar é muita vezes obrigatório por Lei, tanto como prova futura, como para documentar atividades. Arquivar garante assim uma boa resposta às necessidades, preenchendo os requisitos legais, facilitando a gestão de dados e reduzindo os custos operacionais.

Até recentemente, arquivar bases de dados relacionais numa perspetiva de preservação a longo prazo mostrava-se praticamente impossível devido à falta de standards (padrões). O formato SIARD foi desenvolvido a fim de se tornar um standard (padrão), corrigindo essa falha. SIARD foi então desenvolvido como parte do projeto ARELDA dos Arquivos Federais Suíços (SFA).

4.1 A SOLUÇÃO SUÍÇA

Para solucionar o problema, os Arquivos Federais Suíços (SFA) criaram o formato SIARD, um formato livre, normalizado e publicado. Foi ainda criado o SIARD SUITE, uma ferramenta para a conversão de bases de dados para o formato SIARD. Através deste podemos armazenar e aceder ao conteúdo das bases de dados, incluindo metadata e relações, oferecendo assim uma solução no que toca à preservação de bases de dados num panorama de longo prazo.

4.2 CONCEITO

Para assegurar que o armazenamento de dados seja acessível no futuro, o formato SIARD e o SIARD SUITE usam normas ISO: SQL 1999, UNICODE e o mais importante de todos XML. Em regra geral, todos os dados são armazenados num conjunto de caracteres Unicode, sendo que

durante a extração das bases de dados que suportam outros conjuntos de caracteres é realizado o mapeamento para os caracteres Unicode correspondentes. O SIARD não arquiva sinónimos uma vez que não fazem parte da normalização SQL 1999.

Por todas estas razões, o formato livre SIARD é uma das soluções mais aceites a nível mundial na preservação a longo prazo das bases de dados relacionais, apresentando-se como uma solução viável e prática.

4.3 ESTRUTURA DO FORMATO SIARD

Neste formato a informação das bases de dados é divida em dois componentes: metadados, que descrevem a estrutura da base de dados arquivado e os dados primários, que representam o conteúdo das tabelas. Os metadados fornecem ainda informações acerca de onde encontrar os dados primários no arquivo.

Os metadados e os dados primários são armazenados juntos apenas num arquivo ZIP (standard ZIP-64) não comprimido com a extensão ".siard". Os dados primários são armazenados na pasta content e os metadados na pasta header. A estrutura do arquivo SIARD pode ser observada na figura 7.

Os metadados estão disponíveis num único ficheiro, metadata.xml na pasta header, sendo que o arquivo está construido de uma forma hierárquica como uma base de dados relacional.

O código 4.1 mostra um extrato de um ficheiro metadata.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="metadata.xsl"?>

<siardArchive xmlns="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.
    xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="
    1.0" xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/1.0/
    metadata.xsd metadata.xsd">
  <dbname>accounting</dbname>
  ...
  <databaseUser>ACCOUNTING</databaseUser>
  <schemas>
```

```
    <schema>
      <name>ACCOUNTING</name>
      <folder>schema0</folder>
      <tables>
        <table>
          <name>Kontenplan</name>
          <folder>table1</folder>
          <description/>
          <columns>
            <column>
              <name>KONTO</name>
              <type>CHARACTER VARYING(255)</type>
              <typeOriginal>varchar(255)</typeOriginal>
              <nullable>true</nullable>
            </column>
            ...
          </columns>
          <rows>199</rows>
        </table>
        ...
      </tables>
    </schema>
  </schemas>
  <users>
    <user>
      <name>ACCOUNTING</name>
    </user>
  </users>
  <roles>
    <role>
      <name>public</name>
      <admin/>
    </role>
  </roles>
  <privileges>
    <privilege>
      <type>SELECT</type>
      <object>TABLE ACCOUNTING."Kontenplan"</object>
      <grantor>dbo</grantor>
      <grantee>ACCOUNTING</grantee>
      <option>ADMIN</option>
```

```
        </privilege>
        ...
      </privileges>
    </siardArchive>
```

Código 4.1: Exemplo metadata.xml SIARD

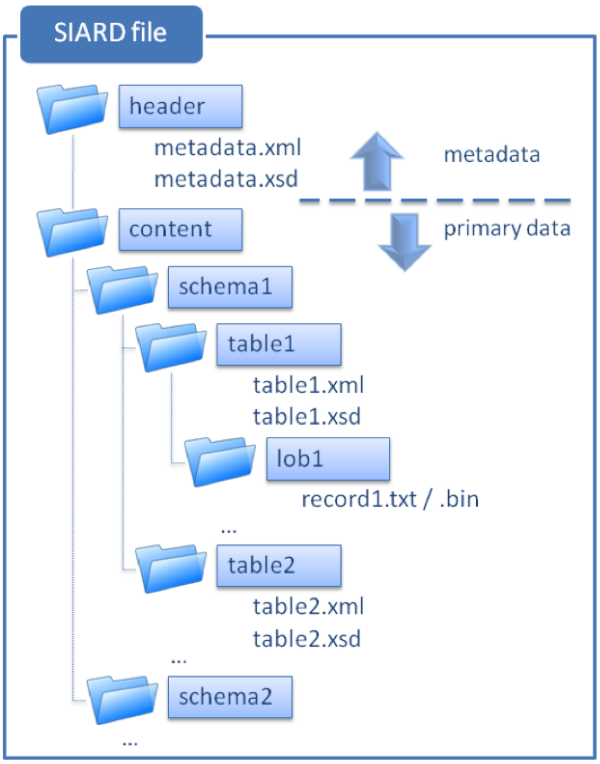


Figura 7: Estrutura do arquivo SIARD. Aldeias (2011).

A tabela 1 descreve o conteúdo de cada nível. Um 'sim' na coluna 'opcional' indica que o item é opcional, ou seja, é não obrigatório.

A pasta header contém também o ficheiro metadata.xsd, um arquivo xsd que se expressa precisamente como o XMLSchema para validar o documento XML (metadata.xml).

A Figura 8 fornece uma visão geral do XMLSchema que o documento XML tem de cumprir.

Pode encontrar uma descrição completa do formato SIARD em Innovation and Preservation (2008).

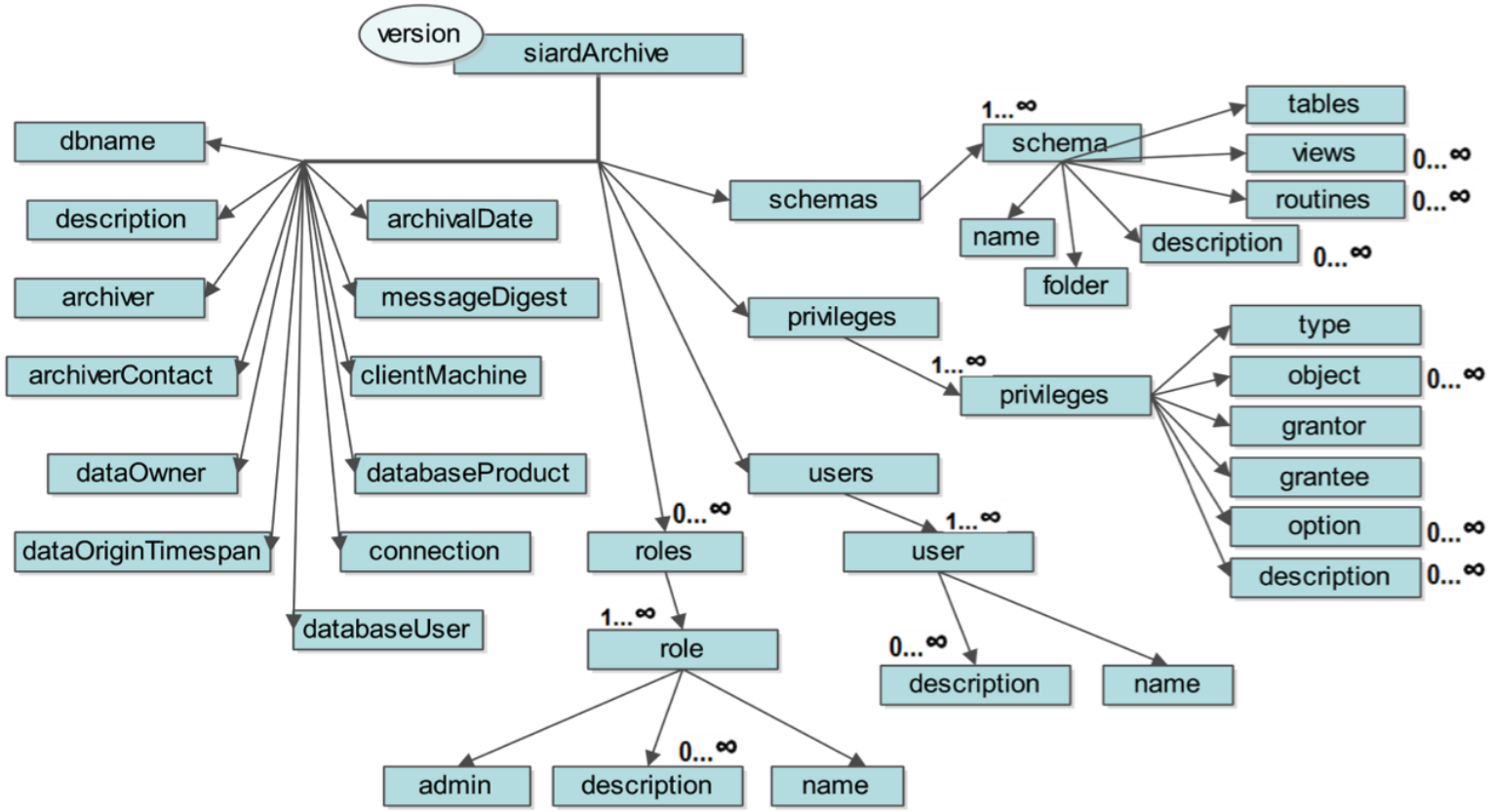


Figura 8: SIARD Schema (metadata.xsd). Freitas (2012).

Identificador	Opcional	Descrição
version	não	Versão do formato SIARD atualmente ”1.0”.
dbname	não	Nome (identificação curta) da base de dados.
description	sim	O significado e o conteúdo da base de dados como um todo.
archiver	sim	Nome da pessoa que procedeu ao arquivamento dos dados da base de dados.
archiverContact	sim	Dados de contacto (telefone, e-mail) da pessoa que procedeu ao arquivamento dos dados da base de dados.
dataOwner	não	Titular dos dados da base de dados; a instituição ou pessoa que, no momento do arquivamento, tem o direito de conceder direitos de uso dos dados e que é responsável pelas obrigações legais, tais como as diretrizes de proteção de dados.
dataOriginTimespan	não	Data de origem dos dados da base de dados; uma hora aproximada em texto.
producerApplication	sim	Nome e versão do programa que criou o arquivo SIARD da base de dados.
archivalDate	não	Data de arquivamento; Data em que os dados primários foram arquivados.
messageDigest	não	Código hexadecimal da mensagem digest sobre a pasta content com um prefixo que indica o tipo de Algoritmo Digest (MD5 ou SHA1). O message digest facilita uma rápida verificação da integridade dos dados primários.
clientMachine	sim	DNS nome do computador (cliente) em que o arquivo foi criado.
databaseProduct	sim	Produto e versão da base de dados, em que o arquivo dos dados primários foi feito.
connection	sim	A string de conexão usada para o arquivo dos dados primários.
databaseUser	sim	UserId do utilizador do SIARD-tool usado para arquivar os dados primários da base de dados.
schemas	não	Listas de shemas da base de dados.
users	não	Lista de utilizadores da base de dados.
roles	sim	Lista de funções (roles) da base de dados.
privileges	sim	Lista de privilégios e funções dos utilizadores.

Tabela 1: Lista de metadados presente no ”metadata.xml”.

Schema Metadados		
Identificador	**Opcional**	**Descrição**
name	não	Nome do schema da base de dados.
folder	não	Nome da pasta onde se encontra o schema, pasta essa que se encontra dentro da pasta content do arquivo SIARD.
description	sim	Descrição do significado e do conteúdo do schema.
tables	não	Lista de tabelas na base de dados.
views	sim	Lista de consultas armazenadas na base de dados.
routines	sim	Lista de rotinas (procedimentos armazenados anteriormente) no schema.

Tabela 2: Lista de metadados do Schema presente no ”metadata.xml”.

Identificador	**Opcional**	**Descrição**
name	não	Nome da tabela no schema.
folder	não	Nome da pasta onde se encontra a informação da tabela.
description	sim	Descrição do significado e do conteúdo da tabela.
columns	não	Lista de colunas na tabela.
primaryKey	sim	Chave primária da tabela.
foreignKeys	sim	Lista de chaves estrangeiras da tabela.
candidateKeys	sim	Lista de chaves candidatas da tabela.
checkConstraints	sim	Lista de restrições verificadas na tabela.
triggers	sim	Lista de triggers da tabela.
rows	não	Número de dados na tabela.

Tabela 3: Lista de metadados das tabelas presente no ”metadata.xml”.

4.3.1 *Dados primários no arquivo SIARD*

Os dados primários da base de dados relacional podem ser encontrados na raiz da pasta content no arquivo SIARD. Se este arquivo estiver vazio significa que contém apenas as definições de metadados que descrevem a estrutura da base de dados.

Os dados primários de cada tabela são arquivados no arquivo SIARD na pasta content, numa subpasta do schema ao qual a tabela pertence. SIARD gera os nomes schema1, schema2, schema3..., automaticamente para as pastas de schema e table1, table2, table3... para as pastas de tabelas. São criados dois arquivos (table.xml e table.xsd) para cada tabela da base de dados. BLOB's e CLOB's (Binary ou Character Large Objects que contém todo o tipo de informação) também são arquivados, armazenados em pastas geradas automaticamente (por exemplo, lob1, lob2, etc) seja em ficheiros TXT ou BIN (record1.text, ou record1.bin, etc), o seu caminho está referenciado no XML da tabela correspondente.

Um dos principais benefícios de ter um ficheiro XML para cada tabela é reduzir o tamanho de cada arquivo XML. Desta forma, os dados estão distribuídos entre vários ficheiros o que aumenta a eficiência de análise e consulta de dados e pode ser extremamente útil para a análise e consulta de arquivos em várias tabelas simultaneamente, resolvendo assim uma consulta, envolvendo mais de um arquivo XML (tabela).

Um breve exemplo de uma tabela convertida no formato SIARD é dado na Figura 4.2 .

```
<?xml version="1.0" encoding="utf-8"?>
<table
  xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table0
      .xsd table0.xsd"
  xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <c1>20020101</c1>
    <c4>1010</c4>
  </row>
  <row>
    <c1>20020101</c1>
    <c4>1010</c4>
    <c5>8001</c5>
```

```
        <c6>699.8</c6>
        <c8>0.0</c8>
        <c9>0.0</c9>
        <c10>0.0</c10>
      </row>
    </table>
```

Código 4.2: Exemplo table0.xml SIARD

4.4 SIARD SUITE

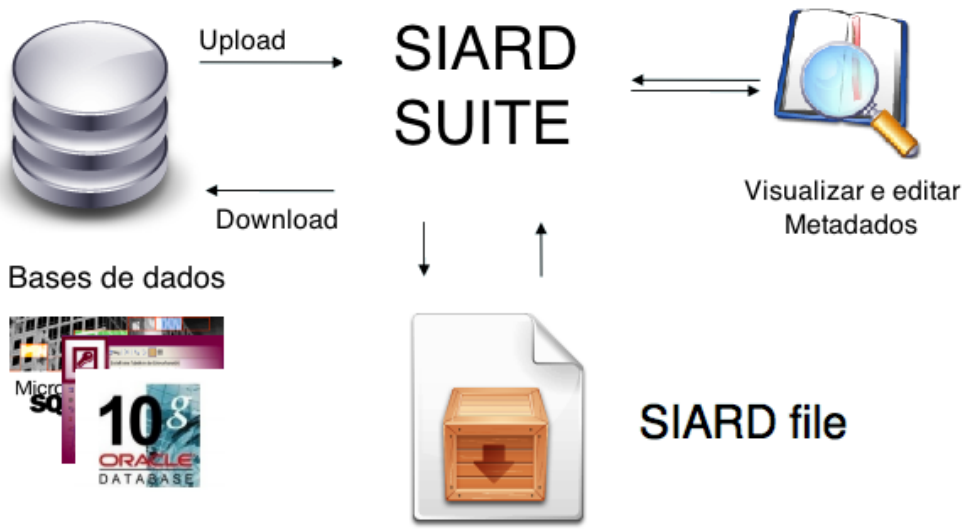


Figura 9: SIARD SUITE.

O software SIARD Suite converte as bases de dados numa coleção de arquivos XML, preservando o conteúdo, as relações e os metadados. Através da interface do SIARD SUITE é possível visualizar os dados primários e editar os metadados. SIARD Suite (figura 9) é composto por três componentes principais:

- SiardEdit permite aos utilizadores:
  - Editar os metadados;

  - Criar um arquivo SIARD com metadados novos;
  - Fazer a correspondência dos metadados de arquivos diferentes;
  - Editar e completar os metadados já existentes;
  - Visualizar e classificar os seus dados primários.

- SiardFromDb é uma ferramenta de migração, em que os utilizadores podem:
  - Converter bases de dados (Oracle, Microsoft SQL Server e Microsoft Access) no formato SIARD.
  - Criar um arquivo SIARD com metadados novos;
  - Criar um arquivo de SIARD completo (com metadados e dados primários no formato SIARD), ou:
  - Gerar um arquivo SIARD vazio (ou seja, sem dados primários).

- SiardToDb permite aos utilizadores carregar arquivos SIARD em qualquer um dos sistemas de suporte de bases de dados: Oracle, Microsoft SQL Server ou o Microsoft Access. Por exemplo, é possível converter uma base de dados Oracle num arquivo SIARD e, em seguida, carregar o arquivo SIARD numa nova base de dados Microsoft SQL Server. Principais funções:
  - Facilitar a pesquisa dentro de uma determinada base de dados;
  - Criar uma instância de uma base de dados (com tabelas etc.) através do arquivo SIARD;
  - Navegar e pesquisar nas bases de dados.

4.5 CONCLUSÃO

- O formato SIARD fornece um formato livre para a preservação a longo prazo de bases de dados relacionais, bem como o Suite de SIARD para fazer as conversões e o armazenamento das mesmas;
- O formato SIARD é baseado em padrões ISO como XML, SQL:1999 e UNICODE;
- SIARD Suite foi desenvolvido pelos Arquivos Federais Suíços (SFA) e foi integrado no âmbito do projeto de investigação europeu PLANETS. O SFA declarou o formato SIARD

como formato utilizado para o arquivamento das suas bases de dados, sendo assim o formato oficial do Governo Federal Suíço.

- SIARD Suite suporta os formatos mais comuns usados em bases de dados: Oracle, Microsoft SQL Server e o Microsoft Access;
- SIARD Suite pode arquivar bases de dados de larga escala;
- SIARD Suite é uma plataforma que funciona com Java 1.5 (ou superior), em Windows, Linux e Mac OS X;
- A interface SIARD Suite é auto-explicativa e fácil de usar, estando disponível em inglês, alemão, francês e italiano.

5

ONTOLOGIA

O termo ”Ontologia”, com origem na Filosofia, surge aquando da tentativa, pela parte dos povos ancestrais deparados com variados entraves, de encontrar a essência das coisas através da mudança. “Ontologia” deriva, assim, do grego ”ontos”, ser, e ”logos”, palavra, e é usada para denominar o ramo da metafísica que estuda as teorias sobre a natureza da existência. Este significado, para a filosofia, foi introduzido no século XVII e surge mais tarde, a partir do século XX, como uma área de investigação nas ciências da computação, tornando-se extremamente relevante nas comunidades da inteligência artificial e da engenharia do conhecimento. É ainda de salientar que este termo possui um sentido dissemelhante nos sistemas de informação e na inteligência artificial (IA), comparativamente ao tradicionalmente adotado na filosofia. Punuru (2007)

São diversas as definições deste termo apresentadas ao longo dos tempos, existindo contradições. De forma simplificada, a ontologia define a terminologia / vocabulário usado para descrever e representar uma área de conhecimento, tal como Física ou Filosofia, podendo a mesma ser usada por pessoas e aplicações para a troca de informações sobre a área em questão.

A ontologia é um modelo de dados representativo de um conjunto de definições de conceitos e da relação entre eles sobre um tema em particular, permitindo, assim, aos seus utilizadores partilharem a mesma terminologia e o mesmo significado, simplificando a sua comunicação. Guarino (1998)

Para que uma ontologia seja facilmente interpretada e usada por agentes de software, torna-se essencial estabelecer a sintaxe e formalismos semânticos. O problema de heterogeneidade semântica é ultrapassado pelo uso de ontologias capazes de explicar o conhecimento implícito e explícito. H. Wache et al. (2001)

5.1 IMPORTÂNCIA DAS ONTOLOGIAS

O uso de ontologias possibilita o desenvolvimento de sistemas mais inteligentes, dado que o conhecimento é representado formalmente de modo a suportar extração explicita e implícita deste. Sowa (2002)

Programadores e analistas de sistemas depreenderam que mais importante que a sofisticação dos softwares, as funcionalidades e os seus processos, eram os dados usados pelos sistemas.

As ontologias têm uma ampla gama de utilizações, entre as quais, e segundo Noy and McGuinness (2001):

- partilhar uma compreensão comum da estrutura de informação entre pessoas ou softwares,
- possibilitar a reutilização de conhecimento de domínio,
- elaborar suposições de domínio explicitas,
- analisar conhecimento do domínio.

De acordo com Grüninger and Lee (2002) as ontologias são utilizadas para aprimorar a comunicação entre sistemas informáticos implementados, entre humanos e, também, entre humanos e sistemas informáticos implementados; para dedução computacional: para planos de representação internos e planeamentos de informação; para análise de estruturas internas, algoritmos, inputs e outputs de sistemas implementados em teorias e termos conceptuais; para reutilização (e organização) de conhecimento: para estruturação ou organização de bibliotecas ou repositórios de planos e informações de domínio. Enfatiza-se assim, a utilidade das ontologias para uma melhor interação entre humanos e sistemas informáticos, devido às limitações práticas associadas às ciências da computação, que consiste na dificuldade de permutação entre a inteligência humana e inteligência computacional.

5.2 UTILIZAÇÕES DAS ONTOLOGIAS

As ontologias têm tido um interesse crescente em áreas como representação do conhecimento, integração de informação, recuperação de informação, comércio eletrónico e Web Semântica. Staab and Studer (2009) Nesta última, as ontologias são de grande utilidade, incluindo:

- classes para representar conceitos gerais de qualquer área,
- relações identificadas entre os objetos,
- propriedades ou atributos dos objetos descritos.

Destaque-se que as ontologias são um meio efetivo de representar, com orientação para as máquinas e processamento automático, a semântica expressa em documentos Web. As ontologias ao fornecerem uma base comum que garante a coerência dos dados, facilitam a pesquisa de informação e integração de dados de diferentes comunidades. Através da Semantic Web, as ontologias podem ter diversas aplicações: Saias (2003)

- Wikis, Blogs e Portais Web: São na maioria das vezes uma página Web especialmente concebida para exibir informações de diversas fontes, de uma forma uniforme, sendo que os documentos são concebidos para leitura em que a principal preocupação prende-se nas fontes, cores e tamanhos das letras. Todavia, estes podem enriquecer os seus documentos com uma ontologia que represente a sua informação, o que abriria caminho para a utilização de processos automáticos para a consulta destas páginas Web.
- Repositórios multimédia: o uso de uma ontologia poderia ser bastante aproveitada numa galeria virtual, associando a cada imagem ou filme um conjunto de informações como o título, ano, autor e tema. Assim sendo, esta informação além de ficar expressa textualmente, teria uma mais valia, uma eficiência superior numa pesquisa automática por parte das aplicações ou agentes.
- Sistemas e aplicações: Uma ontologia pode aplicar-se a vários serviços desde traduções de textos até a serviços relacionados com áreas da educação e medicina. A computação ubíqua, por exemplo, envolve mobilidade e transparência da tecnologia para o utilizador. Os dispositivos devem automaticamente identificar-se perante outros, num ambiente de grande interoperabilidade. Esta apresenta-se como mais uma área em que se pode aplicar uma ontologia a fim de normalizar os conceitos e facilitar a troca de informação.

5.3 CONSTRUÇÃO DE ONTOLOGIAS

Esta secção é dedicada ao modo como efetivamente se pode construir uma ontologia usando uma linguagem Semantic Web.

5.3.1 *Processos Manuais*

Quando são usados processos manuais na construção da ontologia, tem de existir maioritariamente uma preparação prévia, como estudos ou reuniões com profissionais da área. Esta preparação consiste na seleção dos conceitos a incluir, como os descrever e quais as relações que se estabelecem entre eles. Aquando da construção, os dados já se encontram organizados hierarquicamente, sobre um conjunto de conceitos e suas características. Saias (2003)

O passo seguinte consiste na escolha da linguagem para a representação da ontologia que vamos criar, existe várias opções desde Linguagens Clássicas (OCML, Ontolingua) a Linguagens Markup (OIL, RDF, OWL). As linguagens de representação de ontologias estão descritas na secção 5.6. O desenvolvimento da ontologia pode efetuar-se usando apenas um editor de texto, caso se conheça bem a linguagem que se vai usar, contudo foram desenvolvidas ferramentas para facilitar este processo.

Uma das ferramentas mais utilizadas, que conta já com mais de duzentos e trinta mil utilizadores registados em projetos relacionados com Web Semântica, é o Protégé. Esta aplicação é desenvolvida na Universidade de Stanford com a colaboração da Universidade de Manchester, sendo grátis e de código aberto, apresenta uma vasta gama de funcionalidades, disponibilizadas por uma interface gráfica desenvolvida em Java. Com o Protégé é possível trabalhar com várias linguagens, incluindo OWL e RDF,mostrando-se uma ferramenta muito completa, que inclui opções mais avançadas que vão para além do necessário para um utilizador comum. O utilizador pode ainda guardar classes e instâncias numa das várias linguagens suportadas, incluindo OWL e RDF.

No exemplo que se segue é evidenciado o uso do webprotégé, sendo este uma boa solução para quem não quer ou não pode instalar a versão desktop no seu computador. Representa-se ainda a classe Instituição, tendo como subclasses Arte, Casamento, Defesa e Educação, figura 10.

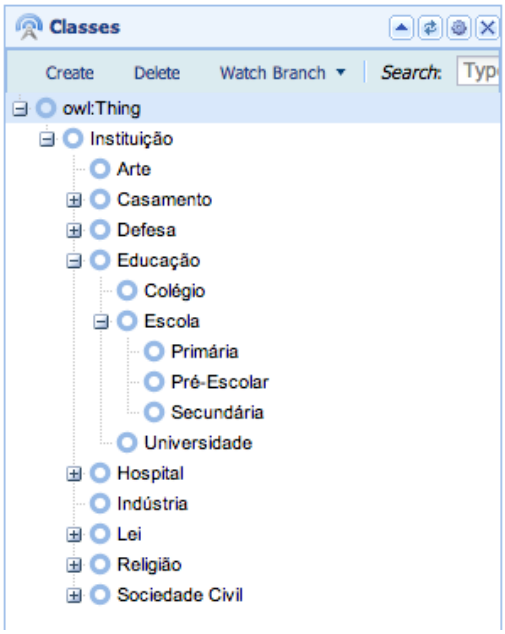


Figura 10: WebProtégé - Classe Instituição, com as subclasses Arte, Casamento, Defesa e Educação.

Para criar uma classe, basta utilizar o botão 'Create' do painel, que conduz a uma janela onde se pode definir o nome da classe, como se pode verificar pela figura 11.

Figura 11: WebProtégé - Criar uma nova classe.

Se selecionar uma das classes pode definir o nome, o IRI, propriedades e anotações (figura 12). As classes Arte, Casamento, Defesa e Educação são construídas a partir da sua superclasse Instituição, para isso basta apenas selecionar a classe Instituição e utilizar o botão 'Create' para definir a nova classe que será adicionada à superclasse Instituição.

É possível efetuar muitas outras opções. Um guia de utilização está disponível em `http://protegewiki.stanford.edu/wiki/WebProtegeUsersGuide`.

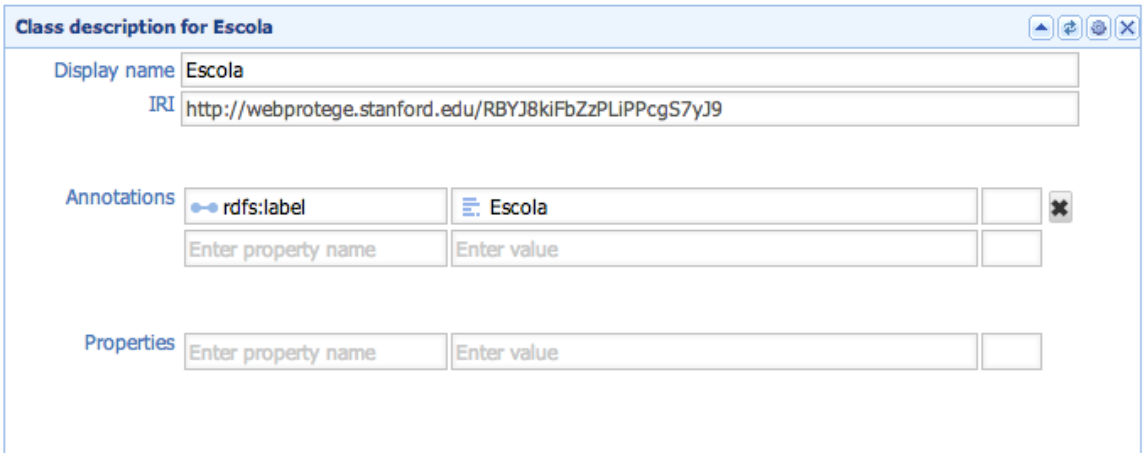


Figura 12: WebProtégé - Descrição da classe Escola.

A secção seguinte revelará mais informações a respeito de ferramentas para a construção de ontologia.

5.3.2 *Processos Semiautomáticos*

Nesta secção são analisados e descritos alguns trabalhos que tentam minimizar a intervenção humana na construção de ontologias, pois como já anteriormente foi realçado, a construção de uma ontologia de raiz é um processo demorado, complicado, em que a ajuda do humano é indispensável. São descritos alguns trabalhos em diferentes áreas, com diferentes objetivos e com grau de automatização diferente. Saias (2003)

Poronto

Poronto é uma ferramenta para construção semiautomática de ontologias a partir de textos em português na área da saúde. A diferença desta abordagem para as demais existentes é a não necessidade do utilizador realizar previamente a anotação do *corpus* linguístico, o que torna a ferramenta mais fácil e simples de se usar. Zahra et al. (2013)

Para ser de mais fácil acesso a todos os interessados, a ferramenta está disponível através de uma página Web.

O processo de criação semiautomática de ontologias está dividido em duas fases: a criação do *corpus* e a criação da ontologia.

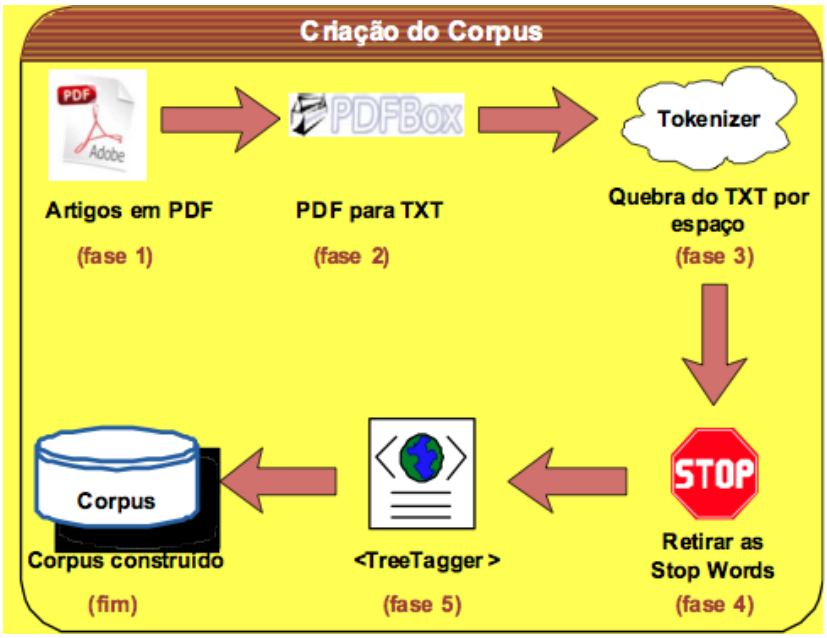


Figura 13: Processo da criação do corpus da ferramenta. Zahra et al. (2013)

A criação do *corpus* está dividida em 5 fases, de acordo com a Figura 13:

1ª fase: o utilizador envia os artigos que deseja processar em formato PDF;

2ª fase: os artigos são transformados em documentos de texto filtrados (sem os marcadores padrões da extensão PDF), é utilizado o PDFBox para este processo;

3ª fase: é efetuado um pré-processamento dos textos, onde o texto é dividido por espaços para posteriormente ser feito o processamento das anotações linguísticas com o TreeTagger[1];

4ª fase: as Stop Words são removidas;

5ª fase: os textos são processados com o TreeTagger[1].

Depois de concluída a fase de construção do *corpus*, pode-se dar início à criação da ontologia. Esta é dividida em oito fases, de acordo com a Figura 14:

1 mais informações sobre TreeTagger podem ser encontradas em Schmid (1994).

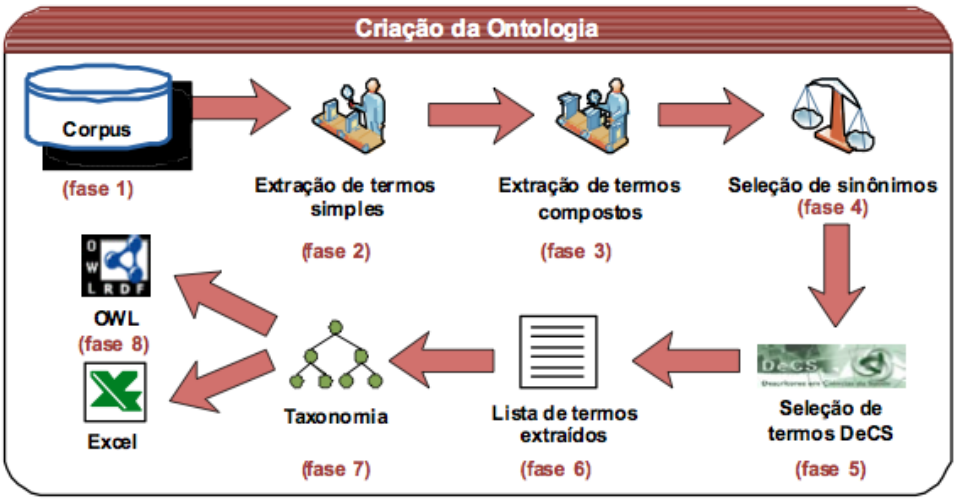


Figura 14: Processo da criação da ontologia. Zahra et al. (2013)

1ª fase: o utilizador preenche os seguintes filtros para processar o *corpus*:

- quantidade mínima de vezes em que um termo simples aparece no *corpus*;
- número mínimo de termos compostos;
- número mínimo de vezes em que um termo composto aparece no *corpus*;
- seleção dos termos compostos, inclusão ou não no resultado dos termos compostos;
- inclusão no resultado apenas de termos marcados pelo TreeTagger[1] como substantivos ou se todos os termos;
- utilização ou não da medida tf-idf[2] como medida de seleção;
- utilização ou não da medida de entropia, como medida de seleção.

De acordo com Wiener (1965) "a soma de informação num sistema é a medida do seu grau de organização; a entropia é a medida do seu grau de desorganização; um é o oposto do outro".

2ª fase: os termos simples são extraídos e é aplicado: tf-idf[2] e entropia.

3ª fase: os termos compostos são extraídos com base em regras expressas por sequências de tipos morfológicos e é aplicado: tfi-df[2] e entropia.

1 mais informações sobre TreeTagger podem ser encontradas em Schmid (1994).
2 mais informações sobre tf-idf podem ser encontradas em Manning and Schütze (1999).

4ª fase: é efetuada uma pesquisa por sinónimos dos termos na lista do OpenThesaurusPT para assim facilitar o critério de seleção do termo pelo utilizador.

5ª fase: é feita uma pesquisa para verificar se os termos extraídos possuem correspondência na lista de DeCS, e assim facilitar o critério de seleção do termo pelo utilizador.

6ª fase: são listados os termos simples e os compostos, extraídos pela ferramenta, assim como alguns sinónimos para estes termos e se o termo está ou não incluído na lista de descritores. A partir dessa lista deve ser selecionado pelo utilizador os termos mais relevantes a serem inseridos na ontologia.

7ª fase: o utilizador pode optar pela organização dos termos anteriormente selecionados numa taxonomia, que é construída com um método baseado em termos compostos.

8ª fase: ao selecionar a opção do Menu "Exportar para..." o utilizador pode exportar o resultado para o formato XLS ou OWL. Este projeto está disponível para download em https://code.google.com/p/poronto/.

Onto.pt

O projeto Onto.PT tem como objetivo a construção automática de uma ontologia lexical para a língua portuguesa, através da exploração do texto em recursos textuais, estruturada de forma semelhante à WordNet de Princeton. Gonçalo Oliveira and Gomes (2010)

A informação é extraída de recursos textuais em português, onde se incluem:

- Thesaurus
- Dicionários
- Enciclopédias
- Corpora

A abordagem para a construção do Onto.PT consiste num procedimento automático com três fases, que se descrevem de seguida:

- **Extração de relações**: É construído manualmente um conjunto de gramáticas com padrões textuais que indicam um conjunto pré-definido de relações semânticas. O texto é em seguida processado por um analisador sintático que utiliza as gramáticas na extração automática de instâncias de relações, que são representadas em triplos t = (a, R, b), onde 'a' e 'b' são palavras, 'R' é o nome da relação entre 'a' e 'b'. A abordagem utilizada para extração é inspirada na construção do PAPEL. Consultar mais informação em Gonçalo Oliveira and Gomes (2010).

  Exemplos de definições extraídas:

  - candeia s.f. utensílio doméstico rústico usado para iluminação, com pavio abastecido a óleo
  - espiga s.f. parte das gramíneas que contém os grãos
  - inquietar v.t. causar ansiedade
  - severo adj. grave , crítico

  Através dos padrões textuais sublinhados é possível extrair, por exemplo, os seguintes triplos:

  - utensílio HIPERONIMO DE candeia
  - iluminação FINALIDADE DE candeia
  - espiga PARTE DE gramínea
  - grão PARTE DE espiga
  - inquietar CAUSADOR DE ansiedade
  - grave SINONIMO DE severo
  - crítico SINONIMO DE severo

- **Descoberta de synsets**: Tendo em atenção apenas as relações de sinonímia, há uma tendência à formação de aglomerados (clusters) de palavras. Nesta fase, esses aglomerados são detetados automaticamente e aproximados a synsets, de acordo com um procedimento semelhante ao apresentado em Gonçalo Oliveira and Gomes (2010). O resultado é um thesaurus, onde cada conceito é representado por um conjunto de palavras sinónimas, à imagem de uma wordnet.

  Synsets são um grupo de elementos de dados que são considerados semanticamente equivalentes para efeitos de obtenção de informação.

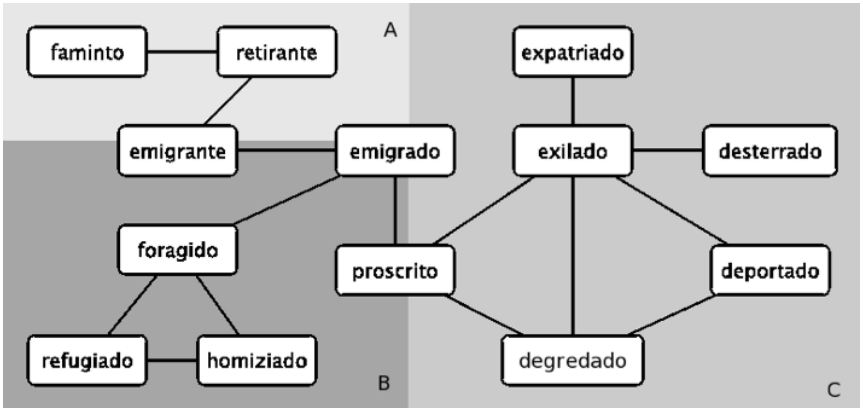


Figura 15: Rede de sinonímia em torno da palavra 'emigrado' e aglomerados identificados Gonçalo Oliveira and Gomes (2010).

A figura 15 apresenta um exemplo de uma rede de palavras formada por relações de sinonímia, extraídas de forma automática, onde existem várias ambiguidades. Cada ligação entre duas palavras indica que foi extraída uma relação de sinonímia entre ambas e fundos com diferentes tonalidades de cinzento identificam aglomerados. É possível identificar três conceitos que, apesar de próximos, são diferentes:

- (A) alguém que muda de local em busca de melhor sorte;
- (B) alguém que foge do seu país por estar a ser perseguido;
- (C) alguém que foi expulso do seu país.

Verifica-se que algumas palavras mantêm alguma ambiguidade e pertencem a dois ou três aglomerados. Podemos então concluir que tudo isto vai ao encontro da realidade das linguagens naturais, onde a mesma palavra pode efetivamente ter mais que um significado.

- **Integração das relações**: Na última fase, é feita a tentativa de associar os argumentos das restantes relações extraídas, aos synsets descobertos anteriormente, de acordo com um dos métodos apresentados em Gonçalo Oliveira and Gomes (2010). Os triplos que relacionam palavras passam a relacionar conceitos, sendo descritos por palavras. Um dos métodos utilizados na realização desta tarefa procura o par de synsets mais semelhante dentro de todos os pares possíveis de synsets (Sa, Sb), Sa contém o termo a, e Sb contém o termo b. A semelhança é calculada com base nas vizinhanças dos termos dos synsets numa rede formada por todos os triplos extraídos.

Triplos de termos	**Triplos de synsets**
aparelho **Hiperónimo-de** televisor	apresto, utensílio, petrechos, instrumento, apetrechos, aparelho **Hiperónimo-de** tv, televisão, televisor
extensão Hiperónimo-de território	superfície, dimensão, extensão; espaço, área **Hiperónimo-de** território, área
ângulo **Parte-de** triângulo	ãngulo, face, lado **Parte-de** triângulo, trilateral
técnica **Membro-de** marketing	arte, técnica **Membro-de** marketing
edição **Finalidade-de** programa	edição, lançamento **Finalidade-de** programa, aplicativo

Tabela 4: Exemplos da integração de relações no thesaurus.

Na tabela 4 mostra alguns exemplos de relações entre termos, incluídos no PAPEL, e a sua correspondência após a integração num thesaurus, o TeP Gonçalo Oliveira and Gomes (2010).

Ontolearn

Navigli and Velardi (2004) desenvolveram um método para extração de ontologias de domínio a partir de sítios da Web, a nível do turismo.

O método consiste em três fases que estão esquematizadas na figura 16, as três fases são:

Extração de terminologia: através da análise gramatical feita nos documentos, são extraídas listas com termos, termos estes marcados com as seguintes etiquetas: SN(Sintagma nominal), adjetivo-SN e preposição-SN.

Interpretação semântica: é determinado o conceito / sentido correto para cada componente de um termo complexo, identificando as relações semânticas existentes entre os componentes do conceito para assim construir um conceito complexo. Um exemplo prático de um conceito complexo que utiliza a abordagem de inclusão de cadeia de caracteres e é formado após o procedimento de desambiguação:
'serviço de transporte público' : (p. ex. serviço $\rightarrow$ serviço de transporte $\rightarrow$ serviço de transporte público).

Integração na ontologia: os conceitos gerados pelo OntoLearn são usados para melhorar e atualizar o WordNet, criando uma ontologia de domínio. O processo consiste em:

- após as árvores de conceitos de domínio serem anexadas, manual e automaticamente aos nodos certos do WordNet, todos os ramos que não contêm um nodo do domínio são removidos da hierarquia do WordNet.
- um nodo intermediário do WordNet é removido sempre que as seguintes condições são encontradas:
  1. não tem nodos irmãos;
  2. tem somente um hipónimo direto;
  3. não é a raiz da árvore de conceitos do domínio;

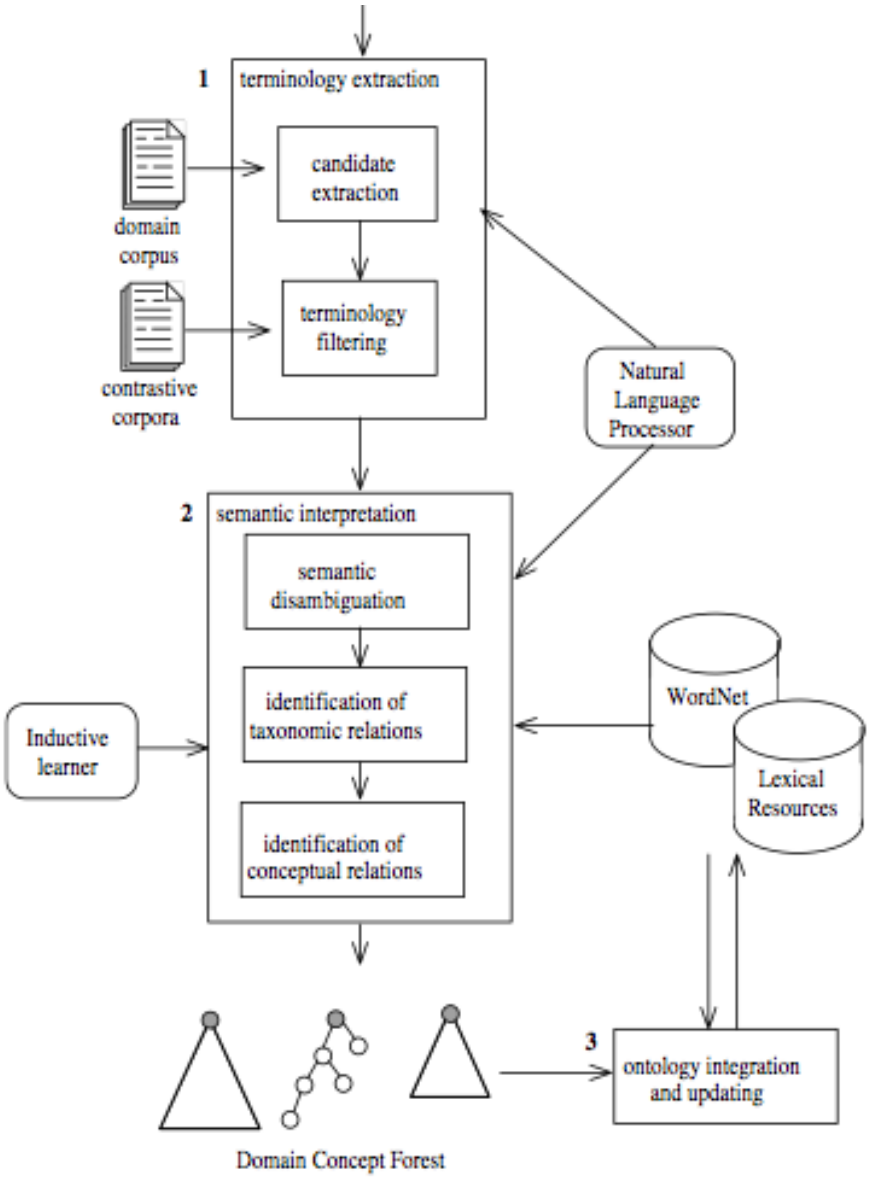


Figura 16: Arquitetura do método Ontolearn. Navigli and Velardi (2004)

4. não está a uma distância ≤ 2 de um nodo raiz da WordNet (para preservar uma ontologia de topo mínima).

Foi efetuado um teste, em que o OntoLearn extraiu 14.383 termos candidatos na 1º fase (extração de terminologia) de textos retirados de sites relacionados com o turismo. Desses, o sistema derivou 3.840 conceitos que foram avaliados por especialistas do domínio.

5.4 COMPONENTES DE UMA ONTOLOGIA

Atualmente, as ontologias partilham muitas semelhanças estruturais, independente da linguagem em que são expressas. A maioria das ontologias descrevem **indivíduos**, **classes** (conceitos), **atributos** e **relações**. Esta secção descreve cada um destes componentes.

- **Indivíduos** são os componentes mais básicos de uma ontologia e podem incluir objetos concretos como pessoas, animais, automóveis, moléculas, planetas, assim como indivíduos abstratos como números e palavras. Não é obrigatório o uso de indivíduos na criação de uma ontologia, contudo um dos desígnios de uma ontologia é apresentar um meio de classificação de indivíduos, mesmo que estes não sejam explicitamente parte da ontologia.

- **Classes** (conceitos) são grupos abstratos, conjuntos ou coleções de objetos. Estes podem conter indivíduos, outras classes, ou até ambos.

  Alguns exemplos de classes:

  - Pessoa, a classe de todas as pessoas;
  - Vinho, a classe de todos os vinhos;
  - Número, a classe de todos os números;
  - Indivíduo, representando a classe de todos os indivíduos;
  - Classe, representando a classe de todas as classes;
  - Coisa, representando a classe de todas as coisas.

  As Ontologias distinguem-se nos seguintes aspetos: se as classes podem conter outras classes, se uma classe pode pertencer a si mesma, se existe uma classe universal (uma classe que contenha tudo). Algumas vezes existem estas restrições para evitar alguns paradoxos conhecidos. Uma classe pode incluir ou estar incluída em outras classes, por exemplo, Vinho inclui Vinho Tinto, sendo qualquer membro de Vinho Tinto também é membro de Vinho. Esta inclusão é utilizada para criar uma hierarquia de classes, geralmente com uma classe geral como Coisa no topo, e classes específicas como Gazela Vinho Verde 2010 na base.

- **Atributos** são as propriedades que descrevem as classes, cada atributo tem pelo menos um nome e um valor, e é utilizado para armazenar a informação que é específica para o objeto ligado a ele. Por exemplo, o objeto 'Gazela Vinho Verde Branco' tem como atributos:

  - Nome: Gazela
  - Tipo: Tranquilo
  - Cor: Branco
  - Tonalidade: Limão
  - Ano: 2010

  No valor de um atributo são usados tipo de dados, tais como: números, strings, booleanos, entre outros.

- **Relações:** Na maioria dos casos, uma relação é um atributo cujo valor é outro objeto na ontologia. Uma das grandes vantagens das ontologias vem da capacidade de descrever as relações entre objetos. O conjunto de todas as relações descreve a semântica do domínio. Tipos de relações mais comuns:

  - Relação de inclusão (é-superclasse-de, é-um, é-subtipo-de ou é-subclasse-de), é o tipo mais importante de relação, define quais objetos são membros de quais classes de objetos. Por exemplo, "Gazela "é-um "Vinho Verde Banco", que, por sua vez, é-um "Vinho".
  - Relação de adição (é-um) cria uma taxonomia hierárquica, uma estrutura de árvore que descreve que objetos se relacionam com quais outros. Nesta estrutura, cada objeto é um "filho"de uma "classe pai".
  - Relação do tipo parte-de, representa como objetos se juntam para formar objetos compostos. Por exemplo, para incluir objetos como Rolhas na ontologia, diríamos que "Rolha é-parte-de Gazela Vinho Verde"já que uma rolha é um dos componentes do Gazela Vinho Verde.

  Além das relações mais comuns anteriormente referidas, as ontologias por norma incluem outros tipos de relações que ajustam ainda mais a semântica do modelo. Estas relações são específicas do domínio e são utilizadas para responder a alguma particularidade da ontologia.

5.5 BIBLIOTECAS DE ONTOLOGIAS

A expansão da utilização das ontologias na Internet levou ao surgimento de serviços onde estão disponíveis listas ou diretórios de ontologias com mecanismos de procura (SPARQL). Estes mecanismos foram chamados de bibliotecas de ontologias.

Bibliotecas estáticas ou geridas por pessoas:

- A DAML Ontology Library disponibiliza ontologias em DAML.
- Ontolingua ontology library é um ambiente colaborativo para procurar, criar, editar, modificar e usar ontologias.

Os serviços abaixo são diretórios e mecanismos de procura. Eles incluem crawlers que pesquisam a Web procurando por ontologias.

- O Swoogle é um diretório e mecanismo de procura para todos os recursos RDF disponíveis na Web, incluindo ontologias.
- Ontaria é um 'diretório onde se pode pesquisar e navegar por dados da Web semântica', que foca em vocabulários RDF com ontologias OWL.

5.6 LINGUAGENS DE REPRESENTAÇÃO DE ONTOLOGIAS

Nesta secção é apresentado o panorama geral sobre as linguagens que são utilizadas para representação das ontologias. É necessário referir que existem várias linguagens específicas para representar as ontologias e outras que não foram criadas para representar as ontologias mas podem ser utilizadas para o fazer. Será feita primeira uma perspetiva histórica sobre a evolução das linguagens, de seguida serão apresentadas as linguagens mais importantes, no final da secção será feita uma apresentação mais detalhada da linguagem OWL.

5.6.1 *Evolução das Linguagens de Representação de Ontologias*

As linguagens de representação de ontologias são utilizadas principalmente para desenvolver ontologias, pois permitem expressar o conhecimento relativo a domínios específicos.

No campo das Ontologias existe uma vasta variedade de linguagens que podem ser utilizadas, podendo ser divididas em dois grupos: Linguagens Clássicas/Tradicionais e Linguagens Markup. Faria (2009)

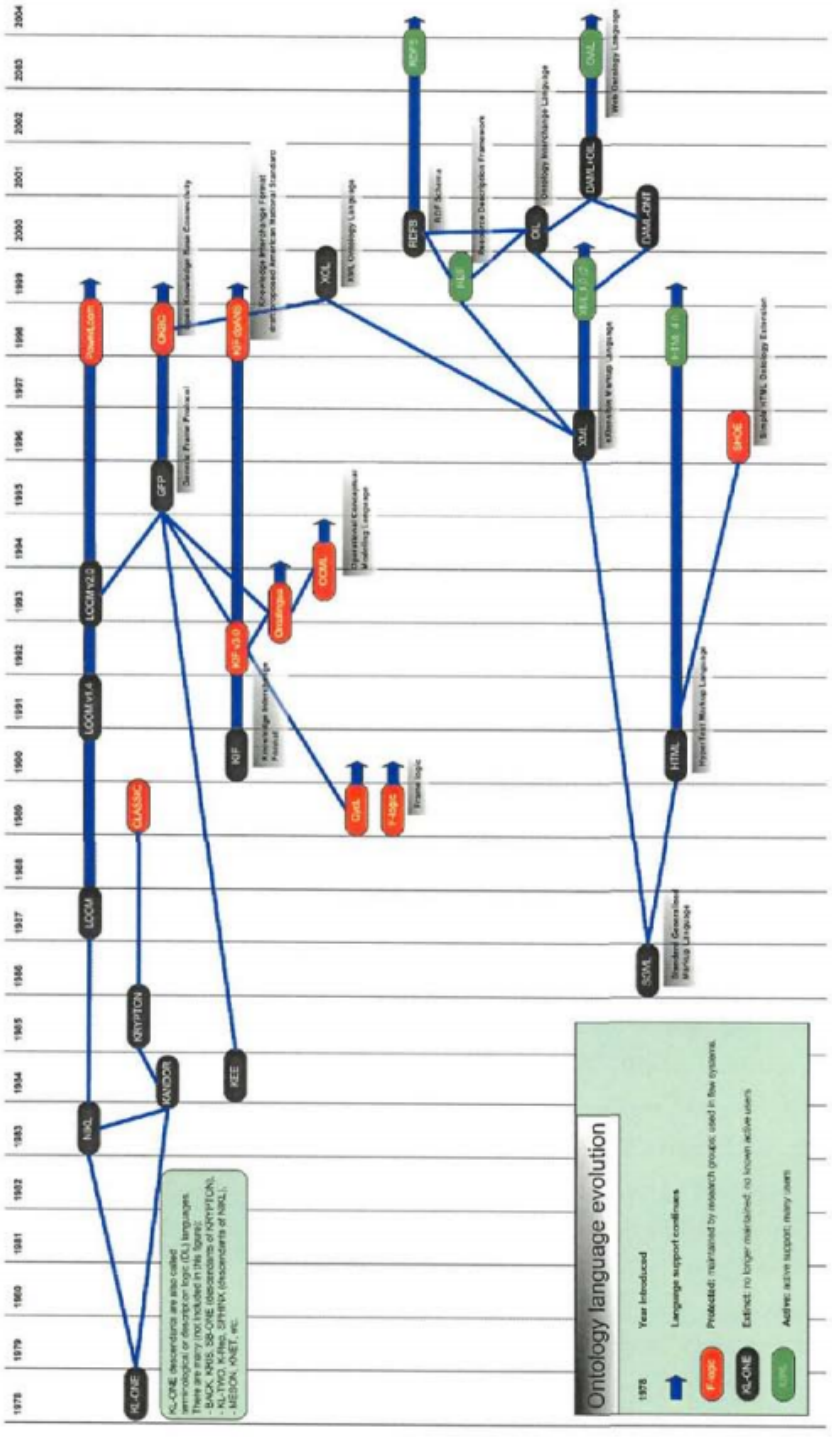


Figura 17: Evolução das Linguagens de Representação de Ontologias. Corcho (2004)

A figura 17 apresenta uma perspetiva histórica das linguagens de representação de ontologias, assim como o ano em que as linguagens mais importantes foram introduzidas. São ainda referenciadas linguagens que já não são usadas (assinaladas a preto), linguagens mantidas por grupos de investigação (assinaladas a laranja) e linguagens ativas (assinaladas a verde). Corcho (2004)

As linguagens utilizadas na especificação de ontologias podem ser divididas em dois grupos:

- Linguagens Clássicas ou tradicionais, nas quais se incluem as linguagens de Lógica Descritiva, Frames, e Lógica de Primeira Ordem. Alguns exemplos são: LOOM, CycL, Ontolingua, OCML, Flogic, KIF e OKBC.
- Linguagens Markup, que são linguagens Web standard ou baseadas nestas. Alguns exemplos das linguagens utilizadas para a representação das ontologias são: SHOE, XOL, RDF, OIL, DAML+OIL e OWL.

Linguagens Clássicas

Algumas das linguagens clássicas já não são utilizadas, algumas por vezes ainda mantidas por grupos de investigação. As linguagens mais relevantes são apresentadas, tal como se segue:

- CycL Foxvog (2010) (Cyc Language) foi criada pela MCC (Microelectronics and Computer Technology Corporation), mostrando-se como uma linguagem formal, baseada em frames e em lógica de primeira ordem. O vocabulário de CycL é constituído por termos, que podem ser divididos em constantes, termos não atómicos, variáveis e um pouco de outros objetos. Estes são combinados com expressões significativas da linguagem, possibilitando assim a elaboração de asserções na base de conhecimento Cyc2. Cyc é uma ampla base de conhecimento, baseada no senso comum.
- KIF Genesereth (1991) é uma linguagem desenvolvida para a troca de conhecimento entre sistemas, mesmo que estes sejam diferentes (criados por programadores diferentes, em alturas diferentes e com diferentes linguagens de programação). É uma linguagem de Lógica de Primeira Ordem com uma sintaxe simples e algumas extensões menores para o suporte de raciocínio sobre relações. Embora a linguagem KIF seja uma linguagem expressiva, é uma linguagem de baixo nível para representar ontologias.

- Ontolíngua Gruber (1992) é baseada em KIF Genesereth (1991), foi criada em 1992 pelo Laboratório de Sistemas do Conhecimento da Universidade de Stanford. Combina paradigmas de frames e predicados de cálculo de primeira ordem.

  A ontolíngua apresenta-se como a linguagem de construção de ontologias utilizado pelo "Ontolingua Server", disponibilizando suporte explícito para a construção de módulos ontológicos que podem ser agrupados, estendidos e refinados numa nova ontologia. Cria ainda uma separação explícita entre a apresentação e a representação de uma ontologia.

```
(define-class female-person (?person)
   "female humans"
   :iff-def (and (human ?person)

                 (= (gender ?person) female)))
```

Código 5.1: Especificação em Ontolíngua

- OCML Tanasescu et al. (2004) foi desenvolvida pelo "Knowledge Media Institute of the Open University" para fornecer as competências da modelação operacional para o projeto VITAL. A linguagem OCML, que é baseada principalmente na linguagem Ontolingua, é também uma linguagem baseada em Frames com uma sintaxe semelhante às linguagens da família Lisp.

- LOOM MacGregor (1991) tem origem num projeto de investigação levado a cabo pelo Instituto de Ciências da Informação da Universidade do Sul da Califórnia. O objetivo do projeto é desenvolver ferramentas avançadas para a representação do conhecimento e raciocínio na inteligência artificial. LOOM é uma linguagem e ambiente para o desenvolvimento de aplicações inteligente. O núcleo desta linguagem é um sistema de representação de conhecimento usado para fornecer suporte dedutivo, à parte declarativa da linguagem LOOM.

- OKBC Chaudhri et al. (1998) é definida por uma linguagem de programação independente. O protocolo, de um modo transparente, suporta ligações em rede, tal como o acesso direto a sistemas de representação de conhecimento e bases de conhecimento.

Na figura 18 mostra as relações entre as linguagens clássicas apresentadas acima. Numa explicação simples, a linguagem Ontolingua é baseada na linguagem KIF integrada com as onto-

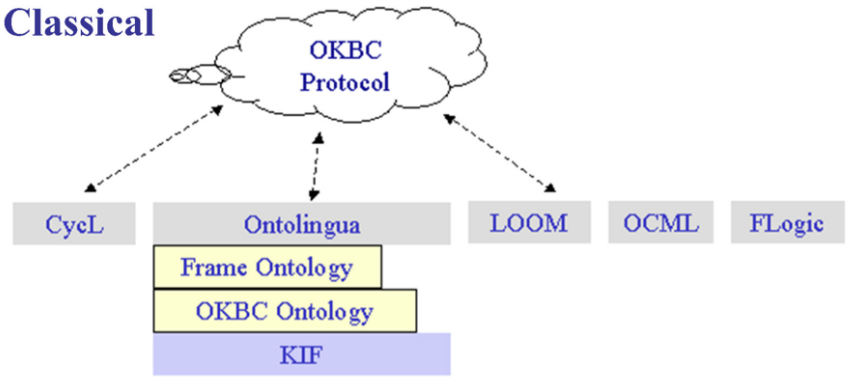


Figura 18: Linguagens de representação de ontologias clássicas. Corcho (2004)

logias Frame e OKBC. O protocolo OKBC foi desenvolvido para permitir a interoperabilidade entre as linguagens Ontolingua, CycL e LOOM.

Linguagens Markup

Existiu um esforço, a partir dos anos noventa, no que toca à tentativa de aumento de investigação acerca de como representar o conhecimento a partir da inteligência artificial, de modo a mostrar-se útil para a World Wide Web, sendo que o resultado passou pela criação de várias linguagens baseadas em HTML ou XML e em várias linguagens de representação do conhecimento baseadas em frames e em abordagens de aquisição de conhecimento.

As linguagens mais importantes baseadas na Web que podem ser utilizadas para a representação das ontologias são apresentadas na seguinte lista:

- SHOE Heflin et al. (1999) foi desenvolvida na Universidade de Maryland. A linguagem SHOE, que é uma extensão do HTML, tem como finalidade incorporar em documentos Web conhecimento semântico processável e disponibilizar tags específicas para a representação de ontologias.
- XML Harold (2001)(eXtensible Markup Language) é um subtipo da linguagem SGML (Standard Generalized Markup Language). XML é uma linguagem que permite a construção

de documentos legíveis para seres humanos e simultaneamente facilmente tratados por máquinas, mostrando-se como um conjunto de regras para a definição de marcadores semânticos, que dividem um documento em partes identificáveis. É ainda uma metalinguagem que define uma sintaxe para ser utilizada na criação de outras linguagens de marcação para um domínio específico, com estrutura e semântica próprias.

O código 5.2 mostra um exemplo de especificação em XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Agenda SYSTEM "C:\Mestrado\Biblioteca.dtd">
<bibliografia>
  <livro>
  <titulo>
         Verdes Campinas
  </titulo>
  <autor webid="http://exemplo.com/#OP5">
          Olimpio Pereira
  </autor>.
  </livro>
</bibliografia>
```

Código 5.2: Especificação em XML

Mais detalhes sobre a linguagem XML podem ser obtidos em Bosak (1997) e Bray et al. (2008).

- RDF Candan et al. (2001) foi desenvolvido pelo W3C como uma linguagem baseada em rede semântica para descrever recursos da Web. Esta linguagem oferece interoperabilidade entre aplicações, que transacionam informação interpretável pelos computadores na Web. O RDF enfatiza as facilidades necessárias para permitir um processamento automático dos recursos da Web.

- RDF(S)Lassila and Swick (1999) também foi desenvolvido pela W3C como uma extensão do RDF. Mostra-se bastante expressiva, pois permite a representação de conceitos, taxonomias de conceitos e relações binárias. Um mecanismo de inferência foi criado para ser utilizado com a linguagem, principalmente para verificar restrições.

```
    <Class ID="Female">
        <subClassOf resource="#Animal"/>
        <disjointWith resource="#Male"/>
    </Class>
```

Código 5.3: Especificação em RDF

- OIL Fensel et al. (2001) (Ontology Inference Layer) é uma proposta para representação Web e uma camada consequente para ontologias, que combina as muito usadas primitivas da modelação das linguagens baseadas em Frames. A linguagem OIL é compatível com o RDF Schema e inclui semânticas precisas para a descrição do significado dos termos.
- DAML é uma extensão das linguagens XML e RDF.
- DAML+OIL McGuinness et al. (2002) é uma versão atualizada da linguagem DAML, que fornece um importante conjunto de construtores para a criação de ontologias e de geração de informação que pode ser lida e interpretada pelos computadores.
- OWL Saha (2007) - Web Ontology Language é a linguagem recomendada pela W3C desde fevereiro de 2004. É uma linguagem que deve ser utilizada quando a informação encapsulada em documentos precisa de ser automaticamente processada por aplicações bem como humanos. Pode ser usada para representar explicitamente o significado de termos e as relações entre estes. Possui mais facilidades para expressar significados e semântica do que XML, RDF e RDF(S) e apresenta facilidades adicionais para a representação de conteúdo da Web interpretável por computador. É uma revisão da linguagem DAML + OIL. Pode encontrar-se uma descrição mais completa sobre a linguagem OWL na secção 5.6.2.

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Colecao">
       <vin:VinhoCor rdf:about="#Tinto" />
        <vin:VinhoCor rdf:about="#Branco" />
        <vin:VinhoCor rdf:about="#Rose" />
   </owl:distinctMembers>
</owl:AllDifferent>
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Colecao">
        <vin:VinhoCorpo rdf:about="#Leve" />
```

```
            <vin:VinhoCorpo rdf:about="#M{\'e}dio" />
            <vin:VinhoCorpo rdf:about="#Cheio" />
      </owl:distinctMembers>
   </owl:AllDifferent>
   <owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Colecao">
            <vin:VinhoSabor rdf:about="#Suave" />
            <vin:VinhoSabor rdf:about="#Moderado" />
            <vin:VinhoSabor rdf:about="#Forte" />
      </owl:distinctMembers>
   </owl:AllDifferent>
   <owl:AllDifferent>
   wine.xml
```

Código 5.4: Especificação em OWL

É necessário ressaltar que se deve ter em atenção a linguagem escolhida para representar a ontologia, pois, em primeiro lugar, deve ser estabelecido o que a aplicação precisa em termos de expressividade. Nem todas as linguagens permitem representar os mesmos componentes e raciocínio do mesmo modo. Deve-se ainda tomar uma boa decisão no que toca a qual linguagem específica usar para representar a ontologia, visto que a mudança de linguagem, depois de criada, teria de ser feita através de uma tradução entre linguagens, traduções essas que ainda não são suficientes para assegurar que a informação não é perdida no processo.

A linguagem XML é atualmente uma das linguagens mais utilizadas , contudo é a menos expressiva se comparada com OWL. A linguagem XML é mais popular no ambiente da Internet, seja na construção de arquivos de configuração, seja para o intercâmbio de dados entre aplicações na Web ou estruturação e armazenamento de dados.

No futuro, Bray et al. (1997) acreditam que, com a contínua expansão da Web e da Web Semântica, a XML será a linguagem universal para representação de dados. Assim, todas as aplicações serão capazes de se comunicar uma vez que elas poderão entender os vocabulários e/ou marcações de outros documentos produzidos por outras aplicações.

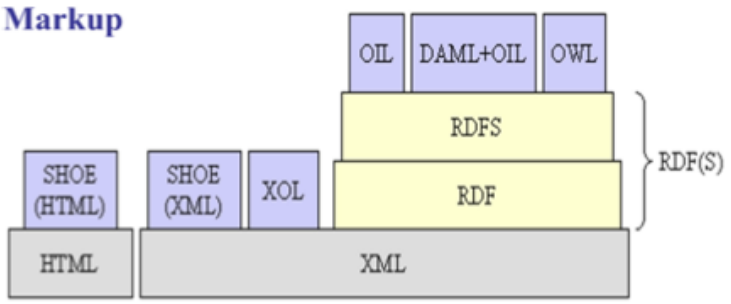


Figura 19: Linguagens de representação de ontologias baseadas na Web. Corcho (2004)

Na figura 19 é apresentado um esquema das linguagens Markup para a representação de ontologias. A linguagem SHOE, era originalmente uma extensão do HTML, mais tarde foi adaptada como extensão do XML, a linguagem XOL é também uma extensão do XML. Pode-se ainda verificar, que as linguagens OIL, DAML+OIL e OWL são evoluções das linguagens derivadas do RDF.

5.6.2 *A Linguagem OWL e suas Extensões*

O acrónimo correto para Web Ontology Language seria WOL ao invés de OWL, contudo OWL foi escolhido pois é um acrónimo facilmente pronunciável que resulta em bons logótipos, e que sugere sabedoria, fazendo honra ao projeto de representação do conhecimento One World Language de William A. Martin, na década de 1970.

De todas as linguagens de representação apresentadas anteriormente, a OWL mostra-se como a escolhida para o caso de estudo. A razão da escolha desta linguagem advém do facto desta estar em muitos aspetos à frente das outras. Foi desenhada para ser usada por aplicações que precisem de processar o conteúdo de informação, ao contrário de ser apenas usada para apresentar informações aos utilizadores. Apresenta-se com uma maior expressividade devido à adição de semântica formal e faz com que o conteúdo da Web seja de mais fácil interpretação do que o suportado pelas linguagens XML, RDF, e RDF(S). Suporta ainda técnicas de raciocínio mais poderosas. Existem muitas ferramentas disponíveis que não fazem apenas algumas tarefas gerais, processando também tarefas de raciocínio sobre as ontologias OWL.

As ontologias criadas com a linguagem OWL permitem representar explicitamente a semântica exata de classes, das suas instâncias e propriedades, dentro de um mesmo domínio.

Os documentos da linguagem OWL tornaram-se uma recomendação formal da W3C em Fevereiro de 2004.Uma candidatura a recomendação de uma segunda versão desta linguagem (OWL2) foi lançada em Junho de 2009, tornando-se recomendação formal da W3C em Outubro de 2009, pois fornece novas extensões à linguagem OWL original, com a adição de uns pequenos, mas úteis, conjuntos de funcionalidades requisitadas pelos utilizadores. Grau et al. (2008)

A linguagem OWL oferece três variantes: OWL Lite, OWL DL, OWL Full. Estas diferenciam-se em termos de poder expressivo e do processamento computacional necessário, o que possibilita aos utilizadores escolherem a que se enquadra mais às suas necessidades:

- OWL Lite - Variante mais simples da linguagem OWL. Para utilizadores que precisam sobretudo de funcionalidades como classificação hierárquica e restrições simples. Embora suporte restrições de cardinalidade, só permite valores de cardinalidade 0 ou 1. Possui ainda poder semântico suficiente para a especificação de ontologias simples, garantindo eficiência do ponto de vista computacional. OWL Lite tem a vantagem de ter menor complexidade formal quando comparada às outras variantes.

- OWL DL - Variante mais elaborada que a Lite, tendo esta como base. OWL DL é assim denominada devido à sua correspondência com a Lógica Descritiva (campo de pesquisa que estudou a lógica que forma a base formal da linguagem OWL), sendo direcionada a utilizadores que querem a máxima expressividade, mas que garanta que todas as conclusões sejam computáveis e que tenham tempo finito. OWL DL inclui todos os construtores da linguagem OWL mas com algumas restrições (por exemplo, uma classe pode ser subclasse de muitas classes, mas uma classe não pode ser instância de outra classe).

- OWL Full - mantém a expressividade da variante DL mas não impõe limitações sobre como os recursos se relacionam. Por esta razão, tem a expressividade máxima e a liberdade sintática do RDF sem garantias computacionais. As propriedades de completude e decisão não são também garantidas.

  Como já referido, cada uma destas variantes é uma extensão da sua antecessora, tanto em relação ao que pode ser expresso, como ao que pode ser concluído. O conjunto seguinte de relações é verdadeiro, já o seu inverso não o é:

- Toda ontologia OWL Lite válida é uma ontologia OWL DL válida.
- Toda ontologia OWL DL válida é uma ontologia OWL Full válida.
- Toda conclusão OWL Lite válida é uma conclusão OWL DL válida.
- Toda conclusão OWL DL válida é uma conclusão OWL Full válida.

No que toca à revisão, em 2009, da extensão OWL, que originou a OWL2, segue-se uma lista das principais funcionalidades que foram introduzidas:

- A descrição de propriedades foi expandida (ex., especificar a disjunção de propriedades);
- O conceito de datatype foi enriquecido, sendo agora possível criar novos tipos de dados de maneira explícita, bem como aplicar um maior leque de restrições;
- Capacidades de meta modelação no OWL DL (ex., é possível usar a mesma designação para uma classe e para uma instância).

Para além das três variantes da linguagem já definidas, o OWL2 define outras três variantes que impõem restrições específicas na linguagem de modo a satisfazerem determinados propósitos dos utilizadores, sendo elas:

- OWL2 EL – Variante com expressividade bastante reduzida, permitindo algoritmos de tempo polinomial para todas as tarefas de raciocínio. Devido a esta capacidade torna-se eficaz para aplicações em que são necessárias grandes ontologias. O acrónimo EL provém da inspiração desta variante na família de lógicas descritivas EL.
- OWL2 QL (Query Language) – Esta variante, impõe restrições que permitem a integração de RDBMS (Relational Database Management System), possibilitando assim que uma ontologia seja interrogada numa base de dados usando a linguagem SQL.
- OWL2 RL (Rule Language) – Apoia-se num subconjunto sintático do OWL2 que seja suscetível de ser implementado usando tecnologias baseadas em regras. O raciocínio em OWL2 RL é escalável sem comprometer em demasia o poder de expressividade da linguagem.

Especificação OWL

SINTAXE

Existem vários formatos sintáticos para especificar a linguagem OWL, mas a principal sintaxe do OWL assenta em RDF XML.

CABEÇALHO

Visto que os documentos OWL são também documentos RDF, têm como raiz o elemento rdf:RDF. Segue-se o elemento owl:Ontology, que pode conter comentários, versões de controlo e a inclusão de outras ontologias já existentes.

```
<rdf:RDF xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2001/XLMSchema#"
  xmlns:owl ="http://www.w3.org/2002/07/owl#">

  <owl:Ontology rdf:about="">
    <rdfs:comment>My owl ontology</rdfs:comment>
    <owl:priorVersion rdf:resource="http://www.ist.utl.pt/myowl_old"/>
    <owl:imports rdf:resource="http://www.ist.utl.pt/tests"/>
    <rdfs:label>Test Ontology</rdfs:label>
  </owl:Ontology>
```

Código 5.5: Especificação de um cabeçalho em OWL

CLASSES

Para definir classes é utilizado o elemento owl:Class. Este elemento mostra-se como uma subclasse de rdfs:Class e existe ainda duas classes pré definidas: a classe owl:Thing e owl:Nothing. Todas as classes são subclasses de Thing e superclasses de Nothing. No seguinte exemplo, pode-

mos constatar a definição da classe sumo como sendo uma subclasse de bebida, sendo ainda de constatar que as duas classes sumo e refrigerante são classes equivalentes (equivalentClass).

```
<owl:Class rdf:ID="sumo">
  <rdfs:subClassOf rdf:resource="#bebida"/>
  <owl:equivalentClass rdf:resource="#refrigerante"/>
</owl:Class>
```

Código 5.6: Especificação de uma classe em OWL

PROPRIEDADES

Na linguagem OWL existem dois tipos de propriedades :

- propriedades de dados
- propriedades de objetos.

As primeiras são expressas como owl:DatatypeProperty e relacionam instâncias a valores tipificados em schema XML (Ex: idade, nome, altura, correio eletrónico) e as segundas como owl:ObjectProperty e relacionam as instâncias das classes com outras instâncias (objetos), exemplos: ”pertence a”, ”é animal de estimação de”. Nestas propriedades é possível restringir o domínio e o tipo de valor da propriedade. Pode definir a propriedade inversa pelo elemento owl:inverseOf, assim como definir propriedades equivalentes pelo elemento owl:equivalentProperty.

```
  <owl:DatatypeProperty rdf:ID="age">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XLMSchema#
       nonNegativeInteger"/>
  </owl:DatatypeProperty>

  <owl:ObjectProperty rdf:ID="isPetOf ">
    <rdfs:domain rdf:resource="#animal"/>
    <rdfs:range rdf:resource="#person"/>
    <rdfs:subPropertyOf rdf:resource="#belongsTo"/>
```

```
    <owl:inverseOf rdf:resource="#hasPet"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="hasDomesticAnimal">
    <owl:equivalentProperty rdf:resource="#hasPet"/>
  </owl:ObjectProperty>
```

Código 5.7: Especificação de propriedades em OWL

RESTRIÇÕES DE PROPRIEDADES

Uma restrição de propriedade é definida pelo elemento owl:Restriction, a qual contém o elemento owl:onProperty associado à restrição pretendida.

Estas restrições podem ser do tipo:

- owl:allValuesFrom: todos os valores pertencem ao tipo especificado;
- owl:hasValue: tem um valor do tipo especificado;
- owl:someValuesFrom: tem alguns valores do tipo especificado;
- owl:maxCardinality: restrição de cardinalidade, valor máximo que pode tomar;
- owl:minCardinality: restrição de cardinalidade, valor mínimo que pode tomar.

No exemplo seguinte é especificado que os cães têm no mínimo um dono e que os cães não gostam de nenhum gato.

```
  <owl:Class rdf:about="#dog">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#isPetOf"/>
        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:
           minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#dislikes"/>
        <owl:allValuesFrom rdf:resource="#cat"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

Código 5.8: Especificação das restrições de propriedades em OWL

PROPRIEDADES ESPECIAIS

A OWL permite especificar que as propriedades podem ter características. Para isso suporta as seguintes propriedades:

- owl:TransitiveProperty
- owl:SymmetricProperty
- owl:FunctionalProperty
- owl:InverserFunctionalProperty
- owl:inverseOf.

```
<owl:ObjectProperty rdf:ID="hasSameMaster">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#dog" />
  <rdfs:range rdf:resource="#dog" />
</owl:ObjectProperty>
```

Código 5.9: Especificação de propriedades com características em OWL

COMBINAÇÕES BOOLEANAS

Na linguagem OWL é possível especificar combinações booleanas de classes como a união, intersecção e o complemento de classes.

- owl:complementOf: as instâncias de uma classe não podem ser instâncias da outra classe complementar;
- owl:unionOf: a nova classe é definida pela união das restantes classes;
- owl:intersectionOf: a classe é definida pela intersecção das classes do argumento.

```
<owl:Class rdf:about="#dog">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:complementOf rdf:resource="#cat"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cat">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#siameseCat"/>
    <owl:Class rdf:about="#persianCat"/>
  </owl:unionOf>
</owl:Class>

<owl:Class rdf:ID="domesticAnimal">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#animal"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isPetOf"/>
      <owl:hasValue rdf:resource="#Person"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Código 5.10: Especificação das combinações booleanas em OWL

ENUMERAÇÕES

Enumerações são especificadas através do elemento owl:oneOf e definem classes listando todos os seus elementos.

```
<owl:Class rdf:ID="Time">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="\#Hours"/>
    <owl:Thing rdf:about="\#Minutes"/>
    <owl:Thing rdf:about="\#Seconds"/>
  </owl:oneOf>
</owl:Class>
```

Código 5.11: Especificação de enumerações em OWL

INSTÂNCIAS

As instâncias das classes são declaradas como na linguagem RDF. Apesar de duas instâncias poderem ter identificadores diferentes, é possível que sejam o mesmo indivíduo. Podemos criar a instância Vinho Branco da classe Vinho das seguintes maneiras:

```
<rdf:Description rdf:ID="vinhoBranco">
  <rdf:type rdf:resource="#vinho"/>
</rdf:Description>

ou

<vinho rdf:ID="vinhoBranco"/>
```

Código 5.12: Especificação de instâncias em OWL

TIPOS DE DADOS

Os tipos de dados usados na linguagem OWL pertencem ao XML e incluem os tipos de dados mais usuais como valores booleanos, valores inteiros, valores decimais, strings, datas. Todavia, os tipos de dados mais complexos permitidos pelo esquema XML não podem ser utilizados pelo OWL.

5.6.3 *SPARQL*

Simple Protocol and RDF Query Language (SPARQL) é uma linguagem para consulta em RDF recomendada pela World Wide Web Consortium desde janeiro de 2008. Esta linguagem permite fazer queries que consistem em padrões triplos, conjunções e disjunções, sendo também um protocolo para mensagens entre os terminais SPARQL (endpoints) e os seus clientes. Estes terminais são aplicações Web que disponibilizam uma interface para pedidos do tipo HTTP GET ou POST, a fim dos clientes poderem aceder a um conjunto de dados RDF.

A sintaxe da linguagem SPARQL é semelhante à sintaxe SQL e disponibiliza um conjunto de comandos como SELECT, ASK, DESCRIBE e CONSTRUCT que permitem retornar dados. Para atualizar ou acrescentar dados é necessário utilizar uma extensão à linguagem denominada SPARUL, que permite assim efetuar comandos como INSERT, MODIFY ou DELETE.

A consulta seguinte visa retornar o nome de todos os países com nome em português presentes na dbpedia.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbonto: <http://dbpedia.org/ontology/>


SELECT *
  WHERE{
```

```
      ?x rdf:type dbonto:Country .
      ?x rdfs:label ?label .
      FILTER ( lang(?label) = "pt" )
    }
  LIMIT 100
```

Código 5.13: SPARQL Query

6

REPOSITÓRIO

Neste capítulo será explicada a arquitetura do sistema, assim como o Layout e a estrutura do repositório web.

6.1 ARQUITETURA DO SISTEMA

O Repositório de Ontologias foi implementado seguindo as orientações do modelo OAIS (figura 20).

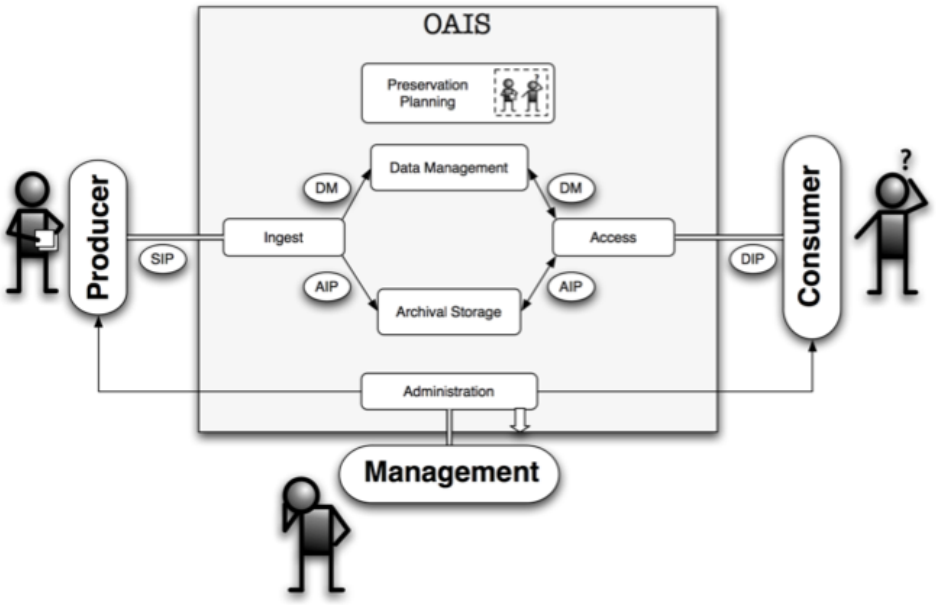


Figura 20: Modelo de referência OAIS.

O modelo OAIS é constituído por 3 mega-processos:

- **Ingestão:** Processo responsável pela inserção de novos materiais a arquivar no sistema;
- **Administração:** Processo responsável pela gestão interna do sistema;
- **Disseminação:** Processo responsável pela disseminação dos objetos arquivados.

O repositório interage com três tipos de utilizadores:

- **Produtor:** Utilizador com permissões de ingestão/ depósito de conteúdos no Repositório de Ontologias.
- **Administrador:** Utilizador com capacidade de mudar e de introduzir alterações ao funcionamento do sistema: criar utilizadores, alterar perfis de acesso, backups, logs, estatísticas;
- **Consumidor:** Tipo de utilizador destinado a consultar e pesquisar informação relativa à informação arquivada.

Existem três tipos de pacotes de informação no sistema:

- **SIP (”Submission Information Package”):** Pacote que é enviado pelo produtor ao sistema para ser processado e arquivado.
- **AIP (”Archival Information Package”):** Pacote arquivado, ou seja, um SIP depois de processado e armazenado torna-se num AIP.
- **DIP (”Dissemination Information Package”):** Pacote oferecido ao consumidor.

Nas secções seguintes iremos detalhar um pouco mais os requisitos dos pacotes de informação que circulam no sistema.

6.1.1 *SIP e o processo de Ingestão*

Começamos por definir o ponto de entrada no sistema, o SIP e o processo de ingestão que lhe está associado. Um SIP pode ser um ficheiro comprimido no formato SIARD ou um ficheiro no formato: OWL ou RDF. O processo de ingestão se receber o ficheiro em formato SIARD, valida, converte para ontologia e armazena. Caso receba uma ontologia (OWL ou RDF), valida e armazena a mesma.

6.1.2 *Validação do SIP*

A validação do SIP é constituída pelas seguintes tarefas:

- Valida se o SIP tem alguma das extensões de ficheiro permitidas (SIARD, RDF ou OWL);
- Caso seja uma ontologia (OWL ou RDF) verifica se já existe no repositório;
- Ficheiros com extensão SIARD sofrem algumas validações que são explicadas no capítulo 7, capítulo destinado à conversão SIARD para ontologia.

6.1.3 *Armazenamento do SIP*

O armazenamento do SIP é constituída pelas seguintes tarefas:

- O armazenamento do ontologia é feito através da RDF API, que utiliza uma base de dados relacional para ser guardada toda a informação das ontologias;
- Todos os ficheiros ingeridos são armazenados no file system numa área gerida pela aplicação web.

Depois do processo de ingestão a informação do SIP é armazenada e convertida num AIP.

6.1.4 *AIP e o armazenamento de ontologias*

AIP é a designação que se dá ao objeto intelectual depois deste ter ficado devidamente arquivado. Neste projeto, o AIP corresponde a uma solução híbrida, com uma parte da informação guardada numa base de dados relacional, outra no file system e outra num ficheiro XML. A componente de registo de logs no sistema será guardada num ficheiro XML.

6.1.5 *DIP e a disseminação / publicação de conteúdos*

Um DIP corresponde à forma como são disponibilizados os conteúdos armazenados. O consumidor / utilizador do sistema tem à sua disposição um website a partir do qual pode explorar os conteúdos armazenados: listar, consultar a informação relativa a um conteúdo, descarregar a informação, entre outros.

Existe ainda uma outra forma de DIP, que consiste na possibilidade de exportação, estando as seguintes disponibilizadas:

- Exportação da base de dados e ficheiros ingeridos;
- Exportação apenas da base de dados;
- Exportação completa da aplicação Web.

6.2 LAYOUT / ESTRUTURA DO REPOSITÓRIO WEB

Para o desenvolvimento do Layout do Repositório foi utilizado HTML e CSS, assim como alguns plugins de JavaScript para certas áreas do repositório desde tabelas com pesquisa, a gráficos. A

figura 21 mostra a página inicial e os menus(áreas) do repositório, caso seja um administrador (perfil com acesso a todos os menus).

Figura 21: Página Home do repositório web.

Na página Home, caso o utilizador não esteja ainda registado, tem a botão 'Registar' que disponibiliza o formulário de registo (figura 22) ficando apenas com a permissões de consumidor. Caso o utilizador já tenha conta criada pode em qualquer página do repositório fazer o login, introduzindo para isso as suas credenciais de acesso (Username e Password) no canto superior direito.

Passamos agora a explicar a estrutura do repositório web, estando este divido nas seguintes áreas / menus:

- Home;
- Ingestão;
- Disseminação;

Figura 22: Repositório Web - Formulário Novo Utilizador.

- Administração;
- Logs / Estatísticas;
- Acerca.

Para a gestão das permissões de acessos às diferentes áreas / menus do repositório foi elaborada e configurada a seguinte estrutura:

- Consumidor
  - Home;
  - Disseminação;
  - Acerca.
- Produtor
  - Home;
  - Ingestão;
  - Disseminação;

  - Acerca.

- Administrador
  - Home;
  - Ingestão;
  - Disseminação;
  - Administração;
  - Logs / Estatísticas;
  - Acerca.

Todas as tentativas de acesso a áreas restritas são bloqueadas e uma mensagem de erro é exibida (Figura 23).

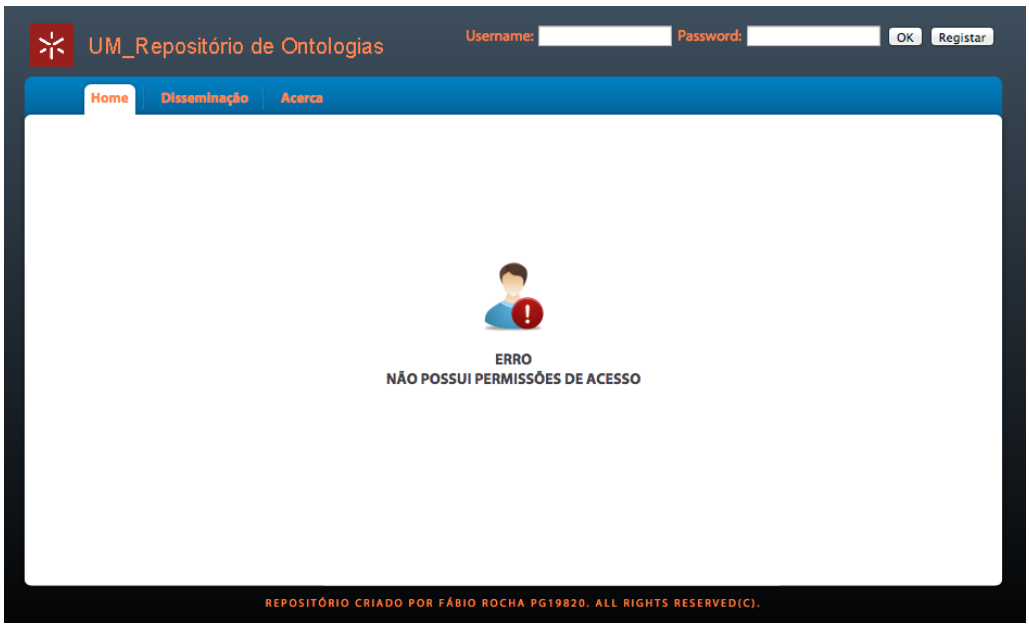


Figura 23: Repositório Web - Mensagem de acesso restrito.

Nas secções seguintes será explicada mais detalhadamente cada área do repositório web.

6.2.1 *Ingestão*

Este é o ponto de entrada do sistema, encontrando-se aqui o processo de ingestão de ontologias (SIP). Este processo só esta disponível caso seja Produtor ou Administrador sendo necessário preencher um formulário online (figura 24).

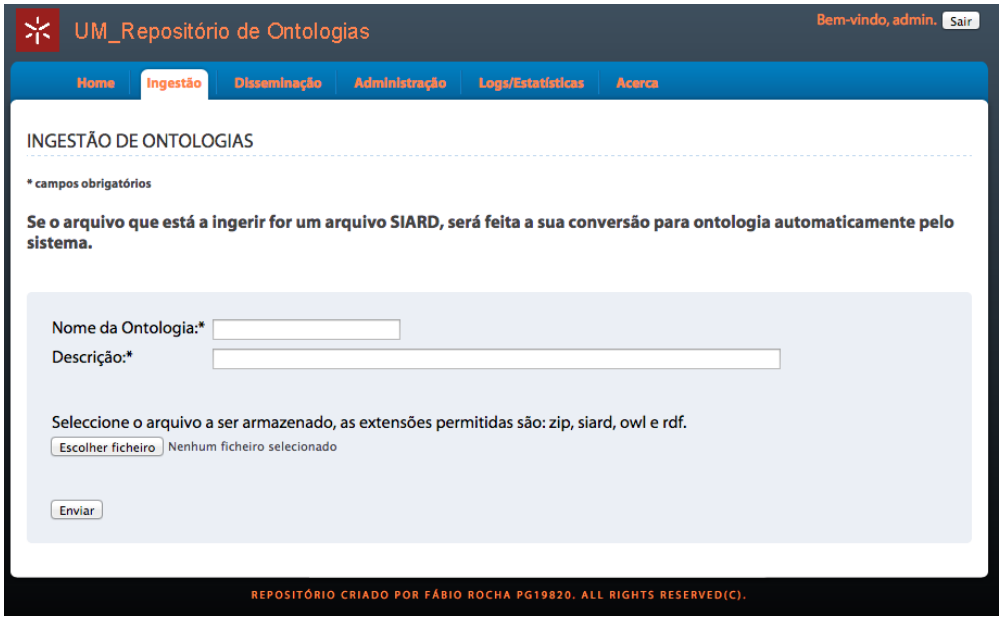


Figura 24: Repositório Web - Página destinada à ingestão de ontologias.

Na submissão do formulário, em caso de algum dos campo obrigatórios não se encontrar devidamente preenchido, uma mensagem de aviso é mostrada (figura 25).

É feita a verificação se o arquivo de entrada é do tipo SIARD e se assim for, é feita a conversão para ontologia, sendo este processo descrito no capítulo 7. Caso se verifique que é um arquivo RDF ou OWL, é verificado se a ontologia já existe no repositório e se já existir, o utilizador é alertado.

Todas as ontologias ingeridas são guardadas no servidor na pasta 'upload'. Esta pasta está estruturada de forma a organizar as ontologias do repositório, em que cada ontologia ingerida é guardada na pasta referente ao utilizador que a inseriu.

Figura 25: Repositório Web - Campo em falta no formulário.

6.2.2 *Disseminação*

Esta é uma área de acesso livre, mesmo sendo um utilizador não registado, onde é apresentada a lista de todos as ontologias do repositório (figura 26).

Nesta listagem o utilizador tem disponível algumas das informações sobre as ontologias, desde o seu nome, nome do ficheiro ingerido, utilizador que inseriu ou até a data de ingestão, sendo possível ainda no caso de ser administrador ou o produtor da ontologia editar ou remover a

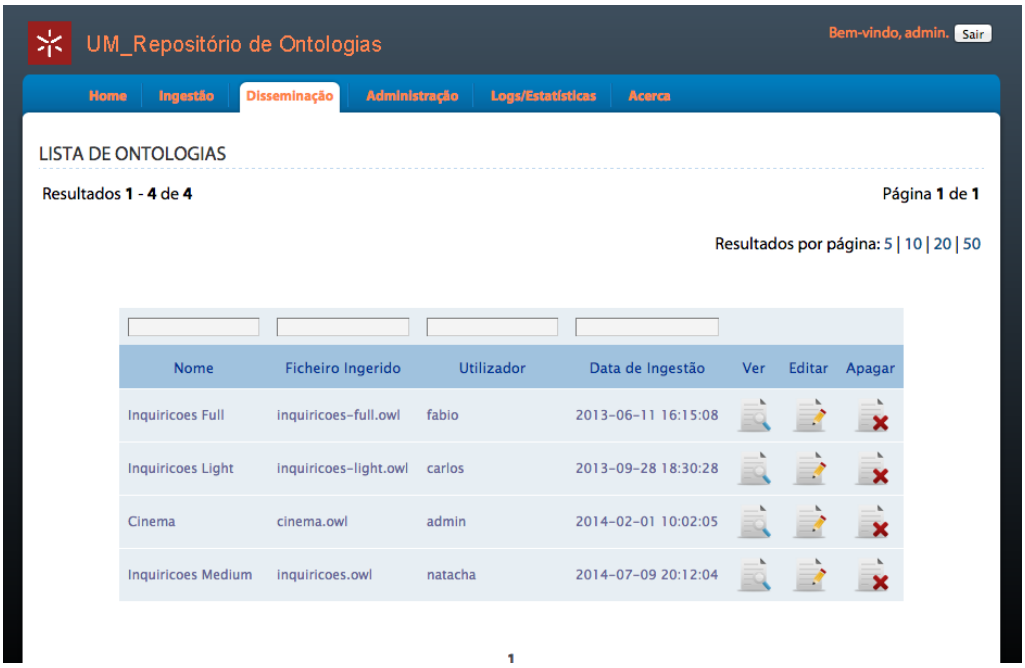


Figura 26: Repositório Web - Lista de Ontologias.

mesma, podendo o utilizador pesquisar na listagem pelos campos anteriormente referidos as ontologias desejadas.

Ao selecionar uma das ontologias é disponibilizado ao utilizador toda a informação sobre a mesma (figura 27) sendo ainda possível ver a lista de querys SPARQL associadas à ontologia e criar uma nova query SPARQL, assim como fazer download da ontologia.

Criar Nova Query SPARQL

Através desta página pode questionar a ontologia com querys SPARQL. Na primeira etapa (figura 28) é apresentada a informação geral sobre a ontologia e um campo para a query SPARQL, ao avançar (figura 29) é mostrada a resposta à query anterior e é questionado se pretende guardar esta query no sistema, tendo para isso que preencher o campo descrição, campo que ajudará a distinguir no futuro as querys e o seu propósito.

Figura 27: Repositório Web - Visualização de uma Ontologia.

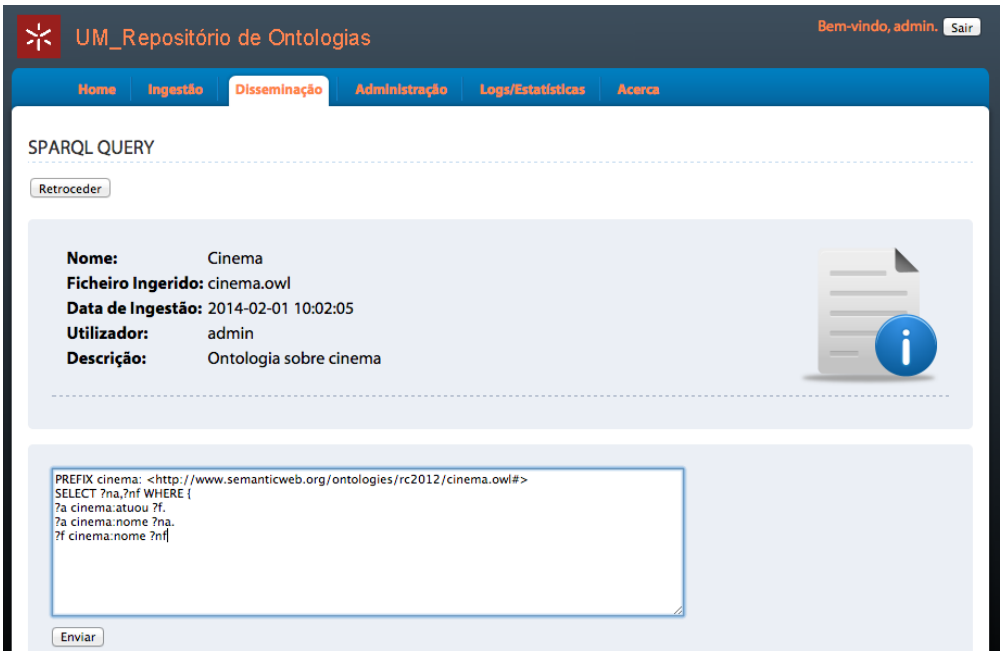


Figura 28: Repositório Web - Formulário Query SPARQL.

Figura 29: Repositório Web - Resultado de uma Query SPARQL.

6.2.3 *Administração*

A partir deste menu do repositório (figura 30), encontra-se a área destinada à administração do repositório, podendo o administrador fazer a gestão de utilizadores, o registo de um novo utilizador, assim como backups do sistema.

Gestão dos Utilizadores

O administrador pode ainda ver todos os utilizadores registados no repositório através de uma listagem, como mostrado na figura 31, podendo ainda editar ou remover os utilizadores existentes.

Figura 30: Repositório Web - Menu Administração.

Figura 31: Repositório Web - Listagem dos Utilizadores.

Os novos utilizadores podem ser adicionados através do submenu 'Inserir Utilizador'. Na figura 32 é apresentado o formulário para a criação do novo utilizador. A única diferença entre este formulário e o formulário disponibilizado no registo por parte um novo utilizador através da botão 'Registar' na Home do repositório (figura 22), é que o administrador pode atribuir qualquer um dos três tipo de utilizador (administrador, produtor ou consumidor), sendo que quando é o próprio utilizador a se registar só tem disponível a permissão de consumidor.

Figura 32: Repositório Web - Formulário Novo Utilizador, quando criado por um administrador.

Quando é criado ou editado um utilizador, algumas verificações são feitas, tais como: se o nome do utilizador já existe, ou se os campos obrigatórios estão preenchidos corretamente. Se a verificação falhar, uma mensagem de erro é exibida (Figura 33). A mensagem da figura 34 é mostrada se o utilizador for inserido ou editado com sucesso.

Backups do sistema

Num repositório desta natureza é de extrema importância os dados que contem, devendo estes estarem seguros através de backups, para deste modo os dados não serem perdidos por alguma

Figura 33: Repositório Web - Formulário Novo Utilizador em caso de erro.

Figura 34: Repositório Web - Mensagem de sucesso na edição ou criação de utilizadores.

situação anómala que possa ocorrer, para isso o administrador do repositório web tem disponível três tipos de backup que pode efetuar:

- Backup do Repositório (SQL e Ficheiros Upload);
- Backup da Base de Dados (SQL);
- Backup total do Repositório (site completo e bd sql).

Através do submenu 'Backup do Repositório' o administrador tem disponível todos os ficheiros ingeridos pelo sistema, assim como a estrutura e a informação da base de dados, isto tudo através sempre de um ZIP, podendo ainda obter apenas o backup da base de dados no submenu 'Backup da Base de Dados', sendo ainda possível um backup completo do repositório web com a estrutura total e a sua base de dados no submenu 'Backup total do Repositório'.

6.2.4 *Logs / Estatísticas*

Neste menu do repositório (figura 35), o administrador tem duas ferramentas para o ajudar a entender o estado e o que esta a acontecer no repositório web, logs com informações que possam ajudar na gestão do sistema e ainda varias estatísticas sobre a utilização do mesmo.

Figura 35: Repositório Web - Menu Logs / Estatísticas.

Logs

Os logs do sistema podem ser acedidos pelo administrador a partir de uma listagem (figura 36), onde pode fazer pesquisas usando todos os campos disponíveis. Nesta listagem é disponibilizado o username, o nome desse mesmo utilizador, a data e hora, qual o tipo de ação efetuada e uma

descrição da mesma. No caso do acesso ao repositório web não ser de um utilizador registado, nos campos username e nome do utilizador é inserido o ip do utilizador que acedeu.

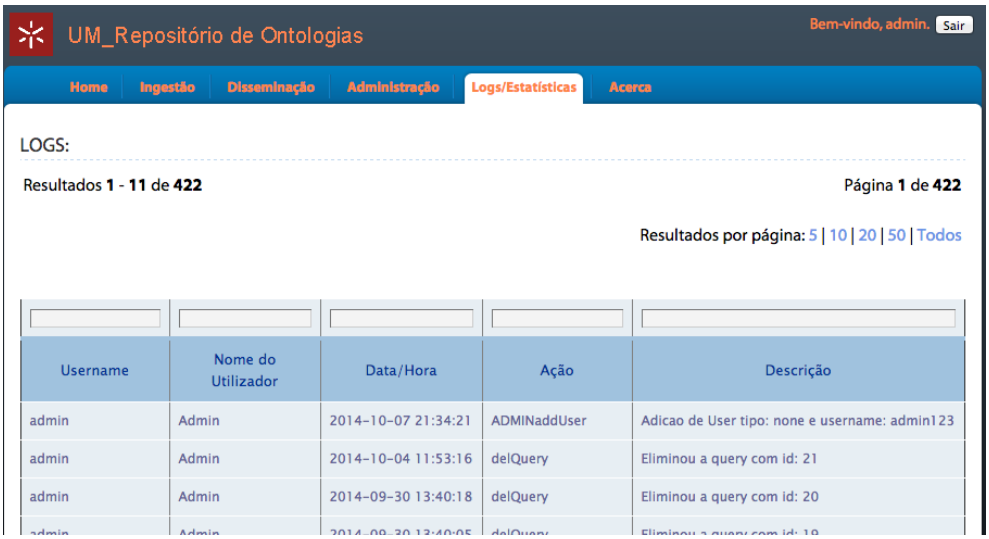


Figura 36: Repositório Web - Logs.

Estatísticas

Foi ainda criada uma área dedicada às estatísticas para que o administrador possa saber mais sobre o uso do repositório web. O administrador tem disponível os seguintes dados:

- Total de Consultas no Repositório;
- Total de Downloads do Repositório;
- Total de Acessos/Login ao Repositório;
- Total de Uploads no Repositório;
- Top 10 Consultas;
- Top 10 Downloads.

Na figura 37 pode ser vista a página de estatísticas. Estas estatísticas são calculadas através dos logs que estão no ficheiro XML, usando filtros XPath, ou a partir da base de dados. Ainda na página de estatísticas, o administrador tem disponível três gráficos, sendo que em dois deles se encontram o 'Top 10 Consultas' e o 'Top 10 Downloads' (figura 38 e figura 39), em que para a geração destes gráficos foi utilizada a biblioteca 'phpGD Bar Graph V1.1', como o nome assim o sugere, é uma biblioteca php.

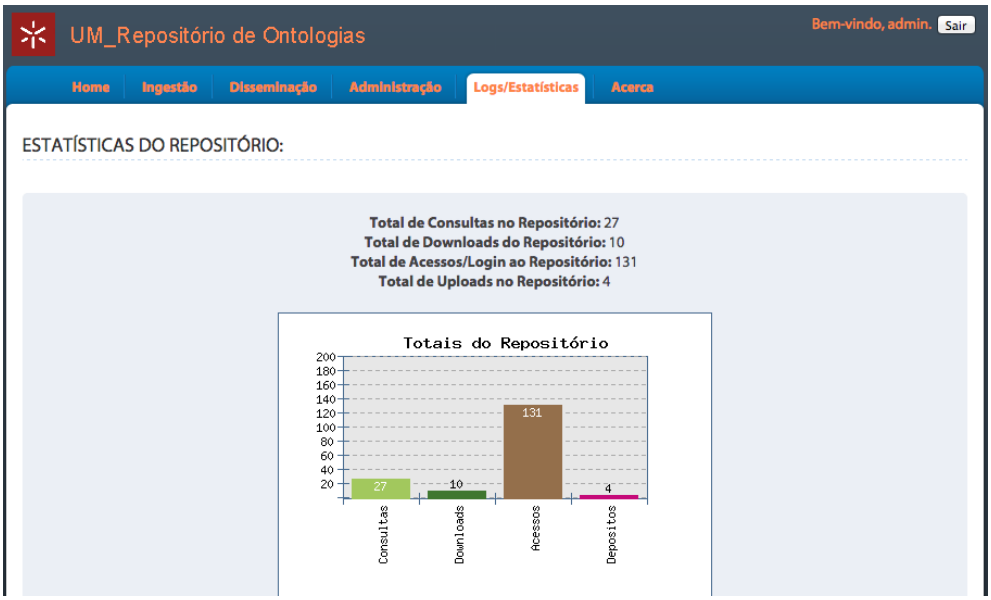


Figura 37: Repositório Web - Estatísticas.

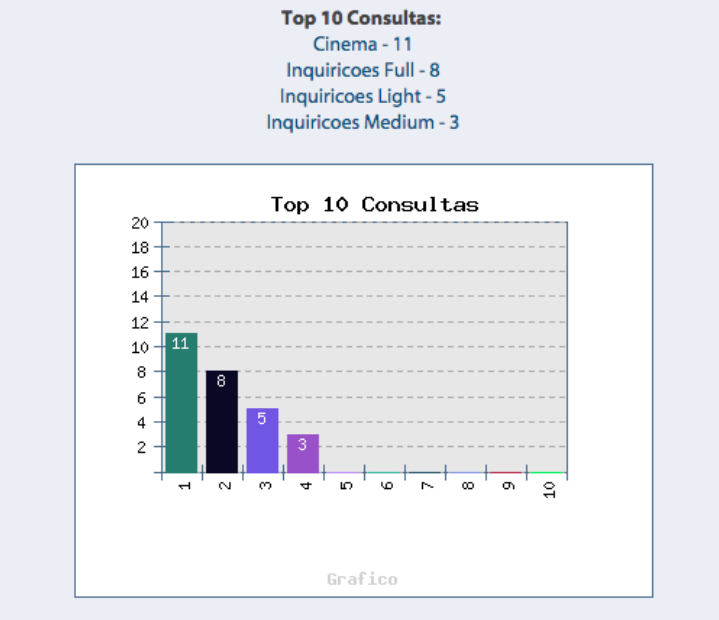


Figura 38: Repositório Web - Gráficos Top 10 Consultas.

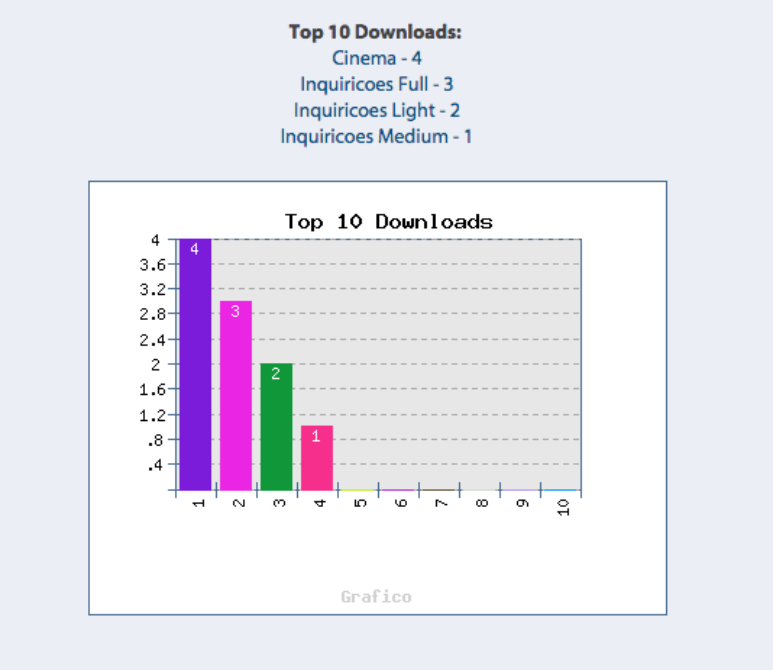


Figura 39: Repositório Web - Gráficos Top 10 Downloads.

7

CONVERSÃO SIARD PARA ONTOLOGIA

Neste capítulo será explicado em detalhe a conversão SIARD para Ontologia. A conversão é efetuada pelo repositório sem intervenção do utilizador sendo uma conversão automática sem que o utilizador possa fazer ajustes (remover tabelas..) manualmente no processo de conversão, porque através do SIARD SUITE o utilizador pode fazer os ajustes necessários de forma a dar mais qualidade à ontologia que será gerada, por exemplo se mudar o nome de uma das tabelas vai automaticamente mudar o nome das classe criada por este processo.

Propriedades SIARD que serão mapeadas para ontologia (OWL):

- **Tabela:** nomes;
- **Colunas:** nomes;
- **Chaves:** chave primaria, chave secundaria;
- **Tuplos:** dados.

Na tabela 5 , é resumido de forma geral o mapeamento entre SIARD e ontologia. Chegamos a este mapeamento através de vários testes com conversões manuais, como ferramenta para a criação das ontologias usamos o Protege, para os casos de estudo foram definidas classes, object properties, data properties e individuals. No desenvolvimento (na linguagem PHP), foi criada uma função para a validação e importação dos dados que estão no ficheiro SIARD para os arrays. Foi ainda criada a função que recebe os arrays com as informações e cria a ontologia a partir destes dados. Estes processos serão explicados em detalhe ao longo deste capítulo.

SIARD	OWL
Tables if(Count Columns = pKeys + fKeys)	Classes Object Property
Primary Keys	Object Properties
Foreign Keys	Individuals Identification
Other Columns	Data Properties
Tuples	Individuals

Tabela 5: Mapeamento SIARD para Ontologia.

Os algoritmos usados para o mapeamento foram uma das fases mais complexas no desenvolvimento do protótipo. Para uma mais fácil compreensão, as partes principais do processo são divididas em pequenos pedaços de código acompanhados por uma explicação mais detalhada, assim como fragmentos do código OWL produzidos também são apresentados para uma melhor compreensão do processo de conversão.

7.1 EXPLICAÇÃO PASSO A PASSO DO PROCESSO DE CONVERSÃO

No primeiro passo do processo é recebido o ficheiro no formato SIARD e é feita a extração dos elementos SIARD. Os elementos extraídos do formato SIARD são guardados em arrays multi dimensionais, podendo a sua estrutura ser consultada no código 7.1. Estes arrays são o ponto de partida para o mapeamento.

- **Array tables:** tem apenas uma dimensão, contem os nomes das tabelas;
- **Array pKeys:** duas dimensões, contem as chaves primarias de cada tabela;
- **Array fKey:** três dimensões, contem as chaves secundarias de cada tabela, referindo a tabela e a coluna;
- **Array columns:** três dimensões, contém o nome das colunas, o tipo e se o campo pode ser nulo para cada tabela;

- **Array tablesData:** três dimensões, contém os tuplos, tendo na estrutura a tabela e coluna a que se refere esse dado.

```
tables = Array{ [1] => table_name_1, ... , [n]=> table_name_n }

pKeys = Array{
  [tb1] => Array{ [pk1] => 'pk_tb1_1', ..., [pkn] => 'pk_tb1_n' },
  ...,
  [tbn] => Array{...}
}

fKeys = Array{
  [tb1] => Array{
    [fk1] => Array{ [table_ref_name] => 'table_ref', [
       table_ref_pkcolumn] => 'table_ref_column'},
    ...,
    [fkm] => Array{...}},
  ...,
  [tbn] => Array{...}
}

columns = Array{
  [tb1] => Array{
    [c1] => Array{ [Name] => 'c1_name', [Type] => 'c1_type', [Null] =>
        'c1_nullable' },
    ...,
    [an] => Array{ ... }},
  ...,
  [tbn] => Array{...}
}

tablesData= Array{
  [tb1] => Array{
    [c1] => Array{ [d1]=> 'd1_data', ..., [dn] => 'dn_data'},
    ...,
    [cm] => Array{...}},
  ...,
  [tbn] => Array{...}
}
```

Código 7.1: Estrutura Arrays Multidimensionais

De seguida, passamos a mapear as tabelas, para cada tabela na base de dados, definimos uma classe na ontologia (tabela 5). Com exceção das tabelas onde todos as colunas constituem uma chave primária composta (combinação de chaves estrangeiras), as tabelas de ligação utilizadas no modelo relacional para os relacionamentos de muitos-para-muitos, não são mapeadas para classes OWL, mas para object properties. Estas object properties têm o domain e range nas classes (tabelas da base de dados) envolvidos na relação. Para cada coluna (chave estrangeira) na tabela uma object property é definida.

Em resumo as tabelas de ligação são mapeados em duas object properties. Estas duas object properties têm nomes diferentes ('has_...' e 'is_..._of') e são inversas uma da outra, pois o domain de uma object property é o range da outra object property e vice-versa. No bloco de código 7.3 mostra as object properties OWL gerados para uma tabela de ligação e no bloco de código 7.4 é disponibilizado um exemplo de uma classe OWL gerada pelo processo.

Bloco de código 7.2 dá uma visão geral do algoritmo que executa estas tarefas, referentes ao mapeamento das tabelas para OWL.

```
FOREACH [ tables[] as table ]
  IF [ ( |columns[table]| = |pKeys[table]| + |fKeys[table])| ) ] THEN
    FOREACH [ columns[table] ]
      NEW 'ObjectProperty'
        Property_Description = 'has_' + fKeys[table][columns[table]][
            pk_table]
        Domain = fKeys[table][next(columns[table])][pk_table]
        Range = fKeys[table][columns[table]][pk_table]
      NEW 'ObjectProperty'
        Property_Description = 'is_' + fKeys[table][columns[table]][
            pk_table] + '_Of'
        Domain = fKeys[table][columns[table]][pk_table]
        Range = fKeys[table][next(columns[table])][pk_table]
      NEW 'InverseObjectProperties'
```

```
            Property_Description = 'is_' + fKeys[table][columns[table]][
               pk_table] + '_Of'
            Property_Description = 'has_' + fKeys[table][columns[table]][
               pk_table]
        END FOR
        non_class[] = table
      ELSE
        NEW 'Classe' table
          'SubClassOf' owl:Thing
        classes[] = table
      END IF
    END FOR
```

Código 7.2: Algoritmo - Classes e tabelas de ligação

```
<owl:ObjectProperty rdf:ID="has_ProcessoFreguesia">
  <rdfs:domain rdf:resource="#processos" />
  <rdfs:range  rdf:resource="#freguesias" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="is_ProcessoFreguesia_Of">
  <rdfs:domain rdf:resource="#freguesias" />
  <rdfs:range  rdf:resource="#processos" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="has_ProcessoFreguesia">
  <owl:inverseOf rdf:resource="#is_ProcessoFreguesia_Of"/>
</owl:ObjectProperty>
```

Código 7.3: OWL - Object Properties para as tabelas de ligação

```
<owl:Class rdf:about="#processos">
  <rdfs:subClassOf>
    <owl:Thing/>
  </rdfs:subClassOf>
</owl:Class>
```

Código 7.4: OWL - Classe

No segundo algoritmo (bloco de código 7.5) é gerado o código OWL para as chaves estrangeiras e colunas das tabelas, com a exceção das tabelas de ligação. As chaves estrangeiras são mapeadas em object properties e as restantes colunas em data properties, inverse object properties são usadas para as relações entre tabelas sendo ainda definida functional object properties para assim manter a consistência do modelo da base de dados relacional na ontologia gerada.

```
FOREACH [ classes[] as table ]
  FOREACH [ fKeys[table] as fkey ]
    NEW 'ObjectProperty'
      Property_Description = 'is_' + fkey['table_ref_name'] + '_Of'
      Domain = fkey['table_ref_name']
      Range = table
    NEW 'ObjectProperty'
      Property_Description = 'has_' + fkey['table_ref_name']
      Domain = table
      Range = fkey['table_ref_name']
    NEW 'InverseObjectProperties'
      Property_Description = 'is_' + fkey['table_ref_name'] + '_Of'
      Property_Description = 'has_' + fkey['table_ref_name']
    NEW 'FunctionalObjectProperty'
      Property_Description = 'is_' + fkey['table_ref_name'] + '_Of'
  END FOR
  FOREACH [ columns[table] as column ]
    IF [ fKeys[table][column['Name']]['table_ref_column'] != column['
       Name'] ] THEN
      NEW 'DataProperty'
        Property_Description = 'has_' + column['Name']
        Domain = class
        Range = data_type
    END IF
  END FOR

END FOR
```

Código 7.5: Algoritmo - Chaves estrangeiras e colunas das tabelas

O bloco de código 7.6 mostra o algoritmo responsável pelo mapeamento dos tuplos das bases de dados. Os tuplos de cada tabela da base de dados são mapeados para Individuals da classe da ontologia a que pertencem. Para cada Individual é associada uma Class Assertion para guardar

a sua chave primária. Data Properties Assertion são utilizados para associar as colunas aos Individuals, com a exceção das colunas que sejam uma chave estrangeira, sendo utilizadas para essas Object Properties Assertion, isto porque esta chave estrangeira foi mapeada num object property.

```
FOREACH [ classes as table]

  FOREACH [ pKeys[table] as pk)
    pk_and_table = pk + '_' + table
  END FOR

  FOREACH [ tablesData[table] as tuple ]

    NEW 'ClassAssertion'
      Description = table
      NamedIndividual = pk_and_table

    FOREACH [ tuple as tuple_column=>tuple_data ]
      IF [ array_key_exists(tuple_column, fKeys[table]) ]
        NEW 'ObjectPropertyAssertion'
          ObjectProperty = fKeys[table][tuple_column]['table_ref_name']
          NamedIndividual = pk_and_table
          NamedIndividual = fKeys[table][tuple_column]['table_ref_name']
              + '_' + tuple_data
      ELSE
        NEW 'DataPropertyAssertion'
          DataProperty = table + '_has_' + tuple_column
          NamedIndividual = pk_and_table
          Literal = tuple_data
      END IF
    END FOR
  END FOR
END FOR
```

Código 7.6: Algoritmo - Tuplos

Para as tabelas que não são mapeados em classes OWL mas sim em object properties por serem tabelas de ligação, existe a necessidade de utilzar object properties assertions (bloco de

código 7.7) para os seus tuplos, para assim manter a consistência do modelo da base de dados relacional na ontologia gerada.

```
 FOREACH [ non_class as table ]
  FOREACH [ columns[table] ]
    FOREACH [ tables_data[table] as tuple ]
      NEW 'ObjectPropertyAssertion'
        ObjectProperty = fKeys[table][columns[table]]['table_ref_name']
        NamedIndividual = fKeys[table][next(columns[table])]['
           table_ref_name'] + '_' + tuple[fKeys[table][next(columns[
           table])]['table_ref_column']]
        NamedIndividual = fKeys[table][columns[table]]['table_ref_name']
            + '_' + tuple[fKeys[table][columns[table]]['table_ref_column
           ']]
    END FOR
  END FOR
END FOR
```

Código 7.7: Algoritmo - Tuplos das tabelas de Ligação

8

CONCLUSÃO

Nothing is impossible, the word itself says 'I'm possible'!

Audrey Hepburn

A preservação da informação é fundamental para as empresas e organizações, sendo a preservação digital a maior preocupação nos tempos que correm, pois hoje em dia a maioria da informação é produzida e guardada digitalmente. Devido à grande evolução a nível de hardware e software existe o problema da preservação digital, pois ficamos dependentes da tecnologia para conseguirmos aceder à informação guardada (ao objeto digital).

Esta temática tem sido alvo de vários estudos, projetos e pesquisas, dando origem por exemplo, ao modelo de referência Open Archival Information System (OAIS) e até mesmo ao formato SIARD estudado nesta dissertação.

Diferentes tipos de objetos digitais requerem diferentes estratégias de preservação, estando esta dissertação centrada na preservação digital mas apenas num tipo de objeto digital, as base de dados relacionais. Ao escolher as bases de dados relacionais como objeto digital a preservar, tivemos que ter em conta o aspecto da preservação, os dados da base de dados, a forma como estão relacionados, a sua estrutura e ainda a semântica da base de dados.

Esta dissertação teve dois grandes focos, a conversão do formato SIARD em ontologias e um repositório que além desta conversão, desse aos consumidores a possibilidade de guardar as suas ontologias (OWL) e questionar as mesmas através de SPARQL.

O repositório foi construido usando o modelo OAIS. A conversão disponibilizada pelo repositório tem uma limitação a nível de escalabilidade visto que as dimensões das base de dados (em formato SIARD) que serão convertidas, podem ser um problema, pois para cada conversão é apenas gerado um único documento OWL. Isto significa que para uma grande base de dados, o sistema precisa de gerar um arquivo OWL enorme, algo que implica enormes capacidades de processamento e, portanto, o consumo de uma grande quantidade de recursos de hardware, um aspeto a melhorar como trabalho futuro.

Para ajudarmos na preservação das bases de dados que se encontram no formato SIARD (a conversão das bases de dados para SIARD pode ser feita através do SIARD SUITE), decidimos fazer a conversão das bases dados (em formato SIARD) para ontologias (OWL), porque com as bases de dados convertidas em ontologias, temos a vantagem de que as ontologias trazem em termos de interpretação dos dados. Sendo que com esta conversão torna ainda possível a interpretação máquina do conhecimento que se encontra nas bases de dados.

Esta estratégia é de facto uma mais valia para futuras interpretações dos dados. As ontologias podem ser adotadas ainda para outros tipos de objetos digitais e desempenhar assim um papel importante na generalidade da preservação digital.

Fazendo agora uma analise do que poderia ser feito em termos de trabalho futuro, ficamos com dois grandes focos: a melhoria do repositório ou ainda a melhoria na conversão SIARD para ontologias.

Começaríamos pela divisão em vários arquivos da ontologia OWL gerado na conversão SIARD para OWL, tornando desta forma o sistema mais escalável.

Em termos de pesquisas futuras pode passar pela tentativa de tornar viável a interpretação máquina das bases da dados que foram transformadas em ontologias, ou explorar ainda os metadados pois as ontologias podem fornecer respostas às perguntas que outros padrões de metadados são incapazes de dar, porque em norma estes possuem estruturas rígidas ao contrário das ontologias.

Ao nível do repositório poderia acrescentar novas funcionalidades, como por exemplo, o produtor de uma ontologia poder escolher se quer a ontologia privada ou publica, melhorar a nível de estatísticas, alargar o arquivamento a outros objetos digitais, assim como novas conversões.

Ao juntar as ontologias com a preservação digital e sendo as ontologias escaláveis a tão diferentes áreas, um grande leque de possíveis pesquisas e trabalhos é assim possível para desta forma ajudarmos a preservação digital. Com esta dissertação esperamos dar um contributo positivo para uma melhor compreensão do problema, que é a preservação digital.

BIBLIOGRAFIA

Carlos Filipe Pereira Aldeias. *Open Archival Information Systems for Database Preservation*. PhD thesis, FEUP, 2011.

H. Besser. Digital Preservation of Moving Image Material. *The Moving Image*, 1(2), 2001.

Jon Bosak. Xml, java, and the future of the web. *World Wide Web Journal*, 2(4):219–227, 1997. URL http://dblp.uni-trier.de/db/journals/wwwj/wwwj2.html#Bosak97.

Stefan Brandl and Dr. Peter Keller-Marxer. Long-term archiving of relational databases with chronos. In *In First International Workshop on Database Preservation - PresDB'07*, 2007.

Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language. *World Wide Web J.*, 2(4):29–66, November 1997. ISSN 1085-2301. URL http://dl.acm.org/citation.cfm?id=274784.273625.

Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml) 1.0 (fifth edition). World Wide Web Consortium, Recommendation REC-xml-20081126, November 2008.

K. Selçuk Candan, Huan Liu, and Reshma Suvarna. Resource description framework: Metadata and its applications. *SIGKDD Explor. Newsl.*, 3(1):6–19, July 2001. ISSN 1931-0145. doi: 10.1145/507533.507536. URL http://doi.acm.org/10.1145/507533.507536.

Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. Okbc: A programmatic foundation for knowledge base interoperability. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 600–607, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL http://dl.acm.org/citation.cfm?id=295240.295747.

Óscar Corcho. *A declarative approach to ontology translation with knowledge preservation*. PhD thesis, Universidad Politecnica de Madrid - Facultad Informatica, 2004.

DANS. Data archiving and networked services (dans) institute, 2012. URL URLhttp://www.dans.knaw.nl/.

DELOS. Network of excellence on digital libraries, 2009. URL http://delos.info/.

Bipin C. Desai. *An Introduction to Database Systems*. West Publishing Co., St. Paul, MN, USA, 1990. ISBN 0-314-66771-7.

ExLibris. Rosetta exlibris, 2012. URL http://www.exlibrisgroup.com/.

Mário Joaquim Firmino Leite Faria. Definição de uma ontologia aplicada ao futebol. Master's thesis, FEUP, 2009.

Adam Farquhar. Planets 1: Integrated services for digital preservation, 2007.

Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. Oil: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, March 2001. ISSN 1541-1672. doi: 10.1109/5254.920598. URL http://dx.doi.org/10.1109/5254.920598.

Miguel Ferreira. *Introdução à preservação digital : conceitos, estratégias e actuais consensos*. Universidade do Minho. Escola de Engenharia, 2006.

Douglas Foxvog. Cyc. Theory and applications of ontology, pages 259–278. Springer, 2010. ISBN 9789048188475. URL http://books.google.pt/books?id=lQS6Abf9wzwC.

Ricardo André Pereira Freitas. *Relational Databases Digital Preservation*. PhD thesis, 2012.

Michael R. Genesereth. Knowledge interchange format. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *KR*, pages 599–600. Morgan Kaufmann, 1991. ISBN 1-55860-165-1.

Anne J. Gilliland-Swetland and Philip B. Eppard. Preserving the authenticity of contingent digital objects: The interpares project. *D-Lib Magazine*, 6(7/8), 2000. URL http://dblp.uni-trier.de/db/journals/dlib/dlib6.html#Gilliland-SwetlandE00.

Hugo Gonçalo Oliveira and Paulo Gomes. Onto.pt: Automatic construction of a lexical ontology for portuguese. In *Proceedings of the 2010 Conference on STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*, pages 199–211, Amsterdam, The Netherlands, The

Netherlands, 2010. IOS Press. ISBN 978-1-60750-675-1. URL http://dl.acm.org/citation.cfm?id=1940526.1940544.

Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. Owl 2: The next step for owl. *Web Semant.*, 6(4):309–322, November 2008. ISSN 1570-8268. doi: 10.1016/j.websem.2008.05.001. URL http://dx.doi.org/10.1016/j.websem.2008.05.001.

Thomas R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical report, 1992.

Michael Grüninger and Jintae Lee. Ontology applications and design. In *commun. ACM*, volume 45, pages 39–41, 2002.

Nicola Guarino. Formal ontology and information systems. In *FOIS*, pages 3–15, June 1998.

T. Vogele H. Wache, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner. Ontology-based integration of information — a survey of existing approaches. In *IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, Seattle, 2001.

Elliotte Rusty Harold. *XML Bible*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2001. ISBN 0764547607.

Jeff Heflin, James Hendler, and Sean Luke. Shoe: A knowledge representation language for internet applications. Technical report, 1999.

Gail M. Hodge. Best practices for digital archiving: An information life cycle approach. Technical report, 2000.

Swiss Federal Archives SFA Unit Innovation and Preservation. Siard format description. Technical report, Federal Department of Home Affairs FDHA, 2008.

Ross King, Rainer Schmidt, Christoph Becker 0001, and Sven Schlarb. Scape: Big data meets digital preservation. *ERCIM News*, 2012(89), 2012. URL http://dblp.uni-trier.de/db/journals/ercim/ercim2012.html#KingSBS12.

Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper. Fedora: An architecture for complex objects and their relationships, August 2005. URL http://arxiv.org/abs/cs.DL/0501012.

Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3c recommendation, W3C, February 1999. URL http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

K. H. Lee, O. Slattery, R. Lu, X. Tang, and V. Mccrary. The State of the Art and Practice in Digital Preservation. *JOURNAL OF RESEARCH-NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY*, 107(1):93–106, 2002.

Robert M. MacGregor. Inside the loom description classifier. *SIGART Bull.*, 2(3):88–92, June 1991. ISSN 0163-5719. doi: 10.1145/122296.122309. URL http://doi.acm.org/10.1145/122296.122309.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-13360-1.

Deborah L. McGuinness, Richard Fikes, James Hendler, and Lynn Andrea Stein. Daml+oil: An ontology language for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, September 2002. ISSN 1541-1672. doi: 10.1109/MIS.2002.1039835. URL http://dx.doi.org/10.1109/MIS.2002.1039835.

Roberto Navigli and Paola Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Comput. Linguist.*, 30(2):151–179, June 2004. ISSN 0891-2017. doi: 10.1162/089120104323093276. URL http://dx.doi.org/10.1162/089120104323093276.

Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. In *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*, 2001.

OpenThesaurusPT. Um projeto open source para a construção de um dicionário de sinónimos para a língua portuguesa. URL http://openthesaurus.caixamagica.pt.

PDFBox. Apache pdfbox - java pdf library. URL http://incubator.apache.org/pdfbox.

Janardhana Punuru. *Knowledge-Based Methods for Automatic Extraction of Domain- Specific Ontologies*. PhD thesis, Department of Computer Science of Louisiana State University, 2007.

Arif Ur Rahman, Gabriel David, and Cristina Ribeiro. Model migration approach for database preservation. In *Proceedings of the Role of Digital Libraries in a Time of Global Change, and 12th International Conference on Asia-Pacific Digital Libraries*, ICADL'10, pages 81–90, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13653-2, 978-3-642-13653-5. URL http://dl.acm.org/citation.cfm?id=1875689.1875702.

José Carlos Ramalho, Miguel Ferreira, Luis Faria, and Rui Castro. Relational database preservation through xml modelling. In *Proceedings of Extreme Markup Languages*, 2007.

José Carlos Ramalho, Miguel Ferreira, Luís Faria, Rui Castro, Francisco Barbedo, and Luís Corujo. Roda and crib a service-oriented digital repository. 2008.

A. Rauber and A. Aschenbrenner. Part of our culture is born digital — On efforts to preserve it for future generations. *TRANS. On-line Journal for Cultural Studies (Internet-Zeitschrift für Kulturwissenschaften)*, 10, July 2001. URL /brokenurl#citeseer.nj.nec.com/article/rauber01part.html.

Goutam Kumar Saha. Web ontology language (owl) and semantic web. *Ubiquity*, 2007 (September):1:1–1:1, September 2007. ISSN 1530-2180. doi: 10.1145/1295289.1295290. URL http://doi.acm.org/10.1145/1295289.1295290.

José Miguel Gomes Saias. Uma metodologia para a construção automática de ontologias e a sua aplicação em sistemas de recuperação de informação. Master's thesis, Universidade de Évora, 2003.

Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, Manchester, UK, 1994. URL http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/.

John F. Sowa. Ontology-based integration of information - a survey of existing approaches. In *Artificial Intelligence of the IBM Systems Journal*, volume 41, pages 331–349, Seattle, 2002.

Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, second edition, 2009.

Vlad Tanasescu, John Domingue, and Liliana Cabral. Ocml ontologies to xml schema lowering. In *First AKT Workshop on Semantic Web Services (AKT-SWS04)*, 2004. URL http://oro.open.ac.uk/23219/.

Kenneth Thibodeau. *Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years presented at The State of Digital Preservation: An International Perspective*. Washington D.C., 2002.

Paul Wheatley. Migration: a camileon discussion paper. In *Ariadne*, 2001.

Norbert Wiener. *Cybernetics, Second Edition: Or the Control and Communication in the Animal and the Machine*. The MIT Press, 1965. ISBN 026273009X.

Faruk Mustafa Zahra, Deborah Ribeiro Carvalho, and Andreia Malucelli. Poronto: ferramenta para construção semiautomática de ontologias em português. *Health Informatics Journal*, 19 (2):52–59, 2013.