PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO EM SISTEMAS E COMPUTAÇÃO

EDERALDO MUNIZ BARRETO JUNIOR

EXTRAÇÃO DE INFORMAÇÕES EM FORMATO TEXTO, IMPLICITAMENTE ANOTADAS PELA SEMÂNTICA IN LOCO, UTILIZANDO PERFIS ADAPTÁVEIS AO DOMÍNIO

Salvador
2011

EDERALDO MUNIZ BARRETO JUNIOR

EXTRAÇÃO DE INFORMAÇÕES EM FORMATO TEXTO, IMPLICITAMENTE ANOTADAS PELA SEMÂNTICA IN LOCO, UTILIZANDO PERFIS ADAPTÁVEIS AO DOMÍNIO

Dissertação elaborada junto ao programa de Mestrado em Sistemas e Computação. Universidade Salvador – UNIFACS, como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação.

Orientador: Prof^{o} André Santanchè

Salvador
2011

FICHA CATALOGRÁFICA
(Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador)

Barreto Junior, Ederaldo Muniz

Extração de informações em formato texto, implicitamente anotadas pela semântica in loco, utilizando perfis adaptáveis ao domínio. Ederaldo Muniz Barreto Junior. – Salvador, 2011.

125 f. : Il.

Dissertação apresentada ao Curso de Mestrado em Sistemas e Computação, Universidade Salvador – UNIFACS, como requisito parcial para obtenção do grau de Mestre.

Orientador Profº. Drº. André Santanchè.

1. Ontologias. 2. Gestão do conhecimento. I. Título

CDD: 004

TERMO DE APROVAÇÃO

EDERALDO MUNIZ BARRETO JUNIOR

EXTRAÇÃO DE INFORMAÇÕES EM FORMATO TEXTO, IMPLICITAMENTE ANOTADAS PELA SEMÂNTICA IN LOCO, UTILIZANDO PERFIS ADAPTÁVEIS AO DOMÍNIO

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador - UNIFACS, pela seguinte banca examinadora:

André Santanchè – Orientador _____
Doutor em Ciência da Computação pela Universidade Estadual de Campinas
Universidade Estadual de Campinas

Daniela Barreiro Claro _____
Doutora em Ciência da Computação pela Université d'Angers, França
Universidade Federal da Bahia - UFBA

Glauco de Figueiredo Carneiro _____
Doutor em Ciência da Computação pela Unifacs pela Universidade Federal da Bahia - Ufba
UNIFACS Universidade Salvador

Salvador, 03 de agosto de 2011.

Dedico este trabalho aos meus pais, a minha esposa e aos meus filhos Gabriel e Daniel Amado Muniz Barreto.

AGRADECIMENTOS

São tantos e tão especiais...

A Deus em primeiro lugar, pelo dom da vida e por permitir que tudo isso se tornasse realidade.

A todos aqueles que me ajudaram direta ou indiretamente na realização desse sonho. Em especial aos meus amigos e colegas Magda Maria Andrade e a Carlos Antônio Guimarães, que contribuíram com discussões, ideias, apoio técnico e diversas formas de incentivo.

Ao Professor Doutor André Santanchè, pela contribuição na definição inicial do objeto pesquisado, sem o que não seria possível a realização deste estudo e pela orientação, ajuda análise, crítica, dedicação, paciência e principalmente, por ter acreditado em mim.

Ao Professor Ph.D. José Augusto Suruagy Monteiro, pelos bons conselhos e serenidade nos momentos mais difíceis.

Por fim, quero manifestar, de forma espacial, a minha gratidão a toda minha família e aos amigos que compartilharam desta jornada.

“Ninguém pode voltar atrás e fazer um novo começo. Mas qualquer um pode recomeçar e fazer um novo fim”.

Chico Xavier.

RESUMO

Diariamente, uma quantidade significativa de conteúdo digital é produzida e uma grande parte dela é composta de informações que, de alguma forma, podem ser melhor estruturadas para facilitar seu inter-relacionamento e tratamento automatizado. Grande parte da informação armazenada nos repositórios digitais (e.g., a Web e as Bibliotecas Digitais) encontra-se na forma de documentos textuais. Considerando o imenso volume de informação disponível nesses repositórios, é muito importante a construção de ferramentas capazes de selecionar automaticamente apenas os dados de interesse de um usuário, facilitando assim o acesso e a manipulação dessas informações. Estudos no contexto de IE – *Information Extraction* ou Extração de Informação – procuram automatizar tarefas de busca e extração de informações dentro dos documentos e tratá-las de forma a apresentar algum tipo de conhecimento novo, relevante e útil para o usuário. Estudos relacionados com IE, no contexto abordado nesse trabalho, vêm sendo desenvolvidos desde 1987. Ainda persistem grandes desafios a serem resolvidos nessa área. As ferramentas de IE ainda são caras, seu desempenho e precisão ainda são relativamente baixos e principalmente são de difícil utilização e adaptação para novos contextos. Diante dessa realidade, esta dissertação apresenta uma proposta de anotação e IE baseada na nossa metodologia denominada Semântica In Loco, configurável de acordo com o domínio relativo do documento, através do conceito de perfis associado às ontologias. Tal proposta mostrou desempenho e precisão acima dos atuais padrões em alguns contextos. Ela é de fácil adaptação e voltada ao usuário final; foi validada através de uma ferramenta capaz de extrair e estruturar em uma ontologia (*Research Card*) dados relativos à fichamento bibliográfico. Tal ontologia explicita a semântica dos registros e é utilizada para a criação de uma base semanticamente inter-relacionada de referências e anotações.

Palavras chaves: Extração de informações. Anotação semântica. Ontologias.

ABSTRACT

A significant amount of digital content is produced each day and a large part of it consists of information that may be better structured or inter-related. Most of the information of digital repositories (e.g., the Web and Digital Libraries) is stored as textual documents. Considering the immense volume of information available in repositories, there is hight interest to build systems able to automatically select only relevant data to the user, thus facilitating the access and manipulation of information. Studies in the context of IE – Information Extraction – are related to automating tasks of information search and extraction within documents and treat them in order to provide some kind of new knowledge, relevant and useful to the user. Studies related to IE, in the context addressed in this work, were developed since 1987. Major challenges still remain to be solved in this area. The IE tools are still expensive, their performance and accuracy are still relatively low and they are especially difficult to use and adapt to new contexts. Therefore, this dissertation presents a proposal for annotation and IE based on the In Loco Semantics methodology, configurable according to the document domain, by using profiles associated to ontologies. This proposal achieved performance and accuracy over current standards in some contexts. It is easy to adapt and designed to end users; it was validated by applying a tool capable of extracting and organizing in an ontology (Research Card) data related to bibliographic cards. Such ontology makes explicit the semantics of records and is applied in the creation of a base for references and annotations semantically inter-related.

Keywords: Information extraction. Semantic annotation. Ontologies.

LISTA DE FIGURAS

Figura 1- Associação (recurso – propriedade – valor) representada como um DLG 27
Figura 2 - RDF do DLG Home Page Ederaldo Junior..... 28
Figura 3 - Diagrama estrutural do OWL 2 com ênfase nas entidades 32
Figura 4 - Ontologia Wine 33
Figura 5 - Ontologia Wine 33
Figura 6 - Ontologia Wine 34
Figura 7 - Ontologia Wine 34
Figura 8 - A evolução do crescimento da quantidade de informações digitais no mundo 47
Figura 9 - Modelo Abstrato de Fonte de Dados de Origem 51
Figura 10 - Modelo Abstrato do Mapeamento 52
Figura 11 - Processo de extração de informações de fontes de dados textuais implicitamente marcadas, utilizando perfis adaptáveis 53
Figura 12 - Arquivo adaptador que é utilizado para configurar o InLocoLoader para o contexto desejado pelo autor 54
Figura 13 - Um dos arquivos de modelo do Microsoft Word 2007 que faz parte do arquivo de perfil pesquisa científica 55
Figura 14 - Transformação de documento + estilo+ conteúdo em tripla RDF 57
Figura 15 - Extração e mapeamento de um cenário para um template no formato de ontologia 58
Figura 16 - Fragmento de esquema do adaptador..... 59
Figura 17 - Fragmento de arquivo adaptador 59
Figura 18 - Modelo do processo interno do InLocoLoader 62
Figura 19 - Diagrama de Casos de Uso do Plug-in InLocoLoader 63
Figura 20 - Diagrama de Classes do plug-in InLocoLoaderDiagrama de Classes do plug-in InLocoLoader 64
Figura 21 - Diagrama sintético com principais elementos do Research Card 70
Figura 22 - Ontologias Cards e Subjects..... 71
Figura 23 - Arquivo de modelo utilizado para anotar conteúdo de artigos..... 73
Figura 24- Arquivo adaptador para o domínio catalogação de publicação..... 76
Figura 25 - Software Protegé utilizado para criar e manter ontologias 77
Figura 26 - Tela de Importação de ontologias e extração das Informações relevantes do documento 77
Figura 27 - Apresentação das informações extraídas do modelo do Word utilizado pelo arquivo de perfil..... 78
Figura 28 - Caixa *Ontology Browser* preenchida com as ontologias importadas que compõem o perfil de saída..... 78
Figura 29 - Resultado final do processo de extração e mapeamento com a apresentação das informações extraídas e mapeadas para as propriedades das ontologias 79

LISTA DE TABELAS

Tabela 1 - Tipos e Funções dos Metadados 20
Tabela 2 - Classes essenciais do RDF-Schema 29
Tabela 3 - Algumas iniciativas pioneiras de Extração de Informação antes da década de 90 . 39
Tabela 4 - Resumo das propriedades da fonte de dados e do processo de extração relevantes para o problema de Extração de Informação 42
Tabela 5 - Resumo das fases de um algoritmo de aprendizagem utilizado em uma extração semiautomática 44
Tabela 6 - Cenário de Uso de Ferramentas de Extração de Informações em relação ao tamanho da Fonte de Dados e o nível de complexidade da formulação das regras de extração 48
Tabela 7 - Relação dos artigos que serão anotados no caso de uso Research Card 68
Tabela 8 - Resumo descritivo das ontologias 71
Tabela 9 – Padrão de anotação utilizadono caso de uso com as propriedades das ontologias do template de saída 74
Tabela 10 - Resumo dos resultados do cálculo dos indicadores Recall e Precision do caso de uso 81
Tabela 11 - Resumo dos resultados do cálculo dos indicadores Desvio Padrão Recall e Desvio Padrão Precision do caso de uso 83
Tabela 12 - Características da Qualidade de Software relacionadas à Usabilidade segundo a ISO/IEC 9126-1 85
Tabela 13 - Tabulação dos resultados da aplicação do formulário de Avaliação da Utilização da Ferramenta de Extração 85

LISTA DE SIGLAS

BDTD	Biblioteca Digital Brasileira de Teses e Dissertações
DCMI	Dublin Core Metadata Initiative
DDEx	Dinamic Document Extraction
IBICT	Instituto Brasileiro de Informação em Ciências e Tecnologia
IE	Information Extraction (Extração de Informação)
IR	Information Retrieval (Recuperação de Informação)
MUC	Message Understanding Conferences.
OBIE	Information Extraction Ontology Based
OWL	Ontology Web Language
RDF	Resource Description FrameWork
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

SUMÁRIO

1 INTRODUÇÃO 14
1.1 JUSTIFICATIVA 14
1.2 DELIMITAÇÃO DO PROBLEMA 15
1.3 HIPÓTESES 17
1.4 OBJETIVOS 17
1.4.1 Objetivo Geral 17
1.4.2 Objetivos Específicos 17
1.4.3 Contribuições 17
1.5 ESTRUTURA DO TRABALHO 18
2 FUNDAMENTAÇÃO TEÓRICA 19
2.1 METADADOS 19
2.2 DUBLIN CORE 21
2.3 ONTOLOGIAS 23
2.3.1 Histórico 23
2.3.2 Conceitos 24
2.3.3 Linguagens de Representação de Ontologias 26
2.4 ANOTAÇÃO 35
2.4.1 Conceito 35
2.4.2 Classificação 35
2.4.3 Anotação Semântica 36
2.4.4 Semântica In Loco 37
2.5 EXTRAÇÃO DE INFORMAÇÃO EM BASES TEXTUAIS 38
2.5.1 Origem da Extração de Informação (IE) 38
2.5.2 Conceito de Extração de Informação 39
2.5.3 Fatores que influenciam na escolha da solução do problema de extração 42
2.5.4 Por que o Interesse em Extração de Informação 46
2.5.5 Cenário de Uso da Extração de Informação 47
2.6 RESUMO DO CAPÍTULO 49
3 EXTRAÇÃO DE INFORMAÇÕES DE FONTES IMPLICITAMENTE MARCADAS 50
3.1 MODELO ABSTRATO 50
3.1.1 Modelo Abstrato da Fonte de Dados de Origem 51
3.1.2 Modelo Abstrato do Mapeamento 52
3.2 METODOLOGIA 52
3.3 PROJETO 53
3.3.1 Produção 56
3.3.2 Extração e Mapeamento 57
3.4 O AMBIENTE DO *INLOCOLOADER* 62
3.4.1 Diagrama de Casos de Uso do InLocoLoader 63
3.4.2 Diagrama de Classes 64
3.5 RESUMO DO CAPÍTULO 64
4 CASO DE USO 66
4.1 PLANEJAMENTO DO CASO DE USO 66
4.2 DESCRIÇÃO DO FERRAMENTAL UTILIZADO NO CASO DE USO 67
4.2.1 Fonte de dados de origem 68
4.2.2 Ontologia Research Card – Cartão de Pesquisa 70

4.2.3 Arquivo de Modelo e Padrão de Anotação **73**
4.2.4 O arquivo adaptador **75**
4.3 EXECUÇÃO DO CASO DE USO 76
4.4 APLICAÇÃO DA FERRAMENTA EM CAMPO 79
4.5 AVALIAÇÃO DO USO DA FERRAMENTA 84
4.6 POSSÍVEIS UTILIZAÇÕES DAS ONTOLOGIAS RESULTANTES DO PROCESSO DE EXTRAÇÃO 86
4.7 RESUMO DO CAPÍTULO 86

5 CONSIDERAÇÕES FINAIS **88**

5.1 ASPECTOS PRINCIPAIS 88
5.2 PRINCIPAIS CONTRIBUIÇÕES 89
5.3 TRABALHOS FUTUROS 90

REFERENCIAS **91**

APÊNDICE A – FORMULÁRIO UTILIZADO NO CASO DE USO PARA EXTRAÇÃO DE INFORMAÇÃO **96**

APÊNDICE B – CÓDIGO FONTE CLASSE CONSTRUTOR.JAVA **98**

APÊNDICE B – CÓDIGO FONTE CLASSE INLOCOLOADER.JAVA **106**

APÊNDICE C – CÓDIGO FONTE CLASSE PROPERTYLEVEL.JAVA **110**

APÊNDICE D – CÓDIGO FONTE CLASSE OBJECTFACTORY.JAVA **111**

APÊNDICE E – CÓDIGO FONTE CLASSE TDOCUMENT.JAVA **113**

APÊNDICE E – CÓDIGO FONTE CLASSE TPROPERTY.JAVA **116**

APÊNDICE F – CÓDIGO FONTE CLASSE ADAPTER.TXT **121**

APÊNDICE G – CÓDIGO FONTE CLASSE ADAPTER.XSD **122**

APÊNDICE H – CÓDIGO FONTE CLASSE ARTICLEW4.XML **123**

APÊNDICE I – CÓDIGO FONTE CLASSE ESQUEMAW4.XSD **124**

APÊNDICE J – FORMULÁRIO DE PESQUISA DE SATISFAÇÃO DO USO DA FERRAMENTA INLOCOLOADER **125**

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

De acordo com dados do Instituto Brasileiro de Informação em Ciências e Tecnologia – IBICT, só no ano de 2010, 6342 teses de doutorado e 18.111 dissertações de mestrado foram publicadas na Biblioteca Digital Brasileira de Teses e Dissertações – BDTD IBCT (2011). Caso levantamento similar seja realizado em outras bibliotecas digitais brasileiras e internacionais, considerando também outros tipos de publicações em formato texto, como por exemplo, artigos científicos, a conclusão será que a quantidade de material digital em formato texto publicado nas bibliotecas digitais é imensa. Essa realidade traz o desafio de como enriquecer a semântica deste enorme volume de informações, de forma a permitir que máquinas nos auxiliem na seleção e relacionamento do conhecimento contido nestas bibliotecas.

Para tentar vencer este desafio, muitas iniciativas vêm sendo desenvolvidas, tanto no que tange à recuperação desses documentos para uma análise posterior pelo leitor, princípio da *Information Retrieval – IR* (Recuperação de Informação), como na extração de informações relevantes dessa imensa massa de publicações, princípio da *Information Extraction – IE* (Extração de Informação). Uma vez de posse de um documento, ou conjunto de documentos, de que se tenha interesse e definido o cenário do que se quer extrair, esses dados são extraídos em um formato de saída desejado em que, por exemplo, podem ser integrados por sistemas de base de conhecimento (TURMO, 2006).

Em muitos contextos a IR e a IE terão papéis complementares. Isto se deve ao fato de que, em muitos contextos, os esquemas sob os quais estão organizados os dados, bem como a respectiva semântica associada, não estão explícitos e/ou estruturados. A IE pode atuar de forma complementar neste sentido, pois procura automatizar tarefas de seleção e extração de informações dentro de documentos, bem como o seu tratamento de forma a apresentá-las de forma mais estruturada, sendo assim ideal para a resolução do problema de tratamento e consumo seletivo da grande quantidade de material digital publicado.

Estudos relacionados à extração de informações de fontes de dados em formato texto nos mostram que os desafios nessa área não são poucos. Segundo Turmo (2006), um dos

requisitos do IE é que o tipo de conteúdo a ser extraído deve ser definido a priori. Tal dependência da tecnologia de IE usualmente gera problemas de portabilidade. A cada novo domínio, novos conhecimentos específicos são necessários a ser usados para a definição dos dados extraídos. Cowie (1996) ressalta ainda o custo de adaptação de um sistema de extração para novos domínios, bem como o tempo necessário para essa adaptação: geralmente meses. Além disso, métodos estatísticos, embora presentes em muitos dos componentes dos sistemas de IE que lidam com fontes de dados de textos em linguagem natural, não são suficientes para atender muitas das tarefas e têm de ser combinadas com abordagens baseadas em conhecimento.

Um recurso que vem sendo bastante utilizado em conjunto com IE é a técnica de anotação semântica. A fonte de dados a ser utilizada pelo processo de IE é previamente marcada por um determinado tipo de anotação e, então, o processo de IE trabalha especificamente extraindo o conteúdo da fonte anotado. Essas técnicas quando usadas em conjunto simplificam bastante o processo de IE, mas quando a fonte de dados é muito extensa, o processo de anotação pode se tornar trabalhoso. Sendo assim, é mais indicada para fontes de dados textuais não legadas, ou seja, aquelas que já tenham sido anotadas durante o processo de produção do texto.

Dentro desse contexto, essa dissertação se propõe a apresentar uma proposta de IE baseada na metodologia para anotações, denominada Semântica In Loco[1] (SANTANCHÈ, 2007), cujo principal diferencial é permitir a anotação do conteúdo concomitante à sua produção de forma pouco intrusiva. Em muitos contextos sua abordagem pode apresentar desempenho e precisão acima dos atuais padrões. Este trabalho contribui definindo uma técnica de adaptação de acordo com o domínio relativo do documento, através do conceito de perfis associados às ontologias.

1.2 DELIMITAÇÃO DO PROBLEMA

Segundo Brown (1996), o mercado de publicação vem se diversificando já há um bom tempo em relação a novas formas de impressão e distribuição das publicações, além do tradicional suporte em papel. Atualmente, existem diversas publicações como livros, revistas e jornais

---

[1] Desenvolvida pelo grupo de pesquisa Semântica In Loco do qual faço parte.

que, além do suporte tradicional em papel, também lançam seus exemplares em CD-ROM, DVDS e também na Internet. Cada meio possui características específicas quanto à sua confecção, apresentação gráfica e distribuição. No entanto, acredita-se que o modelo mais atraente para os dias atuais é mesmo a publicação eletrônica via *World Wide Web*, principalmente em virtude do baixo custo, facilidade de produção, distribuição ilimitada e possibilidade de interação com o leitor. Todas estas vantagens demonstram uma tendência na edição de versões *on-line* para as publicações impressas comerciais e, no caso específico da publicação científica ou acadêmica, isto não é diferente.

Diante desse novo contexto, a oferta de novas publicações tende a crescer a cada dia, decorrendo disso um problema que se agravou com a microinformática e a popularização da Web: o excesso de informações, principalmente em formato texto, conduzindo à seguinte questão de pesquisa:

Como extrair informações relevantes dessa grande quantidade de material em formato de texto, em contínuo crescimento, obedecendo a um formato específico de saída, de maneira eficiente e simples para o usuário e adaptável a diferentes domínios de conhecimento?

Deste modo, um dos desafios consiste no desenvolvimento de pesquisas objetivando a construção de ferramentas que, de alguma forma, auxilie o usuário final na extração de informações relevantes em base de dados textuais, simplificando seu trabalho de busca, seleção e relacionamento de tais informações.

Uma das questões centrais tratadas nesta pesquisa diz respeito à precisão insuficiente relacionada às técnicas tradicionais de extração de informação, dado que estas tentam extrair dados de documentos usualmente escritos em linguagem natural, cujo conteúdo não é propício para processamento automatizado.

Como será apresentada nesta dissertação, a associação de anotações a documentos, dos quais serão extraídos os dados, pode resultar em melhor desempenho e maior precisão, além da produção de um resultado semanticamente enriquecido. Como resultado, é possível obter dados relevantes de documentos em formato texto, de forma transparente, no próprio ambiente de produção do usuário comum. Por fim, trata-se de uma estratégia de extração de informação mais flexível, sendo indicada para fontes de dados textuais não legadas, atuando de forma complementar às estratégias de extração manual e automática, estas mais indicadas para fontes de dados textuais legadas.

1.3 HIPÓTESES

(i) A utilização da Metodologia Semântica In Loco fornece subsídios para a criação de uma solução de Extração de Informação alternativa e complementar as atuais técnicas de Extração de Informação utilizadas no mercado.

(ii) A utilização do conceito de Perfis Adaptáveis ao Domínio possibilita a fácil adaptação da solução de Extração de Informação proposta a novos domínios de conhecimento.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Desenvolver uma solução de IE baseada na metodologia para anotações denominada Semântica In Loco, configurável de acordo com o domínio do documento, através do conceito de perfis associados a ontologias.

1.4.2 Objetivos Específicos

1) Introduzir o conceito de perfis adaptáveis a novos domínios na metodologia de anotação chamada Semântica In Loco.

2) Implementar tal conceito de perfis adaptáveis pela extensão da ferramenta de extração DDEx.

3) Validar a proposta pelo desenvolvimento de uma ferramenta capaz de extrair dados relevantes de documentos e armazená-los em ontologias, bem como sua aplicação em um exemplo.

1.4.3 Contribuições

As principais contribuições deste trabalho são:

- A extensão da metodologia Semântica In Loco para oferecer suporte a perfis associados a ontologias.
- A extensão do software de criação e gerenciamento de ontologias *Protegé* com a funcionalidade de extração de informação de bases textuais, através do *Plug-in InLocoLoader,* capaz de extrair dados relevantes de documentos em formato texto e armazená-los em ontologias.
- Desenvolvimento de um padrão de anotação semântica (formado por um conjunto de metadados e estilos) utilizados no caso de uso para validar a ferramenta InLocoLoader.
- Contribuição na evolução das ontologias Research Card, Card e Subject, aprimoradas para o caso de uso.

1.5 ESTRUTURA DO TRABALHO

Esta dissertação possui cinco capítulos, sendo os mesmos:

O Capítulo 01 apresenta a introdução do trabalho, detalhando a sua justificativa, o problema de pesquisa, objetivos, contribuições e a estrutura do trabalho.

O Capítulo 02 aborda os aspectos conceituais e a fundamentação teórica que subsidiaram o desenvolvimento deste trabalho.

O Capitulo 03 apresenta a metodologia de extração utilizada, detalhando suas várias etapas bem como os vários conceitos associados. É também detalhada cada etapa: partindo de uma entrada, mostrando quais as transformações necessárias para se extrair as informações relevantes desejadas, bem como o formato de saída, que no contexto deste trabalho é formado por ontologias de domínio.

O Capítulo 04 apresenta uma aplicação prática da solução de extração, o plug-in para o Protegé – InLocoLoader, voltado para o domínio específico o de catalogação de publicações.

O Capítulo 05 apresenta a conclusão do trabalho, abordando suas contribuições e limitações, bem como oferecendo uma perspectiva de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são fundamentados os conceitos associados às áreas de pesquisa envolvidas neste trabalho, que são: metadados, o padrão Dublin Core, ontologias, anotação semântica, Semântica In Loco e extração de informações de bases textuais. Também são descritos sucintamente alguns trabalhos relacionados que contribuíram para a pesquisa realizada.

2.1 METADADOS

Na literatura o conceito de metadados assume diferentes siginificados de acordo com o contexto que ele é utilizado. Sendo assim, não há um conceito universal sobre metadados. Ao contrário, trata-se de um conceito ainda em debate nas diversas comunidades nas quais ele é adotado (BREITMAN, 2005).

O ponto de partida comum é o conceito genérico de metadados como dados sobre dados. Este conceito assume diferentes formas e papéis a depender do contexto. No caso específico da Web Semântica, no qual esse trabalho se baseia, o consórcio W3C (World Wide Web Consortium) adota a noção de metadados como descrições de recursos passíveis de ser interpretados por máquinas (MANOLA; MILLER, 2004).

Recurso é um objeto de informação descrito pelos metadados. Um objeto de informação é algo que pode ser tratado como uma entidade discreta em diferentes níveis de agregação (GILLILAND-SWETLAND, 2007), podendo ser tanto um simples dado numérico, quanto um documento, uma página da Web, uma coleção ou um sistema. Para a Web semântica, o termo recurso indica qualquer objeto que pode ser endereçado através de um URI (*Uniform Resource Identifier)* (W3C, 2003), e.g., documentos, *sites* ou sistemas. A descrição de pessoas, objetos ou organizações do mundo real pode ser feita se houver uma página ou URI (e.g., e-mail) associado ao que se quer descrever.

Uma das principais funções dos metadados é a descrição de recursos tendo como objetivo a sua descoberta e localização. Contudo, não é a única função dos metadados no mundo digital. Segundo Rocha (2004), há várias outras funções desempenhadas pelos metadados, como: a descrição do formato do recurso tendo como objetivo capacitar os computadores, para

identificar quais aplicativos podem ser utilizados na sua manipulação. A Tabela 1 abaixo apresenta uma síntese proposta por Gilliland-Swetland (2007) de tipos de metadados e suas respectivas funções.

Tabela 1 - Tipos e Funções dos Metadados

Tipo	**Definição**	**Exemplos**
Administrativo	Metadados utilizados na gerência e na administração de coleções e recursos de informação	▪ Aquisição de informação ▪ Registro de direitos e reprodução ▪ Documentação dos requisitos legais de acesso ▪ Informação de localização ▪ Critério de seleção para digitação
Descritivo	Metadados utilizados para identificar e descrever coleções e recursos de informação relacionados	▪ Registros de catalogação ▪ Auxílio para a procura de informação ▪ Diferenciação entre versões ▪ Índices especializados ▪ Informações para curadoria ▪ Utilização de hiperlinks entre recursos ▪ Anotações por criadores e usuários
Preservação	Metadados relacionados ao gerenciamento para preservação de coleções e recursos de informação	▪ Documentação sobre as condições físicas dos recursos ▪ Documentação sobre as ações tomadas de modo a preservar as versões físicas e digitais dos recursos, e.g., atualização e migração ▪ Documentação de quaisquer mudanças que venham a ocorrer durante a digitalização ou preservação
Técnica	Metadados relacionados à funcionalidade do sistema e como seus metadados se comportam	▪ Documentação sobre hardware e software ▪ Informação relativa à digitação, e.g., formatos, compressão, rotinas de redimensionamento ▪ Registro de tempo de resposta do sistema ▪ Autenticação de dados, e.g., senhas e criptografia
Utilização	Metadados relacionados ao nível e ao tipo de utilização dos recursos	▪ Registros de circulação ▪ Registros de exibição física e digital ▪ Registro de uso e dos usuários dos recursos ▪ Reutilização do conteúdo e informação relativa à multi-versionamento ▪ Logs de busca ▪ Metadados de direitos

Fonte: Tradução livre de Gilliland-Swetland (2007).

Os metadados podem ser classificados também de acordo com a forma que eles se relacionam com o recurso descrito. Eles podem ser armazenados separados dos seus recursos em repositórios, sendo produzidos por sistemas chamados de anotadores, que guiam os usuários na descrição dos recursos (ROCHA, 2004), ou os metadados podem fazer parte do próprio documento que descrevem. Nesse trabalho adotamos a segunda modalidade, através de uma estratégia de padrões de anotação. Por exemplo, os estilos do Microsoft Word, além de formatar a aparência do documento, também foram usados para introduzir a semântica necessária para que aplicações de extração possam ter acesso aos metadados desejados, como será explicado no Capítulo 03.

A evolução natural do uso de metadados na descrição de recursos na Web é a criação de esquemas próprios de metadados para padronizar tais descrições. A arquitetura distribuída da Web conduz naturalmente a uma multiplicação de esquemas, resultando em uma inevitável sobreposição de funcionalidades (ROCHA, 2004). Além disso, em alguns casos é necessária a descrição de recursos que são transversais a vários domínios, sendo um único esquema insuficiente para descrevê-los. Há diversas possíveis abordagens para minimizar o efeito multiplicador e sobreposto dos esquemas, entre elas está a construção de esquemas de base reunindo características comumente encontradas. Uma das iniciativas mais significativas neste sentido é a do Dublin Core.

2.2 DUBLIN CORE

O Dublin Core surgiu da necessidade de se ciar uma forma racional de se lidar com a utilização de esquemas padronizados de metadados para descrição de recursos na Web. Sendo assim, peritos na área de informação juntamente com especialistas em padrões para Web, participaram em 1995 de um congresso na cidade de Dublin no estado de Ohio que, segundo Weibel (1995), tinha como principal objetivo identificar e definir um conjunto de elementos de metadados, para descrição e descoberta recursos da Web.

Diante dos antigos problemas já enfrentados por esses usuários de esquemas de metadados como: sobreposição de esquemas e transversalidade de domínios na descrição de recursos, um conjunto comum mínimo de elementos de grande abrangência foi definido. Esta característica foi fundamental para o seu sucesso.

Segundo DCMI (2007), o Dublin Core tem como objetivo:

- ✓ Simplicidade da criação e manutenção - conjunto pequeno de elementos permitindo que um não especialista facilmente crie registros descritivos simples.
- ✓ Semântica comumente compreendida - definição de metadados de amplo espectro universalmente compreendidos e suportados.
- ✓ Escopo Internacional - desenvolvimento de um padrão de metadados de natureza multilíngue e multicultural no universo de informação eletrônica.
- ✓ Extensibilidade - fornecer mecanismo de extensão para recursos adicionais descobertos, possibilitando diferentes comunidades especificar informações descritivas úteis na Web. Possui flexibilidade e extensibilidade para codificação de semânticas e padrões mais elaborados.

Os 15 elementos do Dublin Core, segundo Rocha (2004), podem ser agrupados em três categorias:

- ✓ Descrição de conteúdo
  - Título, assunto, descrição, linguagem, fonte, recursos relacionados e abrangência (espacial ou temporal);
- ✓ Elementos de propriedade intelectual
  - Criador (responsável intelectual do recurso), publicador (quem tornou o recurso público), contribuidor e direitos autorais;
- ✓ Elementos de instância
  - Data, tipo (ex.: livro, artigo), formato (ex.: pdf. mp3) e identificador (ex.: URI, ISSBN)

Os elementos do Dublin Core são opcionais e podem ser repetidos quando necessário para assegurar que todas as características de um determinado recurso tenham sido representadas a contento (CONTESSA, 2006).

Além dos 15 elementos comuns do Dublin Core, para contemplar a descrição de recursos de comunidades com domínios de conhecimento específicos, que necessitam de descrição mais especificas foi estabelecido o Dublin Core Qualifiers (ROCHA, 2004). Onde os recursos de um determinado domínio são descritos pelos mesmos elementos do Dublin Core, porém são qualificados através de regras próprias desse domínio.

O Dublin Core Qualifiers evoluiu em 2008 para a especificação DCMI Metadata Terms (DUBLIN..., 2008), que engloba as especificações anteriores em um conjunto unificado, em que são explorados elementos mais expressivos da semântica RDF, tal como o uso de classes para definir valor de propriedades. A propriedade *creator*, por exemplo, passa a ter valores no domínio da classe *Agent*, cujos objetos representam pessoas, organizações ou outra entidade capaz de ser criadora de um recurso (DUBLIN..., 2008).

Nesse trabalho, o papel dos metadados, mais especificamente o Dublin Core, será prover a semântica necessária para a extração de informações dos documentos em formato texto, previamente anotada segundo a metodologia de anotação denominada Semântica in Loco, definida na seção 2.4.4.

2.3 ONTOLOGIAS

2.3.1 Histórico

O termo ontologia tem sua origem na Filosofia. Segundo Breitman (2005), a sua primeira estrutura de classificação foi criada por Aristóteles e comentada por Porfírio no século III DC. Enquanto disciplina dessa área, tem como objetivo fornecer sistemas de categorização para organizar a realidade.

No início dos anos noventa, fenômenos como a Web e o barateamento progressivo dos dispositivos de armazenamento contribuíram para o aumento exponencial da oferta de dados disponíveis, fato esse determinante para a grande valorização das técnicas de organização da informação, que têm como objetivo a busca de melhorias na seleção, processamento, recuperação e disseminação desses dados. Diante dessa nova perspectiva, ontologias ganharam um papel de crescente relevância. No contexto da Web foram definidos padrões abertos de grande generalidade e flexibilidade. Deste modo, ontologia começou a ser utilizada em diversos contextos, como na área de recuperação de informações na Web, gestão de conhecimento, elaboração de sistemas educacionais inteligentes, além da área de inteligência artificial (engenharia e representação do conhecimento e o processamento de linguagem natural).

2.3.2 Conceitos

Diante da grande variedade de domínios em que o termo ontologia é utilizado, não há na literatura uma definição universal para ontologia. Segundo o dicionário Aurélio, "Ontologia é a parte da filosofia que trata do ser enquanto ser, do ser concebido como tendo uma natureza comum, que é inerente a todos, e a cada um, dos seres" (FERREIRA, 1999).

A comunidade da Ciência da Computação se apropriou do termo e deu-lhe uma interpretação particular e específica para o seu domínio. Neste trabalho adotamos uma síntese indicada por Guarino et al. (2009) para o conceito de ontologia. Ela define um roteiro de três refinamentos progressivos para a definição. O ponto de partida é a definição de Gruber (1993) amplamente citada: "*An ontology is an explicit specification of a conceptualization*". A partir dela, Borst (1997) substitui a ênfase ao "explícito" em função do "compartilhado": "*An ontology is a formal specification of a shared conceptualization*". Finalmente, Studer et al. (1998) unificam os dois aspectos em uma única definição: "*An ontology is a formal, explicit specification of a shared conceptualisation*". Os três conceitos chave que esta definição traz são:

(i) **Formal** – este é um consenso entre autores; ontologias precisam ser formais no sentido em que são passíveis de serem interpretadas por máquinas.

(ii) **Especificação explícita** – esta é uma característica que tem ganhado força principalmente com o advento da Web; os conceitos, relações e outros elementos da ontologia devem ser explicitamente declarados.

(iii) **Conceitualização compartilhada** – a noção de conceitualização é introduzida por Gruber (1993) se referindo à abstração que se cria do mundo representado. Usualmente esta abstração é compartilhada por uma comunidade.

Segundo Chandrasekaran (1999), o termo pode ser usado com dois significados. De um lado há um vocabulário e do outro um conjunto de conhecimentos descrito por este vocabulário. O vocabulário usualmente é especializado em um domínio ou contexto relevante. O mais importante na ontologia não está restrita ao vocabulário, mas inclui os conceitos que os termos do vocabulário transmitem.

A partir das definições apresentadas, pode-se concluir a importância de uma ontologia na formalização da estrutura de um conhecimento. Com a sua popularização, ontologias têm sido adotadas como entidades centrais de sistemas de representação do conhecimento de um determinado domínio. Ontologias têm se tornado uma das melhores opções para a

representação de vocabulários neste contexto. Por este motivo, na maioria dos casos, a primeira tarefa a ser executada para projetar um sistema de representação de conhecimento eficiente, e seu vocabulário, é realizar uma análise ontológica eficiente do domínio.

Ontologias têm se tornando essenciais para sistemas de *software* que têm por finalidade a busca ou a combinação/integração de informações provenientes de diversas comunidades. Este é o caso das informações contidas na Web, mais precisamente na Web semântica, pois ontologias podem ser usadas para explicitar a semântica dos documentos e permitir que ela seja usada por aplicações Web e por agentes de *software*. Ao usarmos ontologias em aplicações Web ou habilitarmos agentes de *software* a interpretá-las e processá-las, estamos criando as bases para que futuras aplicações possam atuar na Web de forma mais significativa, no sentido de considerarmos uma capacidade maior de execução de tarefas, em um nível conceitual mais próximo do ser humano (SMITH et al., 2004).

Ontologias de domínio podem ser especialmente úteis na anotação de documentos. Nos casos em que é possível delimitar um domínio para um documento anotado, é possível se associar anotações a uma ontologia de domínio que definirá um contexto, eliminando possíveis ambigüidades relacionadas à termos anotados do documento. A restrição de um domínio também facilita e torna mais eficiente o processo de anotação de documentos.

Neste trabalho desenvolvemos uma técnica para associar de forma transparente documentos a ontologias especializadas conforme o seu domínio, de modo que anotações neste documento possam ser relacionadas às ontologias. Esta técnica se baseia na pré-associação do modelo de documento escolhido pelo usuário com as ontologias específicas, que no seu conjunto formam um *perfil*. Além disso, esquemas de ontologias são utilizados como *template* (formato de saída do material extraído) detalhado no Capítulo 4 - caso de uso, em que é criada a ontologia cartão de pesquisa para representar os dados extraídos dos documentos contendo fichamentos associados à literatura científica, que compõem o *cenário* (informações que se deseja extrair), adaptáveis ao domínio através do conceito de *perfis*.

2.3.3 Linguagens de Representação de Ontologias

2.3.3.1 RDF – Resource Description Framework

Segundo Manola e Miller (2004), RDF é uma linguagem que define um modelo de descrição de recursos na Web. Este modelo consiste em um grafo que pode ser mapeado para diversos padrões sintáticos, sendo o XML aquele utilizado para o contexto da Web. Neste caso, o mapeamento RDF é uma aplicação XML, que serve como uma base para uma representação homogênea de metadados. O seu principal objetivo é o de facilitar o intercâmbio de informações que podem ser interpretadas por máquinas entre aplicativos via Web, possibilitando assim a automatização do processamento de recursos (PITTS-MOULTIS; KIRK, 2004).

O princípio por trás do RDF é fornecer interoperabilidade semântica entre aplicativos que trocam informações na rede. Foi desenvolvida pelo W3C (World Wide Web Consortium) como uma linguagem baseada nos princípios de redes semânticas.

Manola e Miller (2004) destacam, entre outras coisas, uma característica muito importante acerca do RDF. Diferentemente das outras linguagens voltadas para Web como o HTML, em que a informação é formatada para ser apenas exibida para pessoas, RDF destina-se a situações em que esta informação precisa ser processada por outras aplicações. Esta capacidade de publicação e troca de dados entre aplicativos diferentes, passíveis de interpretação fora do contexto original de sua criação, caracteriza o RDF como uma linguagem voltada principalmente para ser interpretada por máquinas.

2.3.3.1.1 Declarações RDF

O mecanismo principal para a representação de informações em RDF se baseia na associação de descrições a recursos através de declarações (MANOLA; MILLER, 2004). Uma declaração (*Statement*) atribui um valor a uma propriedade associada a um recurso. Cada propriedade tem um tipo - definido por um vocabulário RDF - e um valor. O valor de uma propriedade pode ser um literal (tal como uma string), um número, ou uma referência para outro recurso.

Uma declaração é composta de três partes:

Recurso – Qualquer objeto que é identificável unicamente por um URI **–** *Uniform Resource Identifier* (Identificador Uniforme de Recurso). URIs constituem um super-conjunto das URLs **–** *Uniform Resources Locators* (Localizador Uniformes de Recursos), não estando restritas a identificadores associados a localizações físicas **–** como as URLs **–** mas também a identificadores chamados URNs, os quais dependem de mecanismos de resolução de nomes para tratá-los. Os URIs podem também identificar livros, elementos em uma página e pessoas individuais, por exemplo.

Propriedade – É uma característica, um atributo ou um relacionamento específico de um recurso. Cada propriedade expressa um relacionamento do recurso com um valor ou com outro recurso.

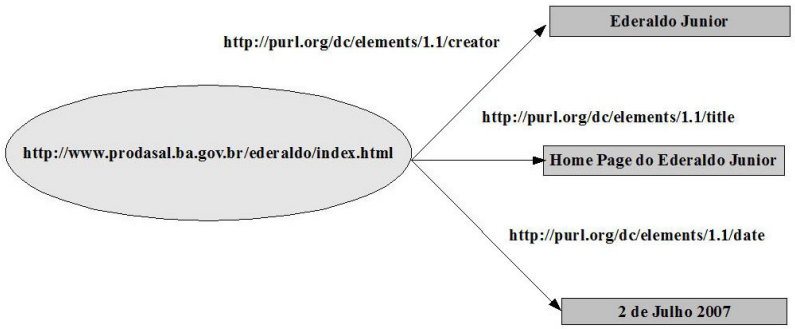
Valor – Deve ser de natureza atômica (string de texto ou número, por exemplo) ou ser uma referência para outro recurso, o qual pode ter suas próprias propriedades/valores.

Uma coleção destas propriedades com os seus respectivos valores, que se referem ao mesmo recurso, é chamada de descrição.

Essas três partes da declaração podem alternativamente ser identificadas como: assunto, predicado e objeto. O recurso descrito é o assunto. A propriedade usada para descrever o recurso é o predicado e o valor da propriedade é o objeto da declaração (MANOLA; MILLER, 2004).

Uma declaração RDF é frequentemente expressa na forma de um Grafo Direcionado e Rotulado (*Directed Labelled Graph* – DLG), em que o recurso e o valor da propriedade são vértices rotulados, enquanto a propriedade é uma aresta rotulada que aponta na direção do valor da propriedade, veja exemplo na Figura 1.

Figura 1- Associação (recurso – propriedade – valor) representada como um DLG



Recursos ou objetos são representados por elipses, enquanto literais são representados por retângulos. O uso de *URIs* é um conceito chave do RDF, ele permite que os elementos representados sejam mais do que simples cadeias de caracteres. Através dos *URIs* cada elemento RDF recebe uma identificação única na Web, possibilitando a construção de uma rede de conceitos e a sua interligação com ontologias diversas. Breitman (2005) afirma que "Grafos são excelentes para transmitir informações entre seres humanos, porém precisamos de uma representação que possa ser processada por máquinas". Por esta razão, o modelo abstrato do RDF é passível de ser mapeado para linguagens como o XML. A Figura 2 ilustra o mapeamento em RDF do DLG da Figura 1 para XML. Trata-se de um mapeamento de um grafo para uma estrutura hierárquica, cuja raiz é o elemento rdf:RDF, contendo um conjunto de descrições subordinadas representadas por elementos rdf:Description.

Figura 2 - RDF do DLG Home Page Ederaldo Junior

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description
 rdf:about="http://www.prodasal.ba.gov.br/ederaldo/index.html">
 <dc:creator>Ederaldo Junior</dc:creator>
 <dc:title>Home Page do Ederaldo Junior</dc:title>
 <dc:date>2 de julho 2007</dc:date>
</rdf:Description>
</rdf:RDF>
```

2.3.3.1.2 RDF Schema

Apesar de o RDF oferecer um conjunto pequeno de elementos predefinidos, ele fornece subsídios para que comunidades independentes possam criar seus próprios vocabulários através do RDF Schema (BREITMAN, 2005), ou seja, novas classes e propriedades particulares ao seu domínio de aplicação.

Segundo Breitman (2005), RDF Schema é um framework para descrever elementos como as classes e propriedades. Estes dois conceitos em RDF são semelhantes àquelas da programação orientada a objetos. A Tabela 2, apresenta um resumo das principais classes do RDF Schema, que são utilizadas para a definição de novas classes, propriedades etc. A classe rdf:Class, por exemplo, atua como uma meta-classe, no sentido em que uma nova classe em RDF será instância de rdf:Class.

Tabela 2 - Classes essenciais do RDF-Schema

rdfs:Resource	Classe que engloba todos os recursos.
rdfs:Class	Classe cuja instância é outra classe (meta-classe).
rdfs:Literal	Classe que engloba todos os literais (cadeias de caracteres).
rdfs:Property	Classe cuja instância é uma propriedade.

Fonte: Desenvolvido pelo autor baseado em Breitman (2005).

No RDF Schema a tag rdfs:subClassof estabelece um relacionamento de herança entre as classes; rdfs:subPropertyof estabelece um relacionamento de herança entre duas propriedades e rdfs:type que estabelece que um recurso é instância de uma classe (MANOLA; MILLER, 2004).

Segundo Breitman (2005), o RDF Schema também permite a indicação de definições relacionadas a termos vocabulário, que podem estar no mesmo arquivo, em outro arquivo, ou em outros lugares na Web. A tag rdfs:seeAlso relaciona um recurso a outro recurso que contém mais informações sobre o primeiro e rdfs:isDefineBy é uma subpropriedade de rdfs:seeAlso, que relaciona um recurso a um local em que sua definição se encontra; esta definição pode estar em RDF.

A linguagem RDF não é direcionada para o consumo humano e sim para ser consumida por computadores. De modo a tornar a informação mais acessível a humanos, o RDF fornece as propriedades rdf:comment, que serve para acrescentar comentários textuais associados a um recurso, e rdf:label – um rótulo que pode associado a um recurso para torná-lo mais significativo para seres humanos. Entre outros objetivos, rdf:label pode ser usado para dar rótulos dos nós dos grafos RDF apresentados a usuários. Tais rótulos podem inclusive ser representados em diversos idiomas.

Quando necessário, na representação de um vocabulário pode-se estabelecer restrições em relação às classes que serão utilizadas como sujeito e como objeto na representação, utilizando para isso as primitivas rdfs:domain e rdfs:range respectivamente (MANOLA; MILLER, 2004).

O RDF é um padrão que pode ser usado como base para a construção de ontologias. As ontologias podem ser expressas mediante diversos formalismos. Um deles foi projetado especificamente sobre o RDF para ser utilizado na Web Semântica; trata-se do OWL – *Web Ontology Language* (SMITH, 2004). OWL é um vocabulário RDF que estende suas capacidades para a definição de relações comumente utilizadas por formalismos voltados à

definição de ontologias. Mas não é a única. Ao longo dos últimos anos foram propostas algumas linguagens para ontologias baseadas em extensões do RDFS. São elas, SHOE, Oil, DAML e DAML+ Oil (HEFLIN et al.,1999).

Para o propósito do nosso trabalho de extração de informações, como veremos no Capítulo 3, OWL será utilizada como formato de saída para o *template* de extração.

2.3.3.2 OWL – Linguagem de Ontologia Web

Segundo Breitman (2005), a OWL é uma linguagem para representar conceitos, seus relacionamentos e axiomas de um determinado domínio, na forma de ontologia. Ela foi criada como uma revisão da linguagem DAML+OIL pelo consórcio W3C, de modo a atender as necessidades das aplicações para Web Semântica.

Atualmente, a OWL conta com duas versões. A primeira – que aqui chamaremos de OWL 1 – teve sua última especificação publicada em 2004 (SMITH et al., 2004) e a segunda – a OWL 2 – tem sua publicação mais recente de 2009 (W3C, 2009). Este trabalho utiliza uma combinação de ambas às especificações da seguinte maneira:

A OWL 2 descreve sua especificação estrutural em UML (MOTIK et al., 2009), portanto, ela foi utilizada como base para a especificação do mapeamento entre os dados extraídos de documentos e ontologias. Foram utilizados apenas aqueles elementos da descrição estrutural que possuem equivalentes em OWL 1 compatível com a OWL DL.

Como a especificação da OWL 2 é retroativamente compatível com a OWL 1, optamos por utilizar neste trabalho a OWL 1 para os aspectos de implementação, por duas razões: (i) o trabalho começou a ser desenvolvido quando a especificação da OWL 2 ainda não tinha sido lançada; (ii) não haviam ferramentas OWL 2 maduras no momento de sua implementação.

A revisão que faremos a seguir diz respeito principalmente à OWL 1, entretanto, apresentaremos o diagrama UML de especificação estrutural da OWL 2, detalhando apenas aqueles elementos que aparecem em ambas.

A linguagem OWL 1 é dividida em três perfis de uso, em ordem crescente de expressividade (SMITH et al., 2004):

- OWL Lite
- OWL DL
- OWL Full

De acordo com Breitman (2005), OWL Lite foi concebida para aqueles casos em que se pretende construir hierarquias de classificação simples, em que não há necessidade da definição de axiomas, nem de se estabelecer relações mais sofisticadas entre os termos. OWL Lite é mais indicado para migração de tesauros e taxonomias para o formato de ontologias.

O OWL Full representa o extremo oposto, em que se alcança o máximo de expressividade do OWL. Entretanto, não é possível se pode garantir que a ontologia resultante é computável e pode decidir inferências em tempo finito. Por esta razão foi criado o OWL DL que impõe restrições ao OWL Full para garantir completude (é computável) e decidibilidade.

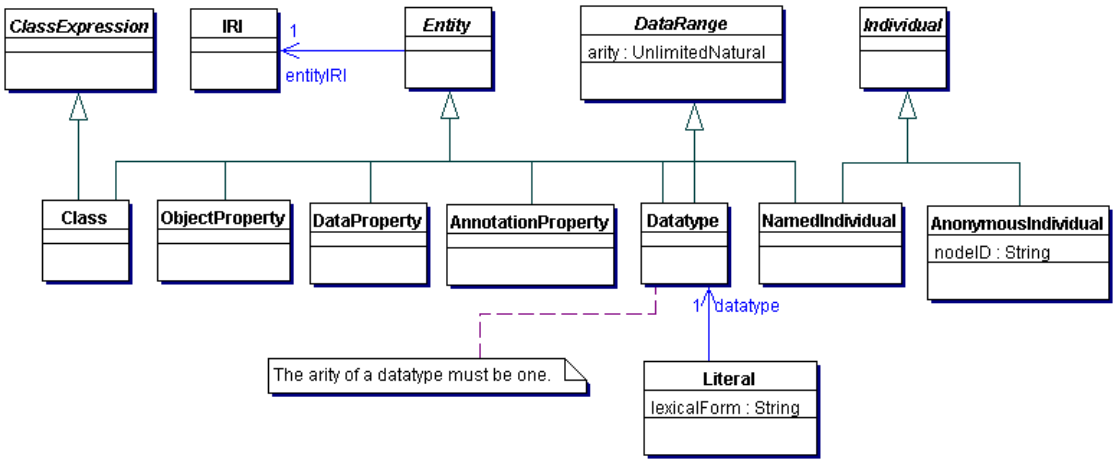
A OWL 2 fez uma revisão dos perfis e os redefiniu em três novos: EL, QL e RL (GOLBREICH; WALLACE, 2009). Seu detalhamento foge do escopo deste trabalho.

A seguir apresentaremos os elementos chave do OWL por meio de um exemplo extraído do guia do OWL (SMITH et al., 2004). Serão retomados alguns elementos apresentados anteriormente em RDF, de modo a apresentar as especificidades do OWL, bem como permitir sua ilustração através do exemplo.

2.3.3.2.1 Diagrama estrutural do OWL 2

A especificação estrutural do OWL 2 é realizada em UML. Para isto foi usado a Meta Object Facility – MOF (OMG, 2010). De um modo simplificado a MOF é uma especificação da OMG que permite a especificação de meta-modelos usando a diagramação do UML, ou seja, a partir dos diagramas UML é possível especificar como linguagens definem seus modelos. A especificação OWL 2 utilizou este recurso. A **Erro! Fonte de referência não encontrada.** apresenta um dos diagramas estruturais do OWL 2 com ênfase em entidades (MOTIK et al., 2009). Nas subseções a seguir detalharemos apenas alguns elementos deste diagrama, que serão usados neste trabalho.

Figura 3 - Diagrama estrutural do OWL 2 com ênfase nas entidades



Fonte: Motik et al. (2009).

2.3.3.2.2 Entidade

Entidade (*Entity* no diagrama) é um elemento primário do OWL do qual deriva grande parte de sua estrutura. Como está ilustrado no diagrama, cada entidade é um elemento que pode ser identificado por uma IRI. O conceito de IRI estende o de URI, apresentado anteriormente, acrescentando recursos de internacionalização.

2.3.3.2.3 Classes

Tal como o RDF, o OWL estabelece um mecanismo para a declaração de classes (*Class* no diagrama). A classe é uma entidade que define o mecanismo fundamental para agrupar e classificar conceitos básicos de um domínio, que vão servir de base para a construção de taxonomias. Em OWL cada individuo pertence implicitamente a uma classe genérica owl:Thing. Então toda classe genérica definida pelos usuários será uma subclasse da classe owl:Thing.

Como ilustra a Figura 4, em seu mapeamento para RDF as classes em OWL são definidas por owl:Class:

Figura 4 - Ontologia Wine

```
<owl:Class rdf:ID="Vintage">
<owl:Class rdf:ID="WineGrape">
<owl:Class rdf:ID="Wine">
   <rdfs:subClassOf
rdf:resource="&food;PotableLiquid" />
</owl:Class>
```

Fonte: Smith (2004).

As três primeiras linhas o código da Figura 4 definem classes relacionadas a tipos de vinhos: Vintage, WineGrape e Wine respectivamente. Em seguida, a classe Wine é apresentada como herdeira de PotableLiquid (líquido potável). Note que, se as classes Vintage e WineGrape não forem explicitamente associadas como herdeiras de nenhuma outra classe, elas serão implicitamente herdeiras de owl:Thing.

A OWL 2 define duas estratégias de mapeamento para XML. Uma baseada em RDF/XML que é compatível com o OWL 1 e outra diretamente para XML a partir do seu modelo estrutural. Neste trabalho será usado a estratégia de mapeamento para RDF/XML.

2.3.3.2.4 Indivíduos

A definição de indivíduos (*Individual* no diagrama) é um recurso da OWL que possibilita categorizar objetos do mundo em classes, bem como o seu relacionamento com outros indivíduos, utilizando-se propriedades. A declaração de um indivíduo em OWL se dá através da sua identificação (IRI) e associação como membro de uma classe. Veja exemplo na Figura 5 da declaração de um indivíduo em RDF/XML. A propriedade `rdf:type` indica que o indivídio Almadem (vinho Almadém) pertence à classe `Wine` (vinho). Indivíduos que não são associados explicitamente a classes, são implicitamente da classe owl:Thing.

Figura 5 - Ontologia Wine

```
<owl:Thing rdf:about="#Almadem">
   <rdf:type rdf:resource="#Wine"/>
</owl:Thing>
```

Fonte: Smith et al. (2004).

2.3.3.2.5 Propriedades

A declaração de propriedades é utilizada para sistematizar características a ser descritas em todos os elementos de uma classe. Por exemplo, todos os vinhos têm safra. Cada indivíduo da classe dá um valor à propriedade. Por exemplo, o vinho Almadem (exemplo anterior) pode ter valor de safra: 1995.

Em OWL as propriedades podem ser de dois tipos:

- **Object Property** – relaciona objetos de duas classes. No exemplo apresentado na Figura 6 em RDF/XML, a *object property* hascolor (tem cor) associa uma instância da classe vinho a uma instância da classe cor.

  Exemplo:

  Figura 6 - Ontologia Wine

```
<owl:ObjectProperty rdf:about="#hascolor">
<rdfs:domain rdf:resource="#Wine"/>
<rdfs:range rdf:resource="#color"/>
</owl:ObjectProperty>
```

  Fonte: Smith et al. (2004).

- **Data Property** – relacionamento entre instâncias de classes e literais expressos em XML Schema. Referências a esses datatypes devem ser feitas indicando-se a IRI base do XML Schema: http://www.w3.org/2001/XMLSchema. Em OWL 1 estas propriedades são chamadas de DatatypeProperty e em OWL 2 de DataProperty (vide diagrama na **Erro! Fonte de referência não encontrada.**). Em OWL 2 o valor deste tipo de propriedade é definido como Literal que é uma especialização de Datatype, conforme indicado no diagrama. Veja exemplo abaixo na Figura 7 em RDF/XML, em que a *datatype property* WineName (nome do vinho) é definida como literal String.

  Figura 7 - Ontologia Wine

```
<owl:DatatypeProperty rdf:about="#WineName">
<rdfs:domain rdf:resource="#Wine"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:ObjectProperty>
```

  xsd é uma abreviatura de uma URI de um XML Schema que define formatos de dados válidos para arquivos XML. String é um tipo primitivo

  Fonte: Smith et al. (2004).

Além destes dois tipos de propriedade, o OWL define o qualificador AnnotationProperty para propriedades usadas em anotação.

2.4 ANOTAÇÃO

2.4.1 Conceito

Segundo Ramalho (2000), no contexto digital anotação é a tarefa de inserir toda a espécie de códigos em textos eletrônicos, tendo como objetivo orientar sua formatação, impressão ou outro tipo de processamento, tornando explícita a sua interpretação.

A anotação de um documento necessita obedecer a certas regras pré-estabelecidas. Pode ser materializadas por meio de comentários, marcações no texto ou registro de interpretações pessoais (MARSHALL, 1997).

2.4.2 Classificação

Na literatura, encontram-se várias classificações associadas ao tipo de anotação. Para Ramalho (2000), anotação se divide em procedimental e descritiva. Anotação procedimental tem como foco a estética do documento, em que as marcações no documento servem para orientar sua formatação e impressão. No segundo caso, o foco das anotações feitas no documento está em explicitar semanticamente a estrutura do texto, indicando como o documento é composto. Esse tipo de anotação tem diversas vantagens:

- torna-se mais fácil a portabilidade do documento entre plataformas diferentes;
- funções distintas podem ser aplicadas às anotações, a depender do contexto;
- programas diferentes podem utilizar apenas os componentes anotados que lhe interessam, por exemplo, um programa de cadastro pode extrair apenas dados de autor, endereço, etc. contido em documentos para alimentar um banco de dados.

A semântica associada aos elementos, fornecida pela anotação descritiva, é a base para a exploração de semântica de documentos para propósitos que vão muito além de formatação e descrição de estrutura do texto. Por esta razão, a proposta deste projeto envolve um padrão de

anotação que exige um deslocamento da prática de anotação procedimental para a descritiva, que será associada a padrões abertos da Web Semântica.

Marshall (1997) apresenta duas classificações para o processo de anotação. No primeiro caso, associada com a capacidade de interpretação da anotação em implícita e explícita e no segundo, a classificação se dá de acordo com o destino da anotação em formal e informal.

Uma anotação é classificada como implícita quando ela é feita para ser interpretada somente pelo seu autor, enquanto que a anotação explícita é passível de interpretação por diversas pessoas ou mesmo máquinas, apresentando um alto nível de semântica associada.

Uma anotação quando destinada a ser utilizada apenas por pessoas é dita informal. Quando o processo de anotação segue esquemas estruturais e é destinada para ser consumida tanto por pessoas quanto por máquinas, ela é dita formal.

2.4.3 Anotação Semântica

Segundo Popov (2003), a anotação semântica segue um "esquema específico para geração e uso de metadados". O esquema refere-se a entidades nomeadas que podem ser anotadas (e.g., pessoas, localizações ou organizações) e que são referenciadas pelo nome em documentos de texto. No contexto da Web Semântica, Euzenat (2002) estabelece que a anotação tem que ser representada em uma linguagem formal e vinculada ao documento.

Uma vez identificadas as entidades nomeadas no documento, a anotação semântica consiste em interligá-las com as suas descrições semânticas representadas num modelo formal como, por exemplo, uma ontologia. Sendo uma ontologia no padrão RDF/OWL, a identificação da entidade é feita através de uma *URI*, o que permite a ligação com a sua descrição no repositório semântico.

Nesse trabalho, é utilizado um esquema de anotação do tipo descritivo, pois foi preciso identificar semanticamente partes do texto de interesse a serem extraídas; explícita, pois é reconhecida por máquinas no processo de extração; formal, pois a semântica é acessível a máquinas e as anotações passíveis de serem associadas a ontologias.

2.4.4 Semântica In Loco

Semântica In Loco (SANTANCHÈ, 2007) pode ser definida como uma estratégia para a produção de anotações associadas a recursos digitais com semântica inter-operável, que seguem os seguintes princípios (SANTANCHÈ; SILVA, 2010): **(i) Anotação *In Loco*:** o processo de anotação acontece concomitantemente à produção de conteúdo (*in loco*); **(ii) Integração de Metáforas:** as metáforas e modelos utilizados para anotação do conteúdo estão alinhadas com aquelas utilizadas para a produção do mesmo; **(iii) Interoperabilidade:** as estratégias de anotação *in loco* são projetadas de modo que estas anotações possam ser automaticamente extraídas do conteúdo e convertidas para padrões abertos de representação de metadados da Web Semântica; **(iv) Persistência Semântica:** os elementos da anotação in loco são conectados a ontologias unificadoras, as quais irão garantir uma interpretação equivalente em diferentes contextos, fornecendo uma persistência semântica ao longo das transformações.

Uma premissa fundamental da anotação in loco é a compreensão de que qualquer conteúdo digital produzido tem necessariamente uma semântica a ele associada. Os padrões utilizados para representar as anotações, tal como o OWL, não têm o propósito de acrescentar semântica onde não havia, mas apenas representar tais anotações com uma semântica padronizada e extensível.

A implantação da semântica in loco é dividida nas seguintes etapas: (i) seleção dos metadados que serão utilizados para anotar o conteúdo digital; (ii) definição de um padrão para anotação do conteúdo à medida que é produzido, para que possa ser identificado automaticamente por uma ferramenta de extração; (iii) construção de uma ferramenta de extração e conversão.

Trata-se de uma metodologia de anotação aplicável a diversos tipos de conteúdo e apta a ser utilizada por diversas ferramentas de produção de conteúdo, uma vez que se baseia em padrões de anotação, ao invés de formatos ou *plug-ins* específicos. Neste trabalho utilizamos um subconjunto da semântica in loco aplicável a documentos e processadores de texto. Conforme será apresentado posteriormente neste trabalho, a etapa (i) da Semântica in Loco é realizada mediante a seleção de metadados que descrevem informações sobre a autoria e o conteúdo do documento, base para a extração das informações (autor, título, subtítulo, assunto, etc.); a etapa (ii) é baseada no uso de estilos como técnica de anotação; na etapa (iii) será desenvolvido um protótipo para extrair as anotações e associá-las com a ontologia de

domínio “Cartão de Pesquisa”. Deste modo, classificamos as anotações adotadas neste trabalho como anotações semânticas.

2.5 EXTRAÇÃO DE INFORMAÇÃO EM BASES TEXTUAIS

2.5.1 Origem da Extração de Informação (IE)

A ideia de extrair informações relevantes para o usuário, tendo como base documentos no formato texto não é nova. Em sua obra, Cowie (1996) apresenta algumas iniciativas pioneiras entre a década de 50 a 80, veja levantamento na Tabela 3. Na década de 90, alguns acontecimentos como o surgimento da Web foram decisivos para a definição do termo Extração de Informação como ele é utilizado no contexto atual. Com o surgimento da Web, a quantidade de informações disponibilizadas principalmente no formato texto cresceu de forma exponencial, embora a capacidade humana de consumo dessas informações permanece a mesma. Segundo Crespo et al. (1994), a capacidade do ser humano de processar grandes quantidades de informação é bastante limitada. Durante uma série de conferências de interpretação de mensagens que ficaram conhecidas por MUC (KAUFMANN, 1992) (KAUFMANN, 1994) (GRISHMAN, 1996), desenvolvidas e patrocinadas pelo DARPA – Agência de Projeto e Pesquisa Avançada dos Estados Unidos –, que foi usada à expressão Extração de Informação – IE.

Tabela 3 - Algumas iniciativas pioneiras de Extração de Informação antes da década de 90

Sistema de IE	Autor	Data
Projeto dirigido pelo Linguistic String Project group na New York University. Convertia resumos de boletins de pacientes (preenchidos em inglês) para uma forma adequada para utilização como entrada num SGDB da CODASYL (Conference on Data Systems Languages).	Naomi Sager	1950
Extrator de informações de relatórios de vôos produzidos por operação de satélite ao redor do mundo.	daSilva e Dwiggins	1980
Sistema de extração de estruturas canônicas de um guia de campo de descrições das plantas e dos animais. O sistema utilizava informações simples para preencher uma estrutura fixa de registro.	Cowie IE	1981
O programa experimental FRUMP (DEJONG, 1982) trabalha a partir de manchetes de jornal (sequestros, atos de terrorismo, as negociações diplomáticas, etc.). Através dessas manchetes, procura palavras relacionadas com o seu assunto, sendo capaz de prever ou esperar a ocorrência de outras palavras ou frases e assim construir o esboço da história. Assim, só são extraídas, as partes dos textos que estão diretamente relacionados com elementos da manchete, o resto do texto é ignorado.	Gerald deJong	1982

Fonte: Desenvolvido pelo autor com base em Cowie (2005)

2.5.2 Conceito de Extração de Informação

Provavelmente a maior parte das informações produzidas por seres humanos (texto, documentos, páginas Web, etc.) é escrita em linguagem natural. Na Internet, uma grande quantidade de informações semiestruturadas produzidas por máquinas é dirigida ao consumo humano. Neste contexto, a Extração de Informação (*Information Extraction* – IE) tem um papel importante. IE pode ser visto como um processo que parte de uma entrada de texto livre ou um conteúdo semiestruturado, gerando saídas que são dados estruturados de interesse, representados, por exemplo, por k-tuplas ou objetos complexos (CHANG, 2006). IE pode ser utilizada por muitas razões como, por exemplo, para fins de indexação em recuperação de

informações *(Information Retrieval* - IR*)* de aplicações para a Internet, usual em motores de busca como o Google (CUNNINGHAM, 2005).

Apesar de terem funcionalidades fortemente relacionadas, IE é muito diferente de IR:

- Um sistema IR encontra textos relevantes e os apresenta para o usuário;
- Uma aplicação IE analisa textos e extrai apenas a informação específica de interesse para o usuário e as devolve num formato definido pelo mesmo.

Crespo et al. (1994) apresentam algumas abordagens que surgiram do conjunto de trabalhos dos últimos anos, voltadas para recuperar e extrair informações disponíveis na Internet e outros sistemas distribuídos. De um modo geral, o que se pode observar é que quando se faz uma análise tanto das abordagens de IR como de IE, mesmo que estas tenham propósitos diferentes, há um elemento comum que pode melhorar de forma significativa o desempenho de ambas: é o conhecimento da estrutura do texto.

Na Web, principalmente nas grandes bibliotecas digitais como, por exemplo, ACM ou SCIELO, a organização estrutural, bem como a semântica dos componentes da maioria das fontes de texto apresentadas não é explícita. Estes sistemas foram criados para a apresentação de dados direcionados ao consumo por pessoas, em que sua estrutura é descrita por marcações associadas a layout de apresentação, sem conotação semântica relacionada ao conteúdo em si. O tipo de anotação predominante nesses casos é a procedimental, conforme foi apresentado anteriormente.

Esse desconhecimento da organização estrutural do texto é o principal fator limitante tanto no processo de IR como no de IE. Fontes de texto deste tipo, em que sua estrutura é representada implicitamente, são tratadas por autores sob diversas nomenclaturas: desestruturada, parcialmente estruturada, ou implicitamente estruturada (ABITEBOUL, 1997). Várias soluções são desenvolvidas para fontes de texto desestruturadas. Para isto, várias técnicas acessórias, como tratamento linguístico, análise de imagem, gráfico, áudio e vídeo, são utilizadas no intuito de dar suporte à localização de documentos em grandes volumes (CRESPO et al., 1994).

No caso especifico da extração de informações, outras técnicas auxiliares são utilizadas para se inferir a semântica implícita e selecionar dados mais relevantes, dentre elas Fuhr (1995) destaca em sua obra: estatísticas e informações de contexto provenientes de bases de conhecimento, *thesaurus* ou inferidos por princípios gramaticais de linguagem natural.

Segundo Dreilinger (1997), a base de ação da maioria dos mecanismos que utilizam as técnicas citadas anteriormente são usualmente mecanismos de pesquisa Web.

Mesmo diante de tanto esforço, estas técnicas, segundo Hahn (1998), apresentam resultados satisfatórios quando aplicadas a domínios específicos – tal como o de informações tecnológicas, persistindo ainda desafios em aberto, para se alcançar resultados consistentes e escaláveis em domínios mais amplos.

Segundo Klein e Frankhauser (1997), para bases de texto genéricas o acesso a fontes de dados estruturadas é o que tem apresentado resultados satisfatórios e escalonáveis para grandes aplicações. Para isso torna-se essencial a extração da estrutura lógica destes documentos, quando esta encontra-se implícita. Dentro dessa categoria, duas abordagens bem definidas se destacam (CRESPO et al., 1994). A primeira pode ser considerada mais "estática", pois envolve a confecção de algoritmos de extração, transformação e filtragem sob medida para cada categoria de documentos, com a assessoria de um especialista de domínio. A segunda abordagem, mais dinâmica, envolve o desenvolvimento de sistemas capazes de lidar com ligeiras variações de estruturas e evoluções de documentos. Para o processo de adaptação contam com o aprendizado de máquina. Algoritmos que envolvem aprendizagem nem sempre são eficientes e podem ter problemas para lidar com estruturas complexas. Por essa razão, a intervenção humana usualmente é necessária, tornando o processo semiautomático. Tal mixagem demonstrou ser uma das melhores para o tratamento de estruturas implícitas de informações em forma de texto.

Extração de informação baseada em Ontologia (OBIE) é uma modalidade específica da IE, que adota uma ontologia como um dos recursos do sistema. Ontologias podem ter muitos papéis nesse processo, mas dois são essenciais: ajudar na identificação de instâncias da ontologia que podem ser extraídas do conteúdo original, e apoiar na atualização automática da ontologia com novas instâncias extraídas do documento (DAVIES, 2007). Enquanto sistemas de IE convencionais geralmente são baseados em um procedimento padrão de correspondência, que segue um conjunto fixo de regras extração (DAVIES, 2007), sistemas de OBIE podem conseguir melhores resultados adotando ontologias ao invés de estruturas semânticas fixas formadas, por exemplo, de elementos léxicos (MAEDCHE, 2003) (MAYNARD, 2005).

2.5.3 Fatores que influenciam na escolha da solução do problema de extração

Devido a grande variedade no formato das fontes de dados textuais (formato livre, semiestruturado ou estruturado), as suas propriedades têm uma forte influência na escolha da solução para o problema de extração. Outro fator preponderante para esta escolha está relacionado às propriedades intrínsecas ao próprio processo de extração. Na sua obra, Crespo et al. (1994) destaca a importâncias desses dois fatores no processo de extração, reunindo um conjunto dessas propriedades em taxonomias, das quais as mais importantes para esse trabalho estão resumidas na Tabela 4.

Tabela 4 - Resumo das propriedades da fonte de dados e do processo de extração relevantes para o problema de Extração de Informação

Nome da Propriedade	Relacionamento		Significado	Classificação
	Fonte de Dados	Processo de Extração		
Estrutura	X		Classifica a fonte de texto de acordo com a formatação física e a presença de marcação sintática ou semântica na sua organização.	• livre de marcação; • marcação Sintática; • marcação Semântica.
Topologia	X		Trata forma de como é composta a fonte de texto.	• centralizada (em que as partes que compõe a fonte estão todas reunidas de forma centralizada com um livro); • descentralizada (formato é semelhante ao site Web, resultante da integração de peças distribuídas e linkadas).
Estabilidade	X		Determina o quanto uma fonte de texto é suscetível a mudanças na sua forma e ou conteúdo.	• mudança de conteúdo; • mudança na forma; • não sofre mudança.
Automação		X	Trata do quanto o processo de extração de informações depende da intervenção humana.	• manual (regras de extração são especificadas manualmente); • semiautomática (existe algum nível de

				automação como uma ferramenta de assistência, mas é necessária a intervenção humana); • automática (feita pelo computador).

Fonte: Desenvolvido pelo autor com base em Crespo et al. (1994)

Nesse trabalho o principal fator influenciador na escolha da solução do problema de extração está relacionado diretamente às propriedades do processo de extração. A propriedade automação define o quanto da participação do usuário será necessário dentro do processo de extração de informação (CRESPO et al., 1994). Devido à importância dessa propriedade, segue abaixo detalhamento de cada um de seus níveis.

2.5.3.1 Extração Manual

Segundo Crespo et al. (1994), na abordagem manual a participação do usuário, principalmente do usuário especialista, é indispensável, pois ele é responsável pela descrição, em uma linguagem de alto nível, das regras necessárias para o processo de extração. Segundo o mesmo autor, a abordagem manual se baseia na escrita de regras de extração de informação, que são baseadas na análise de similaridades capturadas a partir de rotinas de extração específicas, já utilizadas em outros processos similares, mas não equivalentes.

Mesmo tendo a desvantagem da forte dependência da presença do usuário na condução da extração manual e também, na maioria dos casos, ser um processo complexo, trabalhoso, demorado e sujeita a erros, esse processo apresenta grandes vantagens quando a quantidade de fontes de texto de que se deseja extrair informações são pequenas e estáveis. Segundo Crespo et al. (1994), como as soluções manuais são muito específicas, geralmente seus extratores são muito eficientes e rápidos. Além disso, o controle completo sobre o processo de extração permite o programador lidar com estruturas complicadas, que podem ser muito difícil de ser definidas utilizando a modalidade semiautomática ou automática.

2.5.3.2 Extração Semiautomática

Enquanto a quantidade de fontes de texto para serem extraídas informações for pequena e estável, ou seja, pouco suscetível a mudanças no seu conteúdo e forma, o processo de extração manual, mesmo sendo complexo e trabalhoso, resolve o problema. Mas usualmente a quantidade de fontes de texto tende a crescer de forma exponencial e cada vez mais se apresentar menos estável. Dentro desse contexto, uma solução menos trabalhosa e mais produtiva tornou-se necessária. A extração semiautomática se apresenta como uma evolução natural da extração manual, através da automação de parte do seu processo, utilizando-se para isso algoritmos de aprendizagem.

Segundo Crespo et al. (1994), três fases caracterizam tais algoritmos: treinamento, processamento e retorno. A fonte de texto é submetida a esse algoritmo, em que cada fase tem tarefas específicas bem determinadas, gerando saídas que servem de entradas para a fase subsequente. O processo de extração semiautomático é cíclico e interativo. Da mesma forma que existe o treinamento positivo, há o treinamento negativo, tratando dos elementos que não devem ser extraídos no processo semiautomático. O usuário fornece informação de retorno ao algoritmo em cada ciclo, refinando o processo até atingir o nível de satisfação desejado. A Tabela 5 apresenta um detalhamento do funcionamento do algoritmo de aprendizagem, apresentando cada fase com as suas respectivas entradas e saídas.

Tabela 5 - Resumo das fases de um algoritmo de aprendizagem utilizado em uma extração semiautomática

Fases	Atividades	Entradas	Saídas
Treinamento	• reconhecimento de padrões; • montagem de estratégia de extração.	• conjunto de amostras adequadas para a solução da extração.	• conjunto de estratégias de extração.
Processamento	• aplicação do conjunto de estratégias de extração criadas na fase de treinamento em um novo conjunto de amostras.	• conjunto de estratégias de extração; • novas amostras de texto.	• novo Conjunto de estratégias de extração
Retorno	• fornecer *feedback* do usuário ao algoritmo relacionado a qualidade do processo de extração.	• feedback do usuário.	• ordem para finalizar o processo ou continuar o processo para refinamento.

Fonte: Desenvolvido pelo autor com base em Crespo et al. (1994)

2.5.3.3 Extração Automática

Segundo Grishman (2003), a extração automática é composta de duas fases bem definidas e interdependentes. Na primeira fase, é feita uma extração preliminar de fatos individuais de informações relevantes dispersas na fonte de texto, que servirão como base para a definição da estratégia de extração. Na segunda fase, de posse do resultado obtido na primeira fase, os dados são extraídos utilizando-se de inferência ou de informações fruto da composição dos fatos individuais previamente analisados. Nessa fase também é definido um determinado formato de saída.

Em seus estudos, Grishman (2003) destaca dois conceitos de fundamental importância na compreensão do processo de extração de informação, estabelecidos pela *Message Understanding Conferences (MUC*). *Cenários* são a especificação dos eventos particulares ou relações a ser extraídos e *Template* formato final de saída da extração de informação, geralmente em formato tabular, podendo assumir outros formatos também.

O processo de extração automática é baseado na inferência de padrões de entrada, que serão aplicados nas instâncias candidatas. Grishman (2003) destacou ainda que, uma vez estruturada a entrada, são identificados os vários níveis dos componentes e suas relações. A partir deles são abstraídos os padrões em termos desses componentes e relações. Este processo começa tipicamente com uma análise léxica e reconhecimento de nomes (identificação de nomes e outras estruturas léxicas especiais, como datas, expressões correntes, etc.). Em seguida é aplicada a analise sintática, ou algum tipo de análise parcial para identificar grupos substantivos, grupos verbais e possíveis estruturas complementares.

Para finalizar o processo de extração, ainda resta a fase de integração e a fase de avaliação dos resultados. A fase de integração resolve relações de correferências e examina e combina os fatos do documento inteiro, enquanto que e a fase de avaliação dos resultados extraídos, verifica qualidade do processo de extração através dos seus indicadores. Segundo Grishman (2003), essa avaliação é feita comparando o *template* resultante da extração, com o *template* teste gerado utilizando o corpo de treinamento, através de um gabarito de respostas chaves deste corpo de treinamento. Os resultados primários são *precision* e *recall.* N_{chave} é o número total de campos do gabarito, $N_{resposta}$ é o número total de campos preenchidos no gabarito do sistema (certos ou errados), e $N_{correta}$ é o número de campos preenchidos corretamente na resposta do sistema (o número de resposta de acordo com a resposta chave). Então ***precision*** $= \boldsymbol{N_{correta} / N_{resposta}}$ e recall $= \boldsymbol{N_{correta}} / N_{chave}$.

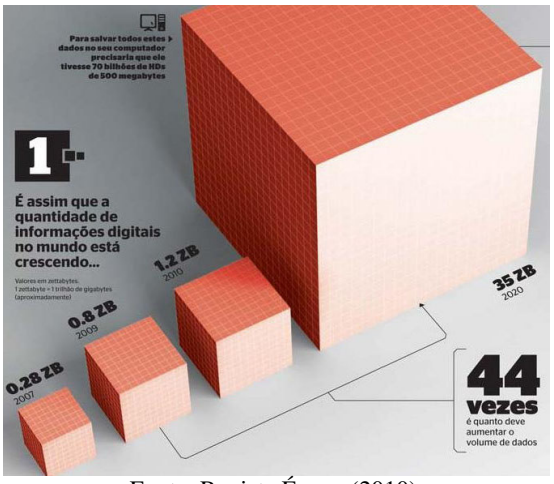
O grande obstáculo à utilização generalizada de sistemas de IE Automática, segundo Grishman (2003), é a limitação do seu desempenho. Segundo Kaufmann (1995), em uma análise feita entre nove sistemas utilizados nas últimas Conferencias de entendimento de mensagens – MUC, cinco dos nove sistemas receberam pontuação na faixa de 51 à 56 (refletindo um *recall* de 43 a 50% e uma *precision* de 59 a 70%)

2.5.4 Por que o Interesse em Extração de Informação

O aumento exponencial da quantidade de informações produzidas e disponibilizadas na sociedade, nos mais diversos formatos, principalmente na forma de texto, é uma tendência mundial, que só tende a se agravar. Segundo Vergotti (2010), se o ritmo atual do crescimento da informação digital no mundo for mantido no atual patamar, chegaremos em 2020 à marca de 35 trilhões de gigabytes de dados armazenados, veja exemplo na Figura 11. Essa marca traz alguns desafios que precisam ser enfrentados de forma definitiva por uma questão de sobrevivência.

Como lidar com a recuperação de documentos de forma eficiente, em se tratando de um universo tão grande de procura? Mesmo resolvido esse primeiro desafio, como a quantidade de documentos produzidos e publicados tendem a continuar crescendo, como lidar com a extração e síntese de informações relevantes desses documentos recuperados, tendo em vista que segundo Crespo et al. (1994), a capacidade do ser humano de processar grandes quantidades de informação é bastante limitada? A resposta para esses desafios é dotar o cidadão dessa nova sociedade digital com ferramentas como a de Extração de Informação – IE e a de Recuperação de Informação – IR, para que a sua capacidade de consumo de informações seja nivelada a atual tendência de aumento. É em face dessa nova realidade que se insere o atual interesse por essa área da informática.

Figura 8 - A evolução do crescimento da quantidade de informações digitais no mundo



Fonte: Revista Época (2010).

2.5.5 Cenário de Uso da Extração de Informação

Levando em conta as vantagens e desvantagens das diversas abordagens relativas ao nível de automação na extração de informações, se faz necessário definir uma estratégia de seleção de qual ferramenta é mais adequado para um determinado tipo de fonte de dados, em relação ao seu nível de automação. Na Tabela 6 abaixo, baseado nos estudos publicados por: Appelt (1999), Cowie (1996), Turmo (2006), e Crespo (1994), apresentamos nossa contribuição definido um cenário de uso adequado para as diversas abordagens de IE, de acordo com seu nível de automação e complexidade da lógica de extração.

Tabela 6 - Cenário de Uso de Ferramentas de Extração de Informações em relação ao tamanho da Fonte de Dados e o nível de complexidade da formulação das regras de extração

Complexidade / Tamanho	Simples	Complexa
Pequena	Como não há grande dificuldade de gerar as regras de extração de forma manual, então a melhor opção será uma ferramenta com nível de automação manual.	A melhor opção será uma ferramenta de extração manual, pois não há material suficiente para a criação de uma base de treinamento para um algoritmo de aprendizagem e o custo relativo não compensa, devido ao tamanho da fonte de dados e a dificuldade de mapear uma fonte de dados complexa para um padrão de extração, o uso de uma ferramenta de extração automática torna-se inviável.
Grande	Caso ela seja relativa a um determinado contexto, a melhor opção será uma ferramenta de extração automática, pois teremos material suficiente para a criação de uma base de treinamento para um algoritmo de aprendizagem e o custo relativo compensa, devido ao tamanho da base.	Apesar de termos uma base com uma quantidade de material, suficiente para uma abordagem com algoritmo de aprendizagem, como existem grandes dificuldades de se mapear essa lógica para um padrão de extração, já que a fonte de dados é complexa, então a melhor opção será uma ferramenta com nível de automação manual.

Fonte: Desenvolvido pelo autor com base em Appelt (1999), Cowie (1996), Turmo (2006) e Crespo (1994)

O nível de automação da ferramenta de extração interfere na forma como a ferramenta vai ser codificada e customizada. Uma maneira de se melhorar isto é trabalhar com a sintaxe ou adicionar semântica a fonte de dados, de modo a facilitar, de alguma forma, o processo de extração de informações relevantes.

Nesse trabalho, desenvolvemos um modelo centrado na ideia de que a melhor maneira de se facilitar a extração dos dados relevantes de uma fonte de dados é adicionar semântica no momento de sua criação, de modo que possamos extraí-las e adaptá-las a novos contextos através de perfis semânticos adaptáveis. Esta lógica não é aplicável a fontes de dados legadas e cujo tamanho seja relativamente grande. Mas para fonte de dados, mesmo que legadas, até

médio porte, ou novas fontes de dados, será mais fácil e eficaz utilizar o modelo proposto de Extração de Informações de Bases Textuais Implicitamente Marcadas e Perfis Adaptáveis.

2.6 RESUMO DO CAPÍTULO

Nesse capítulo, foi feito um resumo de todas as tecnologias necessárias para o entendimento da proposta desta pesquisa, que consiste na extração de informações de bases textuais, implicitamente marcadas, através de perfis semânticos adaptáveis. Os metadados, o esquema Dublin Core, além da metodologia de anotação Semântica in Loco, fornecem os conceitos básicos para a primeira etapa do processo a ser apresentado que é a anotação. Já os conceitos de ontologia e extração de informações balizam a segunda parte do processo que é o IE propriamente dito.

3 EXTRAÇÃO DE INFORMAÇÕES DE FONTES IMPLICITAMENTE MARCADAS

Conforme apresentado no início desta dissertação, este trabalho propõe uma solução de anotação e extração de informações baseada na metodologia para anotações denominada Semântica In Loco, configurável de acordo com o domínio do documento, através do conceito de perfis associados às ontologias, mas para esta finalidade a metodologia Semântica In Loco foi estendida, como veremos neste capítulo.

3.1 MODELO ABSTRATO

Originalmente a metodologia Semântica In Loco realizava a descrição do mapeamento diretamente de padrões de anotação para as ontologias. Como resultado, a descrição tanto do modelo de anotação nas fontes de dados de origem, quanto do mapeamento, eram dependentes de arquiteturas e tipos de documentos específicos, tendo baixa portabilidade.

Uma contribuição desta dissertação é a criação de um modelo abstrato tanto para descrever as fontes de dados, quanto os mapeamentos a serem realizados para ontologias. Tal como o OWL 2, adotamos o UML para realizar a descrição do modelo. Entre outras vantagens isto facilita a descrição de como os itens anotados serão mapeados para a ontologia.

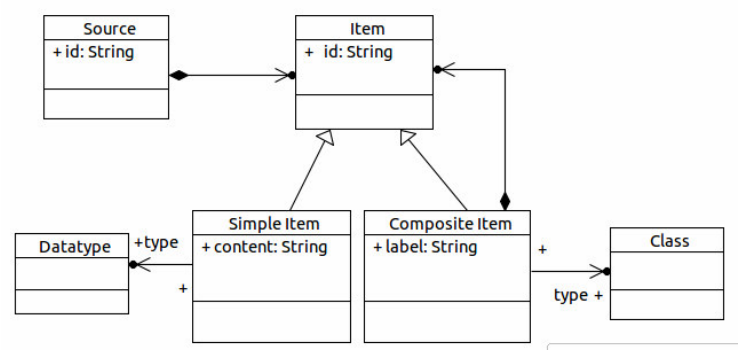
Fundamentalmente o trabalho aqui apresentado pode ser abstraído como um mapeamento entre dois modelos. De um lado está o documento, com itens de dados anotados a serem extraídos e do outro uma ontologia, para onde serão mapeados estes dados. Além destes dois modelos, é necessário se representar também de forma abstrata como será realizado o mapeamento.

Para o modelo de destino adotaremos aquele do OWL 2 apresentado anteriormente. As subseções seguintes descrevem, deste modo, o modelo a ser adotado para a descrição da fonte de dados de extração (documento de origem) e o modelo do mapeamento.

3.1.1 Modelo Abstrato da Fonte de Dados de Origem

O diagrama da **Erro! Fonte de referência não encontrada.** ilustra o diagrama do modelo de dados da fonte de origem. Ainda que a fonte de origem de extração neste trabalho seja sempre composta de documentos texto, o modelo foi concebido de forma genérica, a ser usado pela Semântica In Loco em outros contextos, que atualmente abrange também planilhas eletrônicas e slides.

Figura 9 - Modelo Abstrato de Fonte de Dados de Origem



Como acontece em qualquer processo de extração, os dados a serem recuperados constituem um subconjunto da fonte de dados de origem. A classe `Source` representa genericamente este subconjunto, em que são representadas apenas aquelas partes que serão recuperadas no processo de extração de dados. A fonte é identificada de forma única pelo `id`.

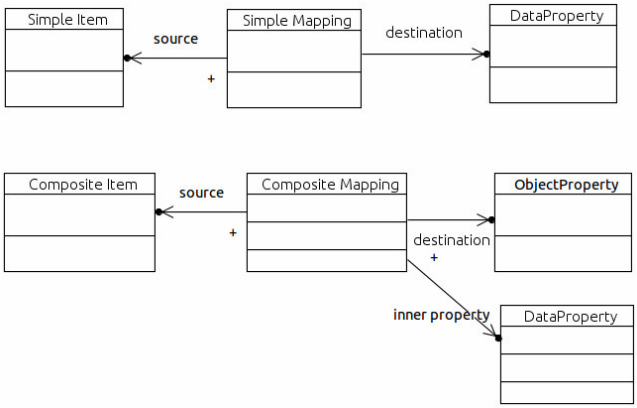
Na Semântica In Loco tal subconjunto de extração é implicitamente identificado através dos padrões de anotação, como será detalhado adiante. Cada parte da fonte de dados delimitada por uma anotação é denominada neste modelo de `item`. Consequentemente, como ilustra o diagrama, uma fonte (`Source`) será uma agregação de itens (`Item`). A classe `Item` se especializa em:

- **`Item simples`** – trata-se de um item de dados literal (atômico) que não possui subitens. Cada item simples é associado a um tipo de dados `Datatype` (importado do modelo estrutural do OWL 2) e um conteúdo, definido pela propriedade `content`.
- **`Item composto`** – agrega um conjunto de outros itens (simples ou compostos). Os itens compostos possuem seu tipo associado a classes a uma classe OWL (`Class` no diagrama). Além disto, usualmente um item composto possui opcionalmente um rótulo (`label`).

3.1.2 Modelo Abstrato do Mapeamento

O diagrama da Figura 10 resume os dois tipos de mapeamentos: simples e composto. Em um mapeamento simples itens da fonte de dados se transformarão em valores dados a propriedades do tipo `DataType` na ontologia destino, de acordo com a seguinte lógica: o mapeamento define qual a propriedade a ser mapeada e o conteúdo do item simples se transforma em valor da propriedade.

Figura 10 - Modelo Abstrato do Mapeamento



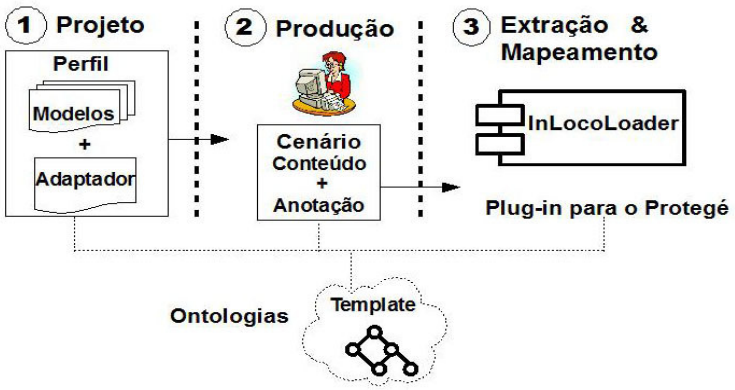
Um mapeamento composto transforma um item composto no valor de uma propriedade `ObjectProperty`. Isto envolve a instanciação de um objeto na ontologia, cuja classe é definida em `Composite Item`. Além disto, um item composto pode ter opcionalmente um `label`, que será convertido no valor de uma propriedade do objeto criado, que deve ser do tipo `DataProperty`. A esta operação chamamos de mapeamento de `inner property`. Seu papel será discutido mais adiante.

3.2 METODOLOGIA

As etapas que compõem a nossa metodologia de anotação e extração de informações de fontes de dados textuais estão sintetizadas no diagrama da Figura 11, que resume um cenário típico da aplicação desta metodologia. Este cenário pode ser organizado em três atividades distintas: **Projeto (1)** – é criado um perfil para a elaboração de documentos em certo domínio, incluindo os seus modelos e um adaptador descrevendo o mapeamento entre elementos do

modelo e ontologias; **Produção (2)** – o conteúdo é produzido e anotado implicitamente pelo autor seguindo um padrão de anotação e baseado em um ou mais arquivos de modelo*;* **Extração e Mapeamento (3)** – o conteúdo original criado na fase de produção é submetido à nossa ferramenta de IE, o *plug-in InLocoLoader,* de onde serão extraídas as informações relevantes que vão ser mapeadas para as propriedades relacionadas às classes de uma ou mais ontologias de domínio, previamente criadas em OWL, conforme mapeamentos disponíveis no arquivo adaptador.

Figura 11 - Processo de extração de informações de fontes de dados textuais implicitamente marcadas, utilizando perfis adaptáveis



Cada uma dessas três atividades será detalhada nas subseções subsequentes.

3.3 PROJETO

Na fase de projeto é definido o contexto para um domínio específico. Tecnicamente, contexto se traduz em um arquivo de perfil formado pelo arquivo adaptador e por um conjunto de arquivos de modelo utilizados por ferramentas de processamento de texto. Apesar de apresentarmos um modelo suficientemente genérico a ser aplicado na maioria dos processadores de texto modernos, neste trabalho validamos a metodologia utilizando o processador de texto Microsoft Word e, para o perfil, arquivos de modelo do Microsoft Word (.dot). O contexto é o passo inicial para a definição do cenário de extração.

O adaptador tem a função de materializar os modelos abstratos da fonte de dados e do mapeamento apresentados na seção anterior. Esta materialização se dá na forma de um arquivo XML, como está ilustrado na Figura 12. O adaptador tem dois papéis simultâneos: o

de identificar os itens da fonte de dados de origem (instâncias das classes Simple Item ou Composite Item apresentadas na seção anterior) e o de representar o mapeamento, conforme modelo abstrato apresentado anteriormente. Este processo de materialização envolve a caracterização de aspectos específicos da Semântica In Loco, tais como padrão de anotação adotados. Além disto, os itens declarados no adaptador devem estar compatíveis com aqueles estabelecidos nos modelos de documento, descritos adiante. Em resumo um adaptador descreve:

- Quais os itens de dados serão reconhecidos no documento de origem;
- Quais os padrões de anotação percebidos no documento anotado;
- Quais as ontologias envolvidas no processo;
- Como cada padrão de anotação produz uma instância de classe ou valor de propriedade nas ontologias envolvidas (mapeamento).

Figura 12 - Arquivo adaptador que é utilizado para configurar o InLocoLoader para o contexto desejado pelo autor

adapter - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

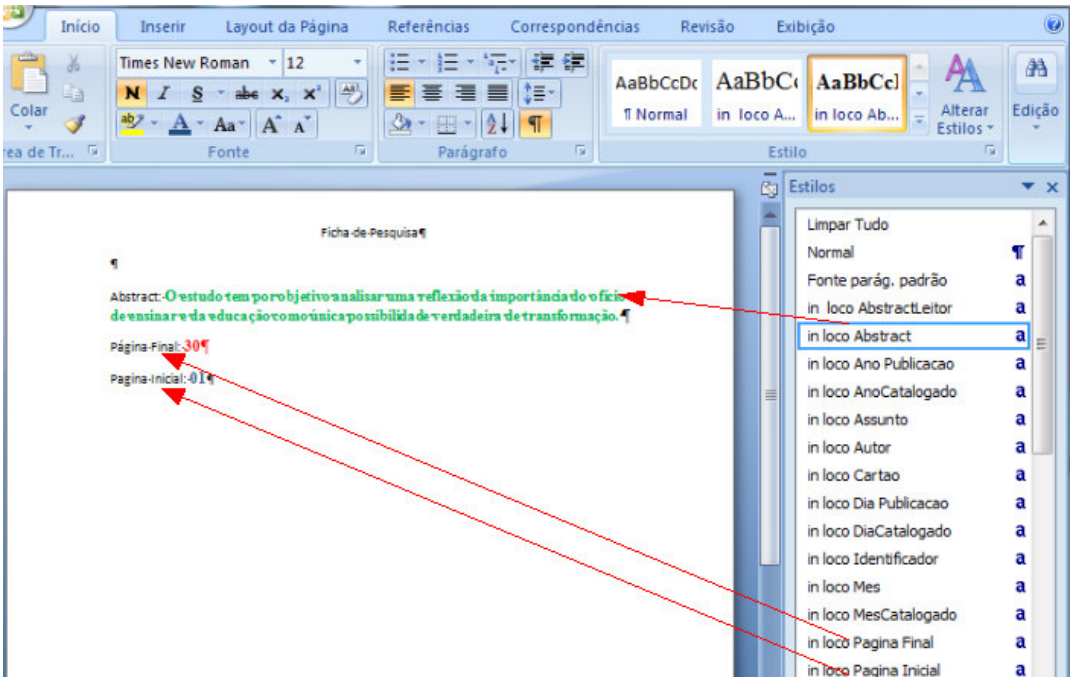
```
<?xml version="1.0" encoding="UTF-8"?>
<map:document style="in loco Identificador" owlclass="card:Article" xmlns:map="http://www.example.or
    <map:property style="in loco Titulo" owlproperty="card:title"
        range="xsd:string">
    </map:property>
    <map:property style="in loco Autor" owlproperty="card:authors"
        range="card:Person" iproperty="card:fullName">
    </map:property>
    <map:property style="in loco Cartao" owlproperty="card:hasCards"
        range="card:ResearchCard">
    <map:property style="in loco AbstractLeitor"
        owlproperty="card:abstractReader" range="xsd:string">
    </map:property>
    <map:property style="in loco DiaCatalogado"
        owlproperty="card:dayCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco MesCatalogado"
        owlproperty="card:monthCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco AnoCatalogado"
        owlproperty="card:yearCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco Referencias"
        owlproperty="card:references" range="card:BibliographicUnit"
        reference="true">
    </map:property>
    <map:property style="in loco Topico" owlproperty="card:hasTopics"
        range="card:CardTopic" iproperty="card:title">
        <map:property style="in loco Recorte"
            owlproperty="card:publicationClip" range="xsd:string">
        </map:property>
        <map:property style="in loco TpPagina"
            owlproperty="card:page" range="xsd:string">
        </map:property>
        <map:property style="in loco Comentarios"
            owlproperty="card:readerComment" range="xsd:string">
        </map:property>
        <map:property style="in loco SubjectTopic"
            owlproperty="card:subjects" range="subj:Subject"
```

Na etapa de extração de informações detalharemos o esquema deste arquivo e a respectiva funcionalidade de cada uma de suas partes. A técnica específica de representar o mapeamento de dados, explicitados por padrões de anotação em um documento, para elementos de uma ontologia é uma contribuição deste trabalho.

Nos arquivos de modelos (.dot) são pré-definidos os subsídios para uso dos padrões de anotação. Como demonstraremos adiante, há várias possibilidades para o uso de padrões de

anotação, mas neste texto optamos por dar ênfase a uma delas, a qual foi predominantemente usada neste trabalho e em experiências práticas: o uso de estilos. Como ilustra a Figura 12, e serão detalhados na próxima subseção, processadores de texto utilizam o recurso de **estilo** como uma forma compacta de associar a um segmento do texto uma configuração pré-definida de formato. Esta estratégia é explorada pela Semântica In Loco como padrão de anotação, para associar mais semântica a trechos do texto. Por este motivo, na elaboração dos modelos é definido um conjunto de estilos, que serão utilizados para anotar conteúdo. Esses estilos marcam as informações relevantes que posteriormente serão extraídas do texto. No processo de mapeamento, elas serão transformadas em valores de propriedades ou instâncias das classes que compõem as ontologias. Os modelos podem dar diferentes graus de liberdade ao usuário que produzirá o conteúdo. Um arquivo de modelo pode ser desde um formulário pré-configurado, onde cada campo do formulário está associado a um estilo, até um arquivo sem campos pré-definidos, com o conjunto de estilos pré-definidos, em que o autor vai usando os estilos para ir marcando partes do texto. Na Figura 13, apresentamos um dos arquivos de modelo utilizado no nosso caso de uso para o domínio de publicações. À direita são apresentados os estilos disponíveis no modelo para uso nas anotações. À esquerda é mostrado o estilo aplicado em um texto.

Figura 13 - Um dos arquivos de modelo do Microsoft Word 2007 que faz parte do arquivo de perfil pesquisa científica



Faz parte desta etapa a construção deste modelo, contendo todos os estilos. Cada rótulo dado a um estilo será usado para demarcar um tipo de item (simples ou composto). Este rótulo será compatível com a identificação dada aos itens no modelo da fonte de dados e consequentemente no adaptador.

3.3.1 Produção

Uma vez definido o perfil para o conteúdo de interesse do autor, passa-se para a fase de produção. Nessa fase, o conteúdo vai ser criado pelo autor e a anotação vai ser realizada de forma implícita. O autor produz o conteúdo utilizando apenas o processador de texto, sem qualquer *plug-in* especial ou software *add-on*. Durante o processo de produção do texto, o autor aplica estilos (por exemplo, título, subtítulo) e implicitamente anota o conteúdo. Como foi referenciado anteriormente, também é possível fornecer modelos com campos predefinidos e com estilos pré-formatados. Neste caso, o autor preenche os campos indicados e não precisa se preocupar com estilos, dado que eles já estão associados aos campos. Os arquivos de modelo que compõe o perfil cumprem dois papéis simultâneos, detalhados a seguir.

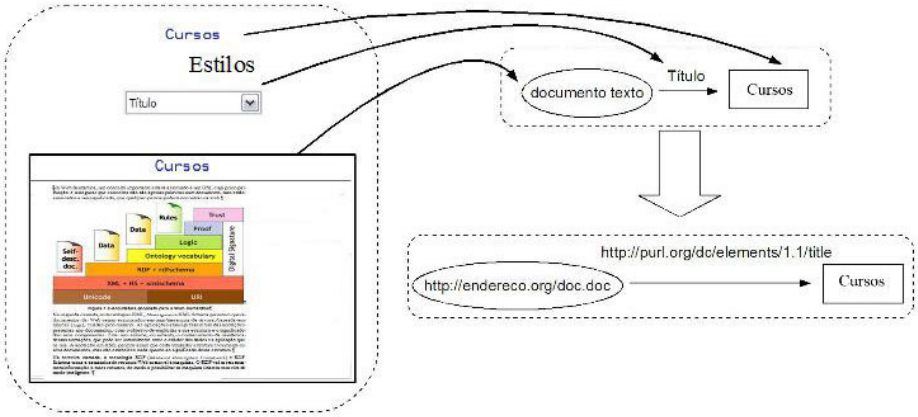
Como mencionado anteriormente, a propriedade estilo do processador de texto é um recurso dessa ferramenta que habilita o usuário a associar um nome para uma configuração específica de formato. Por exemplo, o estilo {Titulo} pode definir a configuração {Times, 16, Negrito, Centralizar}. Nosso padrão de anotação tira vantagem do recurso estilo e estende sua interpretação para nossos propósitos. Além da sua função de formatação, o estilo adquire outra função a de marcação semântica. Assim, para marcamos fragmentos dos documentos utilizando o estilo {Titulo}, por exemplo, o autor realiza duas operações: ele aplica uma formatação pré-configurada para essa parte do texto e simultaneamente associa uma anotação semântica para esse fragmento.

É interessante notar que do ponto de vista do autor nada mudou, ele continua usando o processador de texto Word da mesma forma que ele fazia antes. O conteúdo semântico, agora adicionado pelo estilo, é conseqüência de pré e pós-operações, as quais ocorreram antes e vão ocorrer depois da produção do documento.

Na Figura 14 apresentamos um diagrama ilustrando como é feita a associação entre as anotações e padrões abertos da Web Semântica. As anotações são extraídas do documento e

convertidas para metadados RDF. Cada fragmento do documento, formatado usando um estilo, é convertido para uma tripla RDF seguindo o raciocínio: o endereço do documento se transforma em recurso, o estilo aplicado na propriedade desejada e o fragmento formatado pelo estilo no valor da propriedade. A Figura 15 apresenta um DLG representando a tripla extraída do título do texto "Cursos", formatado pelo estilo {Título}. O estilo {Título} foi transformado numa propriedade RDF http://purl.org/dc/elements/1.1/title a qual é definida no vocabulário Dublin Core.

Figura 14 - Transformação de documento + estilo+ conteúdo em tripla RDF



Este mapeamento entre a anotação e a ontologia é representado pelos dois modelos abstratos da fonte de dados e do mapeamento. Cada segmento marcado por um estilo torna-se um item. O que determina se o item é simples ou composto será o adaptador, especificado anteriormente. No caso de itens compostos, seus subitens serão identificados a partir do contexto, com o uso de um algoritmo descrito na subseção de extração (a seguir).

O exemplo da Figura 14 mostra o caso mais geral. Como será mostrado adiante, o elemento que faz o papel de recurso na tripla nem sempre é o documento. Em casos de itens compostos o próprio objeto OWL criado como resultado do item pode se tornar recurso. Deste modo, o foco de quem é o recurso pode se deslocar.

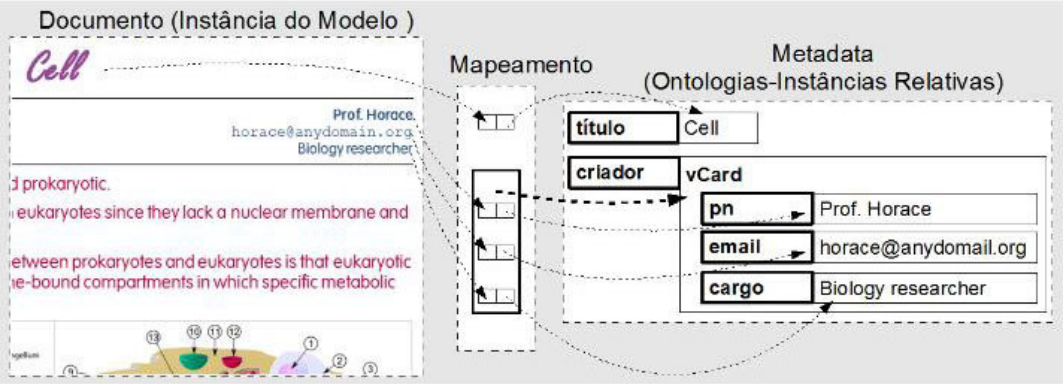
3.3.2 Extração e Mapeamento

Nessa fase serão extraídos os dados relevantes do conteúdo produzido e anotado implicitamente pelo autor e mapeados para um determinado formato de saída. Como visto na

seção 2.5.3.3, respectivamente, *cenário* e *template.* Para o processo de extração é utilizada a ferramenta de IE implementada neste projeto denominada *InLocoLoader*. Nesse trabalho, o *template* é um conjunto de ontologias. Sendo assim, na fase de produção quando criamos os arquivos de modelo e consequentemente os estilos, cada estilo deve corresponder a uma propriedade de uma determinada ontologia de domínio desse conjunto, onde o fragmento marcado por esse estilo será o seu valor.

Na Figura 15 temos um diagrama mostrando o mapeamento das informações extraídas do texto para uma determinada ontologia, sendo que nesse caso o conteúdo foi criado no formato de formulário. Como pode ser observado, o título "Cell" com o respectivo estilo foi mapeado para uma propriedade *título*. Trata-se de um item simples. Os dados do autor do texto (em azul alinhado à direita) serão mapeados para a propriedade *criador* e vão produzir uma instância da classe vCard. vCard é um padrão para a descrição de dados pessoais, que pode ser representado em diversos formatos. Dentre eles há uma versão para RDF (http://www.w3.org/Submission/vcard-rdf/). O sistema de mapeamento é capaz de detectar automaticamente a necessidade da criação de tal instância, uma vez que o item *criador* foi declarado como item composto no adaptador. Deste modo, o nome do autor, seu e-mail e cargo foram mapeados para as respectivas propriedades de vCard, conforme indica a figura. Note-se que, do ponto de vista do RDF, o *resource* associado às propriedades *titulo* e *criador* é o próprio documento, por outro lado, o *resource* associado às demais propriedades é uma instância da classe vCard. Isto demonstra, conforme apontado anteriormente, que o *resource* varia na medida em que as instâncias da ontologia vão sendo construídas. Mais adiante trataremos como isto é configurado no arquivo de mapeamento.

Figura 15 - Extração e mapeamento de um cenário para um template no formato de ontologia



Na seção 3.2 vimos que tecnicamente o perfil é composto de um conjunto de arquivos de modelo e do arquivo adaptador. O adaptador é responsável por duas tarefas importantes: configurar o *InLocoLoader* para uma extração e adaptar essa ferramenta de IE para extrair informações em cada contexto. Para isso, o seu conteúdo é formado por informações relativas tanto ao cenário[2] quanto ao *template*[3].

A fim de detalhar como atua o adaptador, a seguir será apresentada a estrutura de uma parte do adaptador para padrões de anotação baseados em estilo. A Figura 16 apresenta um diagrama do esquema de tal arquivo e a Figura 17 um exemplo com um fragmento do mesmo. O diagrama da Figura 16 utiliza hexágonos para representar elementos. Cada hexágono apresenta o nome de seu elemento, seguido de seus atributos e tipos. A identação indica a relação hierárquica entre os elementos.

Figura 16 - Fragmento de esquema do adaptador

Figura 17 - Fragmento de arquivo adaptador

```
<map:property style="in loco Titulo" owlproperty="card:title"
    range="xsd:string">
</map:property>

...

<map:property style="in loco Topico" owlproperty="card:hasTopics"
    range="card:CardTopic" iproperty="card:title">

    <map:property style="in loco Recorte"
        owlproperty="card:publicationClip" range="xsd:string">
    </map:property>

    ...

</map:property>
```

---

[2] Cenário de Extração, doravante Cenário, refere-se à especificação dos eventos particulares ou relações a ser extraídos, ou seja, ao conjunto de informações relevantes pertencente a um determinado domínio.

[3] Template de Extração, doravante Template, refere-se ao formato de saída da extração de informação de um cenário, que pode ter um formato de texto, tabular ou de ontologia.

Como foi descrito anteriormente, o adaptador tem dois papéis simultâneos, o de identificar itens de dados a ser extraídos e o de caracterizar seu mapeamento. As duas ações são realizadas simultaneamente a partir do elemento `<map:property>` do arquivo adaptador.

Como o exemplo apresentado nas figuras trata de itens identificados através do padrão de anotações baseado em estilos, o elemento `<map:property>` define o item no documento de origem identificando o estilo que o delimita. Além disto, ele representa o mapeamento de um determinado item (definido pelo estilo) para o valor de uma propriedade em uma ontologia. Este elemento define três atributos principais:

- `style` – indica o estilo a ser reconhecido no documento que delimitará um item na fonte de dados
- `owlproperty` – indica a propriedade que receberá o valor na ontologia
- `range` – tipo do valor da propriedade destino

Note que um elemento <map:property> representa simultaneamente uma instância da classe `Item` definida na **Erro! Fonte de referência não encontrada.** e o seu mapeamento – instância da classe SimpleMapping ou CompositeMapping definidas na **Erro! Fonte de referência não encontrada.**. O que determinará se é um item simples ou composto e seu respectivo mapeamento é o fato deste elemento ter ou não sub-elementos, como será detalhado a seguir.

No exemplo, pode-se notar que o estilo "`in loco Titulo`" é mapeado para a propriedade "`card:title`" da ontologia. Dado que seu tipo (range) é string (é um `DataType`) e ele não possui subitens, trata-se de um `item simples` (conforme definido no modelo da **Erro! Fonte de referência não encontrada.**) e o mapeamento é simples e direto. Por outro lado, o estilo "`in loco Topico`" abre o mapeamento para uma instância da ontologia da classe `card:CardTopic` (é do tipo `Class`) e possui sub-elementos. Neste caso, trata-se de um `item composto` (conforme definido no modelo da **Erro! Fonte de referência não encontrada.**). Por se tratar de uma instância que têm propriedades associadas, o esquema XML abre sub-mapeamentos e o `resource` que está sendo descrito torna-se agora a instância criada. Os itens simples e composto são respectivamente associados a um mapeamento simples ou composto, conforme modelo da **Erro! Fonte de referência não encontrada.**.

Existe um atributo opcional extra chamado `iproperty`. Ele aparece apenas em itens compostos, quando o estilo que abre uma instanciação marca um texto. Este texto será

convertido para uma das propriedades do objeto – a que for indicada por iproperty. O iproperty corresponde ao mapeamento da inner property indicada no diagrama da **Erro! Fonte de referência não encontrada.**.

Esta hierarquia de mapeamentos permite a anotação de objetos com estrutura de qualquer complexidade. Um desafio neste contexto é como indicar que as anotações encerram uma instância e retornam para o contexto superior. Atualmente isto é feito quando se encontra algum estilo+propriedade que não corresponde à instância corrente.

No início do arquivo são listados os namespaces de todas as ontologias envolvidas no processo. A criação das ontologias utilizadas no template não faz parte do escopo de nossa ferramenta, mas sendo o InLocoLoader um plug-in para o Protegé, podemos utilizar essa ferramenta para a sua criação. O formato dessas ontologias tem que ser obrigatoriamente o OWL, que é o formato trabalhado pelo nosso *plug-in*.

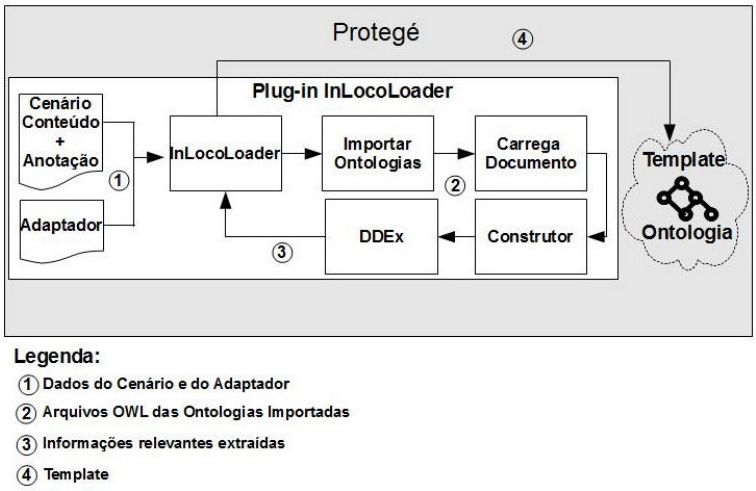
Uma vez definido o cenário e o template, o InLocoLoader vai ler o arquivo adaptador para que ele possa iniciar suas três rotinas principais: importar ontologias, carregar/extrair dados de documentos e preencher ontologias com dados mapeados. Este processo está ilustrado na Figura 18, em que a sua legenda demostra o fluxo interno das informação entre as rotina que compõe o InLocoLoader. A seguir detalhamos como isto é feito na sua implementação desenvolvida na forma de um plugin do Protégé:

- Importar ontologias é uma rotina que prepara o ambiente do Protegé para receber os dados extraídos do cenário. Para isso, ela cria um novo projeto no Protegé e importa as ontologias descritas no arquivo adaptador, garantindo assim que toda vez que importamos algum cenário novo, teremos uma cópia do template antes da importação;

- Carregar documentos é a rotina executada em seguida. Ela aciona uma ferramenta externa que foi adotada neste projeto chamada *DDEx* para ler o cenário e extrair somente o conteúdo marcado pelos seus estilos.

- Mapear as informações extraídas para as respectivas propriedades das ontologias que compõem o template é uma etapa que ocorre concomitante à anterior.

Figura 18 - Modelo do processo interno do InLocoLoader



3.4 O AMBIENTE DO *INLOCOLOADER*

Neste tópico, é detalhado o processo de desenvolvimento do software InLocoLoader. Este software foi desenvolvido como um *plug-in de extensão do* software de criação e gerenciamento de ontologias Protégé, com o objetivo de adicionar-lhe, a funcionalidade de extração de informação em formato texto, convertendo-a para ontologias.

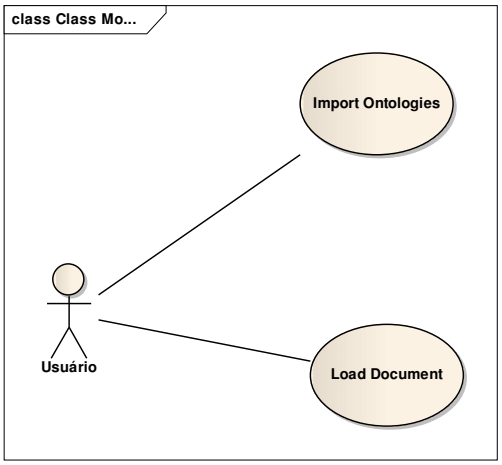
A Semântica In Loco é baseada em dois princípios fundamentais: o princípio da Interoperabilidade e o princípio da Persistência Semântica (detalhados na seção 2.4.4). Uma ferramenta de IE baseada nessa metodologia tem que respeitar esses dois princípios. Baseado nesses pré-requisitos foi escolhido o software Protégé como base para o desenvolvimento do plug-in InLocoLoader. Este software trabalha com padrões abertos da Web semântica como RDF e OWL, atendendo ao princípio da Interoperabilidade e, como sua principal função é a criação e manutenção de ontologias (formato de saída escolhido para o InLocoLoader), também atende ao princípio da Persistência Semântica. Sendo assim, as funções desenvolvidas para o plug-in são exclusivas de extração de informação.

Na próxima seção são apresentados os principais artefatos produzidos durante o processo de desenvolvimento desse software.

3.4.1 Diagrama de Casos de Uso do InLocoLoader

A fase de analise de sistemas evidenciou duas funções principais do *software*, que estão representadas no diagrama de caso de uso da Figura 19. E detalhadas no Quadro 1.

Figura 19 - Diagrama de Casos de Uso do Plug-in InLocoLoader



Como pode ser observado no Diagrama de Casos de Uso apresentado na Figura 19, foi considerado apenas um tipo de pessoa que irá utilizar o ambiente: **Usuário**. O Quadro 1, apresentado a seguir, mostra uma descrição dos casos de uso incluídos na Figura 19.

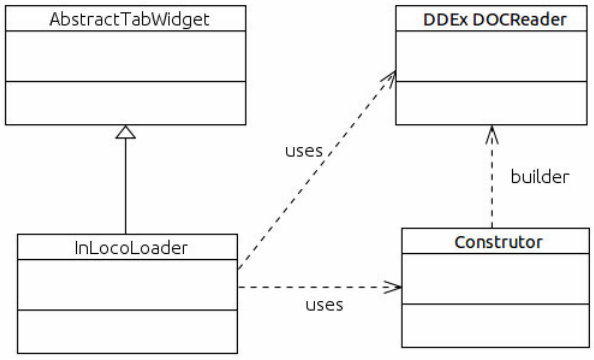
Quadro 1 - Descrição dos Casos de Uso do plug-in InLocoLoader

Nº	Nome do caso de uso	Descrição
1	Import Ontologies	Lê o arquivo adapter.txt e carrega numa nova seção do Protégé; carrega as ontologias que serão utilizadas como saída do processo de extração (*Template*) e seleciona a fonte de dados a ser extraída (Cenário).
2	Load Document	Extrai as informações relevantes do cenário utilizando o componente Java DDEx e armazena essas informações relevantes nas ontologias que compõe o *Template*.

3.4.2 Diagrama de Classes

Uma vez elaborado o diagrama de casos de uso, o próximo passo é a definição das classes a serem utilizadas e seus respectivos relacionamentos. Para isso, a UML propõe a elaboração do Diagrama de Classes. O Diagrama de Classes para o *Plug-in InLocoLoader*, pode ser visualizado na Figura 20. A classe InLocoLoader é a principal. Ela extende a classe AbstractTabWidget que representa um plug-in do tipo aba do Protégé. O InLocoLoader implementa e coordena todas as operações. Ele faz uso do extrator DDEx através da classe DOCReader. A classe Construtor é definida pelo plug-in InLocoLoader. Ela recebe notificações do DOCReader informando itens extraídos do documento e as mapeia para elementos da ontologia.

Figura 20 - Diagrama de Classes do plug-in InLocoLoaderDiagrama de Classes do plug-in InLocoLoader



3.5 RESUMO DO CAPÍTULO

Nesse capítulo foi apresentada a nossa metodologia de extração de informações de fonte de dados em formato texto em três etapas. Na primeira etapa mostramos a preparação do cenário através da adição de semântica ao documento utilizando o nosso padrão de anotação e como mapeamos essas informações para RDF. Na segunda, foi demonstrada como é feita a configuração da nossa ferramenta *InLocoLoader* para gerar o *template* na forma de ontologia, através do arquivo adaptador, e vimos também a importância do mesmo para adaptação da ferramenta para novos domínios, através do conceito de perfis semânticos. Por último foi

apresentado o modelo de processo interno do InLocoLoader demonstrando como é o seu funcionamento interno. Este trabalho também contribuiu com a especificação de um modelo abstrato tanto para a descrição da fonte de dados quanto para o seu mapeamento.

4 CASO DE USO

Este capítulo tem por objetivo apresentar uma aplicação prática da nossa ferramenta de extração, o *plug-in* para o *Protegé* chamado *InLocoLoader*, para um domínio específico o de catalogação de publicações, validando assim a ferramenta. Nesse domínio, temos interesse em catalogar informações retiradas dos diversos tipos de publicações (artigos, monografias, dissertações, teses de doutorados, livros etc.) para futuras pesquisas. Nesse caso de uso, vamos trabalhar com o tipo de publicação Artigo. Para isso vamos seguir o modelo do processo de extração da Figura 11.

4.1 PLANEJAMENTO DO CASO DE USO

A revisão de literatura no domínio da ciência é um trabalho que, além de envolver intensa leitura de uma grande quantidade de material – na forma de artigos, livros e relatórios técnicos, entre outros – passa também, por uma minuciosa fase de catalogação do que foi lido (Fichamento das obras pesquisadas). A questão é que, na maioria dos casos, a relação entre os assuntos tratados nos diversos textos fica na mente do pesquisador que os lê. Com o tempo e o aumento do volume de leitura, informações e relações importantes vão se perdendo na memória. Na medida em que os assuntos e suas relações são explicitados em um esquema passível de ser interpretado por máquinas, torna-se possível o desenvolvimento de programas que dêem assistência no processo de revisão.

Outro aspecto importante é a possibilidade de gerar novas informações a partir de relações preexistentes por inferência (analogia, indução ou dedução), e que de preferência isso seja feito com auxílio do computador. Por exemplo:

- Banco de dados, redes e modelagem de dados são assuntos da área de informática;
- Informática faz parte da área de tecnologia;
- Por Indução, Banco de dados, redes e modelagem de dados são áreas da tecnologia.

Finalmente, a maioria das pesquisas atualmente é conduzida por grupos de estudo. Seguindo esse raciocínio, uma revisão literária em que já exista a possibilidade de colaboração no

processo de fichamento das obras é de fundamental importância para um bom processo de pesquisa.

Diante deste contexto, tendo como objetivo auxiliar o pesquisador a lidar com o processo de pesquisa de maneira mais produtiva é que foi criado esse caso de uso. Ele envolve a aplicação experimental da metodologia com um grupo de usuários, dentro das seguintes condições:

- O estudo foi realizado na COGEL – Companhia de Governança Eletrônica do Salvador, empresa pública municipal que cuida da TIC – Tecnologia da Informação e Comunicação do município do Salvador, durante o mês de agosto de 2011.
- Na Gerência de Projetos da COGEL, um grupo de cinco pessoas trabalharam com uma amostra de vinte artigos listados na Tabela 7, para serem extraídas informações relevantes utilizando o *InLocoLoader*, em que cada uma fará a extração de informações de quatros artigos. Para isso foi utilizado o formulário da Figura 23 feito em Word como modelo contendo vinte campos, marcados pelo padrão de anotação, como cenário e as ontologias *Card* e *Research Card* como template de saída. Todo ferramental necessário para o caso de uso está descriminado na seção 4.2.
- Ao final do experimento foi feita a avaliação da extração dos resultados, utilizando os conceitos de *recall e precision*, abordados na seção 2.5.3.3.
- Ao final do experimento foi feita também a avaliação da utilização da ferramenta, através da aplicação do formulário de avaliação de satisfação.

Nas próximas seções será apresentada: a descrição do ferramental necessário para execução do caso de uso, a execução do caso de uso, a análise e discussão dos resultados obtidos com o caso de uso e as lições aprendidas.

4.2 DESCRIÇÃO DO FERRAMENTAL UTILIZADO NO CASO DE USO

Nessa seção será apresentado o ferramental usado para a confecção do caso de uso, ou seja, a fonte de dados de origem, o arquivo de saída num formato específico (Ontologia), o arquivo de modelo, contendo um padrão utilizado para a atividade de anotação e o arquivo adaptador contendo as informações necessárias para a extração.

4.2.1 Fonte de dados de origem

Num processo de extração de informações em formato texto são extraídas informações relevantes da fonte de dados de origem para um formato específico de saída. Para o caso de uso do *Research Card* foram utilizados vinte artigos como fonte de dados de origem. Na Tabela 7 a seguir estão listados os artigos utilizados no caso de uso, que são arquivos no formato *Word* e arquivos no formato *PDF*.

Tabela 7 - Relação dos artigos que serão anotados no caso de uso Research Card

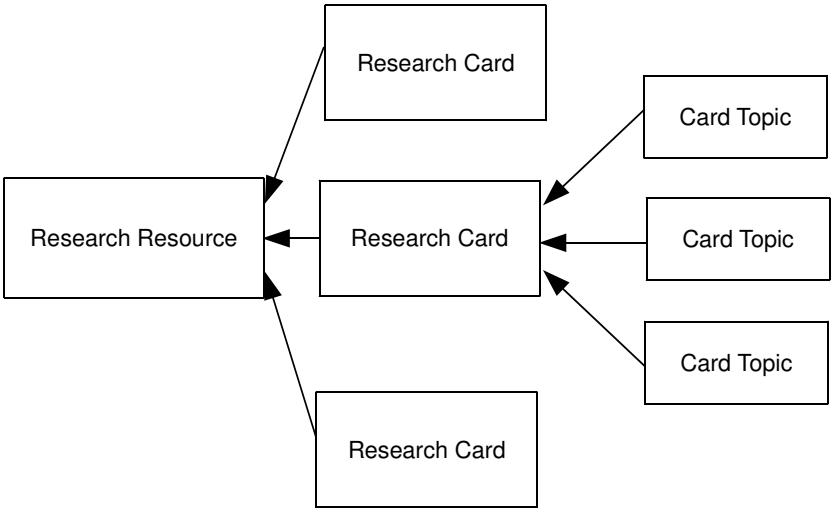
Nº	Título do Artigo	Autores	Formato
1	Design de interface para web semântica Interface design for semantic web	• ELER, Denise.	• Word
2	Uma Arquitetura para Recuperação de Informação Baseada em Semântica e sua Aplicação em Apoio a Jurisprudência.	• Anselmo Maciel Nunes, Renato Fileto.	• PDF
3	Uso de Ontologias para Extração de Informações em Atos Jurídicos em uma Instituição Pública.	• Eduardo Jaime Quirós Batres, Alcione de Paiva Oliveira, Bruno Ventorim Gabrielli, Vinci Pegoretti Amorim, Alexandre Moreira.	• PDF
4	AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages.	• Paul Kogut, William Holmes.	• PDF
5	Adaptive Information Extraction	• Jordi Turmo, Alicia Ageno, Neus Catalá	• PDF
6	Uma Abordagem de Aprendizagem Híbrida para Extração de Informação em Textos Semi-estruturados.	• Edurado F. A. Silva, Flávia A. Barros, Ric. Prudêncio.	• PDF
7	Information Extraction	• Jim Cowie, Wendy Lehnert.	• PDF
8	Information Extraction: Techniques and Challenges	• Ralph Grishman.	• PDF
9	Geração automática de metadados para documentos técnicos científicos	• Bruno Viana Rezende • Marcello Peixoto Bax	• PDF

10	Metadados, Web Semântica, Categorização Automática: combinando esforços humanos e computacionais para a descoberta e uso dos recursos da web	• Rafael Port da Rocha	• PDF
11	Manipulação de uma ontologia desenvolvida em OWL através da utilização da API JENA 2 Ontology	• Paulo Roberto Gonçalves, Parcilene Fernandes de Brito	• PDF
12	Uso de ontologias para gerenciamento e acesso a documentos na web	• Sandro Rigo, Marcirio Silveira Chaves, Renata Vieira	• PDF
13	Desenvolvimento Colaborativo de Ontologias em Redes de Conhecimento Compartilhado	• Daniel Torres Bonatto, Luiz Antonio M. Palazzo	• PDF
14	Classificação de textos baseada em ontologias de domínio	• Sandro J. Rigo, José Palazzo M. de Oliveira, Cristiano Barbieri	• PDF
15	Metadados no Contexto daWeb Semântica	• Alexandre Mori, Cedric Luiz de Carvalho	• PDF
16	Uso De Ontologias para a Extração de Informações em Atos Jurídicos em uma Instituição Pública	• Eduardo Jaime Quirós Batres, Alcione de Paiva Oliveira	• PDF
17	Tecnologia de descoberta de conhecimento em Informações Textuais	• Leandro Krug Wives, Stanley Loh	• PDF
18	Authoring and annotation of desktop les in seMouse	• Oscar Díaz, Jon Iturrioz, Sergio F. Anzuola	• PDF
19	Adaptive Information Extraction	• Jordi Turmo, Alicia Ageno, And Neus Catal	• PDF
20	Intercâmbio de conhecimento na Web	• João Guilherme de Souza Lima	• PDF

Fonte: Desenvolvido pelo autor.

4.2.2 Ontologia Research Card – Cartão de Pesquisa

Figura 21 - Diagrama sintético com principais elementos do Research Card



O diagrama da Figura 21 sintetiza os principais aspectos da ontologia *Research Card* que serão essenciais para a compreensão do que é tratado a seguir. A ontologia define uma classe denominada *Research Resource* que representa qualquer texto científico. O *Research Resource* é especializado em classes específicas, representando cada tipo de texto como, por exemplo, artigo, livro, relatório técnico etc. Cada Research Resource pode ser associado a um ou mais *Research Cards*, que representam fichas de leitura. Tipicamente cada pessoa cria um Research Card associado ao recurso lido, entretanto, como a ontologia prevê colaboração entre diversos leitores, é possível que vários leitores associem um cartão ao mesmo texto. Um Research Card contém dados mais abrangentes sobre o texto e pode estar associado a diversos *Card Topics*. Cada Card Topic registra um aspecto ou trecho específico do texto que foi anotado. Os *Card Topics* se especializam conforme o tipo de anotação a ser feita. Cada um destes elementos pode ser relacionado com os demais, não apenas quando se referenciam ao mesmo texto, mas com *Resources*, *Cards* e *Topics* de outros textos.

A ontologia *subject.owl* representa o domínio de catalogação. Através de suas classes e propriedades mapeamos as informações necessárias sobre o domínio dos textos científicos, que irão compor bases de conhecimento a ser utilizadas em futuras pesquisas. Na Figura 22 temos a ontologia *cards.owl* e *subject.owl* e na Tabela 8 o resumo descritivo da ontologia. No diagrama da Figura 22 classes são representadas por ovais e as ligações entre as classes representam a herança.

Figura 22 - Ontologias Cards e Subjects

Tabela 8 - Resumo descritivo das ontologias

Classe	Conceito	Subclasse	Conceito
Subject	Assunto		
ResearchCard	Cartão de Pesquisa		
CardTopic	Tópico do cartão	Concept	Conceito
		TopicExplore	Tópico a explorar
		RelevantAspect	Aspecto Relevante
Clip	Recorte do texto		
Taxonomy	Ordem hierárquica que relaciona conceitos		
RFC1766	Linguagem (e.g., pt-BR = português do Brasil)		
Entity	Entidade	Person	Pessoa
dcType	Tipos do Dublim	ResearchResource	Recursos

	Core		pesquisados
ResearchResource	Recurso objeto de pesquisa (e.g., texto científico)	BibliographicUnit	Unidade Bibliográfica
		BibliographicCollection	Coleção Bibliográfica
BibliographicUnit	Unidade Bibliográfica (artefato que não se decompõe em unidades menores)	MastersThesis	Dissertação de mestrado
		PhdThesis	Tese de Dotorado
		UnPublished	Artigo não publicado
		TechnicalReport	Relatório Tecnico
		Manual	Manual
		Booklet	
		Book	Livro
		BibliographicPart	Parte de Uma Bibliografia
BibliographicCollection	Coleção Bibliográfica (artefato que se decompõe em unidade menores)	Journal	Revista
		CollectionBook	Enciclopédia
		Proceedings	Anais
BibliographicPart	Parte de uma Bibliografia (artefato que pode ser parte de uma composição maior)	InProceedings	Artigo para Congresso
		Article	Artigo de Revista
		InBook	Capítulo de um Livro
		Misc	NRA

Fonte: Desenvolvido pelo autor.

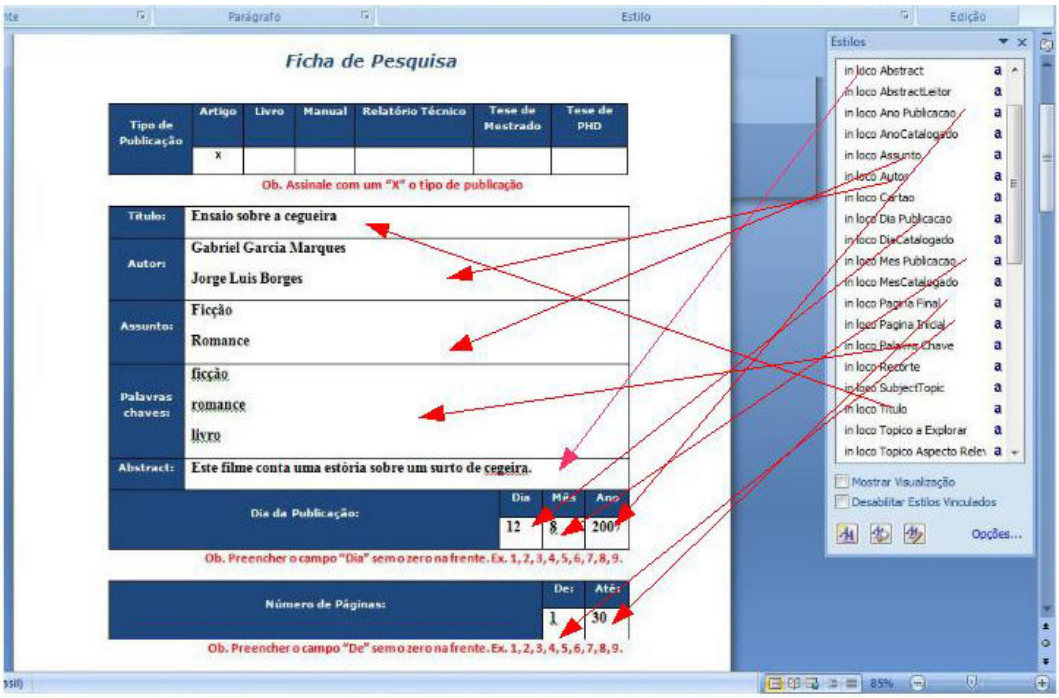
4.2.3 Arquivo de Modelo e Padrão de Anotação

Criar uma ontologia é apenas metade do caminho. Para que esta seja efetivamente usada torna-se necessária a criação de uma metodologia eficiente para registro dos dados provenientes dos textos científicos. Mas para a realização desse registro é necessário seguir a metodologia de extração detalhada na seção 3.2. O modelo do processo de extração é composto de três fases:

- Projeto;
- Produção;
- Extração e Mapeamento.

É na fase de projeto que é criado o cenário que é composto por um conjunto de arquivos de modelo do Word mais o arquivo adaptador. Nesse caso de uso, para a ficha de catalogação é utilizado um arquivo de modelo no formato de um formulário utilizado na fase de produção, representado abaixo na Figura 23, em que cada campo do formulário está associado a uma propriedade de estilo do Word.

Figura 23 - Arquivo de modelo utilizado para anotar conteúdo de artigos



O conjunto de estilos criados no arquivo de modelo, utilizados para o processo de anotação, é o que chamamos de padrão de anotação. O utilizado nesse caso de uso está representado pela Tabela 9.

Tabela 9 – Padrão de anotação utilizadono caso de uso com as propriedades das ontologias do template de saída

Item/Estilo	Informação Relevante	Ontologia	Propriedade
in loco Titulo	Título do artigo em questão	Card.owl	title
in loco Autor	Autor(es) do artigo	Card.owl	authors
in loco Assunto	Assunto do artigo	Card.owl	subject
in loco Palavra Chave	Palavras chaves do artigo	Card.owl	keywords
in loco Abstract	Resumo do artigo	Card.owl	abstract
in loco Dia Publicacao	Dia da publicação do artigo	Card.owl	dayPublished
in loco Mes Publicacao	Mês da publicação do artigo	Card.owl	monthPublished
in loco Ano Publicacao	Ano da publicação do artigo	Card.owl	yearPublished
in loco Pagina Inicial	Nº da página inicial	Card.owl	pageFinal
in loco Pagina Final	Nº da página Final	Card.owl	pageInitial
in loco Cartao	Nº do cartão associado ao artigo	Card.owl	hascards
in loco AbstractLeitor	Resumo do leitor do artigo nesse cartão	Card.owl	abstractReader
in loco DiaCatalogado	Dia que o artigo foi catalogado no cartão	Card.owl	dayCatalogued
in loco MesCatalogado	Mês que o artigo foi catalogado no cartão	Card.owl	monthCatalogued
in loco AnoCatalogado	Ano que o artigo foi catalogado no cartão	Card.owl	yearCatalogued
in loco Topico Titulo	Título de um tópico do	Card.owl	title

	cartão que se quer bordar		
in loco Topico Recorte do Artigo	Cópia de parte do texto referente ao tópico	Card.owl	publicationClip
in loco Topico Pagina	Página a qual o tópico se refere	Card.owl	page
in loco Topico Comentarios	Comentários em relação ao tópico	Card.owl	readerComment
in loco SubjectTopic	Assunto do tópico em questão.	Card.owl	Subjects

4.2.4 O arquivo adaptador

Na fase de projeto também é criado o arquivo adaptador. Como visto no Capítulo 3, tecnicamente o contexto se traduz na criação de um modelo representando a fonte de dados de origem e seu mapeamento (conforme modelo abstrato); tal modelo dará origem a um arquivo de perfil formado pelo arquivo adaptador e por um conjunto de arquivos de modelo do Microsoft Word (*.dot*).

Nesse caso de uso, o arquivo Adapter é utilizado para configurar o *InLocoLoader plug-in* para o domínio de publicações científicas. Na Figura 24, apresentamos um recorte desse arquivo. Nele encontramos todos os dados necessários para se criar o cenário específico da extração e principalmente informações para configurar o *template* de saída, que nesse caso de uso é formado por duas ontologias:

- Cards.owl;
- Subject.owl.

Figura 24- Arquivo adaptador para o domínio catalogação de publicação

adapter - Bloco de notas

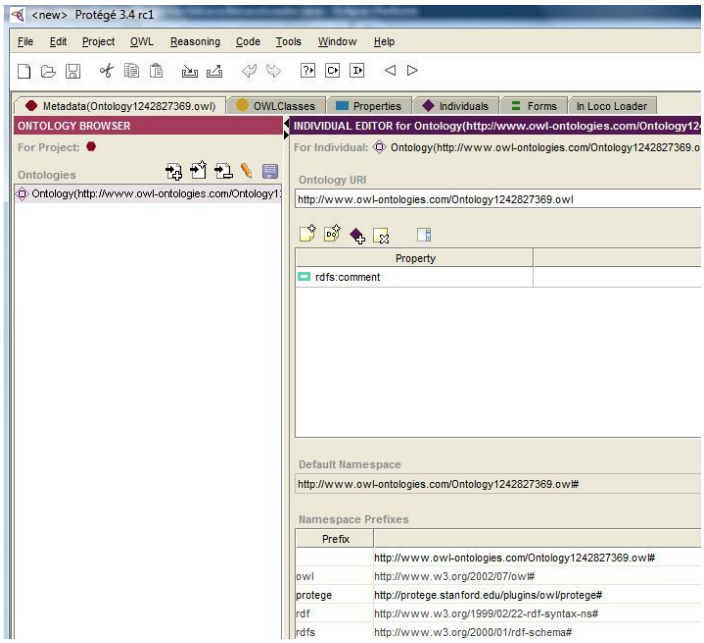
Arquivo Editar Formatar Exibir Ajuda

```
<?xml version="1.0" encoding="UTF-8"?>
<map:document style="in loco Identificador" owlclass="card:Article" xmlns:map="http://www.example.or
    <map:property style="in loco Titulo" owlproperty="card:title"
        range="xsd:string">
    </map:property>
    <map:property style="in loco Autor" owlproperty="card:authors"
        range="card:Person" iproperty="card:fullName">
    </map:property>
    <map:property style="in loco Cartao" owlproperty="card:hasCards"
        range="card:ResearchCard">
    <map:property style="in loco AbstractLeitor"
        owlproperty="card:abstractReader" range="xsd:string">
    </map:property>
    <map:property style="in loco DiaCatalogado"
        owlproperty="card:dayCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco MesCatalogado"
        owlproperty="card:monthCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco AnoCatalogado"
        owlproperty="card:yearCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco Referencias"
        owlproperty="card:references" range="card:BibliographicUnit"
        reference="true">
    </map:property>
    <map:property style="in loco Topico" owlproperty="card:hasTopics"
        range="card:CardTopic" iproperty="card:title">
        <map:property style="in loco Recorte"
            owlproperty="card:publicationClip" range="xsd:string">
        </map:property>
        <map:property style="in loco TpPagina"
            owlproperty="card:page" range="xsd:string">
        </map:property>
        <map:property style="in loco Comentarios"
            owlproperty="card:readerComment" range="xsd:string">
        </map:property>
        <map:property style="in loco SubjectTopic"
            owlproperty="card:subjects" range="subj:Subject"
```

4.3 EXECUÇÃO DO CASO DE USO

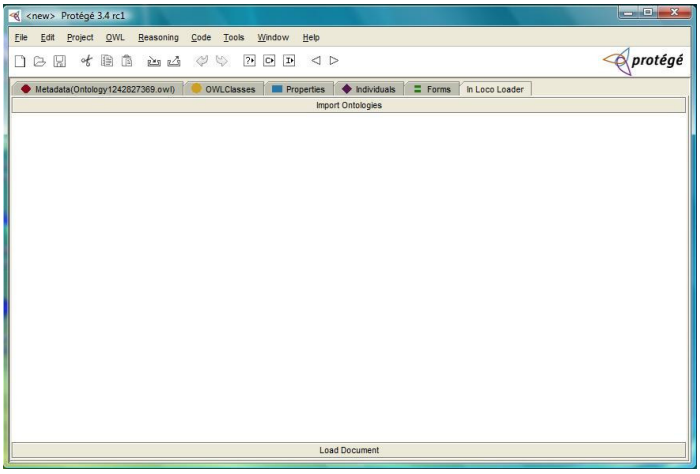
O processo de extração começa com a carga do programa *Protegé (*vide Figura 23) para o qual desenvolvemos nossa ferramenta de extração o plug-in *InLocoLoader*. O *Protégé* (Protégé, 2008) é um dos programas mais populares para a criação e edição de ontologias. Apesar de possuir um formato próprio para a representação de ontologias, o sistema aceita *plug-ins* que possibilitam não apenas estender a ferramenta, mas customizá-la para trabalhar com outros formatos, a exemplo do OWL. O plug-in OWL se tornou tão popular no Protégé que em sua última versão ele deixou de trabalhar com seu formato próprio e adotou o OWL como formato nativo. Conforme está ilustrado na Figura 25, logo abaixo dos ícones de ferramentas é possível observar uma aba referente ao nosso plug-in, o *InLocoLoader.*

Figura 25 - Software Protegé utilizado para criar e manter ontologias



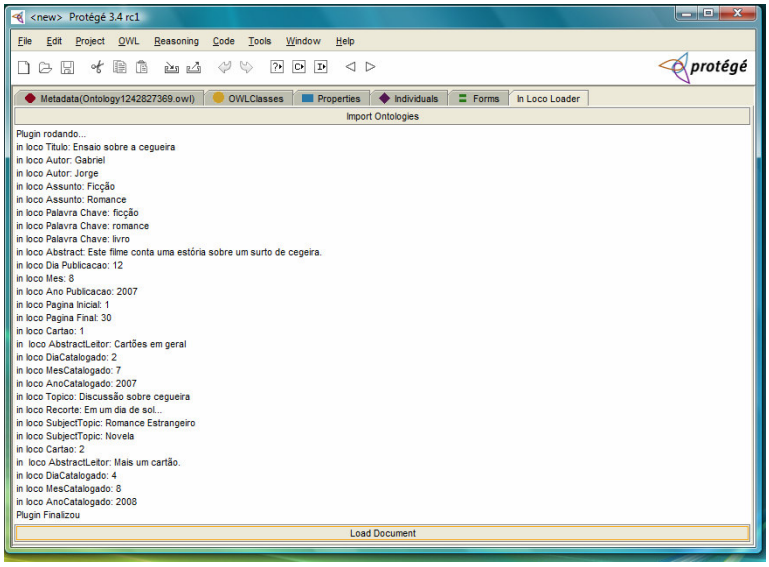
Ainda na Figura 25, pode-se observar na caixa *Ontology Browser* na extrema esquerda, abaixo do conjunto de abas, que ela está vazia. Ao clicar na aba *InLocoLoader* (vide Figura 26) aparece os botões *Import Ontology e Load Documents.*

Figura 26 - Tela de Importação de ontologias e extração das Informações relevantes do documento



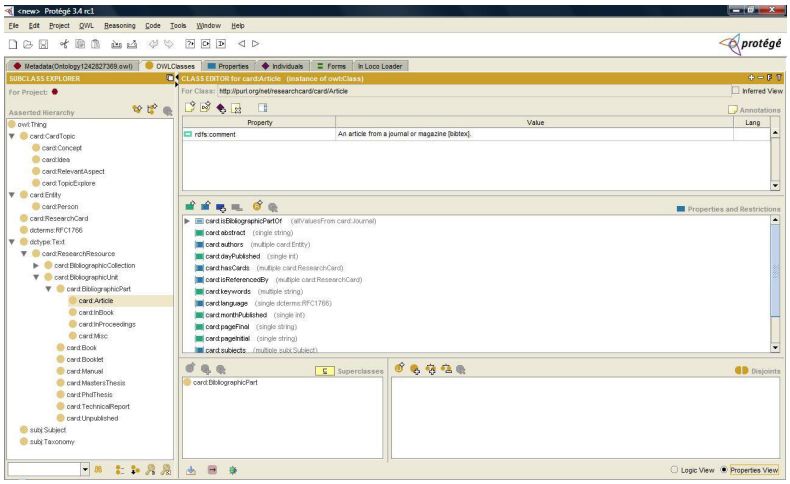
O processo de extração começa com a importação das ontologias que farão parte do *template* de saída. Para isso, clica-se no botão *Import Ontologies*. O próximo passo é executar a rotina de extração, em que o plug-in irá carregar o documento para extrair as informações relevantes marcadas pelos estilos. A Figura 27 ilustra o *plug-in* durante o processo de extração apresentando uma lista com todas as informações extraídas do documento.

Figura 27 - Apresentação das informações extraídas do modelo do Word utilizado pelo arquivo de perfil



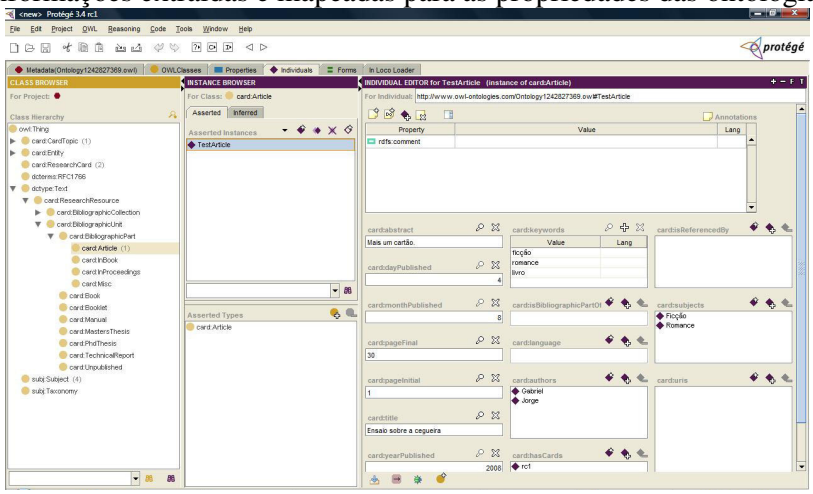
Durante esta fase (vide Figura 28), as informações relevantes são mapeadas para as respectivas ontologias, dentro das propriedades especificas definidas pelo processo de extração.

Figura 28 - Caixa *Ontology Browser* preenchida com as ontologias importadas que compõem o perfil de saída



Na Figura 29 temos o resultado final da extração em que podemos ver através da aba *individuals*, na caixa *Instance Browser* todas as instâncias de artigos importadas pelo plug-in.

Figura 29 - Resultado final do processo de extração e mapeamento com a apresentação das informações extraídas e mapeadas para as propriedades das ontologias



4.4 APLICAÇÃO DA FERRAMENTA EM CAMPO

Neste caso de uso, cinco pessoas diferentes utilizaram a ferramenta *InLocoLoader* e cada uma extraiu informações relevantes de quatro artigos. No total, vinte artigos foram utilizados no caso de uso para a validação da ferramenta. Esses mesmos artigos foram previamente extraídos pelo autor, para a criação dos gabaritos (N_{chave}) que serviram para o cálculo dos indicadores *recall e precision* utilizados no processo de avaliação da ferramenta. As respostas dadas pelos usuários não precisavam ser exatamente iguais às do gabarito. Como muitos dos usuários abreviam parte do texto extraído ou resumiram informações antes de preencher o campo do formulário, foi necessária uma comparação manual dos resultados com o gabarito. O experimento foi conduzido da seguinte forma:

- Em um primeiro momento, as cincos pessoas que participaram do experimento foram apresentadas em conjunto à ferramenta *InLocoLoader*, onde foi explicado além do seu propósito, todo o processo básico de operação.

- Foi apresentado aos cincos participantes do experimento, o formulário de anotação dos artigos (modelo do Word 2007) conforme o Apêndice A. Nesse momento, foi explicado o seu preenchimento. Nesse experimento, o conteúdo de cada campo do formulário tem uma anotação embutida baseada em estilo. O usuário preenche o

campo digitando diretamente o seu conteúdo ou, de forma alternativa, copiando o conteúdo do artigo e colando no seu respectivo campo.

- Durante o experimento, os cincos participantes fizeram cada um, o registro de quatro artigos, num prazo estabelecido de quatro horas, todos os cinco em paralelo.

- Para saber se as informações de um artigo foram corretamente preenchidas/extraídas foi necessário fazer uma análise comparativa com o seu respectivo gabarito ($N_{chave)}$ e para isso foram utilizados os indicadores de desempenho *Recall* e o *Precision*.

O indicador *Recall* mede a taxa de recuperação do processo de anotação/extração, que representa em percentual o quanto foi extraído do que foi anotado corretamente. O indicador *Precision* mede o quão preciso foi o processo de extração, ou seja, o quanto o processo de extração se aproximou da exatidão. Abaixo segue a formulação dos indicadores e no Quadro 2 é apresentada como é a composição desses indicadores.

$$\textbf{\textit{Recall}} = \mathbf{N_{correta/}\ N_{chave}}$$

$$\textbf{Precision} = \mathbf{N_{correta/}\ N_{respostas}}$$

Quadro 2 - Composição dos membros das fórmulas dos indicadores Recall e Precision

Membro da Fórmula	**Significado do membro da fórmula**
N_{chave}	É o numero total de campos do gabarito (conjunto de respostas corretas para o processo de extração). Neste caso de uso, corresponde às vinte possíveis respostas para cada campo do formulário de anotação que serão utilizados no processo de extração. Foi gerado previamente pelo autor, os vinte gabaritos (para cada artigo em questão) para o cálculo dos indicadores de desempenho *Recall e Precision.*
$N_{correta}$	Representa a quantidade de campos do formulário de anotação extraídos, em conformidade com o seu gabarito (N_{chave}).
$N_{respostas}$	Representa o total de campos do formulário preenchidos e que foram extraídos (em conformidade ou não com o seu gabarito).

Na Tabela 10 abaixo, é apresentado o resumo da execução do caso de uso, com o cálculo dos indicadores *Recall e Precision* para cada um dos vinte artigos utilizados, bem como o título de cada artigo e o nome da pessoa que extraiu as informações.

Tabela 10 - Resumo dos resultados do cálculo dos indicadores Recall e Precision do caso de uso

Nº	Operador	Titulo do Artigo	Nchave	Nresposta	Ncorreta	Recall (%)	Precision (%)
1	Usuario1	Design de interface para web semântica	20,0	20,0	20,0	100,0	100,0
2	Usuario1	Interface design for semantic web	20,0	18,0	18,0	90,0	100,0
3	Usuario1	Uma Arquitetura para Recuperação de Informação Baseada em Semântica e sua Aplicação em Apoio a Jurisprudência.	20,0	20,0	20,0	100,0	100,0
4	Usuario1	Uso de Ontologias para Extração de Informações em Atos Jurídicos em uma Instituição Pública.	20,0	20,0	18,0	90,0	90,0
5	Usuario2	AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages.	20,0	15,0	13,0	65,0	86,7
6	Usuario2	Adaptive Information Extraction	20,0	17,0	16,0	80,0	94,1
7	Usuario2	Uma Abordagem de Aprendizagem Híbrida para Extração de Informação em Textos Semi-estruturados.	20,0	20,0	18,0	90,0	90,0

Nº	Operador	Titulo do Artigo	Nchave	Nresposta	Ncorreta	Recall (%)	Precision (%)
8	Usuario2	Information Extraction	20,0	20,0	18,0	90,0	90,0
9	Usuario3	Information Extraction: Techniques and Challenges	20,0	16,0	13,0	65,0	81,3
10	Usuario3	Geração automática de metadados para documentos técnicos científicos	20,0	20,0	15,0	75,0	75,0
11	Usuario3	Metadados, Web Semântica, Categorização Automática: combinando esforços humanos e computacionais para a descoberta e uso dos recursos da web	20,0	20,0	18,0	90,0	90,0
12	Usuario3	Manipulação de uma ontologia desenvolvida em OWL	20,0	19,0	18,0	90,0	94,7
13	Usuario4	através da utilização da API JENA 2 Ontology	20,0	18,0	15,0	75,0	83,3
14	Usuario4	Uso de ontologias para gerenciamento e acesso a documentos na web	20,0	20,0	20,0	100,0	100,0
15	Usuario4	Desenvolvimento Colaborativo de Ontologias em Redes de	20,0	20,0	16,0	80,0	80,0
16	Usuario4	Conhecimento Compartilhado	20,0	19,0	19,0	95,0	100,0
17	Usuario5	Classificação de textos baseada em ontologias de domínio	20,0	18,0	16,0	80,0	88,9

Nº	Operador	Titulo do Artigo	Nchave	Nresposta	Ncorreta	Recall (%)	Precision (%)
18	Usuario5	Metadados no Contexto daWeb Semântica	20,0	19,0	17,0	85,0	89,5
19	Usuario5	Uso De Ontologias para a Extração de Informações em Atos Jurídicos em uma Instituição Pública	20,0	20,0	20,0	100,0	100,0
20	Usuario5	Tecnologia de descoberta de conhecimento em Informações Textuais	20,0	15,0	13,0	65,0	86,7

Fonte: Desenvolvido pelo próprio autor.

Com base na Tabela 10 calculamos o valor médio *Recall* de 85,3% e um *Precision* de 91%. Como estamos utilizando o valor médio desses indicadores de desempenho, calculamos também o desvio padrão dos dois indicadores de desempenho.

Tabela 11 - Resumo dos resultados do cálculo dos indicadores Desvio Padrão Recall e Desvio Padrão Precision do caso de uso

Nº[4]	Recall (%)	(Recall - Média Aritmética Recall)^2	Precision (%)	(Precision - Média Precision)^2
1	100,0	217,6	100,0	80,8
2	90,0	22,6	100,0	80,8
3	100,0	217,6	100,0	80,8
4	90,0	22,6	90,0	1,0
5	65,0	410,1	86,7	18,9
6	80,0	27,6	94,1	9,7
7	90,0	22,6	90,0	1,0
8	90,0	22,6	90,0	1,0
9	65,0	410,1	81,3	95,3
10	75,0	105,1	75,0	256,3
11	90,0	22,6	90,0	1,0
12	90,0	22,6	94,7	13,9
13	75,0	105,1	83,3	58,9
14	100,0	217,6	100,0	80,8
15	80,0	27,6	80,0	121,2

[4] Caso necessite saber o título do artigo, ele se encontra na Tabela 10, ambas possuem mesma numeração.

Nº[4]	Recall (%)	(Recall - Média Aritmética Recall)^2	Precision (%)	(Precision - Média Precision)^2
16	95,0	95,1	100,0	80,8
17	80,0	27,6	88,9	4,5
18	85,0	0,1	89,5	2,4
19	100,0	217,6	100,0	80,8
20	65,0	410,1	86,7	18,9

Fonte: Desenvolvido pelo autor, média Aritmética Recall= 85,3% e média Aritmética Precision= 91,0%

Dos resultados retirados da Tabela 10 e da Tabela 11, obtivemos um Recall médio de 85,3% com uma variação em torno da média (desvio padrão) de 11,8%. Com relação ao *Precision* obtivemos um *Precision* médio de 91,0% com uma variação em torno da média (desvio padrão) de 7,6%. Sendo assim, nosso caso de uso apresentou um *Recall* variando entre (73,5% a 97,1%) e um *Precision* variando entre (83,4% a 98,6%).

Durante a execução do caso de uso (às quatro horas estabelecidas para a extração das informações) pudemos observar, acompanhando o processo de registro de alguns artigos dos cinco usuários, que a principal fonte de erro no processo de extração foi originada durante o processo de anotação. A maioria devido a campos que foram deixados em branco, pois campos em branco são considerados divergentes do gabarito. O processo de copiar e colar informações da fonte de dados para o formulário de anotação também causou problemas de extração. Quando o conteúdo era colado no campo do formulário seu respectivo estilo era sobreposto pelo estilo da fonte de dados de origem. No momento da cola, ao invés de habilitar o campo do formulário para receber o conteúdo, o usuário habilitava a caixa de texto do mesmo. Como o estilo da fonte de dados original não fazia parte do arquivo adaptador, o conteúdo não era extraído. Em alguns casos, durante o processo de cola, por distração do usuário, ele colava o conteúdo copiado no campo errado do formulário. Nesse caso, o conteúdo extraído é armazenado numa propriedade errada da ontologia, nesse caso o processo de extração acontece de forma correta, só que o conteúdo da propriedade da ontologia apresenta um conteúdo errado, comprometendo os valores dos indicadores *Recall e Precision.*

4.5 AVALIAÇÃO DO USO DA FERRAMENTA

Para a avaliação do uso da ferramenta, foi aplicado ao final do experimento o formulário de pesquisa de satisfação (Anexo J). Este formulário foi baseado nas características de Qualidade

de Software (Usabilidade) detalhadas na Tabela 12 a seguir, seguindo a norma ISSO/IEC 9126-1.

Tabela 12 - Características da Qualidade de Software relacionadas à Usabilidade segundo a ISO/IEC 9126-1

Característica	Pergunta Chave
Intelegibilidade	É fácil entender o conceito e a aplicação?
Apreensibilidade	É fácil aprender a usar?
Operacionalidade	É fácil de operar e controlar?

Fonte: Desenvolvido pelo autor

A tabulação dos resultados, do processo de avaliação da utilização da ferramenta no experimento está detalhada na Tabela 13 a seguir. Nesta tabela, temos as respostas dos cinco usuários participantes do experimento, em que cada um respondeu as perguntas chaves apresentada na Tabela 12.

Tabela 13 - Tabulação dos resultados da aplicação do formulário de Avaliação da Utilização da Ferramenta de Extração

Operador	Intelegibilidade	Apreensibilidade	Operacionalidade
Usuario1	Sim	Sim	Sim
Usuario2	Sim	Sim	Sim
Usuario3	Sim	Sim	Não
Usuario4	Sim	Não	Sim
Usuario5	Sim	Sim	Sim
Total	100%	80%	80%

Fonte: Desenvolvida pelo Autor.

Diante dos resultados apresentados na Tabela 13, observa-se que a ferramenta de extração *InLocoLoader* foi avaliada com resultados acima de 70% para as principais características relacionadas a Qualidade de Software no quesito Usabilidade. Segundo os usuários participantes do caso de uso, a ferramenta de extração não atingiu a pontuação máxima em

relação à facilidade de uso, devido a falta de um módulo de extração de informação em lote, em que várias fontes de dados pudessem ser extraídas ao mesmo tempo.

4.6 POSSÍVEIS UTILIZAÇÕES DAS ONTOLOGIAS RESULTANTES DO PROCESSO DE EXTRAÇÃO

Uma das etapas mais importantes de um processo de produção científica (artigo, monografia, dissertação ou tese) é a revisão da literatura. Em alguns casos há necessidade de se realizar uma revisão sistemática mais precisa. Nessa etapa é feita toda a leitura e a minuciosa catalogação de tudo que foi lido e o respectivo fichamento das obras, que serão citadas no novo projeto. Na maioria das vezes, as pessoas fazem consultas na Internet, nos sítios de busca especializados como Google Acadêmico, em bibliotecas digitais, como ACM; vão a bibliotecas físicas e registram os dados dessas obras em fichas digitais, ou até mesmos nas antigas fichas de cartolina, gerando assim seu banco de informações sobre citações de um determinado domínio. Essa metodologia é limitada e sujeita a erros, pois a relação entre os assuntos tratados nos diversos textos fica na cabeça do pesquisador, que os lê e com o aumento do volume das obras lidas, essa informação acaba se perdendo.

A principal utilização da ontologia *Research Card*, *template* de saída desse nosso caso de uso, é justamente ser um local onde se podem compartilhar informações relevantes sobre obras científicas dos mais diversos assuntos catalogados, de forma a assegurar que as relações entre essas informações serão registradas e preservadas, tanto numa revisão da literatura, como numa revisão sistemática. Além dessa função de compartilhamento das informações relevantes e suas relações do domínio de publicação científica, as ontologias desse caso de uso representam um levantamento desse domínio, constituindo-se num elemento fundamental para a construção de uma base de conhecimento para a área de publicações científicas.

Este trabalho contribuiu para o refinamento e consolidação da ontologia *Research Card* que, anteriormente se limitava a um esboço utilizado para fins didáticos.

4.7 RESUMO DO CAPÍTULO

Nesse capítulo foi apresentado um estudo de caso (*Research Card*) para validar a nossa ferramenta de extração *InLocoLoader*. Para isso, utilizamos o domínio de publicação, mais

especificamente artigos científicos, e definimos um formato de saída baseado em ontologias. Foram utilizados vinte artigos durante o processo de extração para validar a ferramenta. Todo o processo foi ilustrado, através da impressão das telas do Protegé no decurso de todo o processo. Foi feita uma análise dos resultados obtidos no processo de extração do caso de uso baseado nos indicadores de desempenho *Recall e Precision,* bem como uma análise do uso da ferramenta.

5 CONSIDERAÇÕES FINAIS

5.1 ASPECTOS PRINCIPAIS

Este capítulo visa tecer considerações finais sobre os resultados obtidos nessa pesquisa, destacando as principais contribuições resultantes, bem como as limitações e temas que não foram suficientemente explorados e que podem suscitar novos estudos.

O presente trabalho partiu da seguinte questão norteadora: Como extrair informações relevantes dessa grande quantidade de textos, publicada nas bibliotecas digitais na Web e em outros repositórios em contínuo crescimento, obedecendo a um formato específico de saída, de maneira eficiente e simples para o leitor e adaptável a diferentes domínios de conhecimento? Para responder essa questão foram estabelecidas as hipóteses de que a Metodologia de Anotação Semântica in Loco forneceria a base para a construção de uma solução de extração de informações alternativa e complementar às técnicas tradicionais de extração e que o conceito de perfis adaptáveis a domínio estenderia essa metodologia, possibilitando a essa solução, uma maior versatilidade de adaptação a novos domínios, difícil de ser encontrada nas tradicionais técnicas de extração (Extração manual, extração semiautomática e automática).

A ferramenta de IE *InLocoLoader* foi validada utilizando-se para isso o caso de uso (Research Card), desenvolvido para o domínio de publicações científicas. Durante o processo de validação desse caso de uso, as hipóteses levantadas foram confirmadas, a ferramenta de IE mostrou-se operacional, extraindo informações previamente anotadas implicitamente no texto, utilizando o padrão de anotação desenvolvido (através dos metadados e estilos produzidos). Configurando-se numa solução alternativa para os casos de fontes de dados não legadas[5] e numa solução complementar nos casos de fontes legadas[6].

---

[5] Fonte de dados não legadas, à anotação é feita em conjunto com a produção do conteúdo de forma implícita.

[6] Fonte de dados legadas: o processo de anotação é feito numa fase posterior a fase de construção do conteúdo.

Com base no exposto nas análises efetuadas no caso de uso Research Card, observa-se que os objetivos foram atingidos, uma vez que:

- ✓ Ficou evidenciado o uso do princípio da interoperabilidade da Semântica In Loco no caso de uso, pois a ferramenta de IE *InLocoLoader* utiliza o OWL - um dos padrões abertos da Web Semântica, para mapear as anotações realizadas na fonte de texto de origem, para as ontologias Research Card e Card, que compõem o *templete* de saída;
- ✓ O mesmo ocorreu com o princípio da Persistência Semântica. Como os elementos das anotações feitas no caso de uso foram conectados a ontologias unificadoras, elas irão garantir uma interpretação equivalente em diferentes contextos, fornecendo uma persistência semântica ao longo das transformações.

Entretanto, é fundamental que o *InLocoLoader* seja validado em diferentes domínios e por diferentes grupos, para que possa ser efetivamente testada a sua capacidade de adaptação a novos domínios utilizado o conceito de perfis. Dessa forma será possível uma melhor avaliação dos limites e possibilidades proporcionadas pelo uso do *InLocoLoader*.

Finalmente, esse trabalho evidencia que a utilização conjunta da Metodologia de Anotação Semântica in Loco e o conceito de Perfil associado a domínio permitiu o desenvolvimento de uma solução de extração de informação (InLocoLoader), alternativa e complementar às soluções tradicionais de extração de informação adaptável a novos domínios.

5.2 PRINCIPAIS CONTRIBUIÇÕES

- A extensão da metodologia Semântica In Loco para oferecer suporte a perfis associados a ontologias.
  - Como parte desta extensão foi definido um modelo abstrato para a caracterização de fontes de dados para extração, bem como um modelo para especificar o mapeamento destas fontes com ontologias.
  - Definição de um modelo de adaptação baseado em uma especificação XML de identificação de itens de extração e seu mapeamento (arquivo adaptador).
- A extensão do software de criação e gerenciamento de ontologias *Protegé* com a funcionalidade de extração de informação de bases textuais, através do *Plug-in*

InLocoLoader, capaz de extrair dados relevantes de documentos em formato texto e armazená-los em ontologias.

- Desenvolvimento de um padrão de anotação semântica (formado por um conjunto de metadados e estilos) utilizados no caso de uso para validar a ferramenta InLocoLoader;
- Contribuição na criação das ontologias Research Card, Card e Subject, aprimoradas para o caso de uso.

5.3 TRABALHOS FUTUROS

Como trabalhos futuros será de grande relevância o desenvolvimento de uma interface gráfica para criação e alteração do adaptador (arquivo *Adapter.xml),* utilizado no processo de migração da solução de extração para novos domínios. Além disso, a adaptação da ferramenta para a possibilidade da realização de extrações através do processamento em lote das fontes de dados, tornando optativo o trabalho relativo à carga unitária de cada fonte de texto de origem para a ferramenta de IE *InLocoLoader*.

REFERENCIAS

ABITEBOUL, S. Querying semistructured data. In: DATABASE theory - ICDT '97, nº 6, 1997, Delphi, Greece. Technical Report. Berlin/ Heidelberg: Stanford InfoLab Publication Note, 1997. p.1-18.

BORST, W. N. **Construction of engineering ontologies for knowledge sharing and reuse**. Enschede: Universiteit Twente, 1997. 243p.

BREITMAN, K. K. **Web semântica:** a internet do futuro. Rio de Janeiro: LTC, 2005. 188p.

BREITMAN, K. K.; LEITE, J. C. S. P. Ontologias: como e porque criá-las. In: JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA, 1., 2004. **Anais...** Porto Alegre: Editora da Sociedade Brasileira de Computação, 2004, p. 3-53.

BROWN, L. D. E. Electronic publishing programs. **Issues in science and technology librarianship,** jan. 1986. Disponível em: <http://www.library.ucsb.edu/istl/C.-H>. Acesso em: 15 jul. 2008.

CHANG, C. H. et al. **A survey of web information extraction systems.** 10. ed. New York: IEEE TKDE, 1996. 1411-1428p.

COWIE, J.; LEHNERT, W. Information extraction. **Communications of the ACM**, New York, p.80-91, 1996.

CONTESSA, D. F. **Um serviço de geração de metadados compatível com o padrão OAI para o sistema JEMS**. 2006. 83 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006.

CRESPO, A. et al. **A survey of semi-automatic extractions and transformation**. Califórnia: Stanford Publication, 1994. Disponível em: <http://www-db.stanford.edu/ crespo/publications/> Acesso em: 12 mar. 2008.

CUNNINGHAM, H. **Information extraction**. Automatic encyclopedia of language and linguistics 2. ed. Sheffield UK: [s.n.], 2005. 22p.

DAVIES, J. S. **Semantic web technologies:** trends and research in ontology based systems. London: Wiley, 2006. 308p.

DREILINGER, D; HOWE, A. Experiences with selecting search engines using metasearch. **ACM Transactions on Information Systems**, New York, 1997. 95 – 222p.

DUBLIN CORE. **DCMI Metadata Terms**. New York: Dublin Core Publication, 2008. Disponível em: <http: //dublincore.org/ documents /dcmi-terms/> Acesso em: 4 out. 2007.

DUBLIN CORE. **Dublin Core Qualifiers**. New York: Dublin Core Publication, 2008. Disponível em: <http: //dublincore.org/ documents/2000/ 07/11/dcmes-qulifiers/> Acesso em: 4 out. 2007.

DUBLIN CORE. **Expressing Qualifiers Dublin Core in RDF/XML**. New York: Dublin Core Publication, 2009. Disponível em: <http: //dublincore.org/ documents/dcq-rdf-xml/> Acesso em: 4 mar. 2009.

EUZENAT, J. Eight questions about semantic web annotations. 2. ed. **Intelligent System**, v. 17, n. 2, p. 55-62, 2002.

FENSEL, D. **Ontologies:** silver bullet for knowledge management and eletronic commerce. 2. ed. New York: Springer Verlag NY, 2003. 162p.

FERREIRA, A. B. **Dicionário Aurélio eletrônico:** século XXI. 3. ed. Rio de Janeiro: Nova Fronteira ; Lexicon Informática, 1999.

KLEIN, B.; FRANKHAUSER, P. Error tolerant document structure analysis. In: INTERNATIONAL JOURNAL ON DIGITAL LIBRARIES, 4., Darmstadt, **Proceedings …** Darmstadt, Germany: Springer, 1998. v.1, 344-357p.

FUHR, N. Probabilistic Datalog — a logic for powerful retrieval methods. In: RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 18, Seattle, **Proceedings …**, Washington, United States: SIGIR, 1995. 282-290p.

GILLILAND-SWETLAND, Anne J. Setting the stage: defining metadata. In: INTRODUCTION TO METADATA: PATHWAYS TO DIGITAL INFORMATION, 2., 2000, Los Angeles. **Proceedings …** Los Angeles: Getty Information Institute, 2006, p.1- 12.

GÓMEZ-PÉREZ, A; FERNANDEZ-PÉREZ, M.; CORCHO, M.. **Ontological engineering**. *ECAI2004 tutorial summary,* 1999. p.1 – 42.

GRISHMAN, R. ; SUNDHEIM, B. Message understanding conference-6: A brief history. In: COMPUTATIONAL LINGUISTICS, 16., 1996, Copenhagen. **Proceedings ….** Copenhagen: Association for Computational Linguistics, 1996. p. 466-471.

GRISHMAN, R. Information extraction: techniques and challenges. In: TH INT'L CONF. ON COMPUTATIONAL LINGUISTICS, 16., 2003. **Proceedings …**Copenhagen, Heidelberg: Springer Berlin / Heidelberg, p. 10-27, 2003. .

GRUBER, T. R.. **A Translation approach to portable ontology specifications**. Stanford: Stanford University, 1993, p.199 -220.

GRUNINGER, M.; FOX, M. Methodology for the Design and Evaluation of Ontologies. In: KNOWLEDGE SHARING, 5.,Canada, **Proceedings…** Canada, 1995. v5, n. 4, p. 1-10.

GUARINO, N.; OBERLE, D.; STAAB, S. What Is an Ontology? IN: HANDBOOK ON ONTOLOGIES, 2., 2009, Heidelberg. **Proceedings …** Heidelberg: Handbook on Ontologies, 2009. p. 1-17.

GOLBREICH, C. ; WALLACE, E. K. **OWL 2 Web Ontology Language New Features and Rationale -- W3C Recommendation** 27. 2009. Disponível em: <w3.org/TR/2009/REC-owl2-new-features-20091027/> Acesso em: 27out. 2009.

HABING, T. C.; COLE, T. W.; MISCHO, W. H. Qualified Dublin Core using RDF. Sci-Tech Journal Articles In: DUBLIN CORE AND METADATA APPLICATIONS, 4., Tokyo Japan, 2001. **Proceedings …** Tokyo Japan : Dublin Core Publication , 2001, v. 1, p. 1-5.

HAHN, U.; SCHNATTINGER, K. Towards text knowledge engineering. In: INNOVATIVE APPLICATIONS OF ARTIFICIAL INTELLIGENCE, 15., Madison, Wisconsin. 1998. **Proceedings …** Madison, Wisconsin, United States: American Association for Artificial Intelligence, Menlo Park, CA, 1998, p. 524-531.

IBCT. **Série Histórica Consolidada Por Data de Publicação na BDTD**. 2010. Disponível em: <http://bdtdj.ibict.br/indicadores/graficoSHCDP.jsp? cod1=&cod2=&cod3=> Acesso em: 10 ago. 2010.

KAUFMANN, M. Proceedings of the Fifth Message Understanding Conference (MUC-5), **Journal of Artificial Intelligence Research,** Virginia, v.2, n.1, p.89-110, 1994.

KAUFMANN, M. Proceedings of the Fourth Message Understanding Conference (MUC-4). **Journal of Artificial Intelligence Research,** Virginia, v. 19, n.3, p.1-19, 1992.

MAEDCHE, A.; NEUMANN, G.; STAAB, S. Bootstrapping an ontology-based information extraction system. In: INTELLIGENT EXPLORATION OF THE WEB, P. S. SZCZEPANIAK, J. SEGOVIA, J. KACPRZYK, AND L. A. ZADEH, EDS. STUDIES, 5, Germany . **Proceedings...** Germany: Physica-Verlag GmbH, Heidelberg, Germany, v. 5, n.1 p. 345-359, 2003.

MANOLA, F.; MILLER, E. **RDF Primer – W3C Recommendation 10 February 2004**. Disponível em: <w3.org/TR/2004/REC-rdfprimer-20040210/> Acesso em: 8 jun. 2007.

MARSHALL, C. Annotation: from paper books to the digital library. **ACM International Conference**, Philadelphia, v. 3, n. 1, p.131-140, 1997.

MOTA, Matheus Silva. **Ferramenta para extração de estruturas e metadados baseada na semântica in loco e padrões de anotação.** 2008. 40 f. Monografia (Graduação) – Curso de Bacharelado em Ciência da Computação) - Universidade Salvador, Salvador, 2008.

MAYNARD, D. ; YANKOVA, M. Ontology-based information extraction for market monitoring and technology watch. In: END USER APECTS OF THE SEMANTIC WEB, 2., 2005, Heraklion. **Proceedings…** Heraklion, Crete: [s.n], 2005. p. 1-10.

MOTIK, B.; PATEL-SCHNEIDER, P. F.; PARSIA, B. **OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax -- W3C Recommendation.** Disponível em: <w3.org/TR/2009/REC-owl2-syntax-20091027/> Acesso em: 27 out. 2009.

NOY, N.; MCGUINESS, D. **Ontology Development 101:** a guide to creating your first ontology. KSL ,Technical Report, Stadford University, 2001.

OMG. **Meta Object Facility (MOF) Core Specification 2.4**. Disponível em <omg.org/spec/MOF/2.4/Beta2/>. Acesso em: 27 out. 2010.

PITTS-MOULTIS, N.; KIRK, C. **XML black book:** solução e poder. New York: Makron Books, 2004. 1-350p.

POPOV, B.; KIM, K. Semantic annotation platform. Lecture Notes. **Computer Science,** p.15.40, n. 2, 2003.

PROTEGÉ. **Protegé.** Disponível em: **<**http://protege.stanford.edu /> Acesso em: 2 jul. 2008.

RAMALHO, J. C. Anotação estrutural de documentos e sua semântica: especificação da sintaxe, semântica e estilo para documentos. 2000. 287f. Tese (Doutorado em Informática) - Universidade do Minho, Portugal. 2000.

ROCHA, R. P. Metadados, web semântica, categorização automática: combinando esforços humanos e computacionais para a descoberta e uso dos recursos da web. **Em Questão,** Porto Alegre, 2004. v. 10, n. 1, p. 109 -121, jan./jun. 2004.

SANTANCHÈ, A. Otimizando a anotação de objetos de aprendizagem através da semântica In Loco. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 18., 2007, São Paulo. **Anais…**, 2007. p. 526-535.

SANTANCHÈ, A.; SILVA, L. A. M. Document-centered learning object authoring. **IEEE Learning Technology Newsletter**, v. 12, p. 58-61, 2010.

SMITH, M. K.; WELTY, C.; MCGUINNESS, D. L. **OWL Web Ontology Language Guide – W3C Recommendation.** 10 feb. 2004. Disponível em: <http://www.w3.org/TR/2004/REC-owl-guide-20040210/> Acesso em: 12 set. 2007.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge engineering: Principles and methods. **Data & Knowledge Engineering**, 1998, v.25, n.1-2, p. 161-197.

TURMO J. Adaptive information extraction. **ACM Comput. Surv,** v. 38, n. 4, p. 38, 2006.

VERGOTTI, M. ; CAIRO, A. Mais informação do que a gente aguenta. **Época**, Rio de Janeiro, maio 2010. Seção Primeiro Plano. Disponível em: <http://revistaepoca.globo.com/Revista/Epoca/0,,EMI142088-15215,00.html>. Acesso em: 25 maio 2010.

W3C Semantic Web Activity Group. **Architecture domain.** New York: 2003. Disponível em: http://www.w3.org/Addressing/ Acesso em: 5 jan 2007.

USHOLD, M.; GRUNINGER, M. Ontologies: principles, methods and applications**.** **Knowledge Engineering Review**, v. 11, n. 2, p. 93-136, 1996.

W3C OWL WG. **OWL 2 Web Ontology Language -- Document Overview -- W3C Recommendation.** Disponível em: <w3.org/TR/2009/REC-owl2-overview-20091027/> Acesso em: 27 out. 2009.

WEIBEL, S. Metadata: The Foundations of Resource Description. **D-Lib Magazine**, Reston, jul. 1995. Disponível em: <www.dlib.org>. Acesso em: 15 dez. 2002.

APÊNDICE A – FORMULÁRIO UTILIZADO NO CASO DE USO PARA EXTRAÇÃO DE INFORMAÇÃO

Ficha de Pesquisa

Ob. Assinale com um "X" o tipo de publicação

Tipo de Publicação	Artigo	Livro	Manual	Relatório Técnico	Tese de Mestrado	Tese de PHD
	X					

Identificador:	**Nunes2008**
Titulo:	Uma Arquitetura para Recuperação de Informação Baseada em Semântica e sua Aplicação no Apoio a Jurisprudência
Autor:	**NUNES M ANSELMO** **FILETO RENATO**
Assunto:	**Recuperação de Informação** **Anotação Semântica**
Palavras chaves:	**Anotação** **Recuperação** **Ontologia**
Abstract:	**Recuperação de informação baseada em mecanismos como buscas por palavras-chaves apresenta problemas de precisão e revocação. Este artigo explora o uso de ontolgias e anotações semânticas para tratar esses problemas, considerando aplicações em jurisprudência. Ele apresenta uma arquitetura de aplicação para extração de informação de documentos baseados em semântica e faz uma análise comparativa de ferramentas para a geração de anotação semânticas, um modulo fundamental da arquitetura proposta.**

Dia da Publicação:	Dia	Mês	Ano
	7	**11**	2005

Ob. Preencher o campo "Dia" sem o zero na frente. Ex. 1, 2, 3, 4, 5, 6, 7, 8, 9.

Número de Páginas:	De:	Até:
	1	11

Cartão				Nº
				5
Abstract do Leitor:	Uso de ontologias e anotação semântica para melhorar a precisão em recuperação de informação em aplicação de jurisprudência.			
Dia da Catalogação:		Dia	Mês	Ano
		7	6	2009
Referências	Marques2007 Manoel2007			

Ob. Preencher o campo "De" sem o zero na frente. Ex. 1, 2, 3, 4, 5, 6, 7, 8, 9.

Tópico	Título	Recupera
	Conceito	X
	Aspecto Relevante	
	Tópico a Explorar	
	Idéia	
Recorte do artigo	A anotação semântica de um documento descreve o seu conteúdo pela associação a conceitos e instâncias descritos na ontologia. Os padrões e formalismos da web semântica permitem que esses metadados baseados em semântica (ontologias e anotações) sejam processáveis por computador. Desta forma, a especificação e a resolução de consultas passam a se beneficiar das relações semânticas descritas na ontologia, possibilitando o tratamento de ambigüidades e a recuperação de documentos semanticamente relacionados aos termos expressos na consulta.	
Página	2	
Comentários	Conceito Interessante para ser citado na Dissertação	
Assuntos	Anotação Recuperação	

Tópico	Título	Extração de Informação
	Conceito	
	Aspecto Relevante	X
	Tópico a Explorar	
	Idéia	
Recorte do artigo	A extração automática de informação de documentos é uma alternativa para a geração de ontologias e anotação semântica	
Página	4	
Comentários	Aspecto Relevante a ser observado	
Assuntos	Extração	

APÊNDICE B – CÓDIGO FONTE CLASSE CONSTRUTOR.JAVA

```
package inLoco;

import inLoco.xmlmap.TDocument;
import inLoco.xmlmap.TProperty;

import java.io.File;
import java.util.Hashtable;
import java.util.List;
import java.util.Stack;

import javax.swing.JTextArea;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;

import ddex.opentext.CharSequenceStyle;
import ddex.opentext.ParagraphStyle;
import ddex.opentext.builder.ITextBuilder;
import edu.stanford.smi.protegex.owl.model.OWLDatatypeProperty;
import edu.stanford.smi.protegex.owl.model.OWLIndividual;
import edu.stanford.smi.protegex.owl.model.OWLModel;
import edu.stanford.smi.protegex.owl.model.OWLNamedClass;
import edu.stanford.smi.protegex.owl.model.OWLObjectProperty;

public class Construtor implements ITextBuilder {

	private static String estilo;
	private static String conteudo;
	private JTextArea text;
	private OWLModel owlm;

	private Stack<OWLIndividual> context = new Stack<OWLIndividual>();
	private OWLIndividual currentInstance;
	private Stack<List<TProperty>> contextProperties = new Stack<List<TProperty>>();
	private List<TProperty> currentProperties;

	private OWLNamedClass articleClass, cardClass, topicClass, subjectClass;
	private OWLIndividual topicInstance;

	private String idLastCard = "rc", idLastTopic = "tp";

	// private Hashtable<String, TProperty> mappings = new Hashtable<String,
TProperty>();

	public void setTextComponent(JTextArea theComponent) {
		this.text = theComponent;
```

```
        }

        public void setOWLModel(OWLModel owlm) {
                this.owlm = owlm;
    readXMLMap();
        }

        public void readXMLMap()
        {
          try {
        JAXBContext jc = JAXBContext.newInstance("inLoco.xmlmap");

        Unmarshaller unmarshaller = jc.createUnmarshaller();

        JAXBElement<TDocument> xml =
(JAXBElement<TDocument>)unmarshaller.unmarshal(new
File(getClass().getResource("xmlmap/articlew4.xml").getPath()));

        TDocument document = xml.getValue();
        currentProperties = document.getProperty();

        /*
        int level = 1;
        for (TProperty p: properties) {
          System.out.println("estilo: " + p.getStyle() + "; propriedade: " + p.getOwlproperty()
+
                        "; tipo: " + p.getRange());
          mappings.put(p.getStyle(), p);
        }
        */
    } catch (JAXBException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

        }

        /* (non-Javadoc)
         * @see ddex.opentext.builder.ITextBuilder#foundCharSequence(int, int,
java.lang.String, int, ddex.opentext.CharSequenceStyle)
         */
        public void foundCharSequence(int arg0, int arg1, String arg2, int arg3,
                        CharSequenceStyle arg4) {
                if (!arg4.getName().contains("Fonte")) {
                        estilo = arg4.getName();
                        conteudo = arg2;
                        text.append(estilo + ": ");
                        text.append(conteudo + "\n");
```

```
// INÍCIO PARTE RELATIVA AO ARTIGO
*****************************************************


                    // Getting the title property

if (arg4.getName().contains("Identificador"))
   currentInstance = articleClass.createOWLIndividual(conteudo);


// TProperty property = mappings.get(arg4.getName());

TProperty property = null;
String docStyle = arg4.getName();
boolean empty = false;
do {
   for (TProperty p: currentProperties) {
      if (docStyle.equalsIgnoreCase(p.getStyle()))
         property = p;
   }
   if (property == null) {
      if (contextProperties.empty())
         empty = true;
      else {
         currentProperties = contextProperties.pop();
         currentInstance = context.pop();
      }
   }
} while (property == null && !empty);


if (property != null)
{
   if (property.getRange().equalsIgnoreCase("xsd:string") ||
      property.getRange().equalsIgnoreCase("xsd:int"))
   {
      OWLDatatypeProperty owlp =
owlm.getOWLDatatypeProperty(property.getOwlproperty());
      if (property.getRange().equalsIgnoreCase("xsd:int"))
         currentInstance.addPropertyValue(owlp, new Integer(conteudo));
      else
         currentInstance.addPropertyValue(owlp, conteudo);
   } else {

      boolean isReference = (property.isReference() == null) ? false :
property.isReference();
      OWLIndividual objInstance;
      OWLObjectProperty owlp;
      if (isReference) {
         objInstance = owlm.getOWLIndividual(conteudo);
```

```
                if (objInstance != null) {
                    owlp = owlm.getOWLObjectProperty(property.getOwlproperty());
                    currentInstance.addPropertyValue(owlp, objInstance);
                }
            } else {
                // pega a classe especificada no range
                OWLNamedClass objClass =
owlm.getOWLNamedClass(property.getRange());
                // cria uma instância da classe especificada
                objInstance = createGetIndividual(objClass, conteudo.trim());

                owlp = owlm.getOWLObjectProperty(property.getOwlproperty());
                currentInstance.addPropertyValue(owlp, objInstance);

                if (property.getIproperty() != null)
                {
                    OWLDatatypeProperty owlip =
owlm.getOWLDatatypeProperty(property.getIproperty());
                    objInstance.addPropertyValue(owlip, conteudo);
                }

                if (property.getProperty() != null) {
                    context.push(currentInstance);
                    currentInstance = objInstance;
                    contextProperties.push(currentProperties);
                    currentProperties = property.getProperty();
                }
            }
        }

    }

                    // Getting the subject property
    /*
                    if (arg4.getName().contains("Assunto")) {
                        createSubject(currentInstance, conteudo);
                    }
                    */

    // Getting the keyword property
    if (arg4.getName().contains("Palavra Chave")) {
        OWLDatatypeProperty keywordProperty =
owlm.getOWLDatatypeProperty("card:keywords");
        currentInstance.addPropertyValue(keywordProperty, conteudo);
    }


                    // Getting the abstract property

                    if (arg4.getName().contains("Abstract")) {
```

```
                                OWLDatatypeProperty abstractProperty =
owlm.getOWLDatatypeProperty("card:abstract");
                                currentInstance.setPropertyValue(abstractProperty, conteudo);
                        }

                        // Getting the dayPublished property
                        if (arg4.getName().contains("Dia")) {
                                OWLDatatypeProperty pageInitialProperty =
owlm.getOWLDatatypeProperty("card:dayPublished");
                                currentInstance.setPropertyValue(pageInitialProperty,
Integer.parseInt(conteudo.trim()));
                        }

                        // Getting the monthPublished property
                        if (arg4.getName().contains("Mes")) {
                                OWLDatatypeProperty monthPublishedProperty =
owlm.getOWLDatatypeProperty("card:monthPublished");
                                currentInstance.setPropertyValue(monthPublishedProperty,
Integer.parseInt(conteudo.trim()));
                        }

                        // Getting the yearPublished property
                        if (arg4.getName().contains("Ano")) {
                                OWLDatatypeProperty yearPublishedProperty =
owlm.getOWLDatatypeProperty("card:yearPublished");
                                currentInstance.setPropertyValue(yearPublishedProperty,
Integer.parseInt(conteudo.trim()));
                        }

                        // Getting the pageFinal property
                        if (arg4.getName().contains("Final")) {
                                OWLDatatypeProperty pageFinalProperty =
owlm.getOWLDatatypeProperty("card:pageFinal");
                                currentInstance.setPropertyValue(pageFinalProperty, conteudo);
                        }

                        // Getting the pageInitial property
                        if (arg4.getName().contains("Inicial")) {
                                OWLDatatypeProperty titleProperty =
owlm.getOWLDatatypeProperty("card:pageInitial");
                                currentInstance.setPropertyValue(titleProperty, conteudo);
                        }

                // PARTE FINAL RELATIVA AO ARTIGO
************************************

                        // INICO DA PARTE REFERENTE AOS CARTÕES
                ****************************************

                        // criando uma instância do cartão
```

```
                    // FIM DA PARTE REFERENTE AOS CARTÕES
************************************

        // INÍCIO DA PARTE REFERENTE AOS TOPICOS
************************************

            // Instânciando um tópico



      }
        }

   // FIM DAS ROTINAS RELATIVAS A TÓPICOS
****************************************


        // *****************    ROTINAS DE USO GERAL
**************************************

        private OWLIndividual createGetIndividual(OWLNamedClass theClass, String id)
        {
           OWLIndividual instance = owlm.getOWLIndividual(id);

           if (instance == null)
        instance = theClass.createOWLIndividual(id);

           return instance;
        }




        public void foundCharSequenceEnd(int index) {
                // TODO Auto-generated method stub

        }

        public void foundCharStyleProperties(int uniqueId, int basedId,
                        String name, CharSequenceStyle cp) {
                // TODO Auto-generated method stub

        }

        public void foundDocumentBegin() {
```

```
		// TODO Auto-generated method stub

	}

	public void foundDocumentEnd() {
		// TODO Auto-generated method stub

	}

	public void foundFilePath(String filePath) {
		// TODO Auto-generated method stub

	}

	public void foundImage(int index, String sugestedName,
			String sugestedExtension, byte[] imgData) {
		// TODO Auto-generated method stub

	}

	public void foundImageNumber(int number) {
		// TODO Auto-generated method stub

	}

	public void foundParagraph(int index, String fullContent,
			int styleUniqueID, ParagraphStyle style) {
		// TODO Auto-generated method stub

	}

	public void foundParagraphCharSequenceNumber(int number) {
		// TODO Auto-generated method stub

	}

	public void foundParagraphEnd(int pageNumbers) {
		// TODO Auto-generated method stub

	}

	public void foundParagraphNumber(int number) {
		// TODO Auto-generated method stub

	}

	public void foundParagraphStyleProperties(int uniqueId, int basedId,
			String name, ParagraphStyle pp) {
		// TODO Auto-generated method stub
```

```
        }

        public void foundStyle(int uniqueId, int basedId, String name,
                        boolean isParagrafStyle, boolean isCharStyle) {
                // TODO Auto-generated method stub

        }

        public void foundStyleNumber(int pageNumbers) {
                // TODO Auto-generated method stub

        }

}
```

APÊNDICE B – CÓDIGO FONTE CLASSE INLOCOLOADER.JAVA

```
package inLoco;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;

import javax.swing.JButton;
import javax.swing.JTextArea;

import ddex.opentext.director.DOCReader;
import edu.stanford.smi.protege.model.KnowledgeBase;
import edu.stanford.smi.protege.widget.AbstractTabWidget;
import edu.stanford.smi.protegex.owl.ProtegeOWL;
import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.model.OWLModel;
import edu.stanford.smi.protegex.owl.model.impl.OWLNamespaceManager;
import edu.stanford.smi.protegex.owl.model.util.ImportHelper;



public class InLocoLoader extends AbstractTabWidget
{
    private static final long serialVersionUID = -3182427134410227151L;

    // DDEx reader:
            private DOCReader leitor;
   // Concrete Builder InLocoLoader

   private Construtor construtor;
  // Visual components:

   private JButton loadOntology;
   private JButton loadDocument;
   private JTextArea text;

   public void initialize()
   {
      // Rotinas do Plug-in InLocoLoader
      this.setLabel("In Loco Loader");

      loadOntology = new JButton("Import Ontologies");
```

```
    loadOntology.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            try {
                importOntologies(getProtegeOntology());
            } catch (URISyntaxException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });

    loadDocument = new JButton("Load Document");
    loadDocument.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            loadDocument(getProtegeOntology());
        }
    });

    // Text Area: It will show the extracted information
    text = new JTextArea();

    setLayout(new BorderLayout());
    add(loadOntology, BorderLayout.NORTH);
    add(loadDocument, BorderLayout.SOUTH);
    add(text, BorderLayout.CENTER);
}

public OWLModel getProtegeOntology()
{
        // recupera a ontologia da memoria do protege
    OWLModel owlm = null;

    try {
        KnowledgeBase kb = getKnowledgeBase();
        if (kb instanceof OWLModel)
            owlm = (OWLModel)kb;
    } catch (Exception error) {
        error.printStackTrace();
    }

    return owlm;
}

public static void importOntologies(OWLModel owlm) throws URISyntaxException
{

    OWLNamespaceManager nsm =
(OWLNamespaceManager)owlm.getTripleStoreModel().getTopTripleStore().getNamespace
Manager();
```

```
    nsm.setPrefix("http://purl.org/net/researchcard/card/", "card");
    nsm.setPrefix("http://purl.org/net/researchcard/subject/", "subj");
    // nsm.setPrefix("http://purl.org/net/researchcard/card-i/", "cinst");

    ImportHelper importHelper = new ImportHelper((JenaOWLModel)owlm);

    URI uriSub = InLocoLoader.class.getResource("subject.owl").toURI();
    URI uriCard = InLocoLoader.class.getResource("card.owl").toURI();
    // URI uriInst = InLocoLoader.class.getResource("card-i.owl").toURI();

    importHelper.addImport(uriSub);
    importHelper.addImport(uriCard);
    // importHelper.addImport(uriInst);

    try {

        // importOntologies importa as ontologias para o protegé
      importHelper.importOntologies(true);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  public void loadDocument(OWLModel owlm)
  {
    text.append("Plugin rodando...\n");

    // Rotinas do DDEx para Extrair o conteudo do arquivo c:\sample.doc
    // marcados pelos estilos in Loco e inclui na ontologia card

    leConf();

    leitor = new DOCReader(getClass().getResource("sample5.doc").getPath());
    construtor = new Construtor();
    construtor.setTextComponent(text);
    construtor.setOWLModel(owlm);
    leitor.build(construtor);
    text.append("Plugin Finalizou\n");
  }

  public void leConf()
  {
      try {
            // FileReader arquivo = new
FileReader(getClass().getResource("adapter.txt").getPath());
     // BufferedReader formatado = new BufferedReader(arquivo);
            BufferedReader formatado =
              new BufferedReader(new
InputStreamReader(getClass().getResourceAsStream("adapter.txt")));
```

```
            String linha = formatado.readLine();
            while (linha != null)
            {
                System.out.println(linha);
                linha = formatado.readLine();
            }

            // arquivo.close();
            formatado.close();
        } catch (IOException erro) {
                erro.printStackTrace();
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {


        // TODO Auto-generated method stub

        File pluginFolder = new File("C:\\Program
Files\\Protege_3.4_rc1\\plugins\\edu.stanford.smi.protegex.owl\\");
        ProtegeOWL.setPluginFolder(pluginFolder);

        edu.stanford.smi.protege.Application.main(args);
    }

}
```

APÊNDICE C – CÓDIGO FONTE CLASSE PROPERTYLEVEL.JAVA

```
package inLoco;

import inLoco.xmlmap.TProperty;

public class PropertyLevel
{
    private int level;
    private TProperty property;

    public PropertyLevel(int level, TProperty property)
    {
        this.level = level;
        this.property = property;
    }

    public int getLevel()
    {
        return level;
    }

    public void setLevel(int level)
    {
        this.level = level;
    }

    public TProperty getProperty()
    {
        return property;
    }

    public void setProperty(TProperty property)
    {
        this.property = property;
    }
}
```

APÊNDICE D – CÓDIGO FONTE CLASSE OBJECTFACTORY.JAVA

```
// This file was generated by the JavaTM Architecture for XML Binding(JAXB)
Reference Implementation, vJAXB 2.0 in JDK 1.6
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the
source schema.
// Generated on: 2009.08.21 at 07:44:23 PM GMT-03:00
//


package inLoco.xmlmap;

import javax.xml.bind.JAXBElement;
import javax.xml.bind.annotation.XmlElementDecl;
import javax.xml.bind.annotation.XmlRegistry;
import javax.xml.namespace.QName;


/**
 * This object contains factory methods for each
 * Java content interface and Java element interface
 * generated in the inLoco.xmlmap package.
 * <p>An ObjectFactory allows you to programatically
 * construct new instances of the Java representation
 * for XML content. The Java representation of XML
 * content can consist of schema derived interfaces
 * and classes representing the binding of schema
 * type definitions, element declarations and model
 * groups.  Factory methods for each of these are
 * provided in this class.
 *
 */
@XmlRegistry
public class ObjectFactory {

    private final static QName _Document_QNAME = new
QName("http://www.example.org/esquema3", "document");

    /**
     * Create a new ObjectFactory that can be used to create new instances
of schema derived classes for package: inLoco.xmlmap
     *
     */
    public ObjectFactory() {
    }

    /**
     * Create an instance of {@link TDocument }
     *
     */
    public TDocument createTDocument() {
        return new TDocument();
    }

    /**
     * Create an instance of {@link TProperty }
     *
```

```
     */
    public TProperty createTProperty() {
        return new TProperty();
    }

    /**
     * Create an instance of {@link JAXBElement }{@code <}{@link TDocument
}{@code >}}
     *
     */
    @XmlElementDecl(namespace = "http://www.example.org/esquema3", name =
"document")
    public JAXBElement<TDocument> createDocument(TDocument value) {
        return new JAXBElement<TDocument>(_Document_QNAME, TDocument.class,
null, value);
    }

}
```

APÊNDICE E – CÓDIGO FONTE CLASSE TDOCUMENT.JAVA

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference
Implementation, vJAXB 2.0 in JDK 1.6
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2009.08.21 at 07:44:23 PM GMT-03:00
//


package inLoco.xmlmap;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for TDocument complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this
class.
 *
 * <pre>
 * &lt;complexType name="TDocument">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence>
 *         &lt;element name="property" type="{http://www.example.org/esquema3}TProperty"
maxOccurs="unbounded"/>
 *       &lt;/sequence>
 *       &lt;attribute name="owlclass" type="{http://www.w3.org/2001/XMLSchema}anyURI"
/>
 *       &lt;attribute name="style" type="{http://www.w3.org/2001/XMLSchema}string" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "TDocument", propOrder = {
    "property"
```

```
})
public class TDocument {

    @XmlElement(required = true)
    protected List<TProperty> property;
    @XmlAttribute
    protected String owlclass;
    @XmlAttribute
    protected String style;

    /**
     * Gets the value of the property property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the property property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getProperty().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link TProperty }
     *
     *
     */
    public List<TProperty> getProperty() {
        if (property == null) {
            property = new ArrayList<TProperty>();
        }
        return this.property;
    }

    /**
     * Gets the value of the owlclass property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getOwlclass() {
        return owlclass;
    }
```

```
    /**
     * Sets the value of the owlclass property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setOwlclass(String value) {
        this.owlclass = value;
    }

    /**
     * Gets the value of the style property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getStyle() {
        return style;
    }

    /**
     * Sets the value of the style property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setStyle(String value) {
        this.style = value;
    }

}
```

APÊNDICE E – CÓDIGO FONTE CLASSE TPROPERTY.JAVA

```
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference
Implementation, vJAXB 2.0 in JDK 1.6
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2009.08.21 at 07:44:23 PM GMT-03:00
//


package inLoco.xmlmap;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for TProperty complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this
class.
 *
 * <pre>
 * &lt;complexType name="TProperty">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence>
 *         &lt;element name="property" type="{http://www.example.org/esquema3}TProperty"
maxOccurs="unbounded" minOccurs="0"/>
 *       &lt;/sequence>
 *       &lt;attribute name="iproperty"
type="{http://www.w3.org/2001/XMLSchema}anyURI" />
 *       &lt;attribute name="owlproperty"
type="{http://www.w3.org/2001/XMLSchema}anyURI" />
 *       &lt;attribute name="range" type="{http://www.w3.org/2001/XMLSchema}anyURI" />
 *       &lt;attribute name="reference"
type="{http://www.w3.org/2001/XMLSchema}boolean" />
 *       &lt;attribute name="style" type="{http://www.w3.org/2001/XMLSchema}string" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
```

```
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "TProperty", propOrder = {
    "property"
})
public class TProperty {

    protected List<TProperty> property;
    @XmlAttribute
    protected String iproperty;
    @XmlAttribute
    protected String owlproperty;
    @XmlAttribute
    protected String range;
    @XmlAttribute
    protected Boolean reference;
    @XmlAttribute
    protected String style;

    /**
     * Gets the value of the property property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the property property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getProperty().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link TProperty }
     *
     *
     */
    public List<TProperty> getProperty() {
        if (property == null) {
            property = new ArrayList<TProperty>();
        }
        return this.property;
    }

    /**
     * Gets the value of the iproperty property.
```

```
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getIproperty() {
    return iproperty;
}

/**
 * Sets the value of the iproperty property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setIproperty(String value) {
    this.iproperty = value;
}

/**
 * Gets the value of the owlproperty property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getOwlproperty() {
    return owlproperty;
}

/**
 * Sets the value of the owlproperty property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setOwlproperty(String value) {
    this.owlproperty = value;
}

/**
 * Gets the value of the range property.
 *
 * @return
```

```
 *     possible object is
 *     {@link String }
 *
 */
public String getRange() {
    return range;
}

/**
 * Sets the value of the range property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setRange(String value) {
    this.range = value;
}

/**
 * Gets the value of the reference property.
 *
 * @return
 *     possible object is
 *     {@link Boolean }
 *
 */
public Boolean isReference() {
    return reference;
}

/**
 * Sets the value of the reference property.
 *
 * @param value
 *     allowed object is
 *     {@link Boolean }
 *
 */
public void setReference(Boolean value) {
    this.reference = value;
}

/**
 * Gets the value of the style property.
 *
 * @return
 *     possible object is
 *     {@link String }
```

```
     *
     */
    public String getStyle() {
        return style;
    }

    /**
     * Sets the value of the style property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setStyle(String value) {
        this.style = value;
    }

}
```

APÊNDICE F – CÓDIGO FONTE CLASSE ADAPTER.TXT

```
Estilos

in loco Abstract, in loco Página Inicial, in loco pagina final,
in loco Titulo, in loco Dia Publicacao, in loco mes Publicacao,
in loco ano Publicacao

Ontologias

"subject.owl"
"card.owl"

Protegé

"C:\\Program
Files\\Protege_3.4_rc1\\plugins\\edu.stanford.smi.protegex.owl\\
```

APÊNDICE G – CÓDIGO FONTE CLASSE ADAPTER.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/adapter"
xmlns:tns="http://www.example.org/adapter" elementFormDefault="qualified">

    <complexType name="TAdapter">
        <sequence>
            <element name="select" type="tns:TSelect" maxOccurs="unbounded"
minOccurs="1"></element>
        </sequence>
        <attribute name="style" type="string"></attribute>
    </complexType>

    <complexType name="TSelect">
        <attribute name="content" type="string"></attribute>
        <attribute name="adapterfile" type="anyURI"></attribute>
    </complexType>

    <element name="adapter" type="tns:TAdapter"></element>
</schema>
```

APÊNDICE H – CÓDIGO FONTE CLASSE ARTICLEW4.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<map:document style="in loco Identificador" owlclass="card:Article"
xmlns:map="http://www.example.org/esquema3"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rc="http://purl.org/net/researchcard/card/card.owl"
xsi:schemaLocation="http://www.example.org/esquema3/esquemaw4.xsd ">
    <map:property style="in loco Titulo" owlproperty="card:title"
        range="xsd:string">
    </map:property>
    <map:property style="in loco Autor" owlproperty="card:authors"
        range="card:Person" iproperty="card:fullName">
    </map:property>
    <map:property style="in loco Cartao" owlproperty="card:hasCards"
        range="card:ResearchCard">
    <map:property style="in loco AbstractLeitor"
        owlproperty="card:abstractReader" range="xsd:string">
    </map:property>
    <map:property style="in loco DiaCatalogado"
        owlproperty="card:dayCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco MesCatalogado"
        owlproperty="card:monthCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco AnoCatalogado"
        owlproperty="card:yearCatalogued" range="xsd:int">
    </map:property>
    <map:property style="in loco Referencias"
        owlproperty="card:references" range="card:BibliographicUnit"
        reference="true">
    </map:property>
    <map:property style="in loco Topico" owlproperty="card:hasTopics"
        range="card:CardTopic" iproperty="card:title">
        <map:property style="in loco Recorte"
            owlproperty="card:publicationClip" range="xsd:string">
        </map:property>
        <map:property style="in loco TpPagina"
            owlproperty="card:page" range="xsd:string">
        </map:property>
        <map:property style="in loco Comentarios"
            owlproperty="card:readerComment" range="xsd:string">
        </map:property>
        <map:property style="in loco SubjectTopic"
            owlproperty="card:subjects" range="subj:Subject"
            iproperty="subj:title">
        </map:property>
    </map:property>
    </map:property>
    <map:property style="in loco Assunto"
            owlproperty="card:subjects" range="subj:Subject"
            iproperty="subj:title">
    </map:property>
</map:document>
```

APÊNDICE I – CÓDIGO FONTE CLASSE ESQUEMAW4.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/esquema3"
xmlns:tns="http://www.example.org/esquema3" elementFormDefault="qualified">

    <complexType name="TDocument">
        <sequence>
            <element name="property" type="tns:TProperty"
maxOccurs="unbounded" minOccurs="1"></element>
        </sequence>
        <attribute name="style" type="string"></attribute>
        <attribute name="owlclass" type="anyURI"></attribute>
    </complexType>

    <complexType name="TProperty">
        <sequence>
            <element name="property" type="tns:TProperty"
                maxOccurs="unbounded" minOccurs="0">
            </element>
        </sequence>
        <attribute name="style" type="string"></attribute>
        <attribute name="owlproperty" type="anyURI"></attribute>
        <attribute name="range" type="anyURI"></attribute>
        <attribute name="iproperty" type="anyURI"></attribute>
        <attribute name="reference" type="boolean"></attribute>
    </complexType>

    <element name="document" type="tns:TDocument"></element>
</schema>
```

APÊNDICE J – FORMULÁRIO DE PESQUISA DE SATISFAÇÃO DO USO DA FERRAMENTA INLOCOLOADER

Formulário de Pesquisa de Satisfação do Uso da Ferramenta InLocoLoader

Data: ___/___/_____

Nome:_____

Característica	Subcaracterística	Pergunta chave para a Subcaracterística
Usabilidade	Intelegibilidade	É fácil entender o conceito e a aplicação?
		Sim: ☐ Não: ☐
	Apreensibilidade	É fácil aprender a usar?
		Sim: ☐ Não: ☐
	Operacionalidade	É fácil de operar e controlar?
		Sim: ☐ Não: ☐

Fonte: Desenvolvido pelo auto