HELLEN CARMO DE OLIVEIRA MATOS

Uma Metodologia para a Construção Semiautomatizada de Ontologias a partir de Documentos Textuais

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Cedric Luiz de Carvalho

Co–Orientador: Prof. Fábio Moreira Costa

Goiânia
2009

HELLEN CARMO DE OLIVEIRA MATOS

Uma Metodologia para a Construção Semiautomatizada de Ontologias a partir de Documentos Textuais

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Ciência da Computação, aprovada em Dia de Agosto de 2009, pela Banca Examinadora constituída pelos professores:

Prof. Cedric Luiz de Carvalho
Instituto de Informática – UFG
Presidente da Banca

Prof. Fábio Moreira Costa
Instituto de Informática – UFG

Profa. Ana Paula Laboissiére Ambrósio
Instituto de Informática - UFG

Profa. Fernanda Lima
Departamento de Ciência da Computação - UNB

Hellen Carmo de Oliveira Matos

Graduada em Sistemas de Informação na UEG - Universidade Estadual de Goiás (2006). Tem experiência na área de Sistemas de Informação, atuando principalmente com os temas: Engenharia de Software, Desenvolvimento de Sistemas com a Linguagem JAVA e Gestão Administrativa. Na comunidade tecnológica participou da Organização de Eventos Tecnológicos (Road Show Microsoft - INETA). Durante o Mestrado, na UFG - Universidade Federal de Goiás, foi bolsista da CAPES e desenvolveu um relatório técnico de Gestão e Representação do Conhecimento.

Dedico este trabalho aos meus amados pais, pois se não fosse pelo imenso esforço que fizeram para que eu pudesse chegar até aqui eu não teria adquirido o conhecimento necessário para a conclusão do mesmo, às minhas irmãs Kelly Carmo e Maryelle Carmo, que me apoiaram incessantemente, sem medir esforços para cobrir minhas faltas em todos os momentos. Dedico também ao meu amado esposo Joel Inácio Matos que com bastante compreensão, paciência e sem medir esforço me apoiou, me auxiliou e sempre esteve ao meu lado para ajudar no que fosse preciso para que eu conseguisse finalizar mais este projeto. E, não menos importante, aos meus avós e todos familiares, que com grande alegria me apoiaram e incentivaram nessa caminhada. E, o mais importante, agradeço a Deus, meu Pai e Amado Senhor, por me privilegiar com essa oportunidade, pela provisão de saúde e intelecto para o cumprimento deste e, por sempre iluminar os meus passos e caminhar comigo.

Agradecimentos

Agradeço ao meu orientador Prof. Cedric Luiz de Carvalho, pelo constante apoio, incentivo, dedicação e amizade essenciais para o desenvolvimento deste trabalho e para o meu desenvolvimento como pesquisadora.

Agradeço ao meu co-orientador Prof. Fábio Moreira da Costa por estar disposto a me apoiar nesta jornada e dedicar parte do seu tempo para me auxiliar na conclusão deste trabalho.

Agradeço a toda a minha família pelo apoio recebido, principalmente do meu esposo, meus pais, minhas irmãs e meu avô.

Aos meus colegas do Instituto de Informática - UFG que sempre me auxiliaram, tirando minhas dúvidas e incentivando-me.

Aos meus sogros José Silva e Maria Inácia que com muito carinho me incentivou para a conclusão dessa caminhada e, nos momentos de cansaço e preocupação me acolheram como filha.

Agradeço novamente, ao meu Criador, meu Deus que colocou todas essas pessoas na minha vida para que fosse possível eu chegar até aqui.

A todos, os meus sinceros agradecimentos.

“Bem-aventurado o homem que acha sabedoria, e o homem que adquire conhecimento;

Porque é melhor a sua mercadoria do que artigos de prata, e maior o seu lucro que o ouro mais fino.

Mais preciosa é do que os rubis, e tudo o que mais possas desejar não se pode comparar a ela.”

Provérbios 3:13-15,
Bíblia Sagrada.

Resumo

Carmo de Oliveira Matos, Hellen. **Uma Metodologia para a Construção Semiautomatizada de Ontologias a partir de Documentos Textuais**. Goiânia, 2009. 156p. Dissertação de Mestrado. Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás.

A organização da informação tornou-se um processo fundamental à medida em que vem crescendo exponencialmente o volume de informações disponíveis, motivado pelo avanço tecnológico, em particular da Internet. Isto originou uma necessidade de meios automáticos de pesquisa e filtragem de informação, como os sistemas de recuperação de informação. Esses sistemas dependem da representação semântica dos documentos, que pode ser feita através de instrumentos de representação de relacionamentos semânticos e conceituais, como as ontologias. O processo de construção de ontologias é complexo e demorado. É difícil saber se uma ontologia é suficientemente abrangente para a informação semântica (informação que possui significado contextual para um sujeito) dos documentos. Atualmente, ontologias são desenvolvidas segundo padrões e métodos da Engenharia de Ontologia. Porém, não existe uma solução universal. Esta dissertação propõe uma metodologia para a construção semiautomatizada de ontologias a partir de um conjunto de documentos textuais. É proposta também a construção de uma ontologia para o domínio de grades computacionais, com o objetivo de validar a proposta. O domínio de grades computacionais é fortemente associado ao compartilhamento de recursos para a resolução de problemas em ambientes distribuídos dinâmicos e heterogêneos. Alocar esses recursos são tarefas essenciais na grade e dependem da qualidade das informações disponíveis. A ontologia construída objetiva prover a descrição dos recursos disponíveis no ambiente de grade, auxiliando na qualificação das informações contidas neste ambiente. Isso irá permitir a descoberta semântica dos recursos da grade para execução de aplicações, de modo eficiente.

Palavras–chave

Ontologia, Metodologia para Construção de Ontologias, Ontologia para Grade Computacional

Abstract

Carmo de Oliveira Matos, Hellen. **A Methodology for the Semi-automatic Construction of Ontology from Textual Documents**. Goiânia, 2009. 156p. MSc. Dissertation. Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás.

Information organization became a fundamental process in face of the exponential increasing of available information, motivated by technology advances, Internet in particular. It demands effective approaches of information filtering and searching, like the information retrieval systems. Such systems depend on the document's semantic representation, which can be implemented through conceptual and semantic relationship representation instruments, as ontologies. The ontology construction is a complex and time-consuming process. It's difficult to know whether a given ontology is wide enough to represent the documents' semantic information. Up to date ontologies are developed according to Ontology Engineering standards and methods, although there is no any universal solution. This work presents a methodology for semi-automatic construction of ontologies. It also demonstrates an ontology construction for grid computing domain, aiming to validate the proposal. Such domain is strongly associated to resources sharing to solve problems on dynamic and heterogeneous environment. The allocation of these resources is an essential task in the grid and depends on the quality of available information. The constructed ontology aims to provide a description of the resources available in a computational grid to help on the qualification of information contained in this environment. This will allow the semantic discovery of grid resources to run applications on an efficient way.

Keywords

Ontology, Methodology for Ontology Construction, Ontologies for Grid Computing

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Códigos de Programas

CAPÍTULO 1

Introdução

As Tecnologias de Informação, em particular a Internet, desempenham um papel muito importante na sociedade atual. Com a Internet, a distância física deixou de ser um problema. Ela permite a geração e troca de grandes quantidades de informações e um aumento do conhecimento coletivo.

O advento da Internet como meio de comunicação ágil, flexível e de baixo custo, e sua adoção em larga escala pelas instituições foram os propulsores das comunidades virtuais. Grupos de pessoas com interesses comuns foram se formando na Web (a porção multimídia da Internet), encorajados, principalmente pelas novas facilidades de comunicação, representadas por ferramentas como e-mail, *chats*, *websites* e outros recursos. Profissionais/estudantes de uma área específica passaram a poder trocar informações relevantes para o seu dia-a-dia, sobre suas "melhores práticas" e sobre a forma como estruturaram seus processos, além de compartilhar soluções para os seus problemas mais comuns. Verdadeiras "comunidades" começaram a se formar em torno do compartilhamento de suas "práticas". Estas comunidades foram chamadas por Lave e Wenger [100] de Comunidades Virtuais de Práticas (CoPs). Além das CoPs, diversos outros tipos de comunidades se estabeleceram na Web, como as comunidades de relacionamento, comunidades de software livre, comunidades de aprendizagem, comunidades empresariais etc.

Um dos produtos desse compartilhamento de informações refere-se a uma grande quantidade de dados sendo armazenados constantemente sem um controle específico de armazenamento e sem muito suporte para uma busca posterior.

1.1 Motivação

A organização da informação tornou-se um processo fundamental à medida em que vem crescendo exponencialmente o volume de dados disponíveis, resultando, muitas vezes, na desorganização de acervos informacionais e, consequentemente, na dificuldade de se encontrar o que se procura em determinado sistema de recuperação de informação.

Pesquisas têm sido desenvolvidas visando o desenvolvimento de mecanismos de indexação, organização e recuperação de informações, com o objetivo de melhorar a

eficácia dos sistemas de recuperação de informação [35]. A utilização de especificações semânticas auxiliam na melhora desse processo de recuperação de informação. Pode-se citar alguns campos de pesquisa que se ocupam da exploração semântica da informação:

- a Web Semântica, que pretende criar metodologias, tecnologias e padrões de metadados para ampliar o escopo das atividades desempenhadas automaticamente [16];
- a utilização da semântica embutida nos próprios documentos com o uso de estruturas da linguagem natural como os sintagmas[1] nominais e verbais [149];
- instrumentos de representação de relacionamentos semânticos e conceituais como as ontologias [163] e os tesauros [38], objetivando evitar problemas relacionados à ambiguidade inerente às palavras da linguagem natural.

Nesta perspectiva, verifica-se um crescente interesse no estudo e desenvolvimento de instrumentos que permitam descrever níveis de relacionamentos semânticos e conceituais mais avançados, contribuindo para a atenção dada às ontologias. A origem das ontologias se dá no campo teórico da filosofia [32], sendo ainda pesquisadas e desenvolvidas como instrumento de representação de conhecimento nos campos das Ciências da Computação e da Informação.

Para a Ciência da Informação, as ontologias são de interesse pela potencialidade que elas têm em organizar e representar a informação. Para a Filosofia, o interesse se dá pela forma como nelas é descrita a realidade [43].

Na Ciência da Computação, ontologias têm sido desenvolvidas no campo da Inteligência Artificial para a definição explícita de conceitos e suas relações, propriedades e restrições expressas formalmente [80]. Atualmente, elas estão começando a se expandir em outras áreas, como a integração inteligente de informação, sistemas cooperativos, produtos de software baseados em agente, e no comércio eletrônico [18].

Segundo Almeida e Bax [2], as ontologias podem melhorar os processos de recuperação de informação ao organizarem o conteúdo de fontes de dados em um determinado domínio.

1.1.1 Contexto

O descontrole da informação e a necessidade de obter-se um melhor desempenho na execução de diversas aplicações dentro das comunidades virtuais têm levado as

[1] **Sintagma** é uma unidade formada por uma ou várias palavras que, juntas, desempenham uma função na frase [8].

organizações virtuais a utilizarem o ambiente de grades computacionais. Essas organizações virtuais compartilham recursos sob políticas próprias e configurações do ambiente de grade [39].

Os ambientes de grade podem ser constituídos por várias comunidades virtuais, diferentes umas das outras, de modo que a forma como seus recursos são disponibilizados, descritos e acessados é bastante heterogênea.

Cada organização virtual possui autonomia sobre seus recursos. Esta autonomia pode acarretar diferentes maneiras para o compartilhamento e disponibilização de seus recursos, o que é feito de acordo com suas políticas administrativas. Essas diferenças podem levar à não-interoperabilidade entre as diversas organizações devido a esta falta de padronização sobre o gerenciamento de recursos em ambiente de grade.

Segundo Foster e Kesselman [60], alcançar a interoperabilidade entre as organizações virtuais, através da habilidade de cooperação, compartilhamento e agregação de recursos computacionais distribuídos, disponibilizando-os como serviços, é o principal objetivo dos ambientes de grades computacionais.

Um mesmo recurso, presente em várias organizações virtuais, pode ter diferentes descrições de suas características. Por exemplo, considere-se um computador que tenha um sistema operacional Linux conforme a distribuição Kurumin. Uma organização virtual poderia descrever este computador como tendo um sistema operacional Linux e outra, como tendo um sistema operacional Kurumin e esta diferença pode acarretar problemas futuros relacionados à pesquisa por recursos.

O ambiente de grade provê mecanismos de cooperação seguros e escaláveis para a descoberta de recursos e negociação de acesso remoto aos mesmos. Isto permite que recursos sejam compartilhados por comunidades científicas em uma escala jamais vista e que grupos geograficamente distribuídos trabalhem em conjunto de maneiras que antes não eram possíveis [106].

O uso de ontologias para a descrição semântica de recursos computacionais tem aumentado nos últimos anos, sem que contudo, exista uma ontologia única para ambientes de grade [22]. Pode-se afirmar que uma ontologia única e consensual para a descrição de recursos computacionais em um ambiente de grade é muito difícil de ser definida e que, provavelmente, não se terá esta ontologia devido à grande variedade de visões que cada organização virtual possui de seus recursos.

O uso de ontologias em grades favorece o compartilhamento e reutilização de aplicações e dados gerando integração de conteúdo das grades. Adicionalmente, essa abordagem pode ajudar na propagação de conhecimento, possibilitando a descoberta semântica dos recursos da grade, e permitir o monitoramento dos diferentes tipos de recursos computacionais do ambiente.

A descrição semântica dos recursos de ambientes de grade foi a motivação inicial

para a investigação acerca da construção de uma ontologia para o domínio de grades computacionais.

1.2 Definição do Problema

As pessoas responsáveis pela elaboração da ontologia deparam-se com problemas que vão muito além das dificuldades técnicas. Na primeira tentativa de criar uma ontologia, em geral, surgem dúvidas como:

- saber se a ontologia deve ser construída manualmente ou não, levando-se em consideração o tempo e os recursos no processo de construção;
- saber se a ontologia construída já é suficiente para o que se pretende.

Mesmo com alguma experiência, ao tentar construir uma nova ontologia relacionada com outro assunto, voltam a surgir as mesmas dificuldades. Não existe uma solução universal. Na prática, devido ao tempo e aos recursos, existe sempre um risco de se obter uma ontologia demasiadamente simples ou, ao contrário, uma ontologia tão elaborada que torna sua utilização pouco viável.

O trabalho aqui apresentado é impulsionado por esta problemática, visando a semiautomatização do processo de construção da ontologia, com foco na redução do tempo e otimização dos recursos disponíveis.

1.3 Objetivo

O objetivo fundamental deste trabalho foi desenvolver um mecanismo para construir ontologias semiautomaticamente, de maneira que estas representem, semanticamente, a informação expressa em um corpus constituído de documentos de um domínio específico. Tal mecanismo é alvo de pesquisas atualmente e, ainda que seus resultados possam ser incompletos, poderá contribuir significativamente para o melhoramento do processo de construção de ontologias.

1.3.1 Objetivo Específico

O objetivo específico desta dissertação é descrever uma metodologia que permita a um não-especialista em um domínio específico, construir ontologias a partir de documentos textuais deste domínio, visando a redução do tempo e a otimização dos recursos disponíveis no processo de construção.

Para demonstrar esta metodologia, foi construída uma ontologia para ambientes de grades computacionais, sendo que as informações recolhidas do corpus poderão ser publicadas dentro de uma comunidade virtual.

1.4 Organização da Dissertação

Quanto à estrutura desta dissertação, o documento está organizado como se segue. Primeiramente, serão abordados temas relacionados aos fundamentos deste trabalho, incluindo: o Capítulo 2, onde são abordados os conceitos relacionados às ontologias; o Capítulo 3, no qual é detalhado o processo de desenvolvimento de ontologias, incluindo uma descrição e uma comparação das metodologias e métodos existentes. O Capítulo 4, apresenta a ferramenta para extração de conhecimento em objetos textuais - Sinapse, utilizada para automatizar o processo de extração de conhecimento em documentos. O Capítulo 5, apresenta a proposta do trabalho. O Capítulo 6, descreve o estudo de caso da construção de uma ontologia para grade computacional. O Capítuo 8, apresenta a conclusão da dissertação e os trabalhos futuros.

CAPÍTULO 2

Ontologias

Ontologias, conforme definido em [129], são vocabulários de termos que podem ser usados em anotações semânticas, cujo significado é formalmente especificado e pode ser acessado e processado automaticamente [167].

Ontologias podem ser vistas como representações explícitas de conceitos relacionados a um domínio e provêm uma estrutura sobre a qual bases de conhecimento podem ser construídas [155]. Uma ontologia define o vocabulário do domínio de um problema e restrições sobre as possíveis combinações de termos na modelagem do domínio [167]. Compartilhar conhecimento comum, reutilizar o conhecimento sobre um domínio, tornar explícitas proposições assumidas sobre um domínio, separar conhecimento sobre um domínio de conhecimento operacional e analisar o conhecimento sobre um domínio são as razões mais comuns para o desenvolvimento de uma ontologia [50].

Ontologias, atualmente, estão sendo utilizadas para diferentes propósitos dentro da Informática, dentre eles: processamento de linguagem natural [118]; *e-commerce* [104]; gerenciamento do conhecimento [91]; e Web semântica [112]. Ontologias também estão presentes em diferentes comunidades como, por exemplo [106]: engenharia do conhecimento, banco de dados e engenharia de software.

Nesta dissertação, o propósito é obter uma metodologia para a construção semiautomatizada de ontologias.

O presente capítulo trata da fundamentação teórica sobre Ontologias, um dos temas centrais desta dissertação. O detalhamento do assunto inclui a definição de ontologia, a apresentação das linguagens e ferramentas para a manipulação de ontologias, o estudo de alguns mecanismos de inferência e definição de estratégias eficientes para recuperação de informações em bases de conhecimento semânticas.

2.1 Definição

O termo ontologia tem origem no grego *ontos*, ser, e *logos*, palavra. Originalmente vem da palavra categoria, definida por Aristóteles, que pode ser utilizada para classificar coisas. O Dicionário Oxford [17] de filosofia define ontologia como:

> *[...] o termo derivado da palavra grega que significa "ser", mas utilizado desde o século XVII para denominar o ramo da metafísica que diz respeito àquilo que existe.*

Em Linguística Computacional, ou seja, no campo de ação da representação formal do conhecimento, uma ontologia pressupõe um enlace entre os símbolos da linguagem natural e as entidades do mundo real que ela representa [171].

De acordo com Sowa [150], o termo ontologia tem um sentido específico quando se trata de organização da informação, diferentemente do adotado na filosofia e em outras áreas. Ontologia é um catálogo de tipos de coisas em que se supõe existir um domínio, na perspectiva de uma pessoa que utiliza uma determinada linguagem.

Uma das definições mais conhecidas para ontologias é apresentada por [80]:

> *Uma ontologia é uma especificação explícita de uma conceitualização.*
>
> *[...] Em tal ontologia, definições associam nomes de entidades no universo do discurso (por exemplo, classes, relações, funções, etc. com textos que descrevem o que os nomes significam e os axiomas formais que restringem a interpretação e o uso desses termos) [...].*

Esta especificação tem sido objeto de grande esforço por parte dos investigadores em Inteligência Artificial, que há várias décadas buscam uma ontologia que ofereça flexibilidade suficiente para conseguir representar o conhecimento complexo registrado na mente humana [171].

O termo conceitualização corresponde a uma coleção de objetos, conceitos, outras entidades, e suas relações que existem em um domínio [68].

A ontologia provê uma linguagem compartilhada para uma comunidade de provedores e consumidores de serviços, sendo eles máquinas (por exemplo, agentes de software) ou pessoas [84]. As ontologias podem ser utilizadas como a espinha dorsal para cada uma das tarefas no ciclo de vida do gerenciamento do conhecimento. Elas servem para estruturar e recuperar as informações de um modo compreensivo e são utilizadas essencialmente para busca, troca e descoberta.

Segundo Guarino [85], ontologia refere-se a um sistema particular de categorias, de acordo com uma certa visão do mundo. Ela refere-se a um artefato de engenharia, constituído por um vocabulário específico utilizado para descrever uma certa realidade. O autor classifica as ontologias sob três aspectos, como se segue:

1. Pelo nível de detalhes, ou seja, pela quantidade de axiomas declarados:
   - Ontologias de referência (*off-line*)
   - Ontologias compartilháveis (*on-line*)

2. Pelo nível de dependência de uma tarefa ou ponto de vista conforme ilustrado na Figura 2.1:

   - Ontologias de alto nível (*top-level ontologies*)
   - Ontologias de domínio (*domain ontologies*)
   - Ontologias de tarefas (*task ontologies*)
   - Ontologias de aplicações (*application ontologies*)

3. Ontologias de representação.

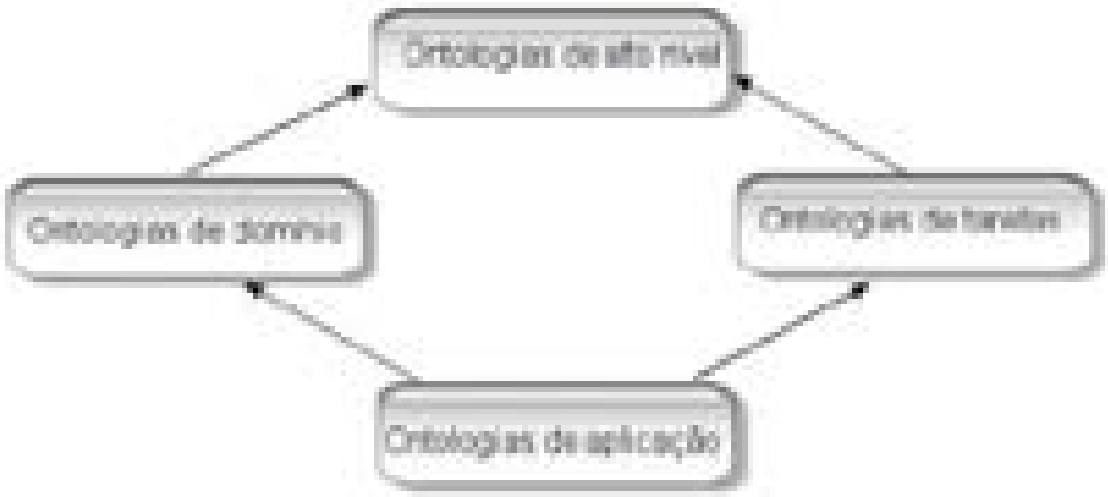

Figura 2.1: *Classificação das Ontologias [84]*

Uma ontologia aproxima o modelo pretendido de uma linguagem lógica. As ontologias de referências são as que se aproximam mais da definição do vocabulário desejado, ou seja, são mais refinadas, podendo assim, ser utilizadas *off-line*. Elas precisam de uma linguagem de alta expressividade e possuem um grande número de axiomas. Já as ontologias compartilháveis, têm um número mínimo de axiomas, são ontologias de alto nível de granularidade (*coarse grained*), e definem um domínio comum que pode ser utilizado por vários usuários [106].

Com relação à classificação pelo nível de dependência de uma tarefa ou ponto de vista tem-se as seguintes definições [84] (Figura 2.1):

- **Ontologias de alto nível** (*top-level ontologies*) descrevem conceitos muito gerais, como espaço, tempo, assunto, objeto, evento, ação, entre outros, os quais são independentes de um problema ou domínio em particular. Parece razoável então, pelo menos, teoricamente, a unificação de ontologias genéricas para grandes comunidades de usuários.

- **Ontologias de domínio** (*domain ontologies*) descrevem o vocablário relativo a um domínio específico, através da especialização de conceitos presentes na ontologia de alto nível.
- **Ontologias de tarefas** (*task ontologies*) descrevem o vocabulário relativo a uma tarefa genérica ou atividade, através da especialização de conceitos presentes na ontologia de alto nível.
- **Ontologias de aplicação** (*application ontologies*) são mais específicas. Descrevem conceitos dependentes de um domínio e de uma tarefa em particular. Estes conceitos sempre correspondem a papéis desempenhados por entidades de domínio enquanto realizando certa atividade, como unidade substituível ou componente disponível.

Finalmente, ontologias de representação constituem um tipo especial de ontologia que descreve a classificação de primitivas usadas por uma linguagem de representação do conhecimento (como conceitos, atributos e relações) [84]. Um exemplo de uma ontologia de representação é a *Frame Ontology* [80], introduzida dentro de um sistema para captura de conhecimento, o Ontolingua [81], com o propósito de habilitar traduções entre diferentes linguagens de representação do conhecimento.

A seguir, tem-se a classificação das ontologias quanto ao assunto de sua conceitualização. De acordo com Van Heijst et al. [166], as ontologias podem ser classificadas como:

- **Ontologias de domínio** (*domain ontologies*) contêm conceitualizações que são específicas para um domínio em particular como por exemplo: eletrônica e medicina.
- **Ontologias genéricas** (*generic ontologies*) são similares às ontologias de alto nível, mas os conceitos definidos são considerados genéricos, ou seja, independentes de um problema ou domínio particular. Tipicamente, ontologias genéricas definem conceitos como: estado, evento, processo, ação, componente, entre outros. Os conceitos, nas ontologias de domínio, são sempre definidos como especializações de conceitos nas ontologias genéricas. Ontologias genéricas são construídas para serem reusadas e estendidas. Um problema é que não existe consenso na comunidade acadêmica sobre o qual é a melhor maneira de expressar o conhecimento geral sobre o mundo que eles estão interessados em representar.
- **Ontologias de representação** (*representation ontologies*) explicam a conceitualização que sustenta o formalismo da representação do conhecimento. São ontologias que agrupam as primitivas de modelagem usadas para formalizar o conhecimento como um modelo de representação. Ontologias de domínio e ontologias genéricas são descritas utilizando primitivas fornecidas pelas ontologias de representação. Esta ontologia define termos como relação, função, classe e outras primitivas usadas na modelagem de ontologias.

- **Ontologias de aplicação** (*application ontologies*) contêm todas as definições que são necessárias para modelar o conhecimento necessário para uma aplicação em particular. Tipicamente, ontologias de aplicação são uma mistura de conceitos que são extraídos das ontologias de domínio e das ontologias genéricas. Ontologias de aplicação não são construídas para o reuso. Esse tipo de ontologia é utilizada para especializar e estender ontologias de domínio ou tarefa para uma dada aplicação.

Pode-se observar que tanto a classificação de Van Heijst et al. [166] quanto a de Guarino [84] são coerentes. Pode-se fazer um mapeamento entre a ontologia de alto nível definida por Guarino e a ontologia genérica definida por Van Heijst et al., percebendo-se que ambas possuem significados similares [106].

Figura 2.2: *Três principais categorias de ontologias quanto ao seu uso [163]*

Uschold e Gruninger [163] definem três categorias principais de ontologias quanto ao seu uso, que estão ilustradas na Figura 2.2:

- **Ontologias para comunicação entre as pessoas e a organização**: a ontologia reduz confusões terminológicas e conceituais dentro da organização. Ontologias permitem o entendimento compartilhado e a comunicação entre as pessoas com diferentes necessidades;
- **Ontologias para interoperabilidade entre sistemas**: a ontologia pode ser usada como uma inter-língua para apoiar a tradução entre diferentes linguagens e representações;
- **Ontologias para engenharia de software**: ontologias podem ser benéficas na especificação, confiabilidade e reutilização durante o processo de engenharia de software. Facilitam o processo de identificação dos requisitos do sistema e o entendimento das relações entre os componentes deste sistema.

Segundo Gruber [80] e Studer et al. [151], na área de Ciência da Computação, uma ontologia é uma especificação explícita e formal de uma conceitualização compartilhada. Esclarecendo os requisitos desta definição:

- **Por especificação explícita**, entende-se que os tipos de conceitos usados e as limitações do uso desses conceitos devem ser definidos explicitamente.
- **Por formal**, entende-se que a ontologia deve ser passível de ser processada por máquinas.
- **Por conceitualização**, que trata-se de um modelo abstrato de uma área de conhecimento ou de um universo limitado de discurso.
- **Por compartilhada**, entende-se que trata-se de um conhecimento consensual apresentado por um grupo, e não por apenas um indivíduo, seja uma terminologia comum da área modelada, ou acordada entre os desenvolvedores dos agentes que se comunicam.

Gruber [80], Studer, Benjamins e Fensel [151] e Uschold e Jasper [161] definem que ontologias podem ter várias formas, desde que tenha um vocabulário de termos e seus significados. Para estes autores, uma ontologia é uma manifestação de um número de participantes que entram em acordo sobre o compartilhamento de um entendimento de um domínio. Esse acordo facilita a comunicação precisa e efetiva sobre o significado dos termos, pois restringe as possíveis interpretações do termo, o que conduz a outros benefícios, como interoperabilidade, reuso e compartilhamento.

Tomando-se por base as definições acima, pode-se concluir que as ontologias objetivam capturar o conhecimento consensual de um modo genérico, podendo ser reusáveis e compartilhadas entre aplicações (software) e por grupos de pessoas, tendo um vocabulário de termos. Além disso, as ontologias são normalmente construídas por um grupo de pessoas em diferentes locais [76].

Uschold e Gruninger [163] propõem o compartilhamento e o reuso de ontologias, além do uso de ontologias para a modelagem de problemas e domínios, fornecendo uma biblioteca para reutilização fácil de classes e objetos de modelagem. Guarino [84] defende a necessidade de se construir ontologias que possam ser reutilizáveis e compartilhadas através de múltiplos serviços e métodos.

Em Maedche [111], é apresentado o triângulo de significados, como se pode ver na Figura 2.3. Segundo ele,“Apesar de nem sempre símbolos conseguirem capturar corretamente a essência de uma referência (ou conceito) ou de um referente (ou coisa), existe uma correspondência entre eles. O relacionamento entre uma palavra e uma coisa é indireto. A ligação só pode ser completada quando um interpretador processa a palavra, o que invoca o conceito correspondente e então liga aquele conceito a uma coisa no mundo”. Este conceito auxilia na definição da estrutura da ontologia como se pode ver mais adiante.

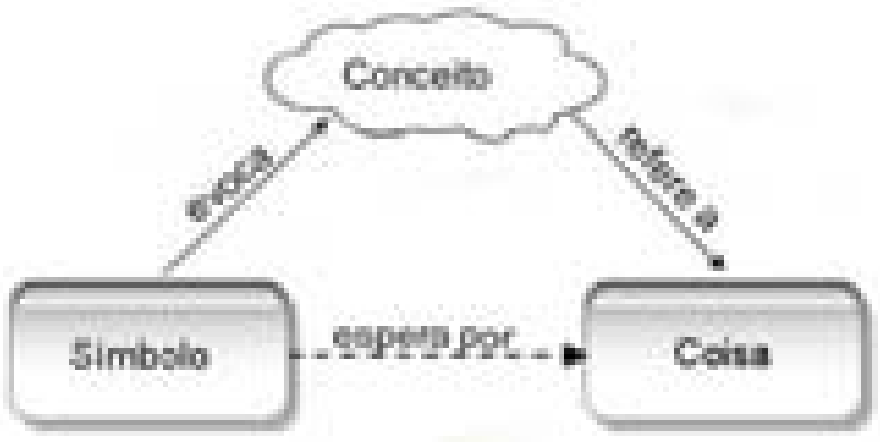

Figura 2.3: *Triângulo de significados [111]*

Continuando, Maedche [111] define: “Uma ontologia é uma teoria lógica constituída de um vocabulário e uma linguagem lógica. Em um domínio de interesses, ela formaliza sinais, descrevendo coisas no mundo, permitindo um mapeamento dos sinais para coisas o mais exato possível. A base do conhecimento deve ser definida sobre uma ontologia, descrevendo circunstâncias particulares.”

Para ilustrar esta utilização, Maedche [111], utiliza o exemplo de duas pessoas pensando sobre o termo “Jaguar”. Inicialmente, estas duas pessoas poderiam ter noções distintas sobre o mesmo símbolo, um imanginando um carro da marca “Jaguar” e outro imaginando o animal da espécie “Jaguar”. Então, o uso de ontologias, neste caso, serviria para que ambos chegassem a um acordo sobre o que se estaria discutindo, podendo ora utilizar uma ontologia que levasse ao carro e ora outra que levasse ao animal, possibilitando assim, a redução de desentendimentos acerca de certos símbolos.

Em [143] encontra-se uma ontologia para qualquer coisa relacionada ao mundo real, tal como ilustrada na Figura 2.4. Segundo o autor, existe uma grande complexidade em se fazer uma descrição completa de alguma entidade do mundo real. Representar tudo o que existe no mundo então, é algo inimaginável. Por esta razão é que se opta por representar domínios restritos de conhecimento, cujos conceitos devem ser elaborados preferencialmente por especialistas da área.

Conforme Knublauch [96], os componentes básicos de uma ontologia são:

- **Classes**: expressam qualquer coisa sobre a qual algo é dito (Figura 2.4); são organizadas em uma taxonomia.
- **Relações**: representam o tipo de interação entre os conceitos de um domínio.
- **Axiomas**: utilizados para modelar sentenças sempre verdadeiras.
- **Instâncias**: utilizadas para representar elementos específicos, ou seja, os próprios dados.

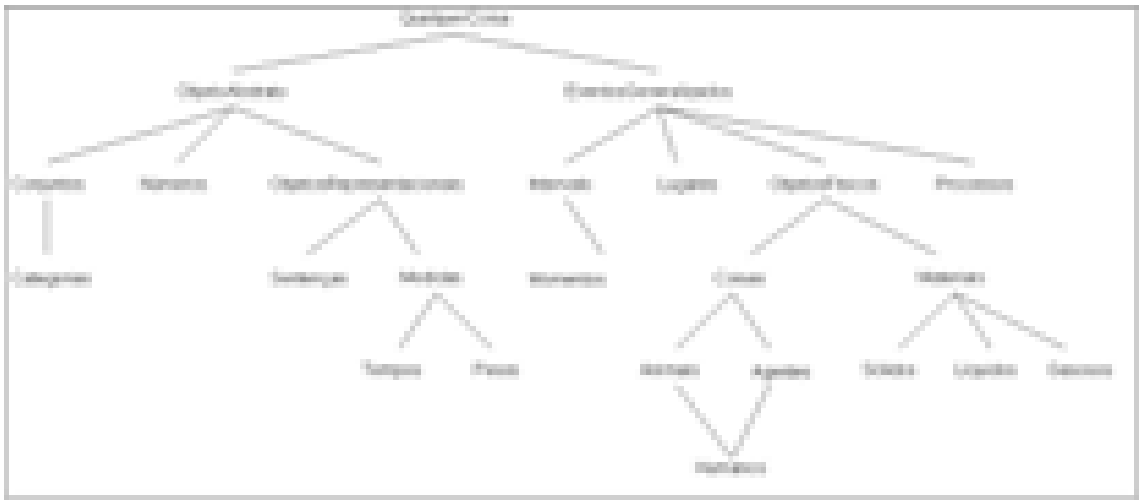

Figura 2.4: *Ontologia de Qualquer Coisa [143]*

Independentemente do domínio representado pela ontologia e das possíveis diferenças existentes entre ontologias, alguns elementos são comuns [29]:

- Existem **objetos** do mundo real a serem representados.
- Objetos têm **propriedades** ou **atributos** que podem assumir valores variados.
- Objetos podem se relacionar com outros objetos.
- **Propriedades** e **relacionamentos** podem mudar com o tempo.
- Existem **eventos** que podem ocorrer em determinados instantes.
- Existem **processos** que podem ocorrer ao longo do tempo.
- O mundo e seus objetos podem estar em diferentes **estados**.
- Eventos podem causar outros eventos ou estados como efeitos.
- Objetos podem ser decompostos ou fazer parte de outros objetos (agregação).

Cada um desses elementos pode ser instanciado, dependendo única e exclusivamente do domínio representado, ou seja, podem não estar presentes na ontologia se não representarem nenhum conceito do domínio[106].

2.2 Linguagens

No começo dos anos 1990, foram criadas linguagens de implementação de ontologia baseadas em IA, pois, basicamente, o paradigma de Representação do Conhecimento lidava com linguagens de ontologia baseadas em lógica de primeira ordem (por

exemplo, *Knowledge Interchange Format* ou KIF [66]), com *frames*[1] ou Quadros combinados com lógica de primeira ordem (por exemplo, Ontolingua [56]), *Operational Conceptual Modeling Language* ou OCML [113] e *Frame Logic* ou FLogic [95] ou com lógica descritiva (por exemplo, Loom [125]).

O crescimento da Internet conduziu à criação de linguagens de ontologia que exploram características da Web. Tais linguagens são chamadas geralmente linguagens de ontologia baseadas em Web ou linguagens de marcação de ontologia (*ontology markup languages*). Essas linguagens estão em fase de desenvolvimento. Os relacionamentos entre elas são mostrados na Figura 2.5.

A Figura 2.5, apresenta os três níveis distintos na arquitetura da Web Semântica, conforme Djuric [52], que introduzem primitivas expressivas: camada de metadados, camada de esquema e camada lógica.

Figura 2.5: *Linguagens na Arquitetura da Web Semântica [52]*

Conforme mostrado na Figura 2.5, a linguagem XML proporciona uma sintaxe comum, bem definida e de fácil processamento. Porém, ela não diz nada sobre os dados que descreve. A semântica dos dados é abordada a partir da camada de metadados (*metadata layer*), através do *Resource Description Framework* (RDF), e da camada de esquema (*schema layer*), com a linguagem RDF *Schema* (RDFS). O modelo RDF define os relacionamentos entre recursos, mas para permitir raciocínio na Web Semântica, seria necessária outra camada. A camada lógica (*logical layer*) introduz linguagens ontológicas, tais como OIL, DAML e OWL, que são baseadas em arquiteturas da camada mais baixa.

No ano de 2001, o W3C (*the World Wide Web Consortium*) criou um grupo de trabalho chamado *Web-Ontology* (WebOnt), com o objetivo de desenvolver uma nova

[1] *Frames* são uma forma de Representação de Conhecimento que armazena informações sobre conceitos ou objetos através de uma estrutura de dados composta por quadros (*frames*) interrelacionados [116].

linguagem de marcação de ontologia para a Web Semântica, chamada *Web Ontology Language* ou OWL [1].

Em [152], é proposta uma classificação das linguagens para construção de ontologias em duas categorias:

- *Linguagens de ontologias tradicionais*, que possuem sua raiz na Inteligência Artificial e na Engenharia do Conhecimento. Esta categoria é subdividida em quatro grupos:
  - No primeiro, estão agrupadas as linguagens que têm como base a lógica de predicados de primeira ordem, onde a CycL é a linguagem representativa deste grupo.
  - No segundo grupo, têm-se as linguagens baseadas em *frame* ou quadro, como Ontolingua, F-logic, CML e OCML. Quadros são o ponto central da modelagem.
  - No terceiro grupo, a base é a lógica descritiva, e a Loom é um exemplo de linguagem que pode definir os conceitos em termos de lógica descritiva para especificar as propriedades que os objetos devem satisfazer para pertencer ao conceito.
- *Linguagens de ontologias para Web*, que seguem padrões W3C, são utilizadas para facilitar o intercâmbio de informações na Internet. A OWL [4] se encaixa nesta categoria.

Na Figura 2.5, estão listadas as linguagens de ontologia agrupadas de acordo com a classificação de [152].

Estes grupos de linguagens possuem diferentes graus de expressividade e diferentes propriedades computacionais e são descritos nas subseções a seguir.

2.2.1 CycL

Linguagem baseada em lógica de primeira ordem, desenvolvida no projeto Cyc [103] com o propósito de especificar ontologias de conceitos comuns, que pudessem dotar os computadores de Inteligência Artificial. Apesar de ter sua sintaxe baseada na lógica de primeira ordem, CycL a estende com o uso de conceitos de lógica de segunda ordem. Predicados são tratados como constantes em expressões.

Para expressar conceitos do mundo real, a linguagem tem, segundo Su e Ilebrekke [152], cerca de 160 termos que podem ser combinados em expressões significativas. Alguns dos conceitos principais da CycL são: constantes, variáveis, fórmulas, predicados e microteorias (conjuntos de fórmulas, que podem participar de outras fórmulas). Microteorias fornecem um contexto para o valor de verdade das fórmulas.

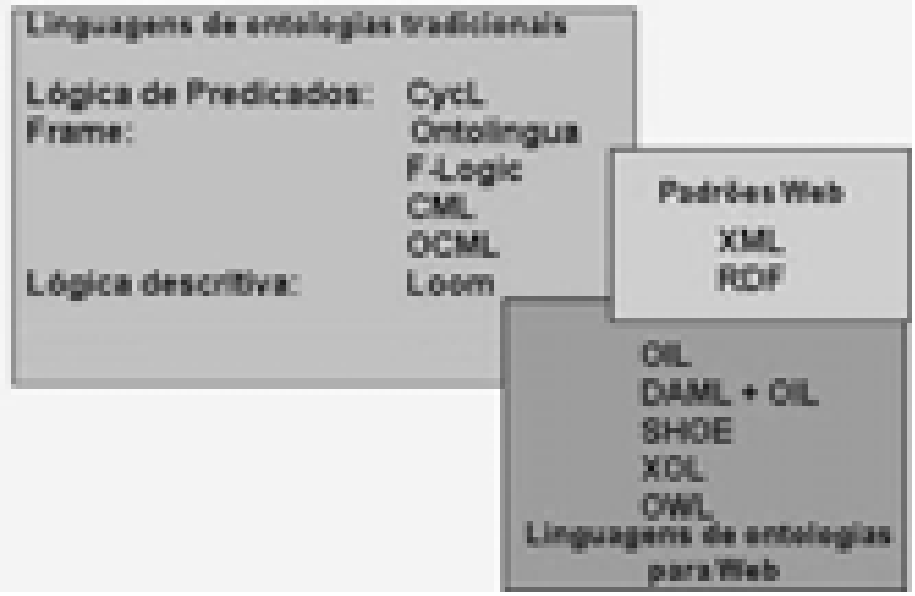

Figura 2.6: *Classificação das linguagens adaptado de [152]*

2.2.2 Ontolingua e KIF (*Knowledge Interchange Format*)

Inicialmente, cabe ressaltar que existe tanto um sistema para descrição de ontologias, quanto uma linguagem de ontologia com o nome "Ontolingua". O sistema é utilizado para a descrição de ontologias compatíveis com múltiplas linguagens de representação [80]. A linguagem Ontolingua é baseada no KIF (*Knowledge Interchange Format*) [67], que tem semântica declarativa, isto é, o significado das expressões existentes na representação pode ser compreendido. Essa linguagem é compreensível logicamente, ou seja, fornece suporte para expressão de sentenças arbitrárias em cálculo de predicados de primeira ordem. Ela provê suporte para a representação de regras de raciocínio não-monotônicas e também, fornece suporte para a definição de objetos, funções e relações.

A linguagem KIF contém uma base de conhecimento clara para o leitor, mas não possibilita o raciocínio automático por meio dela. Desta forma, o Ontolingua traduz definições escritas em KIF para uma forma apropriada, visando desenvolver sistemas de representação do conhecimento [80].

2.2.3 F-Logic (*Frame Logic*)

F-logic é uma linguagem lógica, integrada com paradigmas de orientação a objetos e linguagens baseadas em quadros [95]. Alguns conceitos da orientação a objetos encontram-se representados na F-logic, como: classe, método, tipos e herança. Através da F-logic é possível a representação de conceitos, taxonomias, relações binárias, funções, axiomas, instâncias e regras dedutivas [95].

O Código 2.1, extraído de [3], exemplifica o uso da F-Logic na definição de uma ontologia.

Neste exemplo, é mostrado um vocabulário simples, onde todos os homens e mulheres são pessoas. Pessoa possui os atributos pai, mãe, filha e filho. As regras descrevem o que mais pode ser extraído da ontologia. A primeira regra diz que se X é pai de Y e Y é um homem, então Y é filho de X. As três regras seguintes são similares a esta primeira. O conjunto de fatos instancia os conceitos introduzidos anteriormente. Eles indicam se uma pessoa pertence à classe homem ou mulher e informa sobre as relações pai e mãe entre eles. Na parte final deste exemplo, existe uma consulta que solicita todas as mulheres e seus filhos, que tenham como pai Abraham. A mesma consulta poderia ser escrita como: `FORALL X,Y <- X:woman AND X[son->>Y] AND Y[father->abraham]`

Código 2.1 Exemplo de código em linguagem *F-logic*.

```
1  /*vocabulário*/
2
3  mulher::pessoa.
4  homem::pessoa.
5  pessoa[pai=>homem].
6  pessoa[mãe=>mulher].
7  pessoa[filha=>>mulher].
8  pessoa[filho=>>homem].
9
10 /* regras, consistindo de cabeça e corpo  da regra */
11
12 FORALL X,Y X[filho->>Y] <- Y:homem[pai->X].
13 FORALL X,Y X[filho->>Y] <- Y:homem[mãe->X].
14 FORALL X,Y X[filha->>Y] <- Y:mulher[pai->X].
15 FORALL X,Y X[filha->>Y] <- Y:mulher[mãe->X].
16
17 /* fatos */
18
19 abraham:homem.
20 sarah:mulher.
21 isaac:homem[pai->abraham; mãe->sarah].
22 ishmael:homem[pai->abraham; mãe->hagar:mulher].
23 jacob:homem[pai->isaac; mãe->rebekah:mulher].
24 esau:homem[pai->isaac; mãe->rebekah].
25
26 /* query */
27
28 FORALL X,Y <- X:mulher[filho->>Y[pai->abraham]].
```

2.2.4 CML (*Conceptual Modeling Language*)

Conforme [152], a CML [145] foi desenvolvida para dar suporte ao *framework* CommonKADS e é basicamente uma notação informal para a modelagem do conhecimento. Possui muitas similaridades com o OCML, mas faltam capacidades operacionais [119]. Provê mais primitivas que o OCML na perspectiva estrutural, mas a sua estrutura de modelagem restringe a capacidade de especificação de ontologias.

2.2.5 OCML (*Operational Conceptual Modeling Language*)

OCML é uma linguagem de modelagem, desenvolvida e mantida pelo *Knowledge Media Institue* (KMI) [119]. Possui mecanismos para a definição de relações, funções, classes, instâncias, regras e procedimentos. É similar à Ontolingua, com algumas extensões como a adição de regras dedutivas e de produção, e provê provas de teoremas e mecanismos de avaliação de funções para construções da Ontoligua.

2.2.6 LOOM

Loom é uma linguagem para a representação do conhecimento e, também, um sistema de raciocínio baseado em lógica descritiva [110]. Foi desenvolvida em 1991 pelo ISI (*Information Sciences Institute – University of Southern California*). A lógica descritiva representa o conhecimento de um domínio definindo os conceitos relevantes à ele, ou seja, sua terminologia. Ela utiliza destes conceitos para especificar as propriedades de objetos e indivíduos do domínio, criando uma descrição do mesmo [6].

2.2.7 Telos

Telos é uma linguagem construída para trabalhar vários aspectos da modelagem [120]. Basicamente, é uma linguagem para a representação do conhecimento com um foco em orientação a objetos. Um objeto pode ser um indivíduo (entidades, conceitos, nodos) ou um atributo (relacionamentos, ligações de atributos). Esta linguagem possui um grande poder de expressividade, especialmente pelo fato de indivíduos e atributos serem tratados uniformemente.

2.2.8 RDF (*Resource Description Framework*)

Segundo Lassila e Swick [99], a RDF foi endossada e recomendada pelo W3C como uma linguagem baseada em redes semânticas para descrever recursos da Web. RDF-*Schema* [20] também foi recomendada pelo W3C como uma extensão do RDF com primitivas baseadas em quadro.

A RDF representa relacionamentos entre recursos. RDF-*Schema*, que é um vocabulário com descrição de linguagem para RDF, provê mecanismos para declaração dessas propriedades ou relacionamentos [20].

A combinação de RDF com o RDF-*Schema* é conhecido como RDF(S). A linguagem resultante é mais expressiva e permite a representação de conceitos, taxonomias de conceitos e relações binárias. Algumas máquinas de inferência foram criadas para esta linguagem, principalmente para validar restrições [99].

RDF e RDF(S) são linguagens que têm como objetivo fornecer uma maneira uniforme de representação de metadados em XML [152]. Uma das áreas em que o RDF é bastante aplicado é na descoberta de recursos, onde é capaz de fornecer melhores capacidades aos mecanismos de busca. O maior objetivo da linguagem RDF é definir um mecanismo para descrição de recursos que não faça suposições a respeito de um domínio de aplicação em particular e que não defina as semânticas de nenhum domínio de aplicação. Na linguagem RDF, a definição do domínio deve ocorrer de forma imparcial, uma vez que o mecanismo deve ser apropriado para a descrição de informações a respeito de qualquer domínio [99].

A linguagem RDF, como muitos outros sistemas de modelagem orientados a objetos, é baseada em um sistema de classes. Uma coleção de classes é chamada de *schema*, e estas classes são organizadas em uma hierarquia, provendo extensibilidade através de subclasses [99]. O modelo de dados do RDF consiste de três tipos de objetos: recursos, propriedades (predicados descrevendo os recursos) e valor (objetos atribuindo um valor de uma propriedade a um recurso). Uma definição de recurso em RDF normalmente é vista como um grafo RDF, onde sempre há triplas RDF, como as da Figura 2.7 [106]. O RDF não possui mecanismos para a definição de relacionamento entre estes objetos, mas o RDF-*Schema* possui.

Figura 2.7: *Tripla RDF [106]*

No grafo da Figura 2.7, os nós e arcos representam os recursos, suas propriedades e valores, na forma de declarações contendo sujeito, predicado e objeto [167].

São dois os formatos de arquivos que possibilitam o compartilhamento de dados em RDF: RDF/XML e N3/Turtle [157].

Em um documento RDF/XML existem dois tipos de nós: *resource e property*. Os nós do tipo *resource* são os sujeitos e objetos das sentenças. No Código 2.2, um

exemplo extraído de [157], o nó *Description*, nas linhas 5 a 10, é o único nó de recurso. Os nós de recurso contêm apenas nós de propriedade, que representam afirmações. Há três afirmações neste exemplo, todos com o sujeito `<http://www.princeton.edu>`, na linha 5, e com os predicados `geo:lat`, `geo:long`, e `edu:hasDept`, nas linhas 6, 7 e 8 respectivamente.

Os nós de propriedade, nas linhas 6 e 7, contêm valores literais, como "40.35" e "-74.66", ou uma referência para o objeto recurso usando o atributo `rdf:resource`, na linha 8.

Código 2.2 Exemplo de código em linguagem RDF.

```
1  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#"
2       xmlns:dc="http://purl.org/dc/elements/1.1/"
3       xmlns:geo="http://www. w3.org/2003/01/geo/wgs84\_pos\#"
4       xmlns:edu="http://www.example.org/">
5       <rdf:Description rdf:about="http://www.princeton.edu">
6           <geo:lat>40.35</geo:lat>
7           <geo:long>-74.66</geo:long>
8           <edu:hasDept rdf:resource="http://www.cs.princeton.edu"
9            dc:title="Department of Computer Science"/>
10       </rdf:Description>
11 </rdf:RDF>
```

A notação N3/Turtle é o outro sistema para escrever arquivos RDF, baseada no mesmo modelo abstrato da notação RDF/XML, diferindo principalmente na legibilidade. O exemplo com o Código 2.3, extraído de [157], mostra a mesma informação do exemplo anterior, descrita na notação N3/Turtle.

Código 2.3 Exemplo de notação N3/Turtle.

```
1  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#> .
2  @prefix dc: <http://purl.org/dc/elements/1.1/> .
3  @prefix geo: <http://www. w3.org/2003/01/geo/wgs84\_pos\#> .
4  @prefix edu: <http://www.example.org/> .
5
6
7  <http://www.princeton.edu> geo:lat "40.35" ; geo:long "-74.66" .
8  <http://www.cs.princeton.edu> dc:title "Dept. of Computer Science" .
9  <http://www.princeton.edu> edu:hasDept <http://www.cs.princeton.edu> .
```

A notação Turtle (*Terse RDF Triple Language*) [13], é uma extensão de N3 (N-Triples) [15]. Essas notações são comumente tratadas sem distinção, mas existem diferenças entre elas que podem afetar o resultado de analisadores sintáticos (*parsers*).

Embora disponibilize o modelo e a sintaxe necessários para a descrição de recursos, ou seja, as regras que especificam os elementos de uma sentença, a linguagem

RDF não possibilita a definição de significado dos recursos. Mais especificamente, enquanto propriedades RDF podem ser entendidas como atributos dos recursos, além de representar relacionamentos entre recursos, RDF não provê mecanismos para descrever essas propriedades, nem para descrever o relacionamento entre essas propriedades e outros recursos. Esse é o papel desempenhado pela linguagem RDF-*Schema* [167].

RDF-*Schema* pode ser utilizado diretamente para descrever ontologias, apesar de seu principal propósito não este [152]. RDF-*Schema* provê um conjunto fixo de primitivas de modelagem para a definição de ontologias (classe, recurso, propriedade e relacionamento) e uma maneira padrão de codificá-las em XML. Mas, o RDF-*Schema* possui um poder limitado de expressividade, uma vez que axiomas não podem ser definidos diretamente.

2.2.9 DAML (*DARPA Agent Markup Language*)

DAML (*DARPA Agent Markup Language*) é uma linguagem desenvolvida como uma extensão para XML e RDF. A DAML possui infra-estrutura básica que permite às máquinas realizarem a mesma classificação de inferências que os seres humanos fazem. Um sistema em DAML pode inferir conclusões que não estão explicitamente declaradas. [40, 1].

2.2.10 OIL (*Ontology Inference Layer*)

OIL (*Ontology Inference Layer* é uma camada de inferência e representação baseada na Web, que combina a utilização de modelagem de primitivas provenientes das linguagens baseadas em *frames* com a semântica formal. A sintaxe de OIL é baseada em RDF-*Schema* e provê definição de classes e *slots* (relações), e um conjunto limitado de axiomas [152].

2.2.11 DAML+OIL (DARPA *Markup Language + Ontology Inference Layer*)

A linguagem *DARPA Markup Language + Ontology Inference Layer* (DAML+OIL) foi construída com base na linguagem DAML em um esforço para combinar vários componentes da linguagem OIL. É construída sobre padrões W3C tais como RDF e RDF-*Schema*, e estende estas linguagens com primitivas de modelagem mais ricas, além de possuir semântica clara e bem definida [31].

Existem diferenças entre as linguagens OIL e DAML+OIL devido, principalmente, ao fato de que DAML+OIL foi baseada em RDF. Assim, algumas construções em RDF são possíveis em DAML+OIL, mas não em OIL.

2.2.12 XOL (*Ontology Exchange Language*)

XOL foi inicialmente desenvolvida para tratar ontologias na área de bioinformática [92]. A linguagem é baseada em XML, apesar de poder ser utilizada em ontologias de diversos domínios. É uma linguagem bastante restrita, pelo fato de só possuir definições de conceitos, taxonomias e relações binárias. Foi desenvolvida pelo SRI *International* (*Stanford Research Institute*) [92].

2.2.13 SHOE (*Simple HTML Ontology Extensions*)

SHOE é uma extensão da lingugem HTML (*HyperText Markup Language*) para incorporar conhecimento semântico em documentos encontrados na Web através de anotações em páginas HTML [109]. SHOE provê primitivas de modelagem tanto para especificar quanto para estender ontologias. Cada página declara quais ontologias são utilizadas, o que torna possível a realização de buscas mais inteligentes. SHOE permite a representação de categorias (classes), relações n-árias, regras de inferências e regras para a especificação das ontologias [152].

2.2.14 *Web Ontology Language* (OWL)

OWL é uma linguagem de marcação semântica para publicação e compartilhamento de ontologias na Web. Foi desenvolvida como uma extensão do vocabulário RDF e é derivada da linguagem DAML+OIL [52]. OWL representa o significado de termos em vocabulários e relacionamentos entre eles, facilitando a interpretação do conteúdo Web por máquinas, já que oferece um vocabulário adicional com uma semântica formal, por exemplo, relações entre classes, cardinalidade, características de propriedades.

Enquanto RDF promove a associação e integração de dados distribuídos, OWL possibilita o raciocínio sobre esses dados [64]. OWL também é mais completa do que RDF-*Schema*, pois permite definir tudo que RDF-*Schema* é capaz e mais, como a possibilidade de estabelecer que duas classes são disjuntas, o que não ocorre com RDF-*Schema*. Em OWL, é possível também estabelecer restrições de propriedades herdadas das superclasses nas subclasses, podendo essas tais restrições, terem relações com outras classes.

OWL é dividida em três sub-linguagens, de acordo com o nível de formalidade exigido e a liberdade dada ao usuário para a definição de ontologias: OWL *Lite*, OWL DL e OWL *Full*. Em ordem crescente de expressão, estas três sub-linguagens servem para atender tanto os mais exigentes quanto aqueles que precisam apenas de uma maneira rápida para desenvolver uma ontologia.

OWL *Lite* é a sub-linguagem mais simples sintaticamente. É indicada para usuários que precisam de uma hierarquia de classificação e restrições simples. Por exemplo, dá suporte a cardinalidade. No entanto, essa cardinalidade só pode assumir os valores 0 ou 1. É mais simples fornecer suporte a OWL *Lite* do que a suas irmãs. A OWL DL aumenta a expressividade mantendo a decidibilidade. Sendo um subconjunto da OWL *Full* e fazendo uso da lógica descritiva (*description logics*), de onde vem a extensão DL [96]. A OWL *Full* mantém a expressividade, porém sem garantia de decibilidade, uma vez que instâncias e propriedades podem ser especificadas como classes, diferentemente da OWL DL.

A OWL DL proporciona expressividade máxima, garantia de completude computacional e decidibilidade. Esta linguagem inclui todos os construtores OWL, mas adiciona algumas restrições. As restrições mais significativas são: uma classe não pode ser um indivíduo ou propriedade e uma propriedade não pode ser um indivíduo ou classe [96].

A OWL *Full*, segundo Djuric et al. [52], é uma extensão da OWL DL, que é uma extensão da OWL *Lite*. Desta forma, toda ontologia OWL *Lite* é uma ontologia OWL DL e OWL *Full*, e toda ontologia OWL DL é uma ontologia OWL *Full*.

OWL *Full* pode ser vista como uma extensão ao RDF, ao passo que OWL *Lite* e OWL DL podem ser consideradas extensões a visões restritas do RDF. Todo documento OWL é um documento RDF e todo documento RDF é um documento OWL *Full*. No entanto, nem todo documento RDF é um documento OWL *Lite* ou OWL DL (pois estas linguagens são mais restritas). Por essa razão, deve haver um certo cuidado na hora de migrar documentos RDF para OWL *Lite* ou OWL DL.

Tanto na OWL DL quanto na OWL *Full* existem *tags* para o mapeamento entre ontologias, dentre as *tags*, tem-se a `sameClassAs`, para dizer que uma classe é equivalente a outra, a `sameIndividualAs`, para indivíduos equivalentes e a `differentIndividualFrom`, para indicar que os indivíduos são distintos.

A seguir, o Código 2.4, retirado de [106], representa um pequeno exemplo de uma ontologia expressa em OWL, onde pode-se observar a sua sintaxe básica. A ontologia representada, visualizada em forma de grafo, pode ser vista na Figura 2.8, onde os relacionamentos entre os conceitos são do tipo *is-a*, ou seja, subclasse.

Figura 2.8: *Ontologia descrevendo impressoras [106].*

Observa-se, na Figura 2.8, que todas as classes em OWL são uma subclasse de `owl:Thing`, pois a propriedade `subClassOf` é transitiva. A ontologia representada no Código 2.4, permite ver as definições das classes nas linhas 10, 15, 20 e 21. Duas

propriedades foram definidas, `paginasPorMinuto` nas linhas de 24 a 27 e `resolucaoDPI` nas linhas 28 a 31. Ambas as propriedades possuem como domínio a classe impressora e como `contradomínio` um inteiro, sendo portanto, do tipo `DatatypeProperty`; caso tivessem como domínio outra classe, seriam `ObjectProperty`. Nas linhas de 32 a 35, é definida uma instância da classe HP1200 juntamente com as propriedades desta instância.

Código 2.4 Ontologia descrevendo impressoras em OWL.

```
1  <?xml version="1.0"?>
2   <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
5      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6      xmlns:owl="http://www.w3.org/2002/07/owl#"
7      xmlns="http://laico.cic.unb.br/laico#"
8     xml:base="http://laico.cic.unb.br/laico">
9   <owl:Ontology rdf:about=""/>
10  <owl:Class rdf:ID="Hp1200">
11     <rdfs:subClassOf>
12        <owl:Class rdf:ID="Laser"/>
13     </rdfs:subClassOf>
14  </owl:Class>
15  <owl:Class rdf:about="#Laser">
16     <rdfs:subClassOf>
17        <owl:Class rdf:ID="Impressora"/>
18     </rdfs:subClassOf>
19  </owl:Class>
20  <owl:Class rdf:ID="Recurso"/>
21  <owl:Class rdf:about="#Impressora">
22     <rdfs:subClassOf rdf:resource="#Recurso"/>
23  </owl:Class>
24  <owl:DatatypeProperty rdf:ID="paginasPorMinuto">
25     <rdfs:domain rdf:resource="#Impressora"/>
26     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
27  </owl:DatatypeProperty>
28  <owl:DatatypeProperty rdf:ID="resolucaoDPI">
29     <rdfs:domain rdf:resource="#Impressora"/>
30     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
31  </owl:DatatypeProperty>
32  <Hp1200 rdf:ID="HP1200_Lab-Pos">
33    <resolucaoDPI rdf:datatype="http://www.w3.org/2001/XMLSchema#int">600</resolucaoDPI>
34    <paginasPorMinuto rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10</paginasPorMinuto>
35  </Hp1200>
36 </rdf:RDF>
```

2.3 Ferramentas para o Tratamento de Ontologias

Existem diversas ferramentas para o tratamento de ontologias. Dentre elas destacam-se o OilEd, OntoEdit, DOE, Chimaera e o Protégé (ferramenta utilizada neste trabalho).

2.3.1 OilEd

OilEd é um editor de ontologias cujo objetivo inicial era demonstrar o uso e estimular o interesse na linguagem Oil [10]. Ele utiliza o raciocinador FaCT (*Fast Classification of Terminologies* [89]) para verificar a consistência das ontologias junto com a interface DIG [78], que permite o uso de outros raciocinadores para classificação das ontologias. O OilEd permite exportação para OWL, além de DAML+OIL e RDFS.

2.3.2 OntoEdit

OntoEdit é um ambiente gráfico de desenvolvimento para projeto e manutenção de ontologias [117]. Possui embutida a máquina de inferência FaCT. O processo de desenvolvimento de ontologias no OntoEdit é baseado em uma metodologia própria denominada *On-To-Knowledge*.

O sucessor do OntoEdit é o OntoStudio, que é baseado no ambiente de desenvolvimento Eclipse, da IBM, e está disponível nas versões: profissional e livre (para uso não comercial) [127]. As linguagens suportadas são: OWL e RDF, além de F-logic e OXML. Estas últimas são linguagens otimizadas para processamento de ontologias baseado em lógica. O OntoStudio inclui também um avaliador, chamado *OntoStudio Evaluator*, utilizado na implementação de regras durante a modelagem.

2.3.3 DOE (*The Differential Ontology Editor*)

DOE (*The Differential Ontology Editor*) é um editor simples para criação de ontologias [159]. No DOE são construídas as taxonomias dos conceitos e relações, justificando explicitamente a posição de cada item na hierarquia. Feito isto, o usuário pode então adicionar sinônimos e definições. Considera-se duas taxonomias do ponto de vista semântico extensional. O usuário poderá aumentá-las ou adicionar restrições às relações. A ontologia pode ser traduzida em uma linguagem de representação do conhecimento. As linguagens disponibilizadas são: RDFS, OWL, DAML+OIL, OIL, CGXML [126] (linguagem para especificação de gráficos conceituais). DOE não é um ambiente de desenvolvimento completo de ontologias. Ele oferece técnicas inspiradas em linguística, que atribuem uma definição léxica aos conceitos e relações utilizados e justifica a hierarquia do ponto de vista teórico, que pode ser entendido por pessoas.

2.3.4 Chimaera

Chimaera é um sistema de software que permite aos usuários criarem e manterem ontologias distribuídas na Web [115]. As duas maiores funções que ele oferece são mesclagem de múltiplas ontologias e diagnóstico de várias ontologias ou de uma ontologia individualmente. Também permite o carregamento de bases de conhecimento em diferentes formatos, reorganização de taxonomias, resolução de conflitos de nomes, edição de termos, entre outras opções. Ele pode carregar e exportar arquivos no formato OWL e DAML.

2.3.5 Protégé

Protégé é um ambiente integrado utilizado por desenvolvedores de sistemas e especialistas em um domínio específico para desenvolver e manter ontologias. Através dele é possível descrever os conceitos pertencentes à ontologia, juntamente com seus atributos e relacionamentos [5]. O editor Protégé possui uma arquitetura de metaclasses, documentos de formato padrão utilizados para definir novas classes em uma ontologia, o que o torna facilmente extensível e permite o seu uso juntamente com outros modelos de conhecimento. O Protégé permite interoperabilidade com outros sistemas de representação do conhecimento.

Esta ferramenta foi desenvolvida na Universidade de Stanford - Califórnia (EUA), e está disponível para utilização gratuitamente. Pode ser usada para navegação e gerenciamento da ontologia. Ao contrário de outras ferramentas, o Protégé utiliza a instalação local ao invés de utilizar uma arquitetura cliente-servidor, através da Internet. Além disso, gera saídas em diversas linguagens ontológicas e possibilita aos usuários modificá-lo para ser editor de uma linguagem específica [117].

Segundo o grupo Stanford Medical Informatics, da Stanford University School of Medicine [5], o Protégé é uma ferramenta que permite aos usuários construirem domínios ontológicos e customizarem formulários de entrada de dados. Atualmente, o Protégé é um editor de ontologias e de conhecimentos *Open Source*, que permite duas maneiras principais de modelar as ontologias:

- Através do editor do **Protégé-Frames**: neste editor uma ontologia é considerada como sendo um conjunto de classes organizadas numa hierarquia que representa um domínio de conceitos, um conjunto de *slots* associados à classes para descrever as suas propriedades e relacionamentos, e um conjunto de instâncias dessas classes, que contêm os valores específicos das propriedades [5];
- Através do editor **Protégé-OWL**: este editor é uma extensão do Protégé que aceita a *Web Ontology Language* (OWL). Dada uma determinada ontologia, a

semântica formal da OWL especifica como derivar suas consequências lógicas, isto é, fatos que não estão explicitamente presentes na ontologia, mas envolvidos pela semântica. [168].

Protégé é também uma plataforma que pode facilmente ser estendida para incluir componentes gráficos como tabelas, arquivos de som, imagens e vídeos, e vários formatos de armazenamento, como OWL, RDF, XML, além de tratar HTML.

O Protégé pode ser utilizado como um *plugin* do Eclipse e pode ser utilizado em qualquer máquina que tenha uma JDK (*Java Development Kit* ou Kit de Desenvolvimento Java) instalada. Possui compatibilidade com qualquer banco de dados que tenha driver JDBC como: *Oracle*, *MySQL*, *Microsoft SQL Server* e *Microsoft Access*. Permite múltiplos usuários simultaneamente e tem uma API documentada para auxílio aos usuários [1].

O Protégé possui uma arquitetura integrável a diversas aplicações, através de componentes que podem ser conectados ao sistema. Essa arquitetura propiciou o desenvolvimento de *plugins* que acrescentam novas funcionalidades acopladas às já existentes. Além de *plugins* para modificações na interface com o usuário, a arquitetura do Protégé permite *backend plugins* que lêem e escrevem em diferentes formatos de armazenamento.

Como padrão, a ferramenta armazena suas bases de conhecimento em um formato de arquivo de propósito especial. Atualmente, podem ser criadas classes e instâncias em CLIPS, OIL, Jess, XML, RDF, F-Logic, Prolog, Topic Maps (linguagem padrão ISO para descrever estruturas de conhecimento em páginas da Web), além do armazenamento em bancos de dados relacionais. Essa última capacidade é fundamental para grandes bases de conhecimento, que excedem o tamanho físico da memória principal e precisam de uma maneira eficiente de recuperar a informação em memória secundária [1].

Embora seja uma ferramenta relativamente recente, o Protégé tem evoluído desde o seu aparecimento [69]:

- ONCOCIN (1980s) - Sistema de suporte a decisão clínica (CDSS) para a gestão de pacientes relacionados com a triagem clínica de cancro. Pode ser considerado como a primeira ferramenta de verificação referida na literatura. Esta fazia a verificação de consistência (conflito, redundância, especialização de regras) e de regras em falta [144].
- OPAL (1985) - Interface gráfica para codificar as triagens clínicas de câncer para o ONCOCIN.
- Protégé (1988) - Sistema para definir modelos de triagem para qualquer domínio, para gerar OPAL para o ONCOCIN (CDSS para qualquer domínio de triagem).

- Protégé-II (antes de 1990s) - Ambiente de engenharia do conhecimento (em plataforma NextStep[2]) para definir o modelo e gerar um editor de interface gráfica para qualquer domínio.
- ProtegeWin (meados de 1990s) - Versão Windows que enfatiza a usabilidade; Grupos de utilizadores externos.
- Protégé-2000 (1990s – 2003) - Versão baseada em Java que enfatiza modelos de conhecimento formal, interoperabilidade com outros formalismos (p.ex.: Ontolingua, RDF); Desenvolvimento de arquitetura de expansão de *plug-in's*; Possui código aberto.
- Protégé, v2.0 (versão em 2003) - Desenvolvimento multiusuário; Suporte embutido para XML; Suporte de Web Semântica.
- Protégé, v3.2.1 (versão em 2006) - Suporte OWL; Incorpora o modelo de conhecimento *Open Knowledge Base Connectivity* (OKBC).
- Protégé, v.3.3.1 (versão em 2007) - Suporte OWL 1.0, RDF(S), e *Frames*.
- Protégé, v.4 (beta em 2009) - Suporte OWL 2.0.

2.4 Mecanismos de inferência/racioncínio

A OWL permite que sejam utilizados diversos mecanismos de inferência. Esses mecanismos são utilizados [106]:

- para que novas informações possam ser derivadas de uma ontologia;
- para realização de validações que verifiquem se todas as regras definidas pela ontologia são obedecidas;
- para realizar consultas mais elaboradas através do mecanismo de inferência.

Vários são os mecanismos de inferência que podem ser utilizados com ontologias. A Tabela 2.1 apresenta uma comparação dos mecanismos de inferência existentes.

[2]NextStep é um sistema operacional, lançado em 10 de setembro de 1989, pela NeXT, atualmente parte da *Apple Computer*[139]

	Racer	FaCT	Pellet	Triple
Lógica	DL	DL	DL	Lógica de Horn[1]
Suporta	OWL-DL	OWL-DL	OWL-DL	RDF
Baseada em	Lisp	Lisp	Java	XSB
XML	Sim	Não	Sim	Não
Decidível	Sim	Sim	Sim	Sim
Verificação de Consistência	Sim (OWL-Lite)	Sim	Sim (OWL-Lite)	Não
Interface	DIG, Java, GUI	DIG, Linha de comando	DIG, Java e Jena	Java
Query	Racer query language	-	RDQL	Lógica de Horn
Limitações Conhecidas	-	Não suporta A-Box	-	Só suporta RDF

Tabela 2.1: *Comparativo de mecanismos de inferência*

Mecanismos de raciocínio ou inferência, no paradigma clássico de modelagem, incluem normalmente inferência sobre inclusão de classes em hierarquias (*class subsumption*), verificação de inconsistência de classes, verificação de equivalência de classes, inferência de instâncias e igualdade de instâncias [167].

Conclusões não definidas explicitamente na base de conhecimento podem ser inferidas a partir de regras e axiomas presentes na base, identificando o que é necessariamente verdadeiro e consistente com o sistema de axiomas. Ontologias em OWL podem, deste modo, ser processadas por um mecanismo de inferência para alcançar diferentes metas [167].

Os raciocinadores ou *reasoners* são capazes de fazer inferências sobre objetos de dados utilizando a informação de seus relacionamentos definidos através de uma ontologia, ou seja, verificam se novas informações podem ser derivadas de uma ontologia. Os raciocinadores permitem a verificação ou a validação de uma ontologia, verificando se todas as regras definidas pela ontologia são obedecidas.

O *framework* Jena possui um subsistema de inferência, que foi projetado para permitir que vários mecanismos de inferência possam ser utilizados em conjunto com o próprio *framework* [51]. Esses mecanismos de inferência são utilizados para que informações adicionais possam ser extraídas das ontologias, seja através de axiomas ou de regras associadas ao raciocinador. O *framework* Jena já tem inclusos alguns mecanismos de inferência pré-definidos [51]:

- **Raciocinador transitivo (*transitive reasoner*)**: um mecanismo de inferência simples, que provê implementação de propriedades simétricas e transitivas das propriedades `rdfs:subPropertyOf` e `rdfs:subClassOf`.

[1] Lógica de Horn é uma linguagem de programação paralela elaborada a partir da análise da estrutura básica e da prática da lógica de programação [160].

- **Raciocinador de regras RDFS**: implementa um conjunto configurável de vínculos RDFS.
- **Raciocinadores OWL, OWL Mini, OWL Micro**: implementação de um conjunto útil, porém incompleto da linguagem OWL *Lite*.
- **Raciocinador DAML micro**: usado na DAML API.
- **Raciocinador genérico de regras**: baseado em regras definidas pelo usuário.

Os mecanismos de inferência ou raciocínio possuem também a funcionalidade de validar uma ontologia. Neste caso, na validação é verificado se não há alguma inconsistência com relação à linguagem escolhida para definição da ontologia e também se não há alguma inconsistência com relação à definição da ontologia em si. Um exemplo de validação contra a linguagem é quando se define que uma determinada propriedade terá dois contra-domínios distintos, o que levará ao aparecimento de uma inconsistência na definição dessa propriedade. Outro exemplo de validação é quando uma propriedade é declarada como tendo um contradomínio específico, mas uma instância de uma classe que possua essa propriedade, tenha sido declarada como tendo um contra-domínio de um tipo diferente do declarado.

Uma importante tarefa de inferência, a inclusão de classes *class subsumption*, implica em testar se uma classe é ou não subclasse de outra, com base na descrição das classes. Por exemplo, uma hierarquia declarada manualmente serve como base para que um mecanismo de raciocínio possa inferir e manter uma hierarquia de herança múltipla.

O mecanismo de inferência ou raciocínio também pode executar verificação de consistência. Com base na descrição de uma classe, esse mecanismo pode inferir se essa classe pode ou não ser instanciada, como exposto por Horridge [87]. Adicionalmente, de acordo com Roure [141], uma vez que ontologias codificam relacionamento entre classes de objetos, inferências entre instâncias dessas classes podem ser realizadas.

2.4.1 Linguagens de Consulta

A seguir será feita uma breve descrição de algumas linguagens de consulta de inferência:

- **RQL**: é uma linguagem tipada que segue uma abordagem funcional, que oferece suporte a variáveis que caracterizam expressões generalizadas, tanto para os nós quanto para aos arcos de um grafo RDF [93]. Exemplo:

Código 2.5 Exemplo de uma consulta em RQL

```
1 select X
2 from {X}ministra{Y}metodologia{Z}
3 where Z like "aulas expositivas"
```

Se o objetivo da coordenação de um curso fosse encontrar os professores que ministram disciplinas com a metodologia “aulas expositivas”, a consulta seria representada em RQL conforme o Código 2.5.

- **SeRQL**: significa Linguagem de Consulta RDF *Sesame* ou *Sesame* RDF *Query Language* e é uma linguagem de consulta e transformação baseada em várias linguagens existentes, mais notavelmente RQL, RDQL e N3 [21]. Exemplo:

Código 2.6 Exemplo de consulta SeRQL

```
1  SELECT node FROM    # variáriveis que irão aparecer
2                      # no resultado.
3
4    {node} rdf:type  {<sr:Professor>}, # 1° caminho:
5                                       # Nó do tipo
6                                       # "Profesor".
7
8    {link} sr:source_node {node};      # 2° caminho(I):
9           rdf:type {<sr:Trabalha>};   # Elo com origem no nó
10                                      # e do tipo "Trabalha".
11
12          sr:target_node {?}          # 2° caminho(II):
13           rdf:type {AreaDePesquisa}  # Elo com destino em um nó X
14                                      # cujo tipo é "AreaDePesquisa".
```

Quando se escreve uma consulta em SeRQL, a sintaxe pode até parecer de início com SQL tradicional, porém é semanticamente bem diferente.

Como mostrado no Código 2.6, o principal da consulta em SeRQL está na cláusula FROM onde estabelece o padrão de subgrafo que se está procurando, sempre no formato de triplas do tipo sujeito, predicado e objeto.

Analisando-se o Código 2.6, percebe-se que, no 1^o caminho, descreve-se o padrão *node*, que é uma incógnita dentro da consulta, *type* <Professor>. No 2^o caminho descreve-se *link*, uma 2^a incógnita, *source_node*, *node*, ou seja, um *link* com origem no nó que se pediu. Com ponto-e-vírgula, faz-se um atalho e pode-se suprimir o *link*, mas na verdade está-se procurando *link type* <Trabalha>. Finalmente, completando o segundo caminho, pede-se *link target_node*, *node*, que é um nó parametrizado que se define na hora de executar a consulta e que este nó seja do tipo “AreaDePesquisa”.

Graficamente, o Código 2.6 fica como mostrado na Figura 2.9:

O 1^o caminho: nó do tipo professor.

A 1^a parte do 2^o caminho: Elo com origem no nó e do tipo Trabalha.

A 2^a parte do 2^o caminho: Elo com destino em um nó X, passado como parâmetro, que é do tipo Area de Pesquisa.

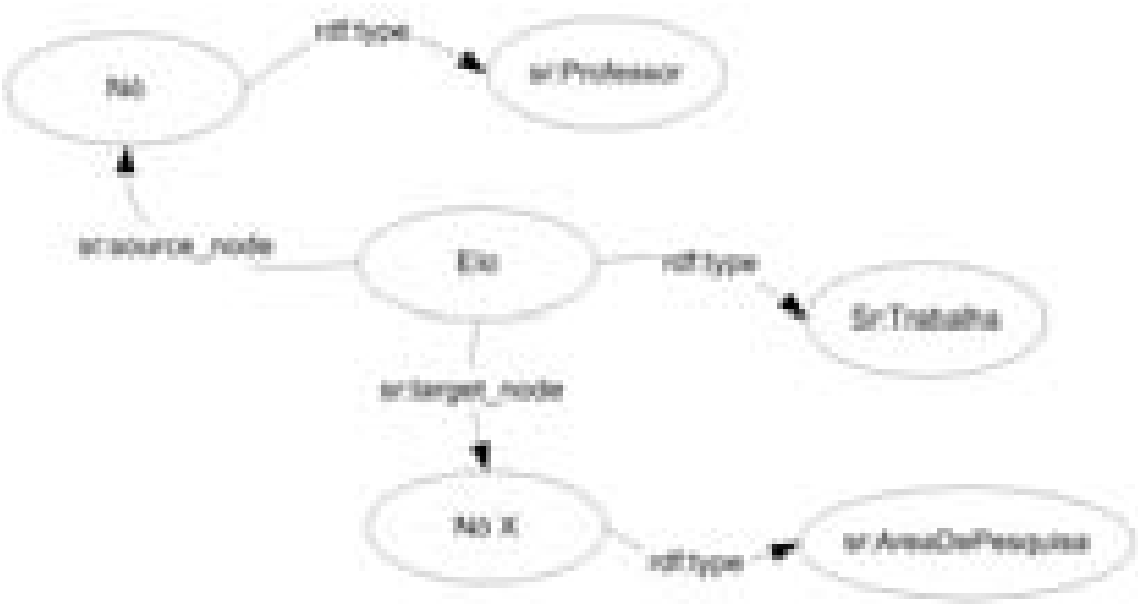

Figura 2.9: *Consulta SeRQL apresentada graficamente*

- **TRIPLE**: o termo Triple denota uma linguagem de regras e de consulta [147]. A linguagem Triple [148] é uma linguagem de regras cujo objetivo é prover racioncínio e transformação sobre conhecimento representado no padrão RDF. A linguagem é baseada na lógica de Horn [160] e possui extensões sintáticas que permitem processar primitivas RDF, como *namespaces*, asserções e recursos. Quando uma consulta é submetida na linguagem Triple, todos os dados RDF e as regras Triple são compiladas em um programa Prolog e submetidas ao sistema de inferência.
- **N3**: a Notation3 (N3) provê uma sintaxe baseada em texto para RDF [15]. Assim, o modelo de dados de N3 segue o modelo de dados da RDF. Adicionalmente, N3 permite a definição de regras, que são denotadas utilizando uma sintaxe especial. Uma asserção RDF, formada por sujeito, predicado e objeto, é escrita na notação N3 da seguinte forma:

  ```
  <sujeito> <predicado> <objeto>
  ```

  Exemplo:

  ```
  <#joão> <#conhece> <#maria>
  ```

  Onde `<#joão>` representa o o sujeito da asserção, `<#conhece>` representa o predicado da asserção e `<#maria>` representa o objeto da asserção.
- **Versa**: tem uma abordagem interessante em que o bloco principal de construção da linguagem é uma lista de recursos RDF. Nesta abordagem, RDF-triples

desempenha um papel chamado de operações transversais, que têm a forma `ListExpr - ListExpr -> BoolExpr`. Estas expressões retornam uma lista de todos os objetos de correspondência tripla. Por exemplo, a expressão transversal `all() - rdfs:label -> *`, retorna uma lista contendo todos os *labels* [86].

2.4.2 A ferramenta Pellet

Pellet é uma ferramenta de código aberto em Java para inferência sobre a linguagem OWL-DL, desenvolvida pela Universidade de Maryland, EUA (*University of Maryland's Mindswap Lab*). Atualmente, Pellet é explorada comercialmente por Clark & Parsia LLC [55]. Pellet atende às recomendações do W3C para casos de testes, definidas em Carrol e Roo [27], para verificadores de documentos OWL:

1. Verificação de sintaxe OWL,
2. Verificação de consistência OWL para OWL *Lite*, OWL-DL e OWL *Full*,
3. Verificação completa de consistência OWL para OWL *Lite*.

Uma verificação completa é decidível em relação à semântica. Uma vez que OWL *Full* não é decidível, não existe verificador de consistência completo para OWL *Full*. Pellet é um verificador sintático e um verificador de consistência completo, isto é, semanticamente decidível, para OWL-DL [167].

Pellet é amplamente utilizada pela comunidade de Web semântica. Ela integra a verificação de consistência ao seguinte conjunto padrão de serviços de inferência lógica, os quais podem ser acessados através de consultas [167]:

- **verificação de consistência** - assegura que uma ontologia não contém fatos contraditórios;
- **satisfatibilidade de conceito** - verifica se é possível a uma classe (conceito) possuir instâncias;
- **classificação** - computa as relações de subclasses entre todas as classes nomeadas para criar a hierarquia completa das classes. A hierarquia de uma classe pode ser utilizada para responder consultas como a obtenção das subclasses de uma determinada classe;
- **realização** - define a classe mais específica (direta) à qual um indivíduo pertence.

O conjunto completo de funcionalidades e características de Pellet inclui a realização de consultas conjuntivas, otimizadas através de algoritmos internos ao mecanismo, usando linguagens como SPARQL (que será apresentada na Seção 2.5.2 como exemplo

de linguagem para recuperação de informações). Pellet também realiza verificação sintática automática através de heurísticas que tentam transformar documentos OWL *Full* em OWL-DL e então processá-los normalmente. Na realização do detalhamento e depuração de ontologias, Pellet considera a totalidade de características OWL-DL (propriedades inversas e transitivas, raciocínio sobre tipos de dados, restrições de cardinalidade, classes enumeradas e declarações de instâncias) [167].

A arquitetura de Pellet é centrada em um pequeno motor de inferência adaptado para sofrer extensões. Esse modelo permitiu o desenvolvimento de interfaces para diferentes ferramentas OWL/RDF, tais como Jena [28] e *WonderWeb* OWL API [12], ou para apoiar aplicações que se comunicam através da interface DIG [11]. Pellet provê diferentes interfaces para análise de ontologias e também implementa interfaces de raciocínio, definidas pelas ferramentas de extensão, para responder às consultas.

2.5 Recuperação de Informações

As bases de conhecimento existentes em grades semânticas e na Web semântica são mantidas para ajudar a resolver os problemas de heterogeneidade e integração de recursos em tais sistemas. Para que sejam efetivamente utilizadas, é necessário definir estratégias eficientes para recuperação de informação. Recuperar informações em bases de conhecimento semânticas envolve, como descrito na Seção 2.4, realização de consultas baseadas em mecanismos de inferência [167].

Na seção 2.4.1, foram apresentadas algumas linguagens de consulta de inferência. Nesta seção, será discutida a utilização de duas diferentes abordagens para o problema de recuperação de informações em bases de conhecimento descritas em OWL-DL. Ambas as estratégias partem da necessidade da realização de consultas sobre bases de conhecimento. Também, ambas desempenham papel relevante no cenário atual da Web semântica e de grades semânticas em razão de sua larga adoção nestes contextos. O objetivo é avaliar as abordagens apresentadas e identificar soluções eficientes para a definição de uma arquitetura de metadados [167]. Serão discutidos, particularmente, o papel de ferramentas e linguagens de consulta empregadas em cada abordagem.

A linguagem de consulta OWL-QL é o ponto de partida da primeira abordagem. OWL *Query Language* (OWL-QL) é uma linguagem de consulta para bases de conhecimento em OWL. OWL-QL possibilita a realização de diálogos baseados em consulta e resposta que podem ser automatizados através de métodos de raciocínio. Serão apresentadas características relevantes de OWL-QL e ferramentas de consulta baseadas em inferência usando esta linguagem.

A segunda abordagem discute a utilização de uma ferramenta de inferência associada a uma linguagem de consulta simples. Especificamente, serão descritas caracte-

rísticas relevantes da linguagem de consulta SPARQL [53, 114].

2.5.1 OWL-QL

OWL-QL é uma linguagem formal que fornece um protocolo de comunicação entre agentes: o cliente, que faz a consulta, e o servidor, que responde a consulta utilizando conhecimento representado em OWL. OWL-QL é uma atualização da linguagem de consulta DAML *Query Language* (DQL) [59].

Nos diálogos de consulta e resposta em OWL-QL, o servidor pode usar métodos de raciocínio automático para derivar respostas e cenários nos quais o conhecimento usado para responder à consulta pode estar em múltiplas bases de conhecimento espalhados na Web semântica. Tais bases de conhecimento podem ou não ser especificadas pelo cliente. OWL-QL permite que o servidor defina as bases de dados a serem usadas em resposta a uma consulta e que o cliente peça ao servidor para identificar a base de conhecimento usada nas respostas.

O protocolo de respostas a consultas em OWL-QL é adaptável para que um servidor possa devolver conjuntos parciais de respostas à medida em que são computadas; e para que um cliente possa definir o número máximo de respostas que o servidor deve enviar no próximo bloco.

Há dois tipos de consultas básicas um OWL-QL: a consulta simples, e a consulta *se-então*. Será usada aqui a sintaxe abstrata da linguagem OWL-QL para representar os exemplos. Esta sintaxe especifica a consulta em linguagem natural, um padrão de consulta, expresso por uma série de sentenças do tipo *propriedade - objeto - valor* com algumas variáveis, cujos valores serão retornados ou não (conforme especificado abaixo do padrão de consulta, como exemplificado nas linhas de 3 a 5 do Código 2.7) como resposta e um padrão que especifica o formado esperado da resposta [54].

Uma consulta OWL-QL simples é um objeto que contém um padrão de consulta com várias sentenças OWL dentro das quais existem algumas referências a URIs que podem ser variáveis. O Código 2.7 apresenta um exemplo de consulta simples, onde o cliente deseja saber o tipo (lei ordinária, lei complementar, decreto, etc) do instrumento jurídico normativo de número 10856. Para responder à consulta *Qual o tipo do instrumento jurídico normativo de nº 10856?*, na linha 1, o servidor iria procurar entre as ontologias disponíveis, por uma que tratasse dos instrumentos jurídico normativos, provavelmente hospedada em um sítio governamental, e procuraria pela instância de número 10856 para então fornecer a resposta.

Código 2.7 Exemplo de consulta simples em OWL-QL [54].

```
1 Consulta:(Qual o tipo do instrumento jurídico normativo de nº 10856?)
2 Padrão de Consulta: {(numero ?inst 10.856) (tipo ?inst ?tipo)}
3 Lista de variáveis que devem ser retornadas: (?tipo)
4 Lista de variáveis que podem ser retornadas: ()
5 Lista de variáveis que não devem ser retornadas: (?inst)
6 Padrão de resposta: {(tipo 10.856 ?tipo )}
7
8 Resposta: ( O tipo do instrumento jurídico normativo de nº 10856 é Lei Ordinária )
9 Instância do Padrão de Resposta: {(tipo 10.856 Lei Ordinária )}
```

Também é possível fazer consultas do tipo *se-então*. Neste tipo de consulta, a mesma vem precedida por premissas que podem ser bases de conhecimento ou referências para outras bases. Deste modo, facilita-se o uso de um operador de implicação, inexistente em OWL. O Código 2.8 ilustra uma consulta com premissa, no caso, a premissa é *João é uma pessoa* e a consulta, *João tem pai?*, na linha 1. Através da informação contida na premissa, linha 2, de que João é uma pessoa e de informações escritas em ontologias de que toda pessoa tem pai e mãe, um agente dotado de capacidade de raciocínio responderia afirmativamente a consulta.

Código 2.8 Exemplo de consulta com premissa em OWL-QL [54].

```
1 Consulta: Se João é uma pessoa, então João tem pai?
2 Premissa: {(type João pessoa)}
3 Padrão de Consulta: {(pai João ?pai)}
4 Lista de variáveis que devem ser retornadas: (?pai)
5 ...
```

Mais detalhes sobre especificação, exemplos de uso, *download* e outras informações relevantes sobre OWL-QL são encontrados em [59].

2.5.2 A Linguagem SPARQL

A linguagem de consulta SPARQL contém capacidades de consultar padrões opcionais ou obrigatórios de grafos com suas conjunções e disjunções, além de possibilitar expressar restrições nos termos RDF que aparecerão no resultado da consulta. O resultado das consultas pode ser um conjunto de grafos em RDF. É recomendada pelo consórcio W3C, funcionando para qualquer fonte de dados que possa ser mapeada em RDF [134]. SPARQL é, ao mesmo tempo, uma linguagem de consulta e um protocolo de acesso.

Diversas ferramentas e APIs são disponibilizadas para prover funcionalidades (como resgatar campos na ontologia que estão na forma de URIs, nodos em branco e valores; extração de sub-grafos RDF da ontologia e a construção de novos grafos

RDF através das consultas SPARQL informadas) baseadas em SPARQL. Uma pequena lista disponível em [53] inclui, entre outros, ARQ, um processador SPARQL para Jena; KAON2, um mecanismo de inferência com suporte parcial para consultas em SPARQL e Pellet, o mecanismo Java de código aberto, descrito na Seção 2.4.2.

SPARQL, assim como RDF, é construída sobre uma tripla consistindo de sujeito, predicado e objeto. Uma tripla padrão SPARQL pode incluir variáveis, que são usadas para indicar itens de dados de interesse que resultarão de uma consulta. Sujeito, predicado e objeto, podem ser substituídos, cada um, por uma variável. Cada tripla que combina com o padrão agregará um valor real do conjunto de dados RDF para cada variável. O exemplo do Código 2.9, extraído de [53], usa variáveis para sujeito e objeto, respectivamente, *element* e *name*. O exemplo apresenta uma consulta que seleciona nome e número dos elementos do grupo 18 da tabela periódica.

Código 2.9 Exemplo escrito em linguagem SPARQL

```
1  PREFIX table: <http://www.daml.org/2003/01/periodictable/PeriodicTable\#>
2      SELECT ?name ?number
3      FROM  <http://www.daml.org/2003/01/periodictable/PeriodicTable.owl>
4       WHERE {
5              ?element table:name ?name;
6              table:atomicNumber ?number;
7              table:group table:group\_18.
8       }
```

A palavra chave *PREFIX* na linha 1, equivale, em sua essência, à definição de um espaço de nomes XML. O rótulo atribuído a um URI pode ser usado em qualquer lugar em uma consulta em substituição ao próprio URI. A cláusula SELECT na linha 2, é usada para definir os itens de dados que serão retornados pela consulta. A palavra chave FROM na linha 3, identifica os dados sobre os quais a consulta vai ser executada. Finalmente, a cláusula WHERE, na linha 4, define o padrão da consulta, isto é, uma coleção de padrões de tripla que identifica a forma do grafo que se deseja obter na combinação resultante da consulta.

SPARQL permite o uso de várias triplas no padrão para construção de consultas mais sofisticadas e complexas. As consultas podem ser construídas a partir de dados incompletos ou ausentes e sobre combinações de alternativas. Adicionalmente, o resultado pode ser ordenado e ter seu tamanho limitado.

2.6 Conclusão

Ontologias podem ser vistas como representações explícitas de conceitos relacionados a um domínio e provêm uma estrutura sobre a qual bases de conhecimento podem

ser construídas [155]. Uma ontologia define o vocabulário do domínio de um problema e restrições sobre as possíveis combinações de termos na modelagem do domínio. As razões mais comuns para o desenvolvimento de uma ontologia são compartilhar conhecimento comum, reutilizar o conhecimento sobre um domínio, tornar explícitas proposições assumidas sobre um domínio, separar conhecimento sobre um domínio de conhecimento operacional e analisar o conhecimento sobre um domínio [50].

Neste capítulo, foram apontadas várias definições e padrões para linguagens e ferramentas utilizadas na construção e validação de ontologias. Essas linguagens e ferramentas, permitem a exploração de projetos concretos. Esses projetos são voltados para a construção e utilização de bases de conhecimento eficientes com o uso de ontologias.

Neste trabalho foram utilizados as ferramentas Protégé, para a edição da ontologia, Pellet, para a verificação de consistência, a linguagem OWL, para a codificação.

CAPÍTULO 3

Desenvolvimento de Ontologias

Do ponto de vista da Engenharia de Ontologias, uma ontologia deve ser projetada para propósitos específicos. Quando se escolhe como representar algum elemento em uma ontologia, estão sendo tomadas decisões de projeto e, assim, são necessários critérios objetivos de avaliação para guiar o processo de projeto. A seguir, são listados os critérios preliminares de projeto de ontologias, propostos por Gruber [81], onde foram assumidos o propósito do compartilhamento de conhecimento e a interoperabilidade entre programas.

1. **Clareza**: uma ontologia deve expressar de maneira clara e objetiva o significado dos termos que ela define. A definição deve ser independente de contexto social ou computacional, utilizando formalismos para atingir este fim. Quando uma definição pode ser declarada na forma de axiomas lógicos, isso deve ser feito. Uma definição completa (um predicado definido pelas condições necessárias e suficientes) é preferível a uma definição parcial (definida somente pelas condições necessárias e suficientes).

2. **Coerência**: Uma ontologia deve ser coerente: isto é, deve permitir inferências que sejam consistentes com as definições. A definição dos axiomas deve ser logicamente consistente. Coerência deve também ser aplicada aos conceitos que são definidos informalmente, como aqueles descritos e exemplificados em linguagem natural. Se uma sentença que pode ser deduzida a partir de axiomas contradiz uma definição ou exemplo dado informalmente, então a ontologia é incoerente.

3. **Extensibilidade**: Uma ontologia deve ser projetada de modo que possa ser estendida e especializada, devendo, preferencialmente, antecipar a utilização de um vocabulário compartilhado.

4. **Mínima influência de codificação**: A conceitualização deve ser especificada em nível de conhecimento sem depender de uma codificação de nível simbólico particular. Influências (de linguagens de codificação) devem ser minimizadas, pois agentes para compartilhamento de conhecimento podem ser desenvolvidos em diferentes sistemas e estilos de representação.

5. **Mínimo compromisso ontológico**: Uma ontologia deve requerer compromissos ontológicos[1] mínimos suficientes para sustentar as atividades planejadas no compartilhamento de conhecimento. Uma ontologia deve fazer um mínimo de imposições possíveis sobre o mundo que está sendo modelado, permitindo liberdade às partes comprometidas com a ontologia, para possibilitar especialização e instanciação quando necessário. Desde que os compromissos ontológicos se baseiem na utilização consistente de um vocabulário, compromissos ontológicos podem ser minimizados através da especificação de uma teoria mais fraca (o que permite a definição de mais modelos) e da definição somente dos termos essenciais para a comunicação de conhecimento consistente com essa teoria.

Neste Capítulo, é apresentado um panorama comparativo que pode servir de apoio na definição de padrões metodológicos para construção de ontologias através da integração de princípios teóricos e metodológicos da ciência da informação. Também são apresentadas as contribuições de metodologias e métodos conhecidos para construção de ontologias.

3.1 Operações de manipulação de ontologias

As ontologias podem ser trabalhadas de diversas formas, de modo a se obter a interoperabilidade entre mais de uma ontologia [106]. A Figura 3.1 ilustra as quatro operações que podem ser realizadas: união, transformação, alinhamento e articulação.

Noy e Musen [123] definem estas quatro operações para integração de ontologias como:

- Figura 3.1(a): **União** – A partir de duas ontologias distintas, obtém-se uma terceira com os termos das ontologias originais, porém sem identificar a qual ontologia o termo pertencia originalmente. Para que a união seja mais eficiente, as ontologias devem representar domínios similares, fazendo com que termos similares sejam unidos.

[1] A noção de compromisso ontológico foi introduzida e discutida por Willard Quine [135]. No sentido quineano do termo, uma teoria acerca de um determinado segmento da realidade ou da experiência é simplesmente uma coleção consistente de crenças ou afirmações, expressas numa determinada linguagem, acerca do segmento em questão. Uma teoria será verdadeira se todas as crenças que a compõem, e logo todas as consequências lógicas dessas crenças, forem de fato verdadeiras. Os objetos com os quais uma teoria está ontologicamente comprometida são precisamente aqueles objetos cuja existência é assumida, de forma explícita ou implícita, pela teoria. Tais objetos formam a ontologia (ou melhor, uma das ontologias) da teoria: um conjunto de entidades a inexistência das quais teria como consequência a falsidade da teoria [135].

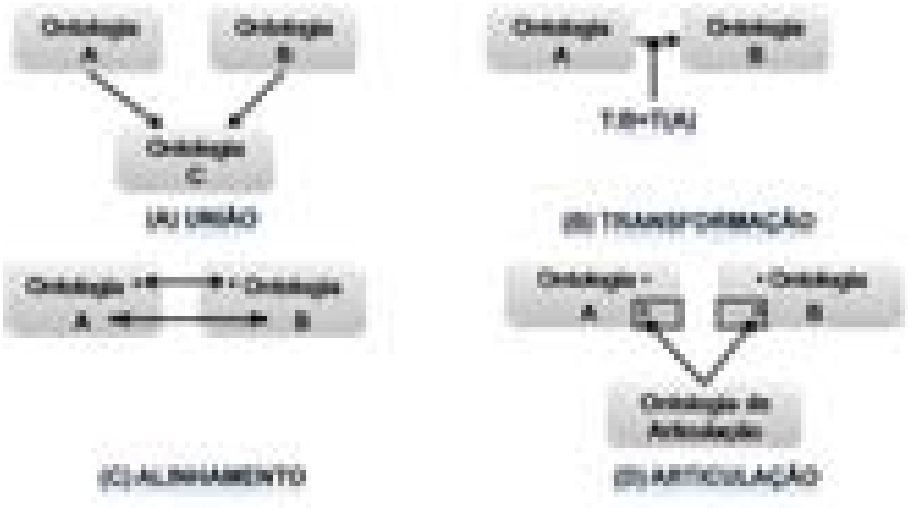

Figura 3.1: *Operações para manipulação de ontologias, adaptado de [123].*

- Figura 3.1(b): **Transformação** – A partir de uma ontologia original, obtém-se uma segunda ontologia transformando os termos da ontologia original em termos baseados em um vocabulário definido por meio de uma fórmula de transformação.
- Figura 3.1(c): **Alinhamento** – No alinhamento, não se cria uma nova ontologia, permanecendo com as ontologias originais, mas são criadas ligações entre os termos de uma ontologia e os da outra, através de mapeamento. Permite-se, assim, a utilização, por uma ontologia, de termos definidos na outra.
- Figura 3.1(d): **Articulação** – Uma ontologia é definida de modo que ela possa realizar o mapeamento entre informações de outras ontologias; normalmente, na articulação não é realizado o mapeamento de todas as ontologias, mas de uma parte das ontologias que seja relevante para determinada aplicação.

3.2 Metodologias e métodos para o desenvolvimento de ontologias

Existe atualmente um conjunto considerável de metodologias e métodos propostos para o desenvolvimento de ontologias. Em Corcho et al. [34], é apresentada uma ampla revisão e comparação das principais metodologias, ferramentas e linguagens para desenvolvimento de ontologias.

A maioria das metodologias analisadas propõe a divisão do desenvolvimento das ontologias em etapas. Outra característica comum é a adoção de métodos para investigação do domínio das ontologias, captura de conhecimento, definição dos componentes das

ontologias e integração de ontologias. A execução desses métodos tanto pode ser realizada manualmente como pode ser auxiliada por ferramentas para automatização de tarefas.

É válido destacar neste ponto a diferença entre os termos *metodologia* e *método*, visto que, na literatura, ambos os termos são usados por vezes de maneira indiscriminada. Contudo, constatou-se que alguns dos objetos a serem identificados e analisados são considerados metodologias, outros são considerados métodos, conforme será visto nas seções a seguir.

Segundo definições do IEEE (*Institute of Electrical and Electronics Engineers*) - 1990 [90] , uma metodologia é "uma série integrada de técnicas ou métodos criando uma teoria geral de sistemas de como uma classe de pensamento pode ser executada". Um método "é um conjunto de processos ou procedimentos ordenados usados na engenharia de um produto ou na realização de um serviço", tendo-se em vista que tais processos são compostos de atividades, que, por sua vez, são compostas por tarefas atribuídas a um ou mais membro do projeto. Geralmente, tarefas relacionadas são agrupadas para formar atividades. Uma técnica é "um procedimento técnico e gerencial usado para alcançar um dado objetivo", o que, em outras palavras, significa um modo pelo qual o método é executado.

Segundo Corcho [33], metodologia e método são conceitos distintos, pois uma metodologia refere-se a conhecimento sobre métodos, isto é, determina "como" e "quando" uma dada atividade pode ser realizada. Assim, uma metodologia é composta de métodos que possuem suas próprias técnicas.

Algumas metodologias aproveitam a robustez da lógica clássica subjacente na abordagem semântica para transformar cenários intuitivos em modelos computáveis. Essas metodologias propõem um conjunto de perguntas dirigidas para a identificação do escopo da ontologia a partir de cenários principais identificados previamente de modo intuitivo.

Abordagens *bottom-up* propõem começar a construção de uma base de conhecimentos para uma aplicação específica e estendê-la posteriormente para uso de novas aplicações em um domínio similar. Em oposição, abordagens *top-down* propõem derivar ontologias de domínio específico a partir de grandes ontologias genéricas. Estas abordagens também podem ser classificadas em relação ao grau de dependência entre a ontologia sendo desenvolvida e sua aplicação final. Sob esta perspectiva, as abordagens são classificadas em dependentes, semi-dependentes e independentes. Metodologias também são classificadas em razão de orientarem a construção de ontologias a partir do zero ou a partir da reutilização de outras ontologias [167].

Em sua tentativa de habilitar o compartilhamento e reuso do conhecimento, Gruber sugere um conjunto de guias para a conversão de um possível sistema monolítico, ou seja, um conjunto de rotinas que interagem entre si, em blocos reutilizáveis [79]. Ele

propõe o uso de três importantes técnicas de decomposição da Inteligência Artificial [79]:

- Separar o conhecimento embutido em programas usando uma linguagem declarativa de representação do conhecimento.
- Identificar classes gerais e relações subjacentes a fatos específicos da aplicação e organizar o conhecimento de modo a habilitar a herança a partir dessa construção.
- Caracterizar tarefas de solução de problemas gerais (ex.: classificação) e classes de inferência e projetar métodos e algoritmos correspondentes.

Geralmente, a prática de construção de ontologias depende fortemente do contexto e dos objetivos de um projeto específico. Por esta razão, as primeiras (historicamente falando) metodologias propostas eram baseadas na experiência obtida enquanto se desenvolvia certo projeto [106].

Ao se construir ontologias, muito pouco existe para ajudar o desenvolvedor na escolha de técnicas e ferramentas apropriadas para suas necessidades. Após a análise do processo de desenvolvimento de um conjunto de ontologias representativas, Uschold [162] criou um *framework* genérico para auxiliar os desenvolvedores a melhor entenderem como escolher as técnicas mais apropriadas para um conjunto de circunstâncias em particular. O *framework* deve identificar passos e técnicas comuns aplicáveis a todos os casos e as condições que requerem passos e técnicas específicas. O *framework* consiste em um conjunto de cinco passos sequenciais, cada passo contendo sub-passos e/ou técnicas para a aplicação dos itens a seguir:

1. Identificação da finalidade da ontologia.
2. Decidir o nível de formalidade (transformar a descrição conceitual em um modelo formal).
3. Identificar o escopo.
4. Construir a ontologia.
5. Avaliar / ciclo de revisão.

O processo de desenvolvimento de uma ontologia não deve ser iniciado antes da identificação clara da finalidade para a qual a ontologia será construída. Este passo inclui atividades como a detecção dos usuários-alvo, a especificação da finalidade relativa à gama de finalidades já identificadas, a elaboração dos cenários de motivação e questões de competência, isto é, questões que a ontologia deve ser capaz de responder para mais esclarecimentos da finalidade e escrita de um documento de requisitos dos usuários. Após a finalização deste passo, informações suficientes devem estar disponíveis para que o

desenvolvedor possa decidir os níveis de formalismo necessários para sua(s) ontologia(s) [106].

Dentro do terceiro passo, o escopo (isto é, o que está e o que não está na ontologia) deve ser identificado e a terminologia (conjunto de termos) – para a informação que a ontologia deve cobrir – precisa ser determinada. Este passo pode ser completado através do uso de uma das seguintes técnicas: cenários motivacionais e questões de competências informais ou *brainstorming* e *trimming* [162]. A determinação da técnica a ser utilizada depende de circunstâncias específicas de cada processo de desenvolvimento de ontologias. O terceiro passo é concluído com um documento contendo os termos e conceitos (estruturados ou não) que a ontologia deve definir.

Não há um consenso quanto a uma metodologia única para o desenvolvimento de ontologias, podendo-se afirmar que as abordagens existentes não são consideradas totalmente maduras em comparação com metodologias da área de engenharia de software ou da engenharia de conhecimento. Constata-se também a ausência de qualquer iniciativa voltada para a unificação das abordagens existentes. Cada grupo define e aplica sua própria abordagem [167].

A seguir, são apresentados alguns projetos, métodos e metodologias existentes que servem como modelo para a construção de ontologias.

3.2.1 Método Cyc

Nos anos de 1980, a *Microelectronics and Computer Technology* (MCC) deu início a criação da Cyc, uma ampla base de conhecimento que considera o conhecimento consensual sobre o mundo, incluindo regras e heurísticas para dedução sobre objetos e eventos do cotidiano [136]. A linguagem de representação da Cyc é a CycL, considerada híbrida por combinar quadros com cálculos de predicado. Tal linguagem possui uma máquina de inferência que permite herança múltipla, classificação automática, manutenção de links inversos, verificação de restrições, busca ordenada, detecção de contradição e módulo de resolução.

A base de conhecimento Cyc foi desenvolvida em 1990 por Douglas Lenat e Ramanathan Guha [136]. Três processos foram considerados em seu desenvolvimento, a saber:

1. extração do conhecimento de senso comum;
2. extração auxiliada por computador;
3. extração gerenciada por computador.

No primeiro processo, o conhecimento requerido para a ontologia foi obtido de forma manual em diferentes fontes, como artigos, livros e jornais. O segundo processo

foi conduzido de maneira automática, isto é, com uso de ferramentas computacionais de processamento de linguagem natural e aprendizado de máquina capazes de usar conhecimento de senso comum suficiente para investigar e descobrir novos conhecimentos. Finalmente, o terceiro processo foi conduzido por um número maior de ferramentas no sentido de gerenciar a extração de conhecimento de senso comum (partes consideradas difíceis de serem interpretadas nas fontes de conhecimento envolvidas) na base Cyc.

3.2.2 Metodologia de Gruninger e Fox

Esta metodologia foi proposta por Michael Gruninger e Mark Fox em 1995 [82], tendo como base para o seu desenvolvimento a experiência obtida no projeto *Toronto Virtual Enterprise* – conhecido como projeto Tove [62], cujos princípios teóricos e metodológicos encontram-se na inteligência artificial.

O objetivo do projeto Tove foi criar um modelo de senso comum sobre empresas, isto é, conhecimento compartilhado sobre o negócio que conduza a deduções de respostas sobre questões acerca do domínio [62]. Para tal, ontologias são criadas no sentido de especificar modelos para organizações públicas e privadas, levando-se em consideração as seguintes características:

- capacidade de fornecer uma terminologia compartilhada para organizações, que possa ser compreendida e utilizada por cada aplicação, isto é, para cada tipo de negócio;
- definição da semântica de cada termo por meio de uma teoria lógica;
- implementação da semântica em um conjunto de axiomas que permita à ontologia deduzir de forma automática respostas às questões comuns no escopo das organizações;
- definição de uma simbologia para representar graficamente termos ou conceitos [83].

A metodologia de Gruninger e Fox foi usada no *Enterprise Integration Laboratory* (Laboratório de Integração de Empresas) da *University of Toronto* (Universidade de Toronto) para o projeto e avaliação de ontologias integradas, incluindo propostas de construção de novas ontologias e extensões de ontologias já existentes. Os seguintes procedimentos foram propostos na metodologia:

1. elaboração de cenários de motivação, que objetivam identificar problemas no ambiente atual;
2. especificação de questões de competência informal, que objetivam especificar em linguagem natural os requisitos que a ontologia deverá ser capaz de atender;

3. concepção da terminologia formal, em que, mediante declarações em lógica de primeira ordem, os conceitos e suas propriedades são organizados em uma taxonomia;

4. especificação de questões de competência formal, em que problemas são definidos de modo consistente perante os axiomas na ontologia;

5. especificação de axiomas formais, que restringem a interpretação dos termos envolvidos nas questões de competência formal;

6. verificação de teoremas completos, que determinam as condições sobre as quais as soluções das questões são completas.

3.2.3 Método de Uschold e King

O método foi proposto inicialmente por Mike Uschold e Martin King em 1995 [164] e estendido em 1996 por Mike Uschold e Michael Gruninger [163] na experiência de desenvolvimento da *Enterprise Ontology*. Tal ontologia foi desenvolvida como parte do projeto Enterprise por meio do Instituto de Aplicações em Inteligência Artificial da Universidade de Edinburgh e parceiros como IBM, Unilever e outros.

Uschold e King [164] consideram os seguintes estágios como sendo necessários a uma metodologia abrangente:

1. identificação do propósito da ontologia, que objetiva identificar a necessidade de construção, o grau de formalismo (desde o informal com uso de linguagem natural até o rigorosamente formal com uso de declarações lógicas) e as classes de usuários da ontologia, incluindo desenvolvedores, mantenedores e usuários das aplicações;

2. construção da ontologia, que se divide em:

   (a) captura ou concepção da conceitualização da ontologia;

   (b) codificação ou implementação através de uma linguagem de representação de ontologias, e

   (c) integração com ontologias já existentes;

3. avaliação da ontologia através dos requisitos especificados;

4. documentação acerca das pretensões da ontologia e das primitivas usadas para expressar as definições na ontologia.

3.2.4 Método Kactus

A ênfase do projeto europeu *Kactus* está na organização de bases de conhecimento que podem ser compartilhadas e reusadas em diferentes sistemas baseados em conhecimento. Para tal, ele utiliza ontologias de domínio para organizar o conhecimento independentemente da aplicação de software que será construída.

Baseando-se no projeto Kactus, Amaya Bernaras e colegas [14] investigaram a viabilidade da reutilização do conhecimento em sistemas de complexidade técnica, como o domínio de redes elétricas, e o papel das ontologias como suporte a tais sistemas. Tal investigação resultou em um método de construção de ontologias, cujos processos envolvidos estariam condicionados ao desenvolvimento da aplicação, ou seja, toda vez que uma aplicação fosse construída, a ontologia, que representa o conhecimento necessário para a aplicação, seria refinada. Tais processos seriam os seguintes:

1. desenvolvimento de uma lista de necessidades ou requisitos que precisam ser atendidos pela aplicação;

2. identificação de termos relevantes para o domínio da aplicação a partir de tais requisitos, construindo, assim, um modelo preliminar;

3. refinar e estruturar a ontologia a fim de obter um modelo definitivo;

4. buscar por ontologias já desenvolvidas por outras aplicações no sentido de sua reutilização. As ontologias reutilizadas demandariam refinamento e extensão para serem usadas na nova aplicação.

3.2.5 Metodologia *Methontology*

A metodologia para construção de ontologias *Methontology* foi desenvolvida no Laboratório de Inteligência Artificial da Universidade Politécnica de Madri entre 1996 e 1997 pelo grupo de pesquisadores Mariano Fernández-López, Asunción Gómez-Pérez, Antônio J. de Vicente e Natalia Juristo [57], [130].

A *Methontology* contempla um conjunto de estágios de desenvolvimento (especificação, conceitualização, formalização, integração, implementação e manutenção), um ciclo de vida baseado em evolução de protótipos [133] e técnicas para realizar as atividades de planejamento, desenvolvimento e suporte. A atividade de planejamento inclui um escalonamento das tarefas e controle sobre as mesmas, no sentido de alcançar a qualidade devida. As atividades de suporte contemplam aquisição de conhecimento, documentação e avaliação, e ocorrem durante todo o ciclo de vida da ontologia.

Os estágios iniciais de desenvolvimento (especificação e conceitualização) implicam um grande esforço dentro das atividades de suporte, como a aquisição de conhecimento e a avaliação. Várias são as razões:

- a maior parte do conhecimento é adquirida no início do processo de construção da ontologia;
- deve-se avaliar corretamente o modelo conceitual para evitar futuros erros no ciclo de vida da ontologia;
- a documentação detalhada deve ser produzida após cada estágio previsto no ciclo de vida.

Para o estágio de conceitualização, Leite et al. [101], [102], fizeram um estudo comparativo das técnicas de Elicitação de Requisitos para o levantamento dos termos da ontologia. Parte desse estudo está ilustrado na Tabela 3.1 [19].

Técnicas	**Entrevistas**	**Leitura Dinâmica de Documentos**	**Questionários**	**Workshop de Requisitos**	**Observação**	**Análise de Protocolo**	**Enfoque Antropológico**	**Base de Requisitos Não Funcionais**
Vantagens	Contato direto com atores, possibilidade de validação imediata	Facilidade de acesso às fontes de informação; volume de informação	Padronização de perguntas; tratamento estatístico	Múltiplas opiniões, criação coletiva	Baixo custo, pouca complexidade da tarefa	Fatos não observáveis, melhor compreensão dos fatos	Visão de dentro para fora contextualizada	Reutilização de conhecimento; antecipação de aspectos implementacionais; identificação de conflitos
Desvantagens	Conhecimento tácito, diferenças culturais	Dispersão das informações; volume de trabalho requerido para identificação dos fatos	Limitação das respostas; pouca interaçao; participação	Dispersao, custo	Dependência do ator (observador); superficialidade decorrente da pouca exposição ao universo de informações	Foco na performance; o que se diz não é o que se faz	Tempo; pouca sistematização	Custo de construção de base; RNF; falsa impressão de completude

Tabela 3.1: *Algumas técnicas de Elicitação de Requisitos [19].*

3.2.6 Método Sensus

A ontologia Sensus foi desenvolvida pelo grupo de linguagem natural do *Information Sciences Institute* - ISI com o propósito de ser usada para fins de processamento de linguagem natural. Esta ontologia foi criada para prover uma estrutura conceitual ampla que pudesse ser utilizada no desenvolvimento de tradutores automáticos [128].

A SENSUS contém tanto conceitos de alto nível quanto específicos, o que justifica a denominação de ontologia de ampla cobertura, possuindo aproximadamente 70 mil conceitos organizados hierarquicamente de acordo com seu nível de abstraçao. No

entanto, esta ontologia foca principalmente nos níveis de abstração alto e médio, deixando a especificação de termos mais concretos - de acordo com o domínio em análise desejado - para as ontologias que serão geradas a partir dela [128].

O método Sensus, baseado na ontologia SENSUS, propõe alguns processos para estabelecer as ligações entre os termos específicos e os termos da ontologia de alto nível [154]. O resultado de tal processo é uma estrutura de uma nova ontologia, que é generalizada automaticamente através de uma ferramenta denominada OntoSaurus [154]. De acordo com o método, os processos envolvidos na construção da ontologia de um domínio específico seriam:

1. identificar termos-chave do domínio;
2. ligar manualmente os termos-chave à ontologia SENSUS;
3. adicionar caminhos até o conceito de hierarquia superior da Sensus;
4. adicionar novos termos para o domínio;
5. adicionar subárvores completas.

3.2.7 Método 101

O método 101 foi concebido por Natalya F. Noy e Deborah L. McGuinness [122] a partir da experiência no desenvolvimento da ontologia de vinhos e alimentos, utilizando o ambiente de edição de ontologias Protégé [88].

O método 101 propõe sete atividades para o desenvolvimento de uma ontologia:

1. Determinar o domínio e o escopo da ontologia;
2. Considerar o reúso de outras ontologias;
3. Enumerar os termos importantes da ontologia;
4. Definir classes e a hierarquia de classes;
5. Definir as propriedades das classes;
6. Definir os valores das propriedades;
7. Criar instâncias.

Tais atividades implicam decisões de modelagem, dentre as quais o método busca enfatizar, além de se encontrarem dentro de um processo iterativo de um ciclo de vida de ontologia.

3.2.8 Método baseado no Léxico Ampliado da Linguagem (LAL)

O grupo de Engenharia de Requisitos da PUC-Rio [19] propôs um processo baseado no Léxico Ampliado da Linguagem (LAL). O processo de construção do Léxico é estruturado e segue princípios sólidos de engenharia de software e técnicas já estabelecidas para captura, modelagem e posterior validação da informação modelada.

O LAL é uma liguagem de representação da Engenharia de Requisitos composta de três entidades básicas: Termo(identificação), Noção(significado do termo) e Impacto(informação sobre o contexto em mãos). Essa linguagem tem por objetivo mapear o vocabulário utilizado no Universo de Informaçao (UdI), ou seja, o contexto no qual o software deverá ser desenvolvido e operado. O UdI inclui todas as fontes de informação e todas as pessoas relacionadas ao software [19].

Os passos deste método de construção de ontologia são:

1. Construção do Léxico(representa todos os conceitos básicos, funções, qualidades e relacionamentos das entidades importantes na própria linguagem do domínio). Dividida em três etapas:
   (a) Levamentamento da Informação;
   (b) Modelagem do LAL;
   (c) Análise do LAL.
2. Mapeamento Léxico - Ontologia. Fase para mapear os termos do tipo:
   (a) Sujeito e Objeto para classes da ontologia.
   (b) Verbo para propriedades.
   (c) Estado para classes ou propriedades.
3. Construção da Hierarquia de Classes. Nesta fase são realizados dois passos:
   (a) Elaboração das listas de classes;
   (b) Construção da hierarquia de classes com as listas elaboradas.

3.3 Análise comparativa das metodologias e dos métodos

Silva et al. [35] fizeram um estudo analítico sobre metodologias e métodos para construção de ontologias mediante análise da literatura sobre metodologias para construção de ontologias e de normas internacionais para construção de software. Através deste estudo, Silva et al. [35] construíram, um panorama comparativo para apoiar a

definição de padrões metodológicos para construção de ontologias. Com base neste estudo e em outras fontes, como [1], [19]e [167], chegou-se a algumas considerações sobre as metodologias e os métodos para construção de ontologias, que estão representadas na Tabela 3.2.

O estudo foi feito com base nas metodologias existentes e já consolidadas, difundidas e bem aceitas na indústria para a área de engenharia de software. Percebe-se certa semelhança entre algumas fases de desenvolvimento de ontologias e fases do processo de desenvolvimento de software. Com base nestas semelhanças, foi criada a Tabela 3.2. Algumas destas semelhanças foram identificadas, principalmente nas atividades de análise de domínio e nas abordagens técnicas para criação de modelos conceituais.

Para verificar a similaridade com a engenharia de software, foi considerado o padrão aceito internacionalmente para desenvolvimento de software, a norma IEEE-1074 [124]. Foram extraídos desta norma, o uso de categorias para procedimentos de análise qualitativa, observando-se assim, características particulares das ontologias (formalização e integração). A seguir, cada categoria, ou seja, cada fase do ciclo de vida é fundamentada de acordo com a norma IEEE-1074 [124] e de acordo com princípios metodológicos para construção de ontologias ([57], [163]):

1. **Gerenciamento do projeto** - ocorrem atividades relacionadas ao início de um projeto, como criação do processo e ciclo de vida, ao planejamento da gestão de um projeto e ao monitoramento e controle do projeto em todo o seu ciclo de vida.

2. **Pré-desenvolvimento** - consiste em analisar idéias ou conceitos de um sistema e, em função de problemas observados no ambiente, alocar requisitos para o sistema antes do início de desenvolvimento do produto. A fase inclui atividades de estudo de viabilidade e análise de requisitos do sistema.

3. **Especificação de requisitos** - abrange as restrições ou regras que o produto deve cumprir em função da definição das necessidades do requisitante. Os requisitos devem servir como documento inicial para a realização das tarefas de modelagem e prototipação, e tal processo é geralmente iterativo.

4. **Modelagem conceitual** - objetiva desenvolver uma representação bem organizada e coerente do sistema que satisfaça os requisitos de produto especificados nas atividades de requisitos.

5. **Formalização** - consiste em transformar o modelo conceitual da ontologia (ou conceitualização) em um modelo formal a fim de definir, de forma precisa, o seu significado. As técnicas empregadas nessa fase são oriundas da área de inteligência artificial, como sistemas de representação baseados em *frames*, redes semânticas, lógica descritiva etc.

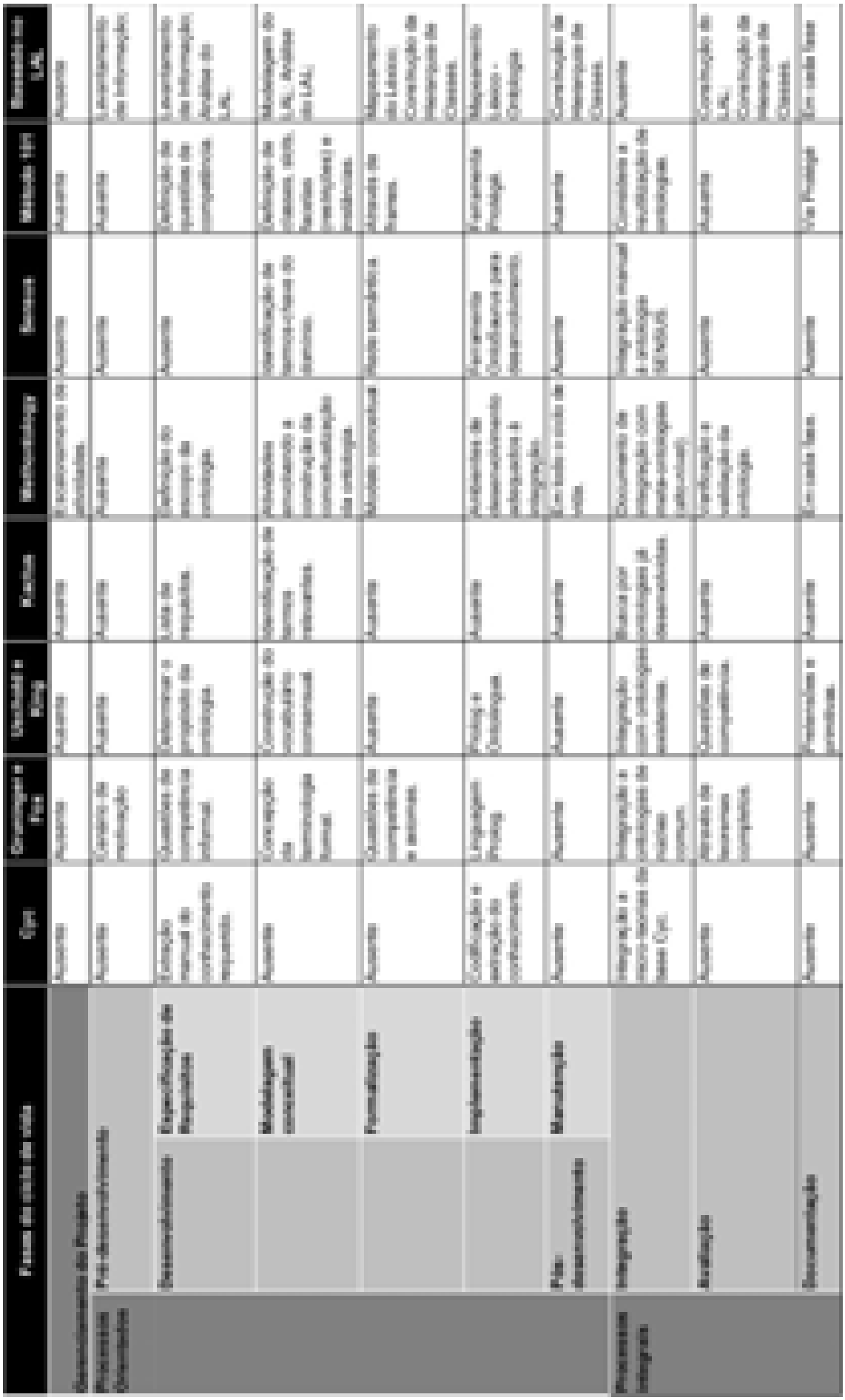

Tabela 3.2: *Tabela comparativa das metodologias e métodos frente às categorias de análise predefinidas [35].*

6. **Implementação** - resulta na transformação da representação do projeto da arquitetura do software em uma linguagem de programação. No caso específico das ontologias, a implementação consiste em mapear o modelo formal em uma linguagem que se adeque às demandas como OWL [4], por exemplo.

7. **Forma de apresentação** - inclui como representar os relacionamentos dos termos, formato de distribuição, tipos de exibição, entre outras.

8. **Manutenção** - considerada uma etapa pós-desenvolvimento, que consiste em identificar problemas e melhorias nos produtos, podendo resultar em novas versões.

9. **Integração** - esta fase considera a reutilização de conceitos existentes em outras ontologias. No processo de integração, as atividades podem ser realizadas durante a fase de modelagem conceitual e implementação da ontologia, sendo considerada, portanto, um processo integral. Ressalta-se que o fato de considerar a busca por ontologias existentes não implica, necessariamente, integração.

10. **Avaliação** - suas atividades são executadas ao mesmo tempo com atividades dos processos orientados ao desenvolvimento do produto, como, por exemplo, condução de revisões e auditorias nos processos, desenvolvimento de procedimento de testes, execução de testes e avaliação de resultados.

11. **Documentação** - desenvolvimento e distribuição de documentação em cada fase para desenvolvedores e usuários envolvidos nos processos, a fim de fornecer, em tempo hábil, informações sobre o produto.

Para cada metodologia e método foi apresentado na Tabela 3.2 um espaço dedicado a cada categoria de análise. Se uma categoria não for pertinente a alguma metodologia ou método, a coluna é preenchida com o valor “Ausente”.

3.4 Apresentação das vantagens e desvantagens dos métodos e das metodologias

A partir das informações dispostas neste Capítulo e visualizadas na Tabela 3.2, chegou-se a algumas considerações sobre as metodologias e os métodos para construção de ontologias analisados (apresentadas na Seção 3.5). A partir dessas considerações, pode-se fazer uma análise comparativa das vantagens e desvantagens dos métodos e metodologias discutidos na Seção 3.2. Esta comparação é ilustrada na Tabela 3.3.

Na comparação, foram considerados os resultados do estudo de Uschold [162], enumerados na Seção 3.2. Esse estudo apresenta uma ajuda ao desenvolvedor para a

		Cyc	Gruninger e Fox	Uschold e King	Kactus	Methontology	Sensus	Método 101	Baseado no LAL
Vantagens	Formalismo		•			•	•	•	•
	Simplicidade			•		•	•		
	Propõe método de extração do conhecimento automática	•							
	Passos bem definidos e objetivos			•		•			
	Quase completa em relação a um ciclo de vida de desenvolvimento.					•			•
	Propõe uma modelagem conceitual e um processo iterativo.							•	•
	Presente no Protégé							•	
	Trabalha com todo o universo da aplicação			•		•			•
	Avalia, refina e garante a estrutura da ontologia		•	•	•				•
Desvantagens	É superficial, segundo Uschold.	•			•		•		
	Ênfase em atividades de desenvolvimento.	•						•	
	Complexo e demorado.		•						•
	Não propõe um planejamento e uma formalização.	•		•					
	Ausência do levantamento inicial do cenário pré-desenvolvimento.	•		•	•	•	•	•	

Tabela 3.3: *Vantagens e desvantagens dos métodos e metodologias.*

escolha de técnicas e ferramentas apropriadas para suas necessidades na construção de ontologias. Também levou-se em consideração a viabilidade conceitual de integração dos diversos padrões e técnicas associados aos temas de engenharia de software e engenharia de ontologias (ver Tabela 3.2). Detalhando a comparação ilustrada na Tabela 3.3:

- As metodologias mostram-se superficiais em relação a detalhes das atividades e dos procedimentos na explicação dos passos para construção de ontologias. É o caso dos métodos Cyc, Kactus e Sensus, os quais parecem considerar que o engenheiro de ontologias já domina o assunto de construção de ontologias e não necessita de detalhes acerca de atividades e procedimentos envolvidos. Já a *Methontology* destaca-se por fornecer, na maioria das vezes, detalhes de como proceder na condução de uma dada atividade e, assim como a metodologia de Uschold e King e o método Baseado no LAL, também destaca-se por trabalhar com todo o universo

da aplicação.

- Os métodos de Gruninger e Fox e Baseado no LAL são complexos e demorados, necessitando um alto nível de conhecimento e experiência por parte do engenheiro de ontologias. Já os métodos de Uschold e King, *Methontology* e Sensus são simples em sua aplicação e entendimento.
- As metodologias *Methontology* e Baseada no LAL são quase completas em relação a um ciclo de vida de desenvolvimento, como apresentado na Tabela 3.2.
- A formalização do processo de desenvolvimento da ontologia, ou seja, a expressão de procedimentos em um determinado formalismo ou linguagem formal, um aspecto importante tratado pela engenharia de software, não é enfatizado nos métodos Cyc, Uschold e King e Kactus.
- Os método Cyc e método 101 desconsideram aspectos importantes relacionados ao gerenciamento do projeto, ao estudo de viabilidade, à manutenção e à avaliação das ontologias produzidas. Eles dão mais ênfase em atividades de desenvolvimento, especialmente à implementação da ontologia.
- O método Cyc e a metodologia de Uschold e King não propõem um planejamento para o processo de construção.
- No método Cyc, na metodologia Uschold e King, no Kactus, na *Methontology*, no Sensus e no Método 101 há ausência do levantamento inicial do cenário pré-desenvolvimento.
- O destaque do método Cyc é a proposta de um método de extração do conhecimento automática.
- As metodologias e métodos de Gruninger e Fox, Uschold e King, Kactus e Baseado no LAL avaliam, refinam e garantem a estrutura da ontologia dando consistência ao artefato.
- O Método 101 destaca-se, para este trabalho, por ser implementado pelo Protégé, ou seja, está nativo no *framework* Protégé como um módulo de "Ajuda" no processo de construção de ontologias. Jutamente com o método Baseado no LAL, destaca-se por propor uma modelagem conceitual e um processo iterativo em toda construção.

3.5 Conclusão

Neste Capítulo, foram apresentadas metodologias e os métodos para construção de ontologias mais representativos na literatura, e também similaridades entre padrões de construção de software e princípios metodológicos empregados na elaboração de ontologias.

Diante do exposto neste Capítulo, percebe-se um cenário relativamente diferente na Engenharia de Ontologia, na qual diversas metodologias têm sido apresentadas e discutidas para construção, reutilização e avaliação de ontologias de software, mas apresentam abordagens e características diversas, sendo direcionadas a diferentes propósitos e aplicações, ou seja, não possuem propostas unificadas, sendo que grupos diferentes utilizam diferentes abordagens.

A partir das informações dispostas na Tabela 3.2, chegou-se a algumas considerações sobre as metodologias e os métodos para construção de ontologias analisados. Tais considerações são enumeradas a seguir [35]:

- Existe uma variedade de estratégias para desenvolvimento de ontologias, comprovando a hipótese de que grupos diferentes apresentam abordagens e características diversas, sendo direcionadas a diferentes propósitos e aplicações [108].
- No contexto das ontologias, algumas abordagens seguem um modelo de ciclo de vida, outras não. Nesse requesito, a que mais se destaca é a *Methontology* por ser praticamente completa em relação a um ciclo de desenvolvimento, não propondo apenas a fase de pré-desenvolvimento. Neste contexto, a construção baseada no LAL destaca-se por apresentar os passos do modelo desordenadamente, fazendo com que sejam repetidos e/ou trocados os processos em vários estágios do seu ciclo.
- Em relação a detalhes das atividades e dos procedimentos para sua condução, algumas metodologias e métodos mostram-se superficiais na elucidação dos passos para construção de ontologias. É o caso dos métodos Cyc, Kactus e Sensus, os quais parecem considerar que o ontologista já domina o assunto sobre construção de ontologias e não necessita de detalhes acerca de atividades e procedimentos envolvidos. Já a *Methontology* destaca-se por fornecer, na maioria das vezes, detalhes de como proceder na condução de uma dada atividade.
- Algumas abordagens dão mais ênfase em atividades de desenvolvimento, especialmente a implementação da ontologia (método Cyc e método 101), desconsiderando aspectos importantes relacionados a gerenciamento do projeto, a estudo de viabilidade, à manutenção e à avaliação de ontologias.

CAPÍTULO 4

Ferramenta para Extração do Conhecimento em Objetos Textuais - SINAPSE

O SINAPSE é um arcabouço computacional direcionado para extração de unidades terminológicas complexas[1], presentes em corpora de domínio de conhecimento específico. Desta maneira, com base em conhecimentos estatísticos e linguísticos é capaz de analisar estruturas textuais e descobrir conhecimentos (conceitos) relevantes implícitos nos mesmos. Como resultado, organiza estes conhecimentos em uma Matriz Atributo x Valor, possibilitando seu pós-processamento [36].

4.1 Uma Visão Gráfica do SINAPSE

O objetivo do SINAPSE é descobrir conhecimentos em objetos textuais. Para a realização desta tarefa, foi implementado no SINAPSE uma técnica de mapeamento por conceitos [36].

Para Wives [170], a idéia fundamental do mapeamento por conceitos consiste em pré-processar um conjunto de documentos, a fim de normalizar seu vocabulário, identificando os conceitos presentes nos documentos e utilizando-os como características representativas dos mesmos. A Figura 4.1 apresenta uma arquitetura geral para o funcionamento do SINAPSE. Nela, pode-se observar que o pré-processador SINAPSE recebe como entrada documentos textuais e uma base de conceitos (conhecimentos). Os documentos textuais serão analisados com base nos conhecimentos disponíveis no ambiente do SINAPSE. Estes conhecimentos são modelados em dois tipos de arquivos. Os conceitos (conhecimentos) estão armazenados em arquivos, em formato OWL. Adicionalmente, existe uma base de regras de reduções gramaticais que é armazenada em arquivos em formato ASCII [105]. Finalmente, como conhecimento de apoio, existem dois arquivos em formato ASCII que contém a *Stoplist*[2] e a lista de preposições [36].

[1] Unidades terminológicas complexas refere-se à quantidade de termos que fazem parte de um conceito. No SINAPSE um conceito tem entre 1 e 5 termos.

[2] Lista de palavras vazias ou dicionário negativo - palavras gramaticais, funcionais de classe fechada

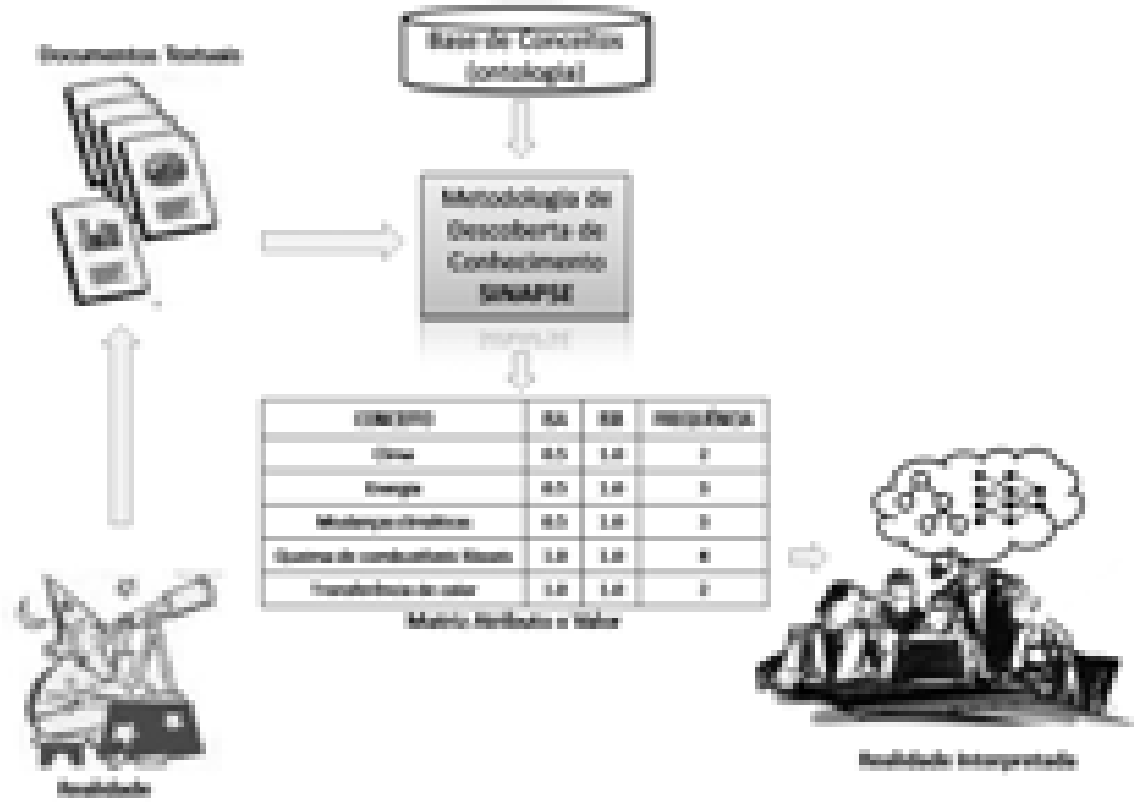

Figura 4.1: *Visão Gráfica de Alto Nível do SINAPSE [36].*

Os resultados da descoberta de conhecimento do SINAPSE são armazenados em um banco de dados numa **Matriz Atributo x Valor**.

A metodologia de descoberta de conhecimento do SINAPSE é estruturada em 7 (sete) etapas como pode ser vista na Figura 4.2, onde se tem:

As três primeiras referem-se à estrutura de planejamento do processo de descoberta de conhecimento, são elas:

- **Definição dos objetivos**: definição do que se deseja obter;
- **Seleção dos textos**: escolha dos textos a serem analisados;
- **Seleção de ontologias**: escolha da ontologia do domínio pretendido.

A etapa de pré-processamento é a parte da metodologia automatizada pelo SINAPSE. Nela, os conhecimentos são descobertos e disponibilizados para as etapas posteriores.

- **Pré-processamento**: descoberta de conhecimento em objetos textuais.

ou instrumentais: são palavras de categoria fechadas (pronome, artigo, preposição, conjunção; algumas classificações incluem aqui verbos auxiliares e modais

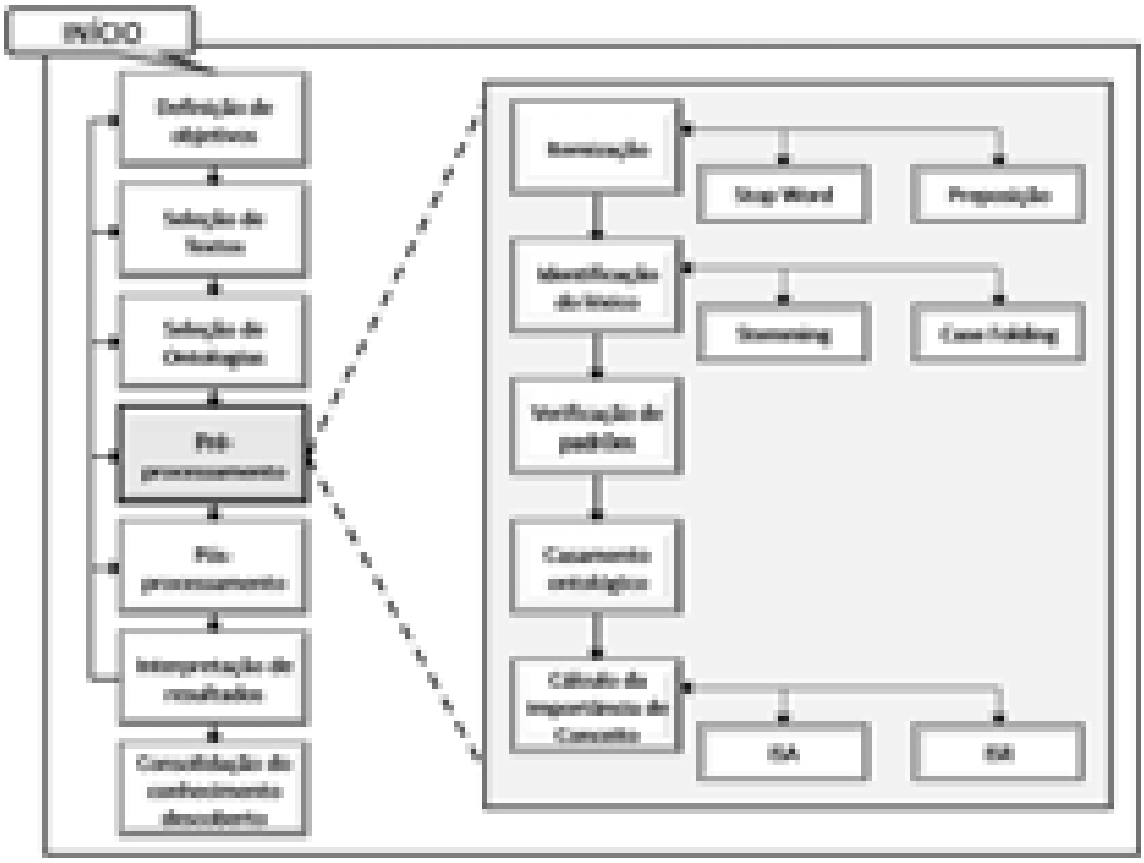

Figura 4.2: *Visão Gráfica da Metodologia SINAPSE [36].*

As três últimas etapas são direcionadas aos objetivos pré-definidos na etapa "Definição dos objetivos". Ela recebe do SINAPSE a **Matriz Atributo x Valor**, já com os conceitos obtidos e respectivos atributos (ISA (Índice de Substantivação Absoluto do conceito), ISR - (Índice de Substantivação Relativo do conceito) e Frequência). Com base nesta matriz as etapas seguintes disponibilizam os resultados esperados.

- **Pós-processamento**: agrupamento, indexação, visualização, resumos, etc.
- **Interpretação de resultados**: verificação de utilidade;
- **Consolidação de conhecimento descoberto**: alteração do estado de conhecimento.

Neste ponto, deve-se observar que o SINAPSE atua como uma infra-estrutura que executa nos bastidores (não é utilizado como uma aplicação *front-end*) da descoberta de conhecimento em texto.

4.2 Domínio de Conhecimento

Wives [170] sugere, que para existir compreensão, análise e processamento de textos, é necessário que se tenha algum conhecimento de apoio (utilização de conheci-

mento através de alguma estrutura, como por exemplo, ontologia) referenciado como *BK* (*Background Knowledge* ou Conhecimento de Domínio ou Conhecimento Semântico). Afirma ainda, que tais conhecimentos podem ser expressos através de regras de produção, regras gramaticais, dicionários morfológicos, redes semânticas ou, até mesmo, ontologias.

No SINAPSE é implementado o conhecimento de apoio (*BK*) através de regras gramaticais do idioma Português do Brasil e de ontologias escritas em formato OWL. O conjunto de regras compreende os sufixos utilizados na formação das palavras. As ontologias definem o conhecimento de mundo no domínio investigado [36].

Para fins organizacionais, pode-se definir uma taxonomia como a organização peculiar de um conjunto de informações para um propósito particular. As taxonomias são implementadas a partir de uma lista de conceitos e categorias, segundo os quais se organizam hierarquicamente os conteúdos (informações e documentos) de interesse [36].

A metodologia de descoberta de conhecimento do SINAPSE requer o uso de textos e ontologias de domínios diferenciados para observação e validação das técnicas empregadas.

4.3 Língua Portuguesa

O SINAPSE é direcionado para o idioma Português do Brasil.

O SINAPSE adotou o uso de conhecimentos linguísticos e estatísticos para a descoberta de conhecimentos em objetos textuais. Assim, quando se utiliza conhecimentos linguísticos, cria-se a necessidade de fixação em idiomas específicos.

4.4 Seleção de Termos mais Importantes

O SINAPSE, no decorrer de seu processo de descoberta de conhecimentos em texto, pode encontrar um conjunto muito grande de conceitos (ou conhecimentos). A quantidade destes conceitos é livre, assim como, o tamanho do texto é livre. Desta forma, a quantidade de conceitos interessantes pode ser proporcionalmente grande. Esta flexibilidade permite analisar textos constituídos de uma única palavra até uma série de livros, onde, a limitação está no ambiente computacional e não no arcabouço [36].

Para que se tenha esta liberdade no tamanho do texto e o aspecto genérico (utilização dos resultados do SINAPSE, após a fase de pós-processamento) do arcabouço, se faz necessária a definição de um limiar. Este limiar representa a quantidade máxima de conceitos selecionados, os quais devem ser os mais importantes dentro do conjunto selecionado, mas, por outro lado, com a máxima qualidade representacional [36].

4.5 Processo de Descoberta de Conhecimento

A descoberta de conhecimento, no modelo implementado pelo SINAPSE, baseia-se na estratégia de realização de rastreamento antecipado dos conhecimentos oriundos de informações não estruturadas no formato textual, baseado em um modelo contextual organizando-os em uma **Matriz Atributo x Valor** [36].

Este modelo, fornece condições para que um mecanismo de busca supere os modelos atuais de mecanismos de buscas. A maioria das organizações apenas exploram as informações estruturadas. Com a Metodologia SINAPSE, a **Matriz Atributo x Valor** torna possível buscas realizando análises em níveis mais elaborados (como por exemplo, o nível semântico) [36].

No ambiente organizacional, encontrar as informações rapidamente quando se precisa delas é em si um diferencial competitivo. Estas informações incluem aplicações internas, sítios da Internet, intranets, extranets, e–mails arquivados, diretórios compartilhados em servidores e desktops, aplicações de GED, CRM, ERP em documentos de qualquer formato [36].

4.6 Textos de Qualquer Tamanho

Para prover facilidades de análises de textos de qualquer tamanho, o SINAPSE procura minimizar o impacto relacionado ao tempo de processamento com as seguintes estratégias [36]:

- Percorrer a lista de léxicos o menor número possível de vezes;
- Não retroceder na lista de léxicos durante a análise.

Assim, o tamanho do texto gera pequenos impactos no tempo de processamento e possibilita a busca por conceitos nestes.

4.7 Metodologia SINAPSE

O SINAPSE adota uma metodologia própria para a descoberta de conhecimentos em textos. Ela é modelada dentro de processos, na etapa de pré-processamento, como forma de preparação do corpus sob análise. Assim, o objetivo metodológico é, a partir da palavra chegar ao conceito, ou seja, ao conhecimento. Este conhecimento deverá ser disponibilizado em uma **Matriz Atributo x Valor** para ser pós-processada por outra aplicação [36].

Esta matriz é uma forma de intermediação para acesso aos conhecimentos descobertos. Neste modelo metodológico aplica-se a verificação em um modelo ontológico

(casamento ontológico) de forma a minimizar as reduções semânticas originárias do processo. Para tanto, esta metodologia executa diversos processos como se pode ver a seguir [36]:

Definição dos objetivos: este é o passo inicial de qualquer processo de descoberta de conhecimento. Nele, o usuário deve definir exatamente o que deseja obter, ou seja, os seus objetivos.

Seleção dos textos: o processamento necessita de informações relacionadas ao contexto de uso pretendido e que sejam relevantes. Para o SINAPSE o tamanho do corpus não é importante. O que deve ser observado é que o tempo para obtenção da resposta é diretamente proporcional ao tamanho do texto.

Seleção das ontologias: o contexto de uso no SINAPSE é definido pela base ontológica. Assim, para cada contexto deve-se ter pelo menos uma ontologia que o represente adequadamente. Uma ontologia é adequada se representar adequadamente o contexto de uso, ou seja, possuir um conjunto de conceitos representativos, preferencialmente, ontologias enriquecidas com variações terminológicas.

Pré-processamento: geralmente, todo processo de descoberta de conhecimento em texto é precedido de uma fase de pré-processamento. Para efeito desta dissertação, foram utilizados somentes os três primeros processos, descritos nos itens a seguir, desta fase de pré-processamento. O SINAPSE, propriamente dito, atua nesta etapa visando descobrir conhecimentos e organizá-los em uma **Matriz Atributo x Valor**, para tanto, executa o seguinte pré-processamento:

1. O **primeiro processo** executado pelo SINAPSE, na busca por conhecimento em texto, é a itemização ou tokenização, onde procura caracterizar o *token* quanto ao padrão linguístico e atributo que se enquadra (uma palavra qualquer, palavra especial - como por exemplo, fórmula matemática, nome próprio, sigla, e-mail e URL). Outra função executada neste processo é a eliminação de todos os caracteres especiais (como por exemplo: "[,.:;!?)]+") do início e do término de cada token. O *token* é a primeira forma identificada pelo SINAPSE e necessita da execução de um sub-processo de limpeza. Após este processo as palavras são armazenadas para serem processadas posteriormente.

2. O **segundo processo**, através do tratamento dos tokens, é a identificação dos itens lexicais com suas respectivas classes gramaticais restritas[3], e, quando for o caso, marcação de atributos lexicais[4] e suas frequências de ocorrências. Este processo utiliza-se de um sub-processo de redução gramatical para obter as classes

[3]Refere-se às classes: Adjetivo, Advérbio, Preposição, Substantivo, Verbo, "I" e "X". Aqui, as classes Substantivo e Adjetivo serão tratadas de forma agrupada.

[4]Refere-se aos atributos especiais (fórmula matemática, etc.), sigla, nome próprio, e-mail e url.

gramaticais restritas. Após esta fase, as palavras e atributos são armazenadas para serem processadas posteriormente.

3. No **terceiro processo**, através do tratamento dos itens lexicais, realiza-se a verificação de padrões linguísticos, percorrendo, o conjunto de itens lexicais para a montagem da lista de candidatos a conceitos. O SINAPSE implementa a verificação de padrões linguísticos para prováveis conceitos (candidatos), com no máximo 5 itens lexicais (termos). Nesta metodologia, o candidato a conceito com mais termos tem maior prioridade. Assim, as verificações de padrões analisam primeiro os padrões de 5 termos, em seguida os de 4 termos, e assim, sucessivamente até os de 1 termo. A cada análise, passa-se a verificar os próximos candidatos. Esta técnica otimiza o processo, pois, não verifica um candidato a conceito mais de uma vez.

   Para cada candidato que esteja dentro dos padrões linguísticos especificados, o SINAPSE faz uma busca nas ontologias em OWL para a realização do casamento semântico. Este casamento é feito em dois níveis. No primeiro nível, verifica se o candidato a conceito tem alguma "classe" (conceito) equivalente dentre as "classes OWL" da ontologia selecionada. Após a verificação de padrões, realiza-se o casamento ontológico para consolidação dos conceitos descobertos.

4. O **quarto processo**, através do tratamento dos candidatos realiza-se o casamento ontológico, que é feito através de mapeamento por conceito. Este casamento é realizado sobre todos os termos candidatos a conceitos, ou seja, termos que estejam dentro de padrões linguísticos e é feita uma verificação na base ontológica se este candidato é "similar" ou igual a um identificador ou variação terminológica.

5. O quinto processo, através do cálculo de importância de conceito ocorre a última etapa do pré-processamento realizado pelo SINAPSE. Ela se subdivide em:

   (a) Cálculo do **ISA**: o *ISA* ou Índice de Substantivação Absoluto do conceito é a execução do cálculo relacionado à importância do conceito em si. Este cálculo é função da quantidade de substantivos presentes neste.

   (b) Cálculo do **ISR**: o *ISR* ou Índice de Substantivação Relativo do conceito é a execução do cálculo da importância em relação ao texto.

   O uso conjunto destas duas medidas possibilita dois aspectos: semântico e o estatístico. O aspecto semântico está relacionado ao fato de se levar em consideração os substantivos presentes no conceito. Já o aspecto estatístico está relacionado ao fato de se levar em consideração a quantidade de vezes que um termo e seus componentes (classe substantivo e indefinido I) aparecem no texto. Após este pré-processamento a **Matriz Atributo x Valor** já está completa, podendo assim, passar

a outras aplicações de Descoberta de Conhecimento em Texto (ou KDT - *Knowledge Discovery from Text*) para pós-processamento.

Pós-processamento: De acordo com os objetivos para a descoberta de conhecimento, o usuário deve selecionar o tipo adequado de pós-processamento. A **Matriz Atributo x Valor** é indicada a diversos tipos (descoberta tradicional após extração, por análise linguística, por análise de conteúdo, por sumarização entre outras), quais sejam, agrupamento de documentos de forma a organizar acessos a estes através de mineração de texto, indexação em sistemas de recuperação de informação, visualização de informações de forma a criar novos mecanismos HCI (*Human-Computer Interaction* ou Interação Homem-Computador), resumos de documentos através da busca ontológica da propriedade "*rdfs:comment*", etc [36].

Interpretação dos resultados: O usuário deve interpretar os resultados apresentados identificando a sua utilidade. Durante esta etapa o usuário pode retornar a qualquer uma das etapas anteriores de forma a obter resultados esperados [36].

Consolidação dos conhecimentos descobertos: Quando da obtenção de resultados úteis o usuário pode utilizá-los para alterar o seu estado de conhecimento [36].

CAPÍTULO 5

Metodologia para Construção Semiautomatizada de Ontologias

Neste Capítulo, é detalhada a metodologia para a construção semiautomatizada de ontologias elaborada nesta dissertação.

O processo de construção de ontologias é complexo e demorado. Atualmente, ontologias são desenvolvidas segundo padrões e métodos da Engenharia de Ontologia, conforme apresentado na Seção 3.2. Não existe uma solução universal.

Segundo López [107], as metodologias de construção de ontologias, apresentadas na literatura, não oferecem grande detalhamento sobre como realizar as atividades iniciais do projeto da ontologia como: criação do processo e ciclo de vida, planejamento da gestão de um projeto e monitoramento e controle do projeto. Assim, justifica-se a necessidade de procedimentos e métodos que orientem e estabeleçam passos bem definidos para a construção de ontologias.

Em muitos casos, técnicas como *brainstorming* (dinâmica de grupo) e entrevistas são sugeridas. Porém, tais técnicas não oferecem aparatos que privilegiem a agilidade e que estabeleçam métricas e regras que possibilitem formalizar e avaliar sua reprodução. A falta de formalismo e agilidade torna difícil a análise dos resultados, estimativa de custos, avaliação da qualidade do processo, comparação entre diferentes execuções nas tarefas da construção e rapidez em todo o processo.

Neste trabalho, é proposta uma metodologia, a partir de algumas metodologias já existentes, como a metodologia proposta por Uschold e King [164], *Methontology* [19] e o método 101 [122]. A ferramenta de extração do conhecimento Sinapse, apresentada no Capítulo 4, foi utilizada na fase de conceitualização (ver Seção 6.2.3). O Sinapse foi utilizado com o objetivo de minimizar a intervenção humana e automatizar tarefas. Essa ferramenta foi utilizada na análise de um *corpus* constituído de documentos de tamanhos variados em língua portuguesa.

Um dos aspectos principais desta nova metodologia é automatizar, mais precisamente, a extração da informação expressa nos documentos textuais do *corpus*. A partir desta extração, formaliza-se os conceitos obtidos semanticamente, aplicando-os na cons-

trução da ontologia.

A metodologia proposta, apresenta uma sequência de passos simples, iterativos e ordenados. Através desses passos, ela visa a produtividade e a melhoria da qualidade no processo de construção de ontologias. A estrutura dos passos propostos orientam o engenheiro de ontologias de maneira objetiva e agrega um maior grau de formalismo em todo o processo de construção.

Para a avaliação da metodologia proposta, foi construída uma ontologia para o domínio das grades computacionais, seguindo os passos propostos. Esta ontologia será descrita no Capítulo 6.

5.1 Metodologia Proposta

Segundo Gasevic et al. [65], para o desenvolvimento de ontologias é necessário um esforço considerável de engenharia, disciplina e rigor; onde princípios de projeto, atividades e processos de desenvolvimento, tecnologias de suporte e metodologias sistêmicas devem ser empregados. Neste sentido, surge a Engenharia de Ontologias preocupando-se com o conjunto de atividades, o processo de desenvolvimento de ontologias, o ciclo de vida de ontologias, os métodos e metodologias para desenvolver ontologias e as ferramentas e linguagens de suporte à construção de ontologias [77].

Segundo Cantele et al. [25], o desenvolvimento e uso de metodologias para construção de ontologias é fundamental, na medida que retiram o caráter subjetivo desta atividade.

Como apresentado na Seção 3.2, não há um consenso da comunidade científica quanto a uma metodologia única ou melhor para o desenvolvimento de ontologias. As metodologias para desenvolvimento de ontologias apresentam ciclos de vida distintos.

Um dos objetivos da metodologia proposta é combinar a Engenharia de Software com a Engenharia de Ontologias para reduzir o tempo de construção de ontologias, melhorar a qualidade e efetividade da metodologia. Para isto, são propostos procedimentos de fácil entendimento e formais, isto é, procedimentos genuínos, claros e positivos.

Para López [107], o que diferencia uma área de estudo na fase de “infância” de outra em fase “adulta” é a aceitação e uso de suas metodologias. A Engenharia de Software pode ser considerada como estando na fase adulta, pois suas metodologias são largamente difundidas, utilizadas e formalizadas. Quanto à Engenharia de Ontologias, antes que as metodologias para o desenvolvimento de ontologias possam ser consideradas maduras e ser aplicadas com êxito, é importante fazer um estudo aprofundado dos problemas a serem resolvidos e analisar quais das metodologias auxiliarão na solução desses problemas.

Para Fernandez-López e Gómez-Pérez [58] em cada metodologia proposta existem atividades que deixam de estar compreendidas. Por isso, segundo os autores e para Sure e Studer [153] e Brusa et al. [23], uma combinação de metodologias se torna pertinente no processo de desenvolvimento de ontologias. Partindo-se deste princípio, o processo proposto combina as melhores práticas das metodologias de Uschold e King [164], da *Methontology* [19] e do método 101 [122].

A partir dos objetivos deste trabalho apresentados na Seção 1.3 e da comparação entre os métodos e metodologias apresentados na Seção 3.4, concluiu-se que a metodologia de Uschold e King [164], a *Methontology* [19] e o método 101 [122] foram as que mais se destacaram diante de um ciclo de vida de construção de uma ontologia e dos aspectos considerados nos estudos de Uschold [162]. Observou-se também, que as três possuem aspectos comuns, que podem ser visualizados na Tabela 5.1. Assim, foram identificados e considerados os aspectos positivos do três métodos para compor a metodologia proposta.

Metodologia Proposta	**Uschold e King**	***Methontology***	**101**
1-Identificar o propósito	Sim	Sim	Sim
2-Identificar o domínio e o escopo	Sim	Sim	Sim
3-Estudar o ambiente	Não	Sim	Não
4-Planejamento	Não	Sim	Não
5-Conceitualização	Não	Sim	Não
5(a)-Capturar os termos importantes	Sim	Sim	Sim
5(a)-Automatizar o processo de captura	Não	Não	Não
5(b)-Enumerar e organizar os termos	Não	Não	Não
6-Formalizar os termos	Não	Sim	Não
7-Implementação	Sim	Sim	Sim
8-Avaliação	Sim	Sim	Sim
9-Documentação	Sim	Sim	Sim

Tabela 5.1: *Comparação da proposta com as metodologias utilizadas para sua composição.*

Portanto, o processo de construção de ontologias proposto neste trabalho é composto dos passos ilustrados na Figura 5.1 e descritos a seguir:

1. **Identificação do propósito**: definir por que construir a ontologia e para que ela será utilizada;

2. **Identificação do domínio e o escopo**: determinar que domínio se deseja cobrir com a ontologia e seu escopo. Para isso é necessário elaborar algumas questões de competência, para as quais a ontologia deve fornecer respostas;

3. **Estudo do ambiente**: analisar o ambiente do domínio que será coberto pela ontologia vendo-o como uma fonte de conhecimento para a etapa de conceitualização;

Figura 5.1: *Metodologia proposta*

4. **Planejar as tarefas a serem executadas no processo de construção**: ao iniciar o projeto de uma ontologia, é necessário planejar as tarefas que serão realizadas. Nesse plano, é necessário descrever recursos e ferramentas que serão necessários;

5. **Capturar os conceitos e relacionamentos textuais**: conceitualização. Nesta fase, é realizado o levantamento dos termos da ontologia, identificando conceitos e relações a representar na ontologia. Para esse levantamento, podem ser utilizadas técnicas tradicionalmente aplicadas pela engenharia de requisitos no processo de elicitação de informação. Algumas dessas técnicas estão listadas na Tabela 3.1. Porém, como o objetivo desta proposta é a agilidade e independência de pessoas, a técnica mais viável para alcançar esse objetivo é a leitura dos documentos para extração do conhecimento neles contido. Para viabilizar a aplicação desta técnica é necessário o auxílio de ferramentas na extração do conhecimento em documentos textuais, como a Sinapse, semiautomatizando esta etapa.

   Este passo pode ser dividido em:

   (a) **Capturar os termos importantes**: fazer uso da ferramenta de extração Sinapse para a captura dos termos, comumente utilizados pelos especialistas do domínio, em documentos textuais. É importante fazer uma lista de termos inicial sem a preocupação com redundâncias ou detalhar exaustivamente seus relacionamentos;

(b) **Enumerar e organizar os termos importantes**: com os termos capturados, enumerá-los e organizá-los de forma a definir textualmente termos-chaves, conceitos e relacionamentos. Com a lista de termos disponível, é possível classificar os elementos de acordo com a compreensão que se tem do domínio. Neste sentido, termos são classificados como classe, relação, propriedade de dados, instância e restrição. Para encontrar os termos-chaves é necessário estabelecer um critério de identificação e numeração do termos capturados pelo Sinapse. O critério indicado é o de ordernar os termos em ordem alfabética, agrupar por radical de mesma base, somar a frequência de todas as palavras com mesmo radical e estabelecer um limite de frequência para cada agrupamento, levando em consideração a classe gramatical de cada palavra.

Para estabelecer o limite mínimo de frequência de cada classe analisada, retira-se um relatório com o total de frequência das classes gramaticais tratadas pelo Sinapse e total geral de todas as palavras do corpus. A partir dos dados obtidos pelo relatório, estabelece-se o limite mínimo analisando o total de palavras, a quantidade de documentos analisados e quantidade de frequências dos termos e calculando a porcentagem mínima de importância dos termos nos documentos.

6. **Formalizar os termos**: nesta estapa, formaliza-se o modelo conceitual da fase anterior através de uma linguagem para descrição de ontologias. Nesta fase, conceitos são definidos através de axiomas que restringem as possíveis interpretações de seu significado e também organizados hierarquicamente através de relações estruturais, tais como “é-um” ou “parte-de”. Lógica de descrição, rede semântica ou modelos baseados em *Frames* são algumas das opções;

7. **Implementação**: codificação da ontologia em uma linguagem de forma a tornar a ontologia passível de processamento automático, ou seja, computável. Implementar a ontologia em uma linguagem como OIL, DAML+OIL e OWL são alguns exemplos;

8. **Avaliação**: antes de disponibilizar a ontologia é necessário avaliá-la, para garantir qualidade e adequação aos padrões (apresentados no Capítulo 2). Os resultados obtidos necessitam de validação por parte de especialistas e ferramentas de inferência para verificar a consistência da ontologia. Se necessário, após a avaliação as etapas de implementação e avaliação podem ser refeitas;

9. **Documentação**: serve para garantir a evolução da ontologia. Da mesma forma que qualquer outro artefato produzido durante o desenvolvimento de software, a onto-

logia precisa ser documentada de forma adequada para permitir que modificações e reuso sejam possíveis no futuro.

A Tabela 5.1 apresenta uma comparação entre a metodologia proposta e as metodologias selecionadas para compor a mesma.

Conforme mostrado na Tabela 5.1, os passos 1, 2, 7, 8 e 9 são comuns às metodologias de Uschold e King, *Methontology* e ao Método 101, porém utilizados em ordem diferente em cada uma delas. Os passos 3, 4 e 6 são tratados somente pela *Methontology*. O passo 5 é comum nas três metodologias selecionadas, embora elas utilizem somente do processo (a), ou seja, o de capturar os termos e depois listá-los. Nenhuma propõe uma automação em alguma etapa. O Método 101 utiliza a etapa de enumeração dos termos como etapa de captura, e a metodologia proposta nesta dissertação apresenta o passo 5(b) como uma sequência do processo de captura semiautomatizado, ou seja, após a extração dos termos com uso da ferramenta Sinapse.

5.2 Conclusão

Este Capítulo apresentou a proposta de uma metodologia que descreve um processo simples, iterativo e ordenado de passos. A Engenharia de Software foi utilizada como um modelo de referência, pois tem um processo de construção de software formalizado e bastante difundido.

A metodologia proposta adota as boas práticas de três metodologias existentes. As metodologias de Uschold e King [164], *Methontology* [19] e do Método 101 [122] foram utilizadas como base nas etapas da proposta, pois apresentaram vantagens significativas diante do objetivo desejado.

A proposta desta metodologia busca incentivar as pessoas a iniciar o processo de construção de ontologias investigando, cuidadosamente, o domínio escolhido. Este cuidado é necessário, pois é nesta etapa, a de estudo do ambiente, que obter-se-á a matéria-prima (documentos textuais) para a construção da ontologia. Com isso, essa proposta motiva um não-especialista no domínio a construir uma ontologia. É válido ressaltar que o processo prospoto neste capítulo é genério, ou seja, independe do domínio.

Na etapa de captura de ontologias, encontra-se a proposta de semiautomatizar o processo de construção com o uso de ferramentas para o processo de extração de termos, como a Sinapse. Esta tarefa é definida como sendo uma tarefa chave para o processo de elaboração de uma ontologia, embora não seja a única [25]. Geralmente esta tarefa é feita por um especialista, mas com o uso de ferramentas automáticas nessa etapa, ele pode ser dispensável.

CAPÍTULO 6

Estudo de Caso - Construção da Ontologia

Como forma de validação da metodologia proposta, foi desenvolvida uma ontologia para o domínio de "Grades Computacionais". Essa ontologia armazena informações sobre os recursos (softwares, hardware e dados) contidos em uma grade. Consultas nesse domínio devem ser semanticamente especificadas com base no vocabulário definido pela ontologia. Desta forma, são eliminadas possíveis interpretações ambíguas na busca e na leitura de informações a respeito do ambiente, pois todos os conceitos usados pelas aplicações têm apenas um significado, tanto para os clientes quanto para os fornecedores das informações.

Neste capítulo, serão apresentados os ambientes de grade computacional, grade de dados e grade semântica na seção 6.1, e na seção 6.2 será mostrado como foi construída a ontologia proposta.

6.1 Grades Computacionais

A necessidade de alto poder de processamento computacional já é comum há algumas décadas em várias áreas do conhecimento como Química, Engenharia, Finanças entre outras. Isso se deve ao avanço científico e tecnológico da humanidade, que tornou o computador uma das principais ferramentas para a produção de inovações. Cada avanço obtido com a ajuda da computação leva à necessidade de um poder computacional ainda maior para poder atingir o próximo nível de conhecimento [97].

Pensando em solucionar o problema de se obter alto poder computacional, os pesquisadores procuraram um novo paradigma que possibilitasse a construção de computadores de alto desempenho acessíveis: os aglomerados ou *clusters*. Um aglomerado (*cluster*) não é nada mais que um conjunto de computadores simples - normalmente PCs - interconectados por uma rede de alta velocidade [7].

A partir do início da década de 1990, houve a criação dos primeiros aglomerados de computadores. A crescente utilização de software livre tornou esses aglomerados de alto desempenho disponíveis a centenas de instituições e milhares de pesquisadores em todo o mundo. Nessa mesma década, a Internet começou a disponibilizar conexões está-

veis de vários megabits por segundo. Assim, veio a idéia de utilizar as linhas de comunicação de grande largura de banda para interconectar aglomerados e supercomputadores localizados em diferentes instituições, em pontos geograficamente distantes [97].

Os sistemas de computação de alto desempenho, distribuídos em redes de escala mundial, tornaram-se uma maneira inovadora de compartilhar recursos computacionais entre várias instituições, de forma a maximizar o uso dos recursos e permitir a construção de uma coleção de máquinas ainda mais poderosas [97].

6.1.1 Grade Computacional

A coleção de aglomerados e supercomputadores geograficamente distantes passou a ser chamada de Grade Computacional (*Computational Grid*) [[60], [61], [158]] em analogia à rede elétrica (*Power Grid*). A grade reflete o objetivo de tornar o uso de recursos computacionais distribuídos tão simples e tão abrangente quanto ligar um aparelho na rede elétrica [97].

Um sistema de computação em grade pode ser definido de maneira bem abrangente como uma infra-estrutura de software capaz de interligar e gerenciar diversos recursos computacionais. Esses recursos, possivelmente, estarão distribuídos por uma grande área geográfica, de maneira a oferecer ao usuário acesso transparente aos mesmos, independentemente de suas localizações. Na maioria dos casos, o principal recurso das grades é a capacidade de processamento, embora alguns sistemas incluam uma ampla gama de recursos, como: dispositivos de armazenamento de grande capacidade, instrumentos científicos e até componentes de software, como bancos de dados [72].

A infra-estrutura de uma grade computacional é organizada principalmente para atender requisitos gerados pela demanda por execução de aplicações e do armazenamento e recuperação dos dados produzidos e consumidos pelas mesmas. De acordo com Roure [47], esses requisitos têm sua origem principalmente no domínio de aplicações complexas como, por exemplo, aplicações científicas (*e-science*), mas continuam válidos no contexto de aplicações para grades de um modo geral.

Os requisitos computacionais das aplicações incluem, entre outros: ciclo de processador, espaço de memória, espaço em disco, políticas de segurança, plataformas, sistemas operacionais, bibliotecas, tamanho dos arquivos de E/S, tempo necessário para transferência de arquivo de entrada, conteúdo de arquivos conforme exigido pelas aplicações, etc.

Apesar da falta de consenso sobre as características de um sistema de Computação em Grade, pode-se listar uma série de aspectos comuns [72]:

- não substituem sistemas operacionais,
- podem integrar recursos distribuídos e descentralizados,

- podem incluir várias plataformas de hardware e software,
- permitem adaptabilidade às políticas locais,
- podem ser utilizados por diversas aplicações.

Principais serviços de uma grade computacional

Sistemas de grades computacionais são hoje em dia implementados como uma camada de *middleware*[1] que executa sobre sistemas operacionais convencionais. Esse *middleware* é responsável por esconder os detalhes e particularidades dos diferentes sistemas operacionais sobre os quais ele opera e também por oferecer serviços de alto nível para a execução de aplicações do usuário. A Figura 6.1 apresenta uma arquitetura genérica implementada pela maioria dos sistemas de grades da atualidade [41].

Figura 6.1: *Arquitetura genérica de uma grade computacional [97]*

O **Agente de Acesso** é o principal ponto de acesso para usuários em sua interação com a grade. Ele é executado em cada nó a partir do qual as aplicações serão submetidas para execução na grade. Além de possibilitar a execução de aplicações, ele deve permitir ao usuário a especificação de requisitos e parâmetros da execução. Deve permitir também que o usuário monitore a execução de suas aplicações e deve auxiliar na coleta dos

[1] *Middleware* é um software que reside entre o sistema operacional e a aplicação a fim de facilitar o desenvolvimento de aplicações, escondendo do programador diferenças entre plataformas de hardware, sistemas operacionais, bibliotecas de comunicação, protocolos de comunicação, formatação de dados, linguagens de programação e modelos de programação[48]

resultados gerados pelas aplicações. Muitos sistemas de grade implementam uma versão Web do agente de acesso, permitindo que o usuário interaja com a grade usando um navegador Web qualquer.

O **Serviço Local de Oferta de Recursos** é executado nas máquinas que exportam seus recursos para a grade e é responsável por executar aplicações nos nós da grade. Para tanto, ele precisa obter o código executável da aplicação, iniciar sua execução (normalmente em um novo processo), coletar e apresentar eventuais erros que possam ocorrer durante ou antes da execução da aplicação e devolver os resultados da execução da aplicação para o usuário que a solicitou. Este serviço é também responsável por gerenciar o uso local dos recursos de um determinado nó e por prover, para a grade, informações sobre a disponibilidade de recursos neste nó.

O **Serviço Global de Gerenciamento de Recursos** é responsável por monitorar o estado dos recursos compartilhados em toda a grade e por responder a solicitações de utilização desses recursos. Para tanto, ele auxilia o Serviço de Escalonamento a efetuar o casamento das requisições por recursos solicitadas pelos usuários com as ofertas de recursos. Ele mantém informações atualizadas dinamicamente sobre quais aglomerados e quais nós estão com recursos disponíveis e dados sobre a utilização destes recursos como, por exemplo, utilização do processador, quantidade de memória RAM disponível, quantidade de espaço em disco disponível, etc. Por esse motivo, ele pode também ser chamado de **Serviço de Informações sobre Recursos**. Em alguns casos, esse serviço pode também ser responsável por detectar falhas nos nós da grade e notificar o agente de acesso sobre erros na execução de aplicações presentes nos nós com problemas.

O **Serviço de Escalonamento** determina, para cada aplicação, onde e quando ela irá ser executada. Ele recebe solicitações de execução de aplicações, obtém informações sobre a disponibilidade de recursos consultando o Serviço Global de Gerenciamento de Recursos e determina a ordem e o local de execução das aplicações.

Finalmente, o **Serviço de Segurança** é responsável por três tarefas principais:

1. proteger os recursos compartilhados, de forma que um nó que exporta seus recursos para a grade não sofra ataques executados por aplicações maliciosas;

2. autenticação dos usuários, de forma que se saiba quem é responsável por cada aplicação e pela sua execução, permitindo que sejam estabelecidas relações de confiança e que usuários sejam responsabilizados pelos seus atos; e

3. fornecer canais seguros de comunicação, garantindo a integridade e confiabilidade dos dados.

A forma como estes serviços são organizados, agrupados e implementados pode variar muito de um middleware de grade para outro, mas de forma geral, todos os principais sistemas de grade oferecem essas funcionalidades [97].

Tipos de grades computacionais

Krauter, Buyya e Maheswaran [98] elaboraram em 2002 uma resenha dos principais sistemas de grades existentes até então e propuseram uma taxonomia identificando diferentes tipos de grades. Para eles, uma Grade de Computação (*Computing Grid*) é um sistema de Computação em Grade que visa a integração de recursos computacionais dispersos para prover uma maior capacidade combinada de processamento aos seus usuários. Já as Grades Computacionais Oportunistas (*Opportunistic Grids* ou *Scavenging Grids*) [73] são um subtipo de grade que utiliza os períodos ociosos de estações de trabalho de funcionários de uma organização, laboratórios acadêmicos, máquinas de estudantes, etc para formar, dinamicamente, uma grade de computação [97].

Grades de Dados (*Data Grids*) são sistemas de computação em grade que objetivam prover mecanismos especializados para publicação, descoberta, acesso e classificação de grandes volumes de dados distribuídos. A computação intensiva de dados, cujo foco está na síntese de novas informações a partir dos dados, que são mantidos em repósitórios, bibliotecas digitais e bancos de dados geograficamente distribuídos são exemplos de grade de dados [48].

Grades de Serviços (*Service Grids*) oferecem serviços viabilizados pela integração de diversos recursos computacionais, como por exemplo, um ambiente para trabalho colaborativo, ou uma plataforma para aprendizado à distância. Uma variante importante deste tipo de grade são os chamados Colaboratórios (*Collaboratories*) [156], laboratórios físicos compartilhados através da Internet de forma que vários pesquisadores e estudantes possam realizar experimentos científicos a distância. Os colaboratórios permitem que laboratórios de alto custo possam ser compartilhados por pessoas localizadas em diferentes cidades, reduzindo os custos e otimizando o uso de recursos escassos.

Grades Semânticas (*Semantic Grids*) são uma extensão das grades computacionais com informações e serviços bem definidos, cujos significados são explicitamente representados, permitindo melhor trabalho em equipe para pessoas e computadores [140].

6.1.2 Grade de Dados

Atualmente, uma grande quantidade de informação é gerada em variados domínios de aplicação, dando origem a grandes coleções de dados. Essas coleções são, em geral, geograficamente dispersas, assim como seus usuários. A combinação dessas coleções com a distribuição geográfica dos recursos e dos usuários demanda sofisticados mecanismos de acesso e compartilhamento dos mesmos [49].

As principais dificuldades para o acesso e compartilhamento de dados em ambientes distribuídos são:

- **Heterogeneidade de sistemas de banco de dados**: existe uma variedade de sistemas de banco de dados no mercado, que diferem tanto na estruturação lógica, tais como o tipo de armazenamento (XML, relacionais, arquivos padrão) quanto no acesso aos protocolos, devido às diversas linguagens de acesso (SQL, OQL, XPath). Este fato torna necessário o desenvolvimento de mecanismos de tradução entre os banco de dados[49];
- **Segurança**: cada instituição possui políticas de segurança independentes para proteger seus dados de usuários não-autorizados, de forma tal que a possibilidade de compartilhamento, em um ambiente distribuído impõe a criação de mecanismos de autenticação e autorização mais complexos, pois, ao possibilitar o acesso aos dados em rede, são herdados problemas de manutenção de segurança das redes de computadores [49];
- **Acesso aos dados com baixa latência**: aplicações como, por exemplo, análise experimental e simulação em física de alta-energia, modelagem climática, engenharia de terremotos e astronomia, são utilizadas por comunidades de centenas de pesquisadores, os quais compartilham conjuntos maciços de dados. Tal compartilhamento impõe a necessidade de transferência de grandes conjuntos de dados para processamento, o que leva a uma alta latência para transferência[7];
- **Confiabilidade**: dada a característica distribuída do ambiente, falhas são comuns. Métodos de recuperação de falhas são necessários para tratar falhas da rede, bem como falhas na transferência dos dados [7];
- **Transparência**: é desejável fornecer aos usuários uma interface uniforme para acesso aos dados distribuídos, de forma a superar aspectos de transparência como, por exemplo, acesso, localização e concorrência [7];
- **Publicação e descoberta dos dados**: dada a possibilidade do compartilhamento dos dados, um aspecto importante é como torná-los públicos. Assim, tem-se que fornecer mecanismos para que os pesquisadores e o público em geral venham a se beneficiar dos dados compartilhados, de modo que se faz necessária a existência de informações que possibilitem a publicação e descoberta desses dados [7].

A fim de superar essas dificuldades, que requerem um alto poder computacional em suas soluções, a computação em grade tem se especializado com o objetivo de fornecer a infra-estrutura para manipular grandes quantidades de dados [94]. Esta especialização é chamada Grade de Dados (*DataGrid*). Chervernak et al. [30] descrevem uma arquitetura geral para grade de dados, enfatizando dois serviços básicas:

- **Serviço de dados** - serviço que provê acesso a grandes conjuntos de dados distribuídos de forma eficiente. Inclui mecanismos para acesso, gerenciamento e execução

de transferências de dados dentro do ambiente da grade, com o objetivo de deixar transparente para os usuário a característica distribuída do ambiente.

- **Serviço de metadados** - fornece mecanismos para nomear, publicar e acessar os diferentes tipos de metadados sobre os dados. Um dos desafios para o ambiente de grade de dados é como identificar e localizar os dados que são de interesse para uma área particular. Uma solução é descrever as características dos dados através de metadados, e usar os metadados para identificar os dados relevantes.

O desenvolvimento de grades de dados é bastante complexo e envolve pesquisas em todas áreas relacionadas aos problemas apresentados nos parágrafos anteriores como: técnicas para o transporte eficiente de dados entre diferentes sítios, replicação e armazenamento de dados, localização de réplicas, escalonamento de execução de aplicações, controle de acesso aos dados e representação semântica dos dados. Neste trabalho, o maior interesse está no gerenciamento dos dados armazenados e na representação semântica dos mesmos [42].

6.1.3 Grade Semântica

Através dos contextos de grades computacionais e grades de dados se introduziu a noção de "Grade Semântica", em 2001 [46]. A visão de Grade Semântica é alcançar facilidade de utilização e automação, facilitando as colaborações flexíveis e computação em uma escala global [141]. Assim, a visão de Grade Semântica ficou estabelecida como a aplicação combinada de tecnologias, de Web Semântica e de Grade como representado na Figura 6.2.

A Grade Semântica é uma extensão da Grade computacional com informações e serviços bem definidos e significados, explicitamente representados, permitindo melhor trabalho em equipe para pessoas e computadores [46]. O uso da palavra "semântica", nesse contexto, não implica, evidentemente, na captura do significado das informações por parte das máquinas, mas em um conteúdo com significado lógico que pode ser mecanicamente manipulado pelos computadores, com finalidades úteis para os seres humanos [167].

A **Web Semântica** tem como objetivo apoiar a cooperação entre os recursos da Web, estabelecendo mecanismos ontológicos e de lógica, usando linguagens como o padrão XML (*eXtensible Markup Language*, RDF (*Resource Description Framework*), OIL (*Ontology Interchange Language*) e DAML (*DARPA Agent Markup Language*) para substituir HTML (*Hypertext Markup Language*) e permitir que páginas Web possuam descrições do seu conteúdo [172].

O objetivo dos **Serviços Web** ou ***Web Services*** é proporcionar uma plataforma aberta para o desenvolvimento, implantação, interação e gestão dos *e-services* distribuídos

Figura 6.2: *Posição da grade semântica no contexto de Grades [141]*

no ambiente. Os Serviços Web permitem a integração dos serviços que executam e residem em lugares diferentes [172].

O objetivo global da **Grade Computacional** é de compartilhar, gerir, coordenar, programar e controlar recursos de computação distribuída, como por exemplo máquinas, redes, dados, software e quaisquer tipos de dispositivos. O ideal da Grade Computacional é que qualquer dispositivo compatível possa ser ligado em qualquer lugar na Grade e os serviços necessários sejam garantidos, independentemente da sua localização, tal como a rede de energia elétrica. As grades computacionais podem utilizar tecnologias Web [172].

A **Grade Semântica**, por sua vez, tenta incorporar as vantagens da grade, Web Semântica e Serviços Web como elementos computacionais autodescritivos. Uma importante arquitetura de grade foi definida na forma da *Open Grid Services Architecture* (OGSA), no qual algumas funcionalidades dos *Web Services* podem ser vistas claramente [172].

Os conceitos de Grade e Web Semântica são fortemente relacionados. A Web Semântica é definida como "uma extensão da Web na qual toda informação possui um significado bem definido, possibilitando que computadores e pessoas trabalhem cooperativamente" [16]. A Grade Semântica procura incorporar a abordagem da Web Semântica à grade computacional, estabelecendo uma analogia na qual a grade semântica está para a grade computacional assim como a Web semântica está para a Web [24]. As visões da Grade e da Web Semântica têm muito em comum, mas podem ser distinguidas porque: a Grade é focada na computação, enquanto o foco da Web Semântica é direcionado para a inferência, a prova e a confiança [142].

A Grade Semântica aborda a forma como a informação é representada, armazenada, acessada, compartilhada e mantida - informação é entendida como dados dotados de significados. Ela deve estar preocupada com a maneira pela qual o conhecimento é adquirido, usado, copiado, publicado e mantido - o conhecimento é entendido como informação aplicada para atingir um objetivo, resolver um problema ou ordenar uma decisão. A Grade Semântica traz implicações para as três camadas conceituais da Grade: conhecimento, in-

formação e computação/dados. Essas camadas complementares fornecem, enfim, acesso a recursos heterogêneos distribuídos globalmente [142].

O conhecimento em Grades Semânticas pode ser visto sob dois aspectos complementares e interdependentes como "conhecimento ou semântica na grade" e "conhecimento ou semântica para a grade" [70]. A primeira visão, que também pode ser entendida como uma visão de "conhecimento para aplicações na grade", está relacionada ao uso de grades e organizações virtuais na exploração de ambientes para resolução de problemas - *Problem Solve Environments* (PSE), como no contexto de *e-science*. A semântica na grade descreve aplicações, dados e conceitos associados a um domínio de problema específico.

O "conhecimento para a grade" possibilita ao *middleware* da grade processar conhecimento sobre disponibilidade e propósito de serviços e recursos da grade, suas políticas de uso, estados e direitos de acesso, em uma extensa lista de possibilidades. Por exemplo, uma arquitetura de grade baseada em um *middleware* flexível e dinâmico deve possibilitar a descoberta e formação dinâmica de organizações virtuais *ad hoc* de recursos dispersos na grade [140].

Definindo mecanismos padrão de criação, nomeação, e localização de serviços, a Grade Semântica pode incorporar a tecnologia *peer-to-peer*, e assim permitir a objetos computacionais autônomos cooperarem igualmente em uma rede e com escalabilidade [172]. Assim, a ligação em redes *peer-to-peer* deve trabalhar não só no nível computacional mas também no nível semântico.

A idéia da Grade Semântica é fazer com que não só os recursos da grade sejam tratados semanticamente, mas também as aplicações da grade que necessitam ter diferentes restrições que podem ser satisfeitas apenas por certos tipos de recursos com capacidades específicas [140].

A Grade Semântica é um local onde os dados estão equipados com um contexto rico e transformado em informação. Essa informação será depois processada e compartilhada por organizações virtuais, a fim de atingir metas e objetivos específicos. Tais informações úteis, como mencionado acima, constituem o conhecimento [142].

A literatura trata o desafio da Grade Semântica com diferentes abordagens. Na prática, trabalhar com grade semântica corresponde a introduzir tecnologias da Web Semântica na Grade como descrito anteriormente [1]. O *background* de conhecimento e vocabulário desse domínio podem ser capturados em ontologias. Também podem ser representados por outros serviços (tal como um RDF *triplestore*) e serviços que suportam capacidades "semânticas", tais como serviços de tesauros, ou mesmo as ontologias, que suportam funções de bibliotecas digitais [70].

A Grade Semântica demanda uma arquitetura ou algum tipo de *framework* sistemático para projetar seus componentes e aplicações [71]. Alguns trabalhos estão sendo feitos nesta direção, os quais serão abordados mais adiante.

6.2 Construção da Ontologia

Nesta proposta, a ontologia deve conter descrição dos diferentes recursos computacionais providos por ambientes de grade. O uso desta ontologia deverá facilitar a recuperação de informações sobre recursos disponíveis em grades computacionais e a integração dos mesmos.

6.2.1 Classificação da Ontologia Produzida

Na Seção 2.1, foram apresentadas algumas classificações de ontologias sob diversos aspectos, como por exemplo: nível de detalhes, nível de dependência de uma tarefa ou ponto de vista e tipo de representação.

A partir dessas classificações, a ontologia construída encaixa-se, pelo nível de detalhes, como ontologia compartilhável, pois define um domínio comum que pode ser utilizado por vários usuários. Quanto à sua generalidade ou nível de dependência, pode ser classificada como ontologia de domínio, pois descreve o vocabulário relativo ao domínio específico de grade computacional e poderá ser reutilizada para confecção de novas ontologias.

A categoria adequada para classificação da ontologia proposta quanto ao seu uso, conforme definida por Uschold e Gruninger [163], também apresentada na Seção 2.1, é de ontologia para interoperabilidade entre sistemas, pois poderá ser utilizada como uma "ponte" entre um sistema de busca de uma comunidade virtual e o sistema da grade.

6.2.2 Linguagem e Ferramentas

A ontologia foi construída na linguagem OWL (*Web Ontology Language*), recomendada como um padrão de linguagem para a construção de ontologias pelo W3C [27]. Uma das vantagens da linguagem OWL é a criação de um vocabulário mais extenso para descrição de propriedades e classes, permitindo descrição de relacionamentos entre classes, cardinalidade, igualdade, características de propriedades, entre outras [4].

Para edição da ontologia, foi usado o editor Protégé 3.3.1 [69], [87], apresentado na Seção 2.3.5. Optou-se pela utilização deste editor, primeiramente, por ser um software livre, bem documentado, atualizado e disponível em diversas plataformas, como Windows, LINUX, Mac OS, Solaris e HP UX (*Hewlett-Packard* UNIX). Em segundo lugar, por ser um dos editores com maior quantidade de *plug-ins* disponíveis, dos quais foram utilizados o Jambalaya, o OWL-Viz, OWL2UML e o Pellet.

Como mecanismo de inferência, foi usada a ferramenta Pellet [55], apresentada na Seção 2.4.2. A escolha desta ferramenta deu-se por ela ser um *plugin* do Protégé, por ser gratuita, por inferir sobre a linguagem OWL-DL, por atender às recomendações do

W3C para verificadores de documentos OWL, e por ser um verificador sintático e um verificador de consistência completo.

6.2.3 Implementação da Ontologia

A ontologia proposta foi implementada seguindo os passos da metodologia descrita na seção anterior. Estes passos serão detalhados a seguir:

1. **Identificar o propósito** - para identificar o propósito, foi necessário responder à pergunta "qual é o objetivo da ontologia?". A resposta a esta pergunta é que o principal objetivo desta ontologia é possibilitar uma representação semântica dos recursos oferecidos por uma grade computacional para que possam ser reutilizados, compartilhados e estruturados, de forma que os usuários da grade possam realizar pesquisas "inteligentes" sobre estes recursos.

2. **Identificar o domínio e o escopo** - para identificar o domínio e o escopo da ontologia, foi necessário responder às questões "o que se deseja representar semanticamente com a ontologia?" e "para que informações a ontologia deve fornecer respostas?". A resposta à primeira questão é que serão representados os recursos de uma grade computacional, assim, o domínio representado é o das grades computacionais. Para obter a resposta da segunda pergunta, no domínio analisado, a ontologia deve poder responder a questões como as relacionadas a seguir:

   - Quais recursos estão disponíveis na grade?
   - Quais computadores possuem plataformas para a execução de uma determinada aplicação?
   - Quais são os tipos de hardwares disponíveis para a execução de uma aplicação?
   - Quais são os softwares disponíveis para implementar ou executar uma aplicação?
   - Quais hardwares adequados para a execução de uma aplicação com um desempenho esperado?
   - Quais middlewares estão disponíveis na grade?

3. **Estudar o ambiente** - definido o propósito e o domínio da ontologia, o próximo passo executado foi uma revisão bibliográfica sobre o domínio de grades computacionais. Na Seção 6.1, foram apresentados os resultados desta etapa. Visando a descrição de recursos computacionais oferecidos nos ambientes de grades, também

foi feita uma pesquisa a respeito dos conceitos que deveriam ser obtidos para descrição geral de recursos computacionais. Como resultado das pesquisas, foi adquirido um corpus, ou seja, um repositório de documentos do domínio.

4. **Planejar as tarefas a serem executadas no processo de construção** - nesta etapa foram definidos os próximos passos para construção da ontologia, as metas a serem cumpridas, a linguagem para implementação e as ferramentas utilizadas em cada passo seguinte. A meta foi a construção de uma ontologia consistente, especificando as relações semânticas que os termos mantêm com as suas classes e subclasses. Os passos são descritos a seguir. A linguagem e as ferramentas foram definidas após uma ampla pesquisa das linguagens e ferramentas existentes para construção e validação de ontologias que estão descritas no Capítulo 2. A Seção6.2.2 apresenta as ferramentas e a linguagem definidas.

5. **Conceitualização** - Foi feita uma investigação de ferramentas para extração dos termos nos documentos contidos no repositório adquirido. Optou-se pela utilização do arcabouço Sinapse, descrito no Capítulo 4.

   Para este processo, foram escolhidos 20 documentos (artigos, capítulos de livros, dissertações e teses) do domínio contidos no corpus. Esses documentos são arquivos no formato ".pdf" obtidos de repositórios da Unb, USP, UFG, UFMA, Rede Nacional de Ensino e Pesquisa, do projeto INTEGRADE, da Sociedade Brasileira de Computação, entre outros . Para a utilização do corpus no Sinapse, foi necessário converter os arquivos ".pdf" para o formato ".txt". Os documentos utilizados foram:

   - Artigos:
     - Integrando Dispositivos Móveis ao Middleware Integrade [75];
     - Ontologias Aplicadas à Descrição de Recursos em Ambientes *Grid* [131];
     - Repositório Seguro de Aplicações Baseado em GSS [45];
     - LiveOurGrid - Estimulando o Uso de Grades Computacionais Através da Experimentação [63];
     - InteGrade: Rumo a um Sistema de Computação em Grade para Aproveitamento de Recursos Ociosos em Máquinas Compartilhadas [74];
     - Selecionador de Recursos Grade Baseado na Integração Semântica de Múltiplas Ontologias [146];
     - Uma Solução Software Livre para Mineração de Dados em Grades Computacionais [165];
     - Conhecendo as Grades Computacionais [44];
     - MAG, Um Middleware de Grade Baseado em Agentes: Estado Atual e Perspectivas Futuras [37];

  - Descoberta de Recursos em Grades Computacionais Utilizando Estruturas P2P [137];
- Capítulo de Livro: Grades Computacionais - Conceitos Fundamentais e Casos Concretos [97];
- Dissertações:
  - Matching Semântico de Recursos Computacionais em Ambientes de Grade com Múltiplas Ontologias [106];
  - Serviço Baseado em Semântica para Descoberta de Recursos em Grade Computacional [1];
  - InteGrade: Um Sistema de Middleware para Computação em Grade Oportunista [72];
  - Protocolos Par-a-Par para Interligação de Aglomerados em Grades Computacionais [138];
  - Uma Proposta de um Sistema de Imagem Única para uso de Computação em Grade em Organizações Virtuais [132];
  - MobiGrid: Arcabouço para Agentes Móveis em Ambiente de Grades Computacionais [7];
- Teses:
  - Armazenamento Distribuído de Dados e Checkpointing de Aplicações Paralelas em Grades Oportunistas [42];
  - Abordagem Semântica Aplicada à Integração e Gerenciamento de Recursos e Aplicações em Grades Computacionais [167];
  - Seleção Distribuída de Recursos em Grades Computacionais Usando Raciocínio Baseado em Casos e Políticas de Granularidade Fina [121];

(a) **Capturar os termos importantes** - para a captura dos termos, executou-se uma seleção de termos relevantes para a base de conhecimento. Esta seleção foi feita utilizando o arcabouço Sinapse, com a identificação dos termos ou elementos úteis para a construção desejada. Este processo é ilustrado na Figura 6.3.

Para obter todos os termos contidos nos documentos, foi necessária uma modificação na estrutura do Sinapse, fazendo com que ele extraísse os termos de todos os documentos, acrescentando-os nos registros do banco de dados. Para essa captura, foram utilizados somente os processos 1, 2 e 3 do pré-processamento do Sinapse, descritos na Seção 4.7, como ilustrado na Figura 6.3.

(b) **Enumerar e organizar os termos importantes** - a partir dos dados obtidos com a varredura do Sinapse, foi possível enumerar e organizar os substantivos e verbos mais importantes no domínio através de relatórios. O modelo desses relatórios está exposto no Anexo A.

O critério utilizado para identificar e enumar os termos mais importantes do domínio, foi o de ordenar os dados em ordem alfabética, agrupar por "Stem" (ou radical) de mesma base, somar a frequência de todas as palavras com o mesmo radical e estabelecer um limite mínimo de frequência para cada agrupamento. Para isto, foi levada em consideração a classe gramatical de cada palavra.

Para encontrar o limite mínimo de frequência de cada classe analisada, foi retirado um relatório com o total de frequência das classes gramaticais tratadas pelo Sinapse e o total geral de todas as palavras do corpus, como mostra o relatório A do Apêndice A. A partir dos dados do relatório, estabeleceu-se um limite de frequência mínima de 10% dos totais de cada classe, pois, abaixo desse valor, o termo se mostra insignificante para o total de palavras existentes nos documentos do corpus. Para os substantivos, o limite é maior que para os verbos, devido às suas aplicações sintáticas nas frases. Assim, foram identificadas as palavras mais relevantes em todos os documentos analisados.

6. **Formalizar os termos** - desenvolveu-se um diagrama, ou uma prévia da rede semântica, definindo as classes e a hierarquia das classes. Para esta etapa, foi executado um processo manual de obtenção das relações entre os termos, utilizando o corpus e os relatórios gerados na fase de conceitualização, como mostra a Figura 6.4. Um dicionário [169] também foi utilizado para garantir as propriedades e conceitos dos termos obtidos.

   Como ilustrado na Figura 6.4, primeiramente, foi feita uma leitura dinâmica[2] dos documentos do domínio. Essa leitura se fez necessário para identificar as relações sintáticas entre os termos sintáticos. O resultado foi uma árvore sintática das frases, da qual foi possível gerar um mapa conceitual.

   Conforme ilustrado na Figura 6.4, para obter-se uma árvore sintática representativa das frases de cada texto, fez-se uma análise semântica parcial, de onde se extraiu triplas de palavras (sujeito, verbo, complemento), que deram origem às instâncias de entidades, ou conceitos, axiomas e ações envolvendo esses conceitos. As extração

[2]Leitura Dinâmica é um método que consiste basicamente na leitura de textos de forma rápida e clara, mantendo o entendimento apesar da velocidade [26].

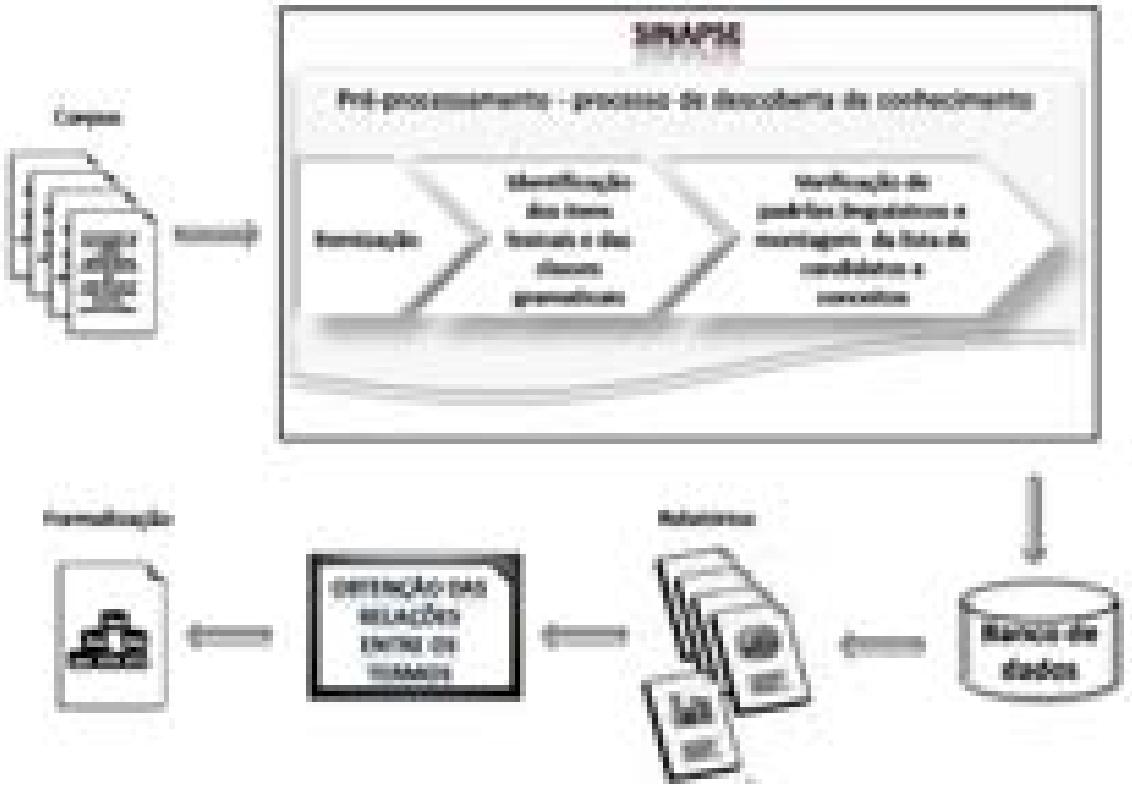

Figura 6.3: *Processo da Conceitualização*

Figura 6.4: *Processo de obtenção das relações entre os termos*

das triplas foi feita a partir de buscas nos documentos das palavras mais relevantes enumeradas no passo anterior.

Na gramática da Língua Portuguesa, as orações, consideradas sintaticamente, são constituídas de duas partes principais: sujeito e predicado [9]. Assim, através das relações sintáticas entre os termos, o próximo passo foi a dedução de algumas relações semânticas entre os respectivos conceitos (classes, subclasses, propriedades e instâncias).

Em seguida, recolheu-se informação sobre o grupo sintático (sujeito ou predicado) em que surge cada substantivo, relativamente ao verbo da frase. Para ajudar a encontrar as propriedades ou relações, a partir da respectiva lista de valores possíveis dos conceitos, foi utilizado um dicionário [169]. Após a identificação do grupo sintático dos termos, as relações entre eles e os axiomas puderam ser deduzidas. O resultado desse processo está ilustrado, de uma maneira genérica, na Figura 6.5. A Figura 6.5 mostra uma rede semântica contendo as classes gerais da ontologia proposta.

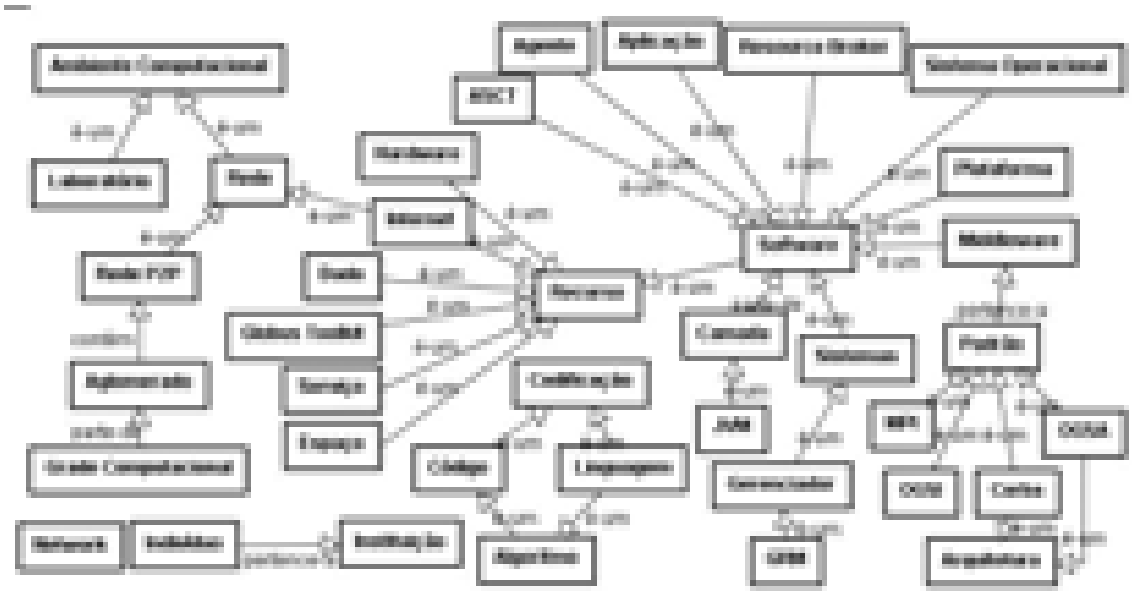

Figura 6.5: *Produto da Formalização: rede semântica com as classes gerais da ontologia proposta*

7. **Implementação** - codificação da ontologia. Neste passo, foi utilizada a ferramenta Protégé para o desenvolvimento da ontologia na linguagem OWL, como especificado na Seção 6.2.2. A partir dos resultados obtidos nas fases de conceitualização e formalização foi implementada a ontologia.

   A Figura 6.6 apresenta uma métrica da ontologia proposta gerada pela ferramenta Protégé. Em resumo, a ontologia é composta por 91 classes (*classes*), 7 restrições

(*restrictions*), 44 propriedades (*properties*), 9 atributos (*datatypes*) e 21 indivíduos (não especificados na métrica).

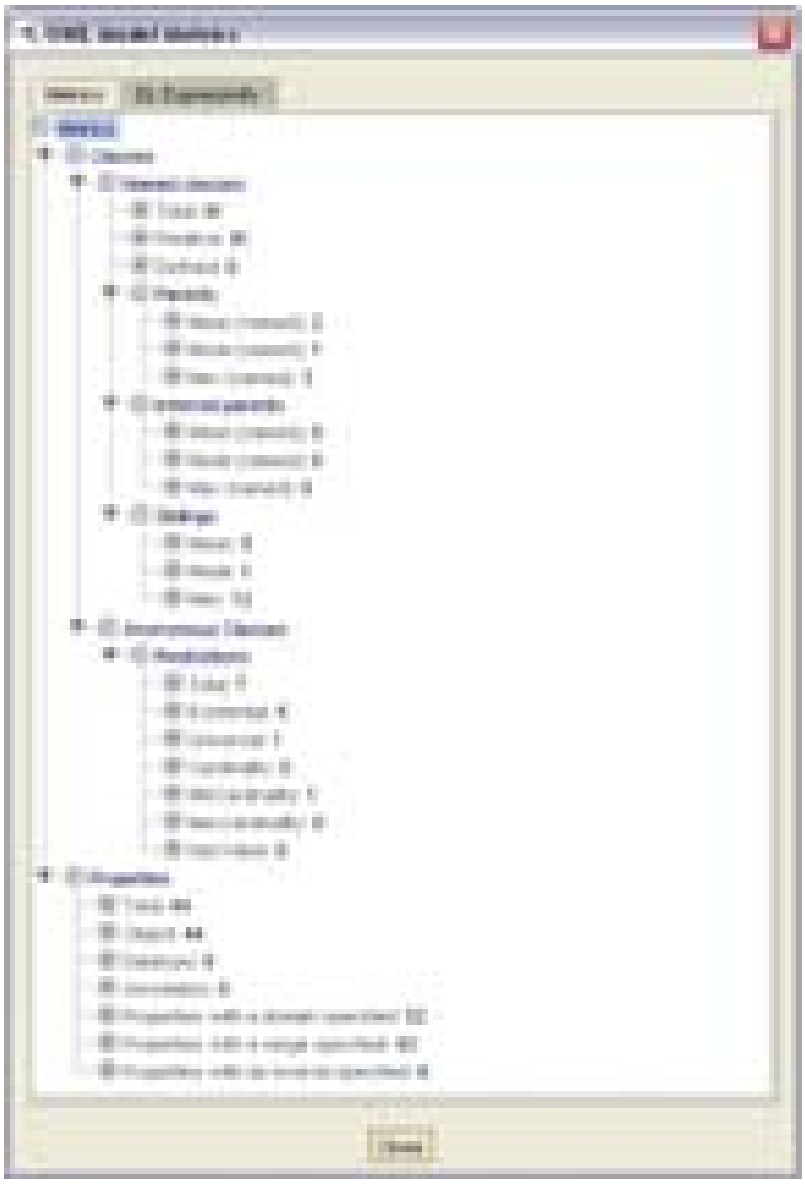

Figura 6.6: *Métrica Geral da Ontologia*

A Figura 6.7, mostra a visão geral da hierarquia de todas as classes declaradas na ontologia, gerada pelo *plugin* OWL Viz do Protégé. O *plugin* OWL Viz mostra uma visão das hierarquias declaradas e inferidas (que neste caso, são as mesmas como será especificado adiante).

Na Figura 6.8, é mostrado o diagrama geral das classes e propriedades. Esse diagrama foi gerado pelo *plugin* OWL2UML do Protégé. O *plugin* OWL2UML gera um diagrama geral de classes com as opções de comentários, propriedades, associações, restrições e generalizações. Para um visual mais claro, o diagrama

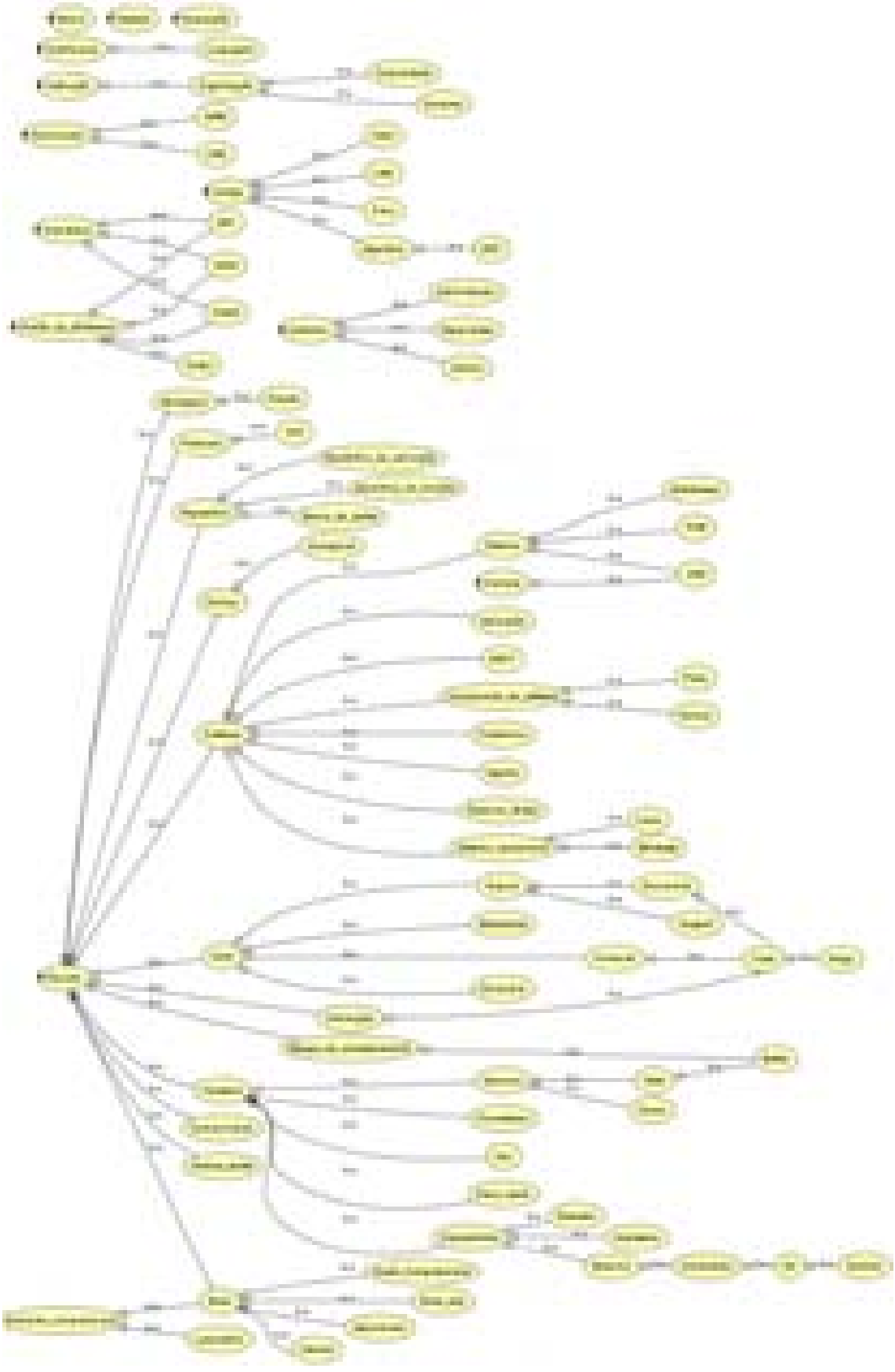

Figura 6.7: *Hierarquia das Classes gerada pelo plugin OWL Viz*

da Figura 6.8 foi gerado somente com as opções de associações, restrições e generalizações.

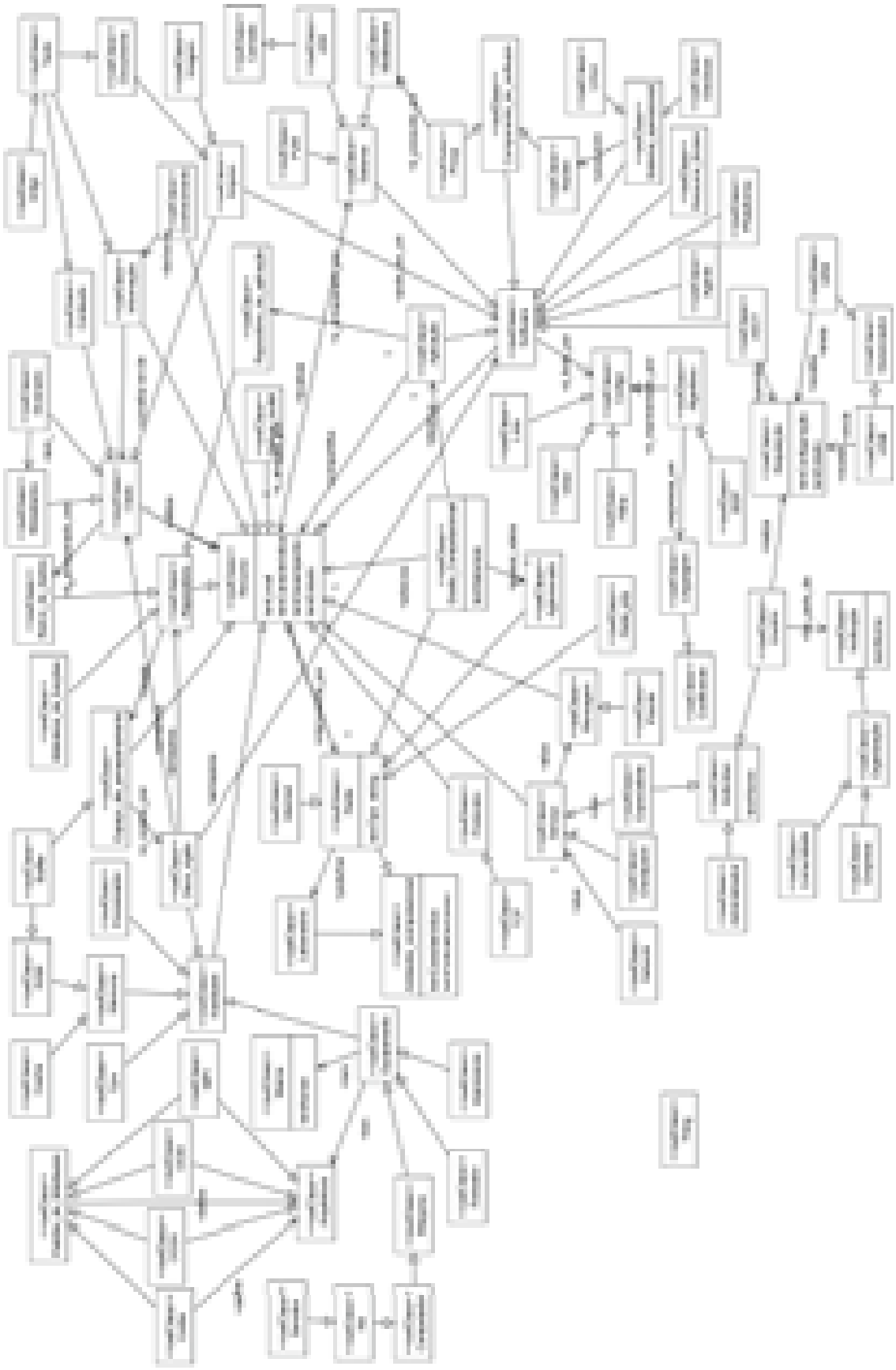

Figura 6.8: *Visão Geral da Ontologia gerada pelo plugin OWL2UML*

8. **Avaliação** - Essa etapa foi iniciada com o processo de validação da ontologia por um mecanismo de inferência. Neste processo, foi utilizada a ferramenta Pellet para a verificação sintática e de consistência completa de forma automática.

   O resultado da verificação automática pela Pellet demonstrou que não houve nenhuma inconsistência ou fatos contraditórios na ontologia implementada.

   Na Figura 6.9, é mostrada a métrica, gerada pelo Protégé, após realizar essa verificação pela Pellet. Em uma comparação com a Figura 6.6, nota-se a diferença que foi circulada na Figura 6.9, sendo que os parentes inferidos antes eram todos 0 (zero).

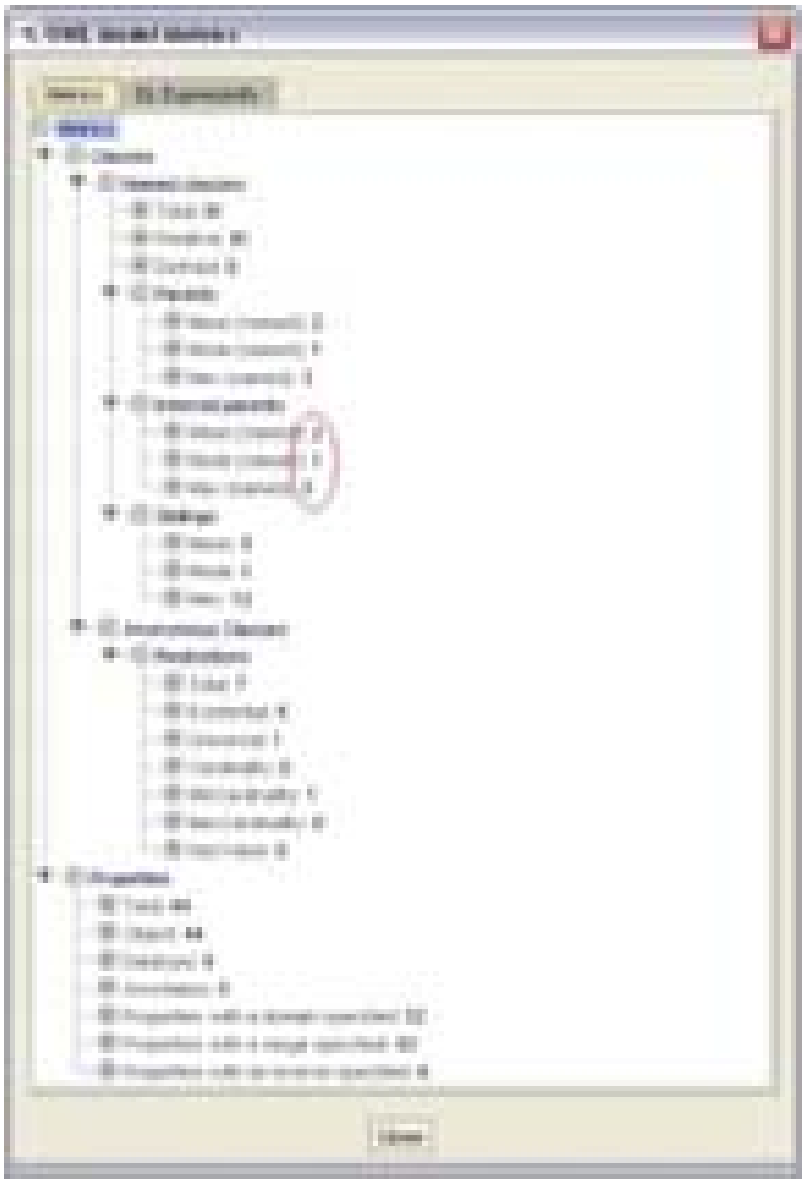

Figura 6.9: *Métricas após a inferência*

Na Seção 6.3.1, é ilustrado o resultado das validações executadas pela ferramenta Pellet, através de testes de consultas (queries) que validam as questões de competências exemplicadas no passo 2 (Identificar o domínio e o escopo).

Após a validação pelo mecanismo de inferência, também foi feita uma validação da ontologia por um especialista em Grades Computacionais. Foi feita uma reunião apresentando todas as classes, hierarquias e propriedades da ontologia para uma avaliação consistente e exata. O especialista identificou poucas incoerências e propôs poucas alterações. Os passos de implementação e avaliação foram novamente executados garantindo assim, a consistência da ontologia.

9. **Documentação** - no Apêndice B, é apresentado o detalhamento da ontologia com apresentação das classes e suas subclasses e, do código OWL e RDF gerado pela etapa de implementação.

6.3 Cenário de Uso

Para a aplicação da ontologia, é proposto um cenário com uma arquitetura em três camadas, como ilustrada na Figura 6.10:

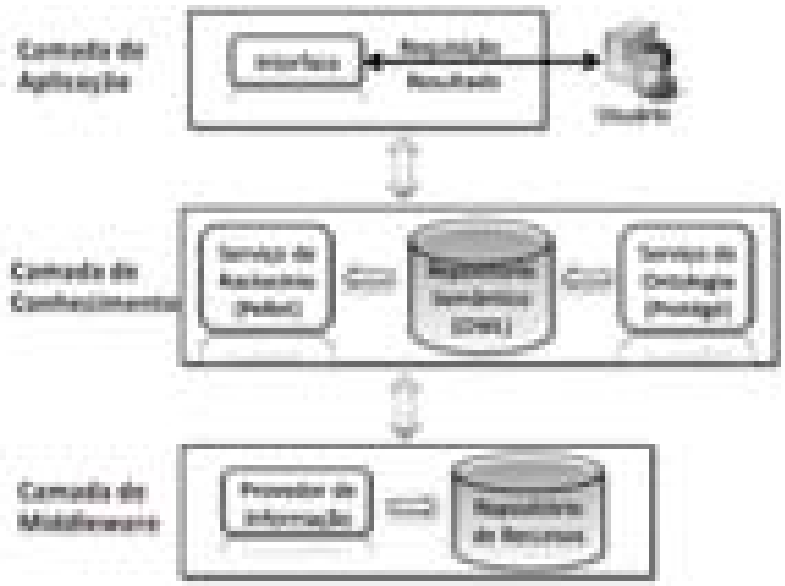

Figura 6.10: *Arquitetura do cenário de uso*

A Figura 6.10 ilustra a arquitetura que apresenta a Camada de Aplicação implementada no topo da Camada de Conhecimento que, por sua vez, se encontra em cima da da Camada de *Middleware*.

- **A Camada de Aplicação** é a que fica no topo da arquitetura, a qual provê o uso dos recursos do ambiente de grade. A interface gráfica do protótipo deve permitir que o requisitante de recursos submeta uma consulta e obtenha o resultado semântico de sua requisição através do serviço de ontologia que utiliza o motor de inferência;
- **A Camada de Conhecimento** é o foco principal deste cenário e provê o serviço de descoberta semântica de recursos de uma grande quantidade de dados agregados da camada de serviços de informação que estão armazenados no Repositório de Recursos. Para o ambiente da grade do cenário apresentado, os usuários e os agentes de software devem ser capazes de descobrir os nós da grade oferecendo os serviços requeridos e com propriedades particulares. No cenário descrito, a Camada de Conhecimento é formada pela ontologia e os componentes de raciocínio. A Camada de Conhecimento é orientada a domínio e utiliza o conhecimento da grade, que foi construído através da ontologia da grade. Serão descritos abaixo os componentes implementados na Camada de Conhecimento:
  - **Serviço de Ontologia** - para este serviço é utilizado o Protégé (Seção 2.3.5) para lidar com os recursos da grade semanticamente. Para cada tipo de informação recuperada do nó da grade, cria-se instâncias dos conceitos apropriados no código da ontologia. Desta forma, um modo em que uma instância será exatamente relacionada com apenas um tipo de conceito e os valores de várias propriedades recuperadas são associadas às respectivas propriedade dos conceitos apropriados no código da ontologia. Neste trabalho, o código da ontologia é definido como a ontologia dos recursos da grade que prevêem a hierarquia de conceitos nos quais algumas propriedades definem suas características. A Seção sec:TemplateDaOntologia do Anexo B mostra o código da ontologia com a hierarquia de conceitos considerada. Este código, que não é definitivo e por isto pode sofrer evoluções, define um mínimo de conceitos e propriedades providos por um serviço de recuperação de informação de recursos em cada elemento da grade.
  - **Repositório Semântico** - o código da ontologia é formado por conceitos e propriedades que correspondem aos valores das instâncias e propriedades que constituem o Repositório Semântico dos recursos da grade. Os valores das propriedades consideradas para definir os conceitos estão armazenados em um repositório de recursos na Camada de *Middleware*. A descrição semântica dos recursos no repositório facilita o uso de mecanismos de inferência para

interagir com a base e recuperar informações semanticamente. O repositório de recursos deve ser atualizado periodicamente, assim, a adição ou remoção de recursos é contabilizada no repositório semântico dinamicamente.

  - **Serviço de raciocínio** - o serviço de raciocínio utiliza o motor de inferência Pellet (Seção 2.4.2). A API do Pellet provê funcionalidades para verificação da validade de espécies, checagem de consistência de ontologias, classificação de taxonomia, checar inferências e responder um conjunto e consultas. A Pellet pode ser utilizado como um raciocinador externo ao Protégé.

- **Na Camada de *Middleware*** é provido o acesso unificado e seguro aos recursos remotos, podendo-se utilizar diferentes *middlewares* de grade. Ela deve conter um **Provedor de Informação**, ou seja, um serviço de informação que contém um modelo abstrato de mapeamento dos recursos da grade. Nesta camada, os dados sobre os recursos da grade capturados pelo provedor de informações serão armazenados em um **Repositório de Recursos**.

6.3.1 Validação

Para validar a ontologia para o cenário descrito nesta Seção, foi simulado um protótipo da camada de conhecimento, que é o foco deste Capítulo, utilizando o Protégé executando inferências com a ferramenta Pellet. Para demonstrar as inferências, foram criados exemplos de instâncias com suas propriedades e executadas algumas consultas (queries). Para as consultas, foram utilizadas as questões de competência exemplificadas na Seção 6.2.3, verificando assim, a consistência da ontologia.

A seguir, serão ilustrados os resultados obtidos com a aplicação das consultas executadas pela Pellet.

A Figura 6.11 verifica a questão “Quais recursos estão disponíveis na grade?”.

As Figuras 6.12 e 6.13 verificam a questão “Quais computadores possuem plataformas para a execução de uma determinada aplicação?”.

A Figura 6.14 verifica a questão “Quais são os softwares disponíveis para implementar ou executar uma aplicação?”.

A Figura 6.15 verifica a questão “Quais hardwares adequados para a execução de uma aplicação com um desempenho esperado? ”.

A Figura 6.16 verifica a questão “Quais middlewares estão disponíveis na grade?”.

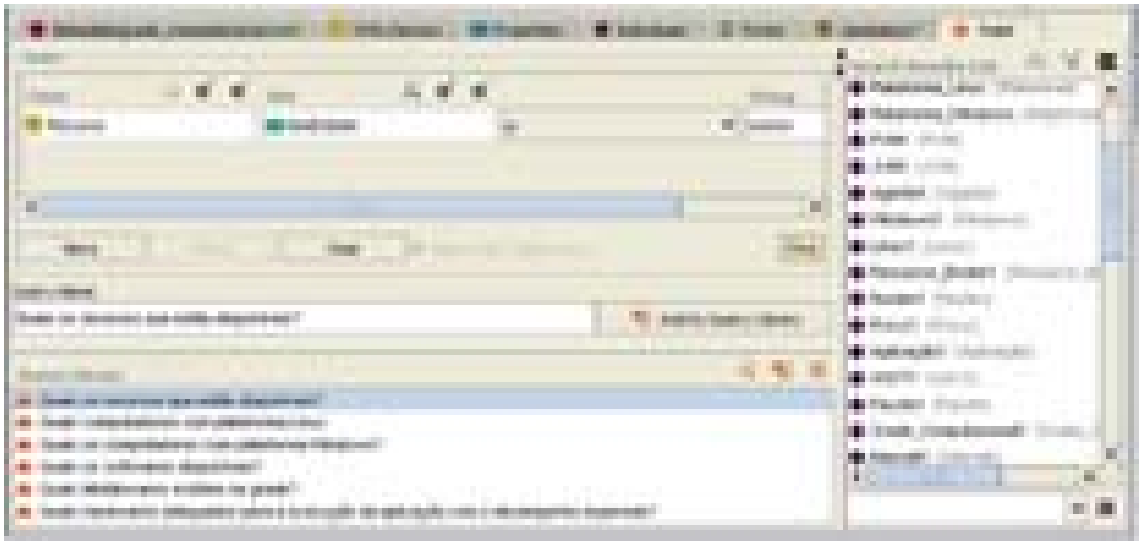

Figura 6.11: *Consulta recursos disponíveis.*

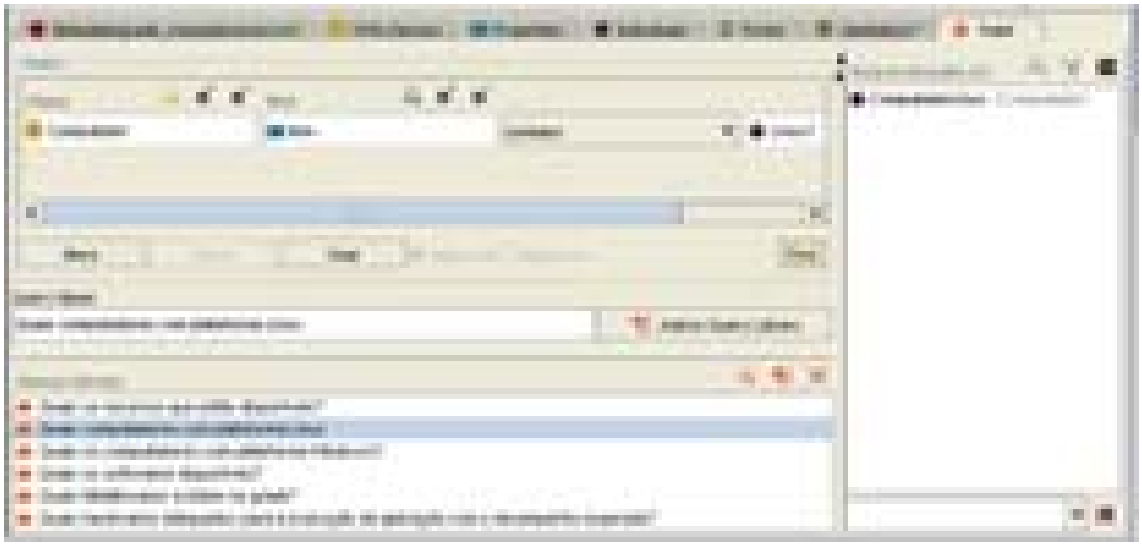

Figura 6.12: *Consulta um computador com plataforma Linux.*

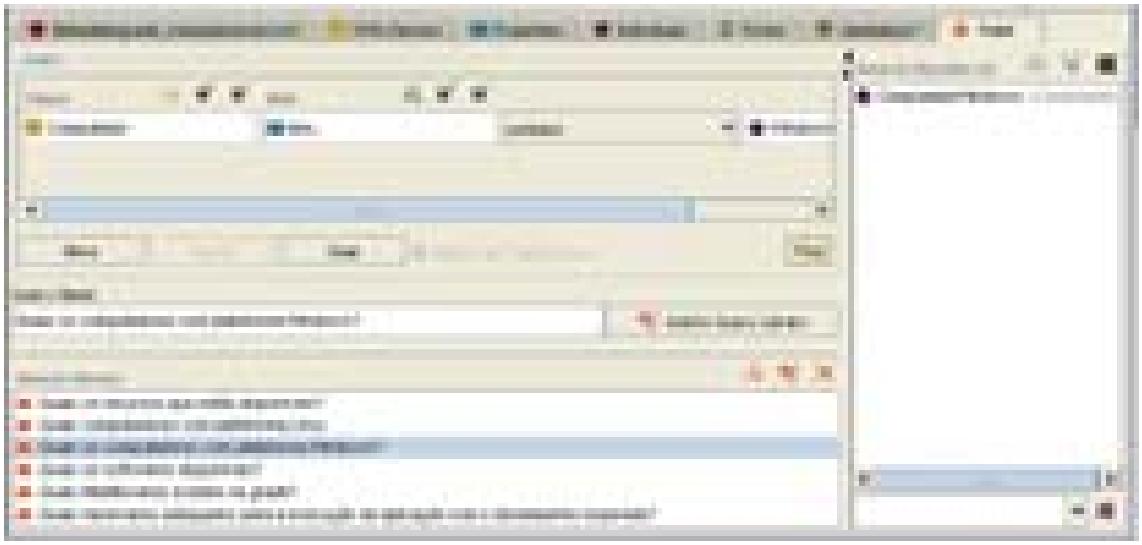

Figura 6.13: *Consulta um computador com plataforma Windows.*

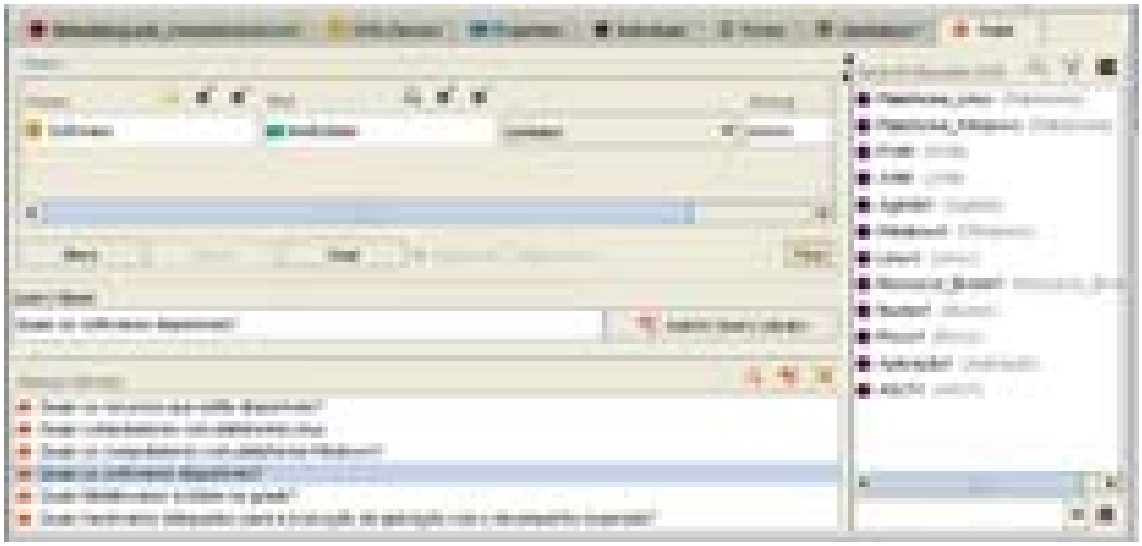

Figura 6.14: *Consulta quais softwares estão disponíveis.*

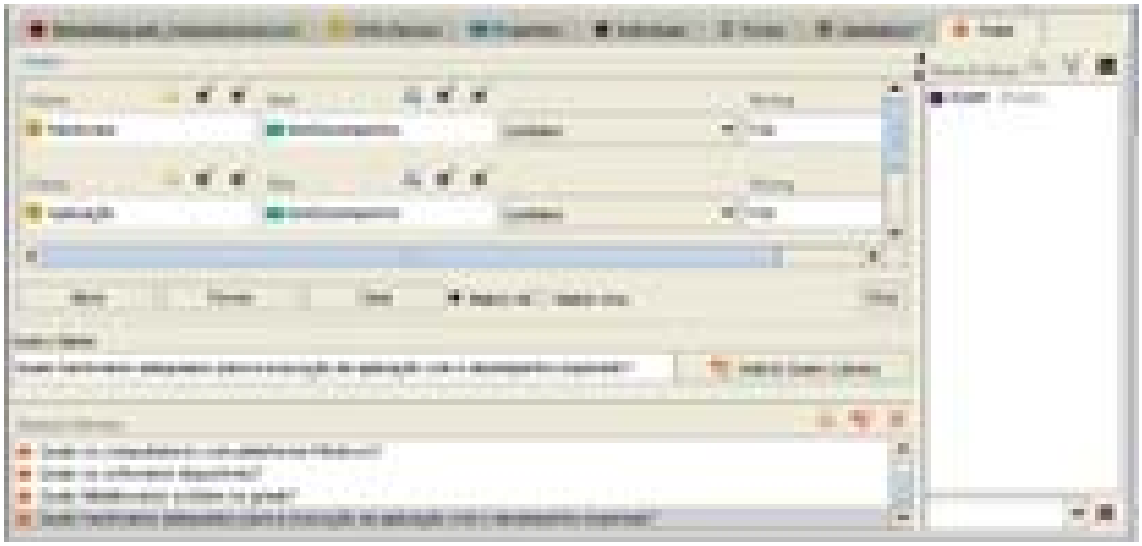

Figura 6.15: *Consulta o hardware com o desempenho adequado para uma aplicação específica.*

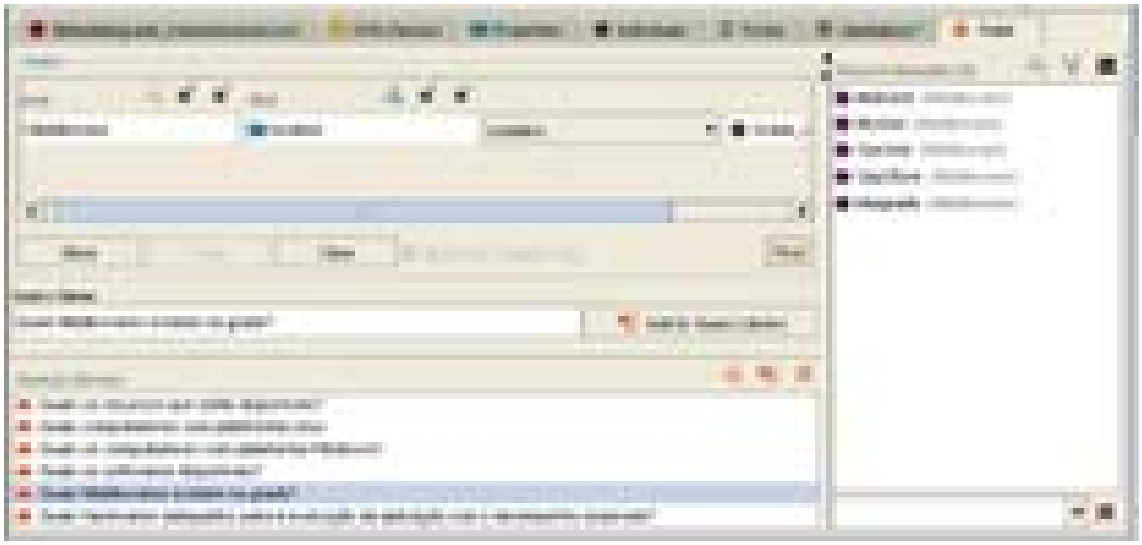

Figura 6.16: *Consutla os middlewares disponíveis na grade.*

CAPÍTULO 7

Conclusão e Trabalhos Futuros

Nesta dissertação, foi proposta uma nova metodologia para a construção semi-automatizada de ontologias. Um estudo de caso também foi realizado. A área escolhida para este estudo foi a de Grades Computacionais. Neste Capítulo, será feito um balanço dos resultados alcançados.

Numa construção totalmente manual de ontologias, existe sempre um risco de se demorar bastante na construção e ainda se obter uma ontologia demasiadamente simples, ou o contrário, uma ontologia tão elaborada que torna sua utilização pouco viável. Nessa construção, além do tempo, o engenheiro de ontologias também se depara com outros obstáculos como a intervenção ou dependência humana e, tarefas exaustivas e minunciosas tais como entrevistas, leitura dinâmica de documentos, aplicação de questionários, observações, análises, captura de termos entre outros. Estes obstáculos podem elevar o tempo da construção e inviabilizar esta atividade.

Estudos apresentados no Capítulo 3, mostraram a necessidade de uma metodologia formalizada, simples e precisa. Estes mesmos estudos também mostraram que não há um senso comum sobre um método ou metodologia ótimos para o desenvolvimento de ontologias. Diante das comparações apresentadas na Seção 5.1, ficou claro que são necessários muitos recursos, como tempo, documentos e pessoas, no processo manual de construção de ontologias e a dificuldade de formalizar o processo de construção.

A implementação de um processo semiautomatizado para a construção de ontologias possibilitou a redução dos riscos e obstáculos com os quais se deparam os engenheiros de ontologias na sua construção.

A metodologia proposta é o resultado da combinação da metodologia de Uschold e King [164],com a *Methontology* [19] e com o Método 101 [122], resultando em uma metodologia produtiva através de passos bem definidos e de fácil aplicação.

A aplicação da metodologia proposta, ou seja, a construção da ontologia para Grades Computacionais, utilizando da metodologia proposta, foi executada em 15 dias, enquanto o processo manual poderia levar meses, devido ao exaustivo e minucioso trabalho de extração dos termos importantes do domínio. Os recursos utilizados foram as ferramentas Sinapse, Protégé e Pellet, o repositório de documentos adquiridos na etapa

de estudo do ambiente e um dicionário. Vale ressaltar que, para essa construção, não foi necessária a presença de um especialista do domínio. A ontologia construída é composta por 91 classes, 7 restrições, 44 propriedades, 9 atributos e 21 indivíduos.

O resultado da aplicação da metodologia proposta, apresentou características esperadas, tais como produtividade e eficiência no seu uso, pois o objetivo de reduzir o tempo na construção de ontologia foi alcançado. Assim, esta metodologia torna-se um atrativo para mensurar o esforço gasto na realização da construção da ontologia e permite futuras análises do custo dessa atividade dentro da Engenharia de Ontologia.

Uma importante característica da metodologia é sua independência de domínio, pois seus processos manuais são bem definidos, o que permite sua aplicação para diversas áreas do conhecimento.

Os objetivos delineados no começo do trabalho foram, na sua essência, atingidos, ainda que existam algumas limitações, como é o caso de ressaltar que os resultados obtidos na construção da ontologia carecem de validação por parte de especialistas. O processo descrito, ainda não substitui todas as etapas do processo manual de construção de ontologias. Funciona antes como uma adequação e melhoramento no mesmo, através da identificação (semiautomática) de conceitos e relações a representar na ontologia.

O fato de trabalhar com a ferramenta Sinapse, uma ferramenta de Processamento de Linguagem Natural, também pode ser uma limitação, pois os resultados nem sempre são rigorosamente corretos. A perda de informação durante o processo de extração executada pelo Sinapse pode afetar, não apenas o processo de construção da ontologia como também, e consequentemente, o processo de avaliação/validação da mesma.

Há um aspeto alheio a este trabalho que acaba por contribuir para resultados não satisfatórios: o fato dos documentos utilizados serem repletos de expressões descritas a partir de diferentes pontos de vista, dificultando assim, a fase de extração de informação do domínio estudado. Além disso, devido a essas diferenças nos textos, pode haver falhas na etapa de conceitualização devida, à captura de termos considerados relevantes mas que são alheios ao domínio.

Por outro lado, o resultado alcançado com a construção de uma ontologia a partir da metodologia proposta foi considerado satisfatório, pois os objetivos esperados foram atingidos.

7.1 Visão Geral da Ontologia

Como apresentado no Capítulo 6.1, ambientes de grade contém uma grande quantidade de recursos heterogêneos para permitir a execução de diferentes tipos de aplicação em diferentes modos. Por serem classificados sob uma grande variedade de

aspectos e características, existe uma grande dificuldade na busca e reutilização das aplicações e dos recursos computacionais em grades [167].

Neste contexto, a ontologia gerada, envolvendo classes e instâncias do domínio de Grades Computacionais, possui classificações que podem ser proveitosas como base de uma estratégia para utilização eficiente da grade em diferentes cenários. Consequentemente, o uso desta ontologia na busca por uma aplicação, ou conjunto de aplicações, ou de recursos contidos no ambiente de grade para reutilização, mostra-se útil nos cenários de recuperação de informações sobre as aplicações e na pré-seleção de recursos para execução de aplicações na grade.

Uma característica importante da ontologia desenvolvida é seu potencial de reutilização para diferentes sistemas de *middleware* de grade, pois ela é um modelo genérico e extensível. Ela é baseada em um conjunto de classes e propriedades com significados bem definidos em contexto simples de aplicações e recursos, que podem ser estendidos e usados em diferentes domínios na grade. Conclui-se, portanto, que a ontologia foi satisfatória.

7.2 Contribuições Esperadas

A proposta agrega uma contribuição à Engenharia de Ontologia ao propor a formalização da tarefa de construção de ontologias, buscando orientar a subjetividade encontrada nas etapas de planejamento, estudo do ambiente, captura e enumeração dos termos.

Este trabalho apresenta uma contribuição na forma de construir ontologias, introduzindo uma metodologia para a construção semiautomatizada de ontologias (classes e respectivas instâncias), diretamente a partir do texto dos documentos em linguagem natural.

Espera-se também, contribuir para a integração, compartilhamento e reutilização de recursos, aplicações e dados para grades computacionais, possibilitando, em consequência, a integração e interoperabilidade de diferentes comunidades virtuais que constituem esses ambientes de grade através do uso da ontologia.

7.3 Trabalhos Futuros

As limitações apontadas neste Capítulo são:

- necessidade de validação da ontologia por parte de especialistas;
- várias etapas do processo, que são complexas e demoradas, ainda são executadas manualmente;

- problemas com o uso da ferramenta Sinapse, como a deficiência de diferenciação de *tokens*, limitações na classificação das palavras gramaticalmente e na quantidade de termos extraídos, provocando assim, lentidão no processo de extração pela ferramenta;
- complexidade do processo de seleção e uniformização do corpus.

Para superá-las, é necessário desenvolver alguns trabalhos sobre os seguintes aspectos:

- Aprimoramento do processo de análise sintática e semântica, com extração automática de n-uplas de termos, de forma a permitir uma melhor identificação dos conceitos e propriedades dentro das orações.
- Aprimoramento da ferramenta Sinapse para aumentar a precisão da extração dos termos no corpus. Este aprimoramento se dará com a ampliação das classes gramaticais representadas pela ferramenta, a caracterização e diferenciação de *tokens* e agilidade no processo de extração.
- Automatização do processo de seleção de documentos do domínio que irão compor o corpus. O processo de busca e seleção dos documentos do domínio é uma tarefa exaustiva e que exige muito tempo. A automatização desse processo pode levar a uma redução considerável do tempo para a construção da ontologia.
- Aplicação do conceito de grades semânticas em comunidades virtuais utilizando da ontologia proposta.

Referências Bibliográficas

[1] ALLEMAND, J. N. C. **Serviço Baseado em Semântica para Descoberta de Recursos em Grade Computacional**. Master's thesis, Universidade de Brasília, Unb, 2006.

[2] ALMEIDA, M; BAX, M. **Uma Visão Geral sobre Ontologias: Pesquisa sobre Definições, Tipos, Aplicações, Métodos de Avaliação e de Construção**. Ciência da Informação, 32(3), 2004.

[3] ANGELE, J; LAUSEN, G. **Ontologies in F-Logic**. In: Staab, S; Studer, R, editors, Handbook on Ontologies. Springer, 2004.

[4] ANTONIOU, G; ANTONIOU, G; HARMELEN, F. V; HARMELEN, F. V. **Web Ontology Language: Owl**. In: Handbook on Ontologies in Information Systems, p. 67–92. Springer, 2003.

[5] AT THE STANFORD UNIVERSITY SCHOOL OF MEDICINE, S. M. I. **The Protégé Ontology Editor and Knowledge Acquisition System**. http://protege.stanford.edu/, último acesso em Janeiro de 2009, 2006.

[6] BAADER, F. **The Description Logic Handbook: Theory, Implementation, and Applications**. Cambridge University Press, September 2007.

[7] BARBOSA, R. M. **MobiGrid: Arcabouço para Agentes Móveis em Ambiente de Grades Computacionais**. Master's thesis, Universidade de São Paulo, USP, 2007.

[8] BECHARA, E. **Gramática Escolar da Língua Portuguesa**. Editora Lucerna, 1 edition, 2001.

[9] BECHARA, E. **Moderna Gramática Portuguesa - Atualizada Pelo Novo Acordo Ortográfico**. Editora Lucerna, 37 edition, 2009.

[10] BECHHOFER, S; HORROCKS, I; GOBLE, C. A; STEVENS, R. **OilEd: A Reasonable Ontology Editor for the Semantic Web**. In: KI '01: Proceedings of the Joint German/Austrian Conference on AI, p. 396–408, London, UK, 2001. Springer-Verlag.

[11] BECHHOFER, S; MÖLLER, R; CROWTHER, P. **The DIG Description Logic Interface**. In: Proceedings of the International Workshop on Description Logics (DL-2003), Rome, Italy, September 5-7, 2003.

[12] BECHHOFER, S; VOLZ, R; LORD, P. W. **Cooking the Semantic Web with the OWL API.** In: Proceedings of the International Semantic Web Conference, p. 659–675, 2003.

[13] BECKETT, D; BERNERS-LEE, T. **Turtle - Terse RDF Triple Language**. W3c team submission, W3C, January 2008.

[14] BERNARAS, A; LARESGOITI, I; CORERA, J. **Building and Reusing Ontologies for Electrical Network Applications**. In: Wahlster, W, editor, Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 96): Budapest, Hungary: August 11-16, p. 298–302. Wiley, 1996.

[15] BERNERS-LEE, T. **Notation 3**. Technical report, W3C, 1998.

[16] BERNERS-LEE, T; HENDLER, J; LASSILA, O. **The Semantic Web: Scientific American**. Scientific American, 284(5):34–43, May 2001.

[17] BLACKBURN, S. **Dicionário Oxford de Filosofia**. Jorge Zahar, Rio de Janeiro, RJ, 1997.

[18] BREITMAN, K. K; DO PRADO LEITE, J. C. S. **Ontology as a Requirements Engineering Product**. Requirements Engineering, IEEE International Conference on, 0:309, 2003.

[19] BREITMAN, K. K. **Web Semântica: a Internet do Futuro**. LTC, Rio de Janeiro, RJ, 2005.

[20] BRICKLEY, D; GUHA, R. **Resource Description Framework (RDF) Schema Specification 1.0**. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, último acesso em Janeiro de 2009, 2000.

[21] BROEKSTRA, J; KAMPMAN, A. **SeRQL: An RDF Query and Transformation Language**. ISWC, August 2004.

[22] BROOKE, J; FELLOWS, D; GARWOOD, K. L; GOBLE, C. A. **Semantic Matching of Grid Resource Descriptions**. In: Proceedings of the European Across Grids Conference, p. 240–249, 2004.

[23] BRUSA, G; CALIUSCO, M. L; CHIOTTI, O. **Towards Ontological Engineering: a Process for Building a Domain Ontology From Scratch in Public Administration**. Expert Systems, 25(5):484–503, 2008.

[24] CANNATARO, M; TALIA, D. **Semantics and Knowledge Grids: Building the Next-Generation Grid**. Intelligent Systems, IEEE, 19(1):56–63, Jan-Feb 2004.

[25] CANTELE, R. C; ADAMATTI, D. F; FERREIRA, M. A. G. V; S., S. J. **Aplicação da Reengenharia de Software na Construção Acelerada de Ontologias**. In: Friedrich, L. F, editor, Proceedings of the Simpósio Brasileiro de Sistemas de Informação SBSI 2005. SBC, Outubro 2005.

[26] CARAVANTES, G. R; LEDUR, P. F. **Leitura Dinâmica E Aprendizagem Aprimorando Sua Eficácia Como Leitor**. Editora AGE Ltda, 1 edition, 2003.

[27] CARROLL, J. J; DE ROO, J. **OWL Web Ontology Language Test Cases**. World Wide Web Consortium, Recommendation REC-owl-test-20040210, February 2004.

[28] CARROLL, J. J; DICKINSON, I; DOLLIN, C; REYNOLDS, D; SEABORNE, A; WILKINSON, K. **Jena: Implementing the Semantic Web Recommendations**. In: WWW Alt. '04: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, p. 74–83. ACM Press, 2004.

[29] CHANDRASEKARAN, B; JOSEPHSON, J. R; BENJAMINS, V. R. **What are Ontologies, and Why do We Need Them?** Intelligent Systems and Their Applications, IEEE [see also IEEE Intelligent Systems], 14(1):20–26, 1999.

[30] CHERVENAK, A; FOSTER, I; KESSELMAN, C; SALISBURY, C; TUECKE, S. **The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets**. Journal of Network and Computer Applications, 23:187–200, 2001.

[31] CONNOLLY, D; VAN HARMELEN, F; HORROCKS, I; MCGUINNESS, D. L; PATEL-SCHNEIDER, P. F; STEIN, L. A. **DAML+OIL (March 2001) Reference Description**. Technical report, W3C, 2001.

[32] CORAZZON, R. **Definitions of Ontology by Leading Philosophers**. In: Proceedings of the Raul Corazzon. Theory and History of Ontology. A Resource Guide for Philosophers, p. . , 2008.

[33] CORCHO, O; LÓPEZ, M. F; PEREZ, A. G. **Methodologies and Methods for Building Ontologies**. In: In: Oscar Corcho, Mariano Fernández López, Asuncion Gomez Perez - Ontological Engineering, p. 107–153, New York, 2003. Springer-Verlag New York, LLC.

[34] CORCHO, O; LÓPEZ, M. F; PÉREZ, A. G. **Methodologies, Tools and Languages for Building Ontologies: Where is Their Meeting Point?** Data Knowl. Eng., 46(1):41–64, July 2003.

[35] DA SILVA, D; SOUZA, R; ALMEIDA, M. **Ontologias e Vocabulários Controlados: Comparação de Metodologias para Construção**. Ciência da Informação, 37(3), 2008.

[36] DA SILVA, J. M. **Sinapse - Uma Metodologia para Extração de Conhecimentos em Objetos Textuais Baseada em Conceito para o Português do Brasil**. Master's thesis, Instituto de Informática-INF, Universidade Federal de Goiás, UFG, 2007.

[37] DA SILVA E SILVA, F. J; LOPES, R. F; DE SOUSA, B. B; VIANA, A. E. B; DE SOUSA, S. A. **MAG, Um Middleware de Grade Baseado em Agentes: Estado Atual e Perspectivas Futuras**. In: Proceedings of the WCGA 2006: Anais do IV Workshop on Computational Grids and Applications. Simpósio Brasileiro de Redes de Computadores (SBRC2006), Curitiba, Maio 2006. Sociedade Brasileira de Computação.

[38] DAHLBERG, I. **Teoria do Conceito**. Ciência da Informação, 7(2):101–107, 1978.

[39] DANTAS, M. **Tecnologias de Redes de Comunicação e Computadores**. Editora Axcel Books, 1 edition, 2002.

[40] DARPA, I. E. O. **The DARPA Agent Markup Language Homepage**. http://www.daml.org/, último acesso em Janeiro de 2009, 2000.

[41] DE CAMARGO, R. Y; GOLDCHLEGER, A; CARNEIRO, M; KON, F. **Pattern Languages of Program Design 5**, chapter The Grid Architectural Pattern: Leveraging Distributed Processing Capabilities, p. 337–356. Software Pattern Series. Addison-Wesley, 2006.

[42] DE CAMARGO, R. Y. **Armazenamento Distribuído de Dados e Checkpointing de Aplicações Paralelas em Grades Oportunistas**. PhD thesis, Departamento de Ciência da Computação do IME/USP, May 2007.

[43] DE OLIVEIRA CHISHMAN, R. L; DA ROSA ALVES, I. M; BERTOLDI, A. **O Conhecimento Semântico Representado em Ontologias Aplicadas à Busca e Extração de Informações na Web.** In: , p. . XI Simpósio Nacional de Letras e Lingüística / I Simpósio Internacional de Letras e Lingüística (SILEL), 2006.

[44] DE PAULA NASCIMENTO, A. **Conhecendo as Grades Computacionais**. CER - Revista Eletrônica de Computação, 1(1), 2006.

[45] DE RIBAMAR BRAGA PINHEIRO JÚNIOR, J; VIDAL, A. C. T; KON, F. **Repositório Seguro de Aplicações Baseado em GSS**. In: Proceedings of the Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, Florianópolis, 2005. SBC.

[46] DE ROURE, D; JENNINGS, N. R; SHADBOLT, N. R. **Research Agenda for the Semantic Grid: A Future e-Science Infrastructure**. Technical Report UKeS-2002-02, National e-Science Centre, December 2001.

[47] DE ROURE, D; JENNINGS, N; SHADBOLT, N. **The Semantic Grid: A future e-Science Infrastructure**. In: Berman, F; Fox, G; Hey, A. J. G, editors, Grid Computing - Making the Global Infrastructure a Reality, p. 437–470. John Wiley and Sons Ltd., 2003.

[48] DE SOUSA, B. B. **Um Serviço de Metadados Integrado ao Middleware de Grade MAG**. Master's thesis, PPGEE/UFMA, July 2006.

[49] DE SOUSA, B. B; DA SILVA E SILVA, F. J; TEIXEIRA, M. M; FILHO, G. C. **MagCat: An Agent-based Metadata Service for Data Grids**. In: Proceedings of the AgentGrid2006: 4th International Workshop on Agent Based Grid Computing. IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'06), Singapore, May 2006. IEEE Computer Society Press. to appear.

[50] DEVEDZIć, V. **Understanding Ontological Engineering**. Commun. ACM, 45(4):136–144, April 2002.

[51] DICKINSON, I. **Jena Ontology API**. http://jena.sourceforge.net/ontology/index.html, último acesso em Janeiro de 2009, 2008.

[52] DJURIC, D; GASEVIC, D; DEVEDZIC, V. **Ontology Modeling and MDA**. Journal of Object Technology, 4(1):109–128, 2005.

[53] DODDS, L. **Introducing SPARQL: Querying the Semantic Web**. http://www.xml.com/pub/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html?CMP=OTC-TY3388567169, último acesso em Janeiro de 2009, Nov 2006.

[54] DRUMOND, L. R; GIRARDI, R. **Uma Análise das Técnicas e Ferramentas para o Desenvolvimento de Aplicações para a Web Semântica**. REIC - Revista Eletrônica de Iniciação Científica, 6(1), 2006.

[55] E PARSIA LLC, C. **Pellet: The Open Source OWL DL Reasoner**. http://clarkparsia.com/pellet, último acesso em Janeiro de 2009, 2007.

[56] FARQUHAR, A; FIKES, R; RICE, J. **The Ontolingua Server: a Tool for Collaborative Ontology Construction**. In: Proceedings of the International Journal of Human-Computer Studies, 1996.

[57] FERNANDEZ, M; GOMEZ-PEREZ, A; JURISTO, N. **METHONTOLOGY: From Ontological Art Towards Ontological Engineering**. In: Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, p. 33–40, Stanford, USA, March 1997.

[58] FERNÁNDEZ-LÓPEZ, M; GÓMEZ-PÉREZ, A. **Overview and Analysis of Methodologies for Building Ontologies**. Knowledge Engineering Review, 17(2):129–156, 2002.

[59] FIKES, R; HAYES, P; HORROCKS, I. **OWL-QL: A Language for Deductive Query Answering on the Semantic Web**. Technical Report KSL 03-14, Stanford University, Stanford, CA, 2003.

[60] FOSTER, I; KESSELMAN, C. **The Grid 2: Blueprint for a New Computing Infrastructure (The Morgan Kaufmann Series in Computer Architecture and Design)**. Morgan Kaufmann, November 2003.

[61] FOSTER, I; KESSELMAN, C; TUECKE, S. **The Anatomy of the Grid: Enabling Scalable Virtual Organizations**. Lecture Notes in Computer Science, 2150, 2001.

[62] FOX, M. S. **The TOVE Project Towards a Common-Sense Model of the Enterprise**. In: IEA/AIE '92: Proceedings of the 5th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, p. 25–34, London, UK, 1992. Springer-Verlag.

[63] FREITAS, A. L; BARBOSA, A. A; GREVE, F. **LiveOurGrid: Estimulando o Uso de Grades Computacionais Através da Experimentação**. In: IN: PROCEEDINGS OF THE IS DA VI ESCOLA REGINAL DE COMPUTAçãO BAHIA-SERGIPE. SBC, Aracaju, 2006.

[64] FROMM, K. R. **UMBC eBiquity: Resource: Introducing Semantic Technologies and the Vision of the Semantic Web**. Semantic Interoperability Community of Practice (SICoP), 2005.

[65] GAAEVIC, D; DJURIC, D; DEVEDZIC, V; SELIC, B. **Model Driven Architecture and Ontology Development**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[66] GENESERETH, M. **Knowledge Interchange Format**. In: Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, p. 238–249. Morgan Kaufman Publishers, 1991.

[67] GENESERETH, M; FIKES, R. **Knowledge Interchange Format, Version 3.0 Reference Manual**. Technical report, Computer Science Department, Stanford University, 1992.

[68] GENESERETH, M. R; NISLSSON, N. J. **Logical Foundations of Artificial Intelligence**. Morgan Kaufmann Publishers, September 1987.

[69] GENNARI, J. H; MUSEN, M. A; FERGERSON, R. W; GROSSO, W. E; CRUBÉZY, M; ERIKSSON, H; NOY, N. F; TU, S. W. **The Evolution of Protégé: An Environment for Knowledge-Based Systems Development**. International Journal of Human-Computer Studies, 58:89–123, 2002.

[70] GOBLE, C; DE ROURE, D; SHADBOLT, N; FERNANDES, A. **Enhancing Services and Applications with Knowledge and Semantics**. In: Foster, I; Kesselman, C, editors, The Grid 2: Blueprint for a New Computing Infrastructure, p. 431–458. Morgan-Kaufmann, 2004.

[71] GOBLE, C. **Towards a Semantic Grid Architecture**. In: Goble, C; Kesselman, C; Sure, Y, editors, Semantic Grid: The Convergence of Technologies, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[72] GOLDCHLEGER, A. **InteGrade: Um Sistema de Middleware para Computação em Grade Oportunista (InteGrade: a Middleware System for Opportunistic Grid Computing)**. Master's thesis, Department of Computer Science - University of São Paulo, Dez 2004. In Português.

[73] GOLDCHLEGER, A; KON, F; GOLDMAN, A; FINGER, M; BEZERRA, G. C. **InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines**. Concurrency and Computation: Practice and Experience, 16(5):449–459, March 2004.

[74] GOLDCHLEGER, A; KON, F; LEJBMAN, A. G. V; FINGER, M; SONG, S. W. **InteGrade: Rumo a um Sistema de Computação em Grade para Aproveitamento de Recursos Ociosos em Máquinas compartilhadas**. Technical report, MAC-IME-USP 2002-08, Outubro 2002.

[75] GOMES, D; DA SILVA E SILVA, F. J; ENDLER, M. **Integrando Dispositivos Móveis ao Middleware Integrade**. In: PROCEEDINGS OF THE I WORKSHOP ON PERVASIVE AND UBIQUITOUS COMPUTING, WPUC 2007, October 2007.

[76] GOMEZ-PEREZ, A; ANGELE, J; FERNANDEZ-LOPEZ, M; CHRISTOPHIDES, V; STUTT, A; SURE, Y. **A Survey on Ontology Tools**. OntoWeb deliverable 1.3, Universidad Politecnia de Madrid, 2002.

[77] GOMEZ-PEREZ, A; FERNANDEZ-LOPEZ, M; CORCHO, O. **Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web, 1st Edition**. Springer-Verlag, Heidelberg, 2004.

[78] GROUP, D. I. **DIG Interface**. http://dl.kr.org/dig/interface.html, último acesso em junho de 2009, 2006.

[79] GRUBER, T. R. **The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases**. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, p. 601–602. Morgan Kaufmann, 1991.

[80] GRUBER, T. R. **A Translation Approach to Portable Ontology Specifications**. Knowledge Acquis., 5(2):199–220, June 1993.

[81] GRUBER, T. R. **Towards Principles for the Design of Ontologies Used for Knowledge Sharing**. Int. J. Hum.-Comput. Stud., 43(5-6):907–928, 1995.

[82] GRÜNINGER, M; FOX, M. **Methodology for the Design and Evaluation of Ontologies**. In: IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995, 1995.

[83] GRUNINGER, M; FOX, M. **The Logic of Enterprise Modelling**. In: Modelling and Methodologies for Enterprise Integration, In J. Brown and D. O'Sullivan, Cornwall, Great Britain: Chapman and Hall, 1995.

[84] GUARINO, N. **Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy**. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1998.

[85] GUARINO, N. **Formal Ontology and Information Systems**. In: , p. 3–15. IOS Press, 1998.

[86] HAASE, P; BROEKSTRA, J; EBERHART, A; VOLZ, R. **A Comparison of RDF Query Languages**. In: , p. 502–517. , 2004.

[87] HORRIDGE, M. **A Practical Guide To Building OWL Ontologies With The Protege-OWL Plugin**. University of Manchester, 1 edition, June November 2004.

[88] HORRIDGE, M; KNUBLAUCH, H; RECTOR, A; STEVENS, R; WROE, C. **A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0**. The Univerrityof Manchester and Stanford University, August 2004.

[89] HORROCKS, I. **The FaCT System**. http://www.cs.man.ac.uk/~horrocks/FaCT/, último acesso em junho de 2009, 2003.

[90] IEEE, C. S. **IEEE Standard Glossary of Software Engineering Terminology**. IEEE Std 610.121990, 1990.

[91] KALFOGLOU, Y; KALFOGLOU, Y; DOMINGUE, J; DOMINGUE, J; MOTTA, E; MOTTA, E; VARGAS-VERA, M; VARGAS-VERA, M; SHUM, S. B; SHUM, S. B. **MyPlanet: an Ontology-Driven Web-Based Personalised news Service**. In: Proceedings of the IJCAI'01 Workshop on Ontologies and Information Sharing, p. 44–52, 2001.

[92] KARP, P. D; CHAUDHRI, V. K; THOMERE, J. **XOL: An XML-Based Ontology Exchange Language**. Technical report, SRI International, 1999.

[93] KARVOUNARAKIS, G; ALEXAKI, S; CHRISTOPHIDES, V; PLEXOUSAKIS, D; VOUTON, F. V. **Rql: a Declarative Query Language for Rdf**. In: , p. 592–603. ACM Press, 2002.

[94] KESSELMAN, C; FOSTER, I. **The Grid: Blueprint for a New Computing Infrastructure**. Morgan Kaufmann Publishers, November 1998.

[95] KIFER, M; LAUSEN, G; WU, J. **Logical Foundations of Object-Oriented and Frame-Based Languages**. Journal of ACM, 42:741–843, July 1995.

[96] KNUBLAUCH, H. **Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protege/OWL**. In: Frankel, D. S; Kendall, E. F; McGuinness, D. L, editors, PROCEEDINGS OF THE 1ST INTERNATIONAL WORKSHOP ON THE MODEL-DRIVEN SEMANTIC WEB (MDSW2004), 2004.

[97] KON, F; GOLDMAN, A. **Grades Computacionais: Conceitos Fundamentais e Casos Concretos**. In: Proceedings of the Tomasz Kowaltowski e Karin Breitman (Org.) Atualizações em Informática 2008, p. 55–104, SBC, 2008. Editora PUC-Rio.

[98] KRAUTER, K; BUYYA, R; MAHESWARAN, M. **A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing**. Software Practice and Experience, 32(2):135–164, 2002.

[99] LASSILA, O; SWICK, R. R. **Resource Description Framework (RDF) - Model and Syntax Specification**. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, último acesso em Janeiro de 2009, 1999.

[100] LAVE, J; WENGER, E. **Situated Learning : Legitimate Peripheral Participation**. Cambridge University Press, 1 edition, September 1991.

[101] LEITE, J. C. S. D. P; FRANCO, A. P. M. **A Strategy for Conceptual Model Acquisition**. In: IEEE International Symposium on Requirements Engineering, p. 243–246. IEEE Computer Society, 1993.

[102] LEITE, J. C. S. D. P; ROSSI, G; BALAGUER, F; MAIORANA, V; KAPLAN, G; HADAD, G; OLIVEROS, A. **Enhancing a Requirements Baseline with Scenarios**. In: RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, p. 44, Washington, DC, USA, 1997. IEEE Computer Society.

[103] LENAT, D. B; GUHA, R. V. **Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[104] LÉGER, A; MICHEL, G; BARRETT, P; GITTON, S; GOMÉZ-PÉREZ, A; LEHTOLA, A; MOKKILA, K; RODRIGEZ, S; SALLANTIN, J; VARVARIGOU, T; VINESSE, J. **Ontology Domain Modeling Support for Multi-Lingual Services**. In: PROCEEDINGS OF THE MKBEEM. ECAI'00 WORKSHOP ON APPLICATIONS OF ONTOLOGIES, p. 25, 2000.

[105] LOPES, A. V. **Introdução à Programação com Ada 95**. Editora da ULBRA, RS, 1 edition, 1997.

[106] LOPES, J. G. R. C. **Matching Semântico de Recursos Computacionais em Ambientes de Grade com Múltiplas Ontologias**. Master's thesis, Universidade de Brasília, Unb, 2005.

[107] LÓPEZ, F. **Overview Of Methodologies For Building Ontologies**. In: Proceedings of the WORKSHOP ON ONTOLOGIES AND PROBLEM-SOLVING METHODS: Lessons Learned and Future Trends, p. 4–1,4–13, 1999.

[108] LÓPEZ, M. F; GÓMEZ-PÉREZ, A; SIERRA, J. P; SIERRA, A. P. **Building a Chemical Ontology Using Methontology and the Ontology Design Environment**. IEEE Intelligent Systems, 14(1):37–46, 1999.

[109] LUKE, S; HEFLIN, J. **SHOE 1.01: Proposed Specification**. University of Maryland, 2000.

[110] MACGREGOR, R. M. **Using a Description Classifier to Enhance Deductive Inference**. In: Proceedings of the Seventh IEEE Conference on AI Applications, p. 141–147, 1991.

[111] MAEDCHE, A. **Ontology Learning for the Semantic Web (The Kluwer International Series in Engineering and Computer Science, Volume 665)**. Springer, February 2002.

[112] MAEDCHE, E; STAAB, S; STOJANOVIC, N; STUDER, R. **Semantic Portal - The SEAL Approach**. In: Proceedings of the Spinning the Semantic Web, p. 317–359. MIT Press, 2003.

[113] MATOUSEK, K; UHL?, J. **On Representing Uncertain Historical Time**. Database and Expert Systems Applications, International Workshop on, 0:105–109, 2004.

[114] MCCARTHY, P. **Search RDF data with SPARQL**. http://www-128.ibm.com/developerworks/xml/library/j-sparql/, último acesso em Janeiro de 2009, May 2005.

[115] MCGUINNESS, D. L; FIKES, R; RICE, J; WILDER, S. **The Chimaera Ontology Environment.** In: AAAI/IAAI, p. 1123–1124. AAAI Press / The MIT Press, 2000.

[116] MINSKY, M. **A Framework for Representing Knowledge**. In: Haugeland, J, editor, MIND DESIGN: PHILOSOPHY, PSYCHOLOGY, ARTIFICIAL INTELLIGENCE, p. 95–128. MIT Press, Cambridge, MA, 1981.

[117] MIZOGUCHI, R. **Tutorial on Ontological Engineering: Part 2: Ontology Development, Tools and Languages**. New Generation Comput., 22(1), 2003.

[118] MORENO, A. O; HERNANDEZ, C. P. **Reusing the Mikrokosmos Ontology for Concept-based Multilingual Terminology Databases**. In: Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000), Athens, Greece, maio 2000.

[119] MOTTA, E. **An Overview of the OCML Modelling Language**. In: Proceedings of the KEML'98: 8th Workshop on Knowledge Engineering Methods e Languages, p. 21–22, 1998.

[120] MYLOPOULOS, J; BORGIDA, A; JARKE, M; KOUBARAKIS, M. **Telos: Representing Knowledge About Information Systems**. ACM Transactions on Information Systems, 8:325–362, 1990.

[121] NASSIF, L. N. **Seleção Distribuída de Recursos em Grades Computacionais Usando Raciocínio Baseado em Casos e Políticas de Granularidade Fina**. PhD thesis, Universidade Federal de Minas Gerais, UFMG, 2006.

[122] NOY, N. F; MCGUINNESS, D. L. **Ontology Development 101: A Guide to Creating Your First Ontology**. Stanford University, November 2008.

[123] NOY, N. F; MUSEN, M. A. **PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment**. In: AAAI/IAAI, p. 450–455, 2000.

[124] OF ELECTRICAL, I; IEEE, E. E. **IEEE 1074-2006 - IEEE Standard for Developing a Software Project Life Cycle Process**. Thomson Reuters, Jan 2006.

[125] OF SOUTHERN CALIFORNIA'S INFORMATION SCIENCES INSTITUTE, U. **Loom Project Home Page**. http://www.isi.edu/isd/LOOM/LOOM-HOME.html, último acesso em Janeiro de 2009, 2006.

[126] OF TECHNOLOGY, D. U. **CGXML**. http://tockit.sourceforge.net/cgxml/index.html, último acesso em junho de 2009, 2002.

[127] ONTOPRISE. **Ontostudio**. http://www.ontoprise.de/, último acesso em Janeiro de 2009, 2007.

[128] ORLEAN, D. A. **Um Processo Unificado para Engenharia de Ontologias**. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, Rio de Janeiro, RJ, Ago 2003.

[129] PATEL-SCHNEIDER, P. F; HORROCKS, I. **Position Paper: a Comparison of Two Modelling Paradigms in the Semantic Web**. In: WWW '06: PROCEEDINGS OF THE 15TH INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, p. 3–12, New York, NY, USA, 2006. ACM.

[130] PÉREZ, A. G; FERNÁNDEZ, M; ET. **Towards a Method to Conceptualize Domain Ontologies**. In: Proceedings of the Workshop on Ontological Eng. (part of ECAI '96: 1996 European Conf. AI). European Coordinating Committee for Artificial Intelligence, 1996.

[131] PERNAS, A. M; DANTAS, M. A. R. **Ontologias Aplicadas à Descrição de Recursos em Ambientes Grid**. Revista Infocomp, 3(2), 2004.

[132] PINHEIRO, F. J. R. **Uma Proposta de um Sistema de Imagem Única para uso de Computação em Grade em Organizações Virtuais**. Master's thesis, Universidade Federal de Santa Catarina, Fundação de Amparo à Pesquisa do Estado de Alagoas, Dez 2005.

[133] PRESSMAN, R. S. **Engenharia de Software**. MCGRAW-HILL, 5 ed. edition, 2002.

[134] PRUD'HOMMEAUX, E; SEABORNE, A. **SPARQL Query Language for RDF**. Technical report, World Wide Web Consortium, January 2008.

[135] QUINE, W. V. O. **On What There Is**. Review of Metaphysics, 2:21–38, 1948/1953.

[136] REED, S. L; LENAT, D. B. **Mapping Ontologies Into Cyc**. In: Proceedings of the AAAI 2002 Conference Workshop on Ontologies for the Semantic Web, Edmonton, Canada, JUL 2002. Cycorp, Inc.

[137] ROCHA, V; NETTO, M. A. S; KON, F. **Descoberta de Recursos em Grades Computacionais Utilizando Estruturas P2P**. In: PROCEEDINGS OF THE 24TH BRAZILIAN SYMPOSIUM ON COMPUTER NETWORKS (SBRC), p. 913–928, Curitiba, Brasil, May 2006.

[138] ROCHA, V. M. **Protocolos Par-a-Par para Interligação de Aglomerados em Grades Computacionais**. Master's thesis, Instituto de Matemática e Estatística - Universidade de São Paulo, Dez 2005. In Português.

[139] RODOSEK, G. D. **Service Management Platform: The Next Step in Management Tools**. In: DSOM '00: Proceedings of the 11th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, p. 59–70, London, UK, 2000. Springer-Verlag.

[140] ROURE, D. D. **Semantic Grid Community Portal**. http://www.semanticgrid.org/, último acesso em Novembro de 2007, 2006.

[141] ROURE, D. D. **A Brief History of the Semantic Grid**. In: Goble, C; Kesselman, C; Sure, Y, editors, Semantic Grid: The Convergence of Technologies, Dagstuhl Seminar Proceedings. Internationales Begegnungs - und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[142] ROURE, D. D; BAKER, M. A; JENNINGS, N. R. **The Evolution of the Grid**. In: Grid Computing: Making the Global Infrastructure a Reality, p. 65–100. John Wiley & Sons, 2003.

[143] RUSSELL, S. J; NORVIG, P. **Artificial Intelligence: A Modern Approach**. Pearson Education, 2003.

[144] SANTOS, J; VALE, Z; RAMOS, C; FERNANDES, M; ROSADO, C; MARQUES, A. **Aplicações Inteligentes em Centros de Controlo: Verificação e Validação**. 5as Jornadas Hispano-Lusas de Ingeniería Eléctrica, July 1997.

[145] SCHREIBER, G; WIELINGA, B; AKKERMANS, H; VAN DE VELDE, W. **CML: The CommonKADS Conceptual Modelling Language**. Technical report, University of Amsterdam, 1994.

[146] SILVA, A. P. C; DA SILVA DIAS, J; DANTAS, M. A. R. **Selecionador de Recursos Grade Baseado na Integração Semântica de Múltiplas Ontologias**. In: Proceedings of the V Workshop de Computação em Grade e Aplicações (WCGA), 2007, Belém, Belém - PA, 2007. Proceedings of the V Workshop de Computação em Grade e Aplicações.

[147] SINTEK, M; DECKER, S. **TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web**. In: ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web, p. 364–378, London, UK, 2002. Springer-Verlag.

[148] SINTEK, M; DECKER, S. **TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web**. In: ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web, p. 364–378, London, UK, 2002. Springer-Verlag.

[149] SOUZA, R. R. **Uma Proposta de Metodologia para Escolha Automática de Descritores Utilizando Sintagmas Nominais**. PhD thesis, Escola de Ciência da Informação, UFMG, Belo Horizonte, MG, 2005.

[150] SOWA, J. F. **Knowledge Representation: Logical, Philosophical, and Computational Foundations**. Brooks/Cole, August 2000.

[151] STUDER, R; BENJAMINS, R; FENSEL, D. **Knowledge Engineering: Principles and Methods**. Data & Knowledge Engineering, 25(1-2):161–198, MAR 1998.

[152] SU, X; ILEBREKKE, L. **A Comparative Study of Ontology Languages and Tools**. In: Proceedings of the Advanced Information System Engineering, 14th International Conference, CAiSE 2002, p. 761–765. Springer Verlag, 2002.

[153] SURE, Y; STUDER, R. **A Methodology for Ontology-Based Knowledge Management**. In: Davies, D. J; Fensel, P. D; van Harmelen, P. F, editors, Towards the Semantic Web, p. 33–46. John Wiley & Sons, Ltd, , 2003.

[154] SWARTOUT, B; RAMESH, P; KNIGHT, K; RUSS, T. **Toward Distributed Use of Large-Scale Ontologies**. AAAI Symposium on Ontological Engineering, 1997.

[155] SWARTOUT, W; TATE, A. **Guest Editors' Introduction: Ontologies**. IEEE Intelligent Systems, 14(1):18–19, 1999.

[156] TA, F. **Collaboratories**. Annual Review of Information Science and Technology (ARIST), 36, 2002.

[157] TAUBERER, J. **What Is RDF**. http://www.xml.com/pub/a/2001/01/24/rdf.html, último acesso em Janeiro de 2009, July 2006.

[158] THAIN, D; TANNENBAUM, T; LIVNY, M. **Condor and the Grid**. In: Berman, F; Fox, G; Hey, A, editors, Proceedings of the Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc., April 2003.

[159] TRONCY, R; ISAAC, A; MALAISÉ, V. **Using XSLT for Interoperability: DOE and The Travelling Domain Experiment.** In: Sure, Y; Óscar Corcho, editors, EON, volume 87 de **CEUR Workshop Proceedings**. CEUR-WS.org, 2003.

[160] UEDA, K. **Guarded Horn Clauses: A Parallel Logic Programming Language with the Concept of a Guard**. In: Programming of Future Generation Computers, p. 441–456. North, 1987.

[161] USCHOLD, M; JASPER, R. **A Framework for Understanding and Classifying Ontology Applications**. In: Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW 99, 1999.

[162] USCHOLD, M. **Building Ontologies: Towards a Unified Methodology**. Edinburgh, Scotland, 1996.

[163] USCHOLD, M; GRÜNINGER, M. **Ontologies: Principles, Methods, and Applications**. Knowledge Engineering Review, 11(2):93–155, 1996.

[164] USCHOLD, M; KING, M. **Towards a Methodology for Building Ontologies**. In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, Held in Conjunction with IJCAI-95, 1995.

[165] VALENTEM, P; PIMENTA, A; DOS SANTOS, D. M. B; NETO, M. M. **Uma Solução Software Livre para Mineração de Dados em Grades Computacionais**. In: Proceedings of the I Concurso de Trabalhos Acadêmicos em Software Livre, 2005, Florianópolis, Florianópolis, 2005. III Congresso Catarinense de Software Livre - SOLISC.

[166] VAN HEIJST, G; SCHREIBER, A. T; WIELINGA, B. J. **Using Explicit Ontologies in KBS Development**. Int. J. Hum.-Comput. Stud., 46(2-3):183–292, 1997.

[167] VIDAL, A. C. T. **Abordagem Semântica Aplicada à Integração e Gerenciamento de Recursos e Aplicações em Grades Computacionais**. PhD thesis, Escola Politécnica da Universidade de São Paulo, 2007.

[168] VIEIRA, R; DOS SANTOS, D. A; DA SILVA, D. M; SANTANA, M. R. **Web Semântica: Ontologias, Lógica de Descrição e Inferências**. In: Goble, C; Kesselman, C; Sure, Y, editors, Proceedings of the Cesar Teixeira; Eduardo Barrere; Iran Abraão. (Org.), volume 1 Ed.,1, p. 127–167. Web e Multimidia: Desafios e Soluções (WebMedia 2005 - Minicursos), 2005.

[169] WEISZFLOG, W. **Michaelis Moderno Dicionário Da Língua Portuguesa**. Melhoramentos, 1 edition, 2004.

[170] WIVES, L. K. **Utilizando Conceitos como Descritores de Textos para o Processo de Identificação de Conglomerados (Clustering) de Documentos.** PhD thesis, Universidade Federal do Rio Grande do Sul - Brasil, 2004.

[171] ZAVAGLIA, C. **Produção de Ontologias Específicas: a modelagem da OntoEco**. GEL: Grupo de Estudos Lingüísticos do Estado de São Paulo, XXXIV:1182–1187, 2005.

[172] ZHUGE, H. **The Knowledge Grid**. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2004.

APÊNDICE A

Relatório Léxicos

Neste apêndice, é mostrado o modelo de duas páginas de um dos relatórios elaborados com os dados obtidos pela varredura nos documentos do domínio proposto neste trabalho. Também é apresentado um relatório com a soma das frequências totais das palavras extraídas pelo Sinapse.

No relatório "*Relatórios Léxicos*", a coluna "Palavra" refere-se aos termos retirados dos documentos analisados. A coluna "Stem" refere-se forma base ou raiz linguística de cada palavra. A coluna "Frequência" refere-se à quantidade de vezes que esta palavra apareceu em todos os documentos analisados. A coluna "CG" refere-se à classe gramatical da palavra, sendo "S" para substantivo, "V" para verbo, "A" para advérbios, "X" e "I" para as outras classes. A coluna "Sigla" marca que a palavra pode ser uma sigla. A coluna "NP" marca que a palavra pode ser um nome próprio.

No relatório "*Relatório dos Totais das Palavras de cada classe gramatical extraídas pelo Sinapse*", é apresentado o total de frequência dos substantivos, verbos, advérbios, das outras classes (indeterminadas e desconhecidas) e um total geral de palavras extraídas dos documentos do corpus.

~~Relatório Léxicos~~

Palavra	Stem	Frequência	CG	Sigla	NP
abordagem	abord	2929	S		
abordagens	abord	830	S		
abordado	abord	29	S		
abordam	abord	40	V		
aborda	abord	124	X		
abordados	abord	103	S		
abordadas	abord	40	S		
abordar	abord	25	V		
abordavam	abord	1	V		
abordando	abord	8	V		
abordaremos	abord	27	V		
abordaram	abord	11	V		
abordada	abord	60	S		
abordará	abord	11	V		
abordamos	abord	36	V		
	Total Stems	**4274**			
Aborda	aborda	2	I		x
	Total Stems	**2**			
Abordadas	abordadas	13	I		x
	Total Stems	**13**			
Abordagem	abordagem	345	I		x
	Total Stems	**345**			
Abordagens	abordagens	83	I		x
	Total Stems	**83**			
Abordaremos	abordaremos	5	I		x
	Total Stems	**5**			
abortada	abort	6	S		
abortará	abort	13	V		
	Total Stems	**19**			
about	about	75	X		
	Total Stems	**75**			
above	abov	29	X		
	Total Stems	**29**			
ABox	abox	169	I		x
	Total Stems	**169**			
abriu	abr	18	V		
abre	abr	14	X		

~~Relatório Léxicos~~

Palavra	Stem	Frequência	CG	Sigla	NP
abrirem	abr	11	V		
abrir	abr	43	V		
abrirá	abr	19	V		
	Total Stems	**105**			
Abraham	abraham	15	I		x
	Total Stems	**15**			
Abramson	abramson	30	I		x
	Total Stems	**30**			
abrangência	abrang	127	S		
abrangente	abrang	202	S		
abrange	abrang	2	X		
	Total Stems	**331**			
Abre-se	abre-se	19	I		x
	Total Stems	**19**			
abreviação	abrevi	11	S		
abreviações	abrevi	15	S		
	Total Stems	**26**			
ABREVIATURAS	abreviaturas	2	I	x	
Abreviaturas	abreviaturas	34	I		x
	Total Stems	**36**			
abriga	abrig	1	X		
	Total Stems	**1**			
abril	abril	11	X		
	Total Stems	**11**			
aBrot	abrot	4	X		
	Total Stems	**4**			
abscissas	absciss	17	X		
	Total Stems	**17**			
absoluta	absolut	10	X		
	Total Stems	**10**			
absorve	absorv	1	X		
	Total Stems	**1**			
abstração	abstr	190	S		
abstrações	abstr	115	S		
abstrai	abstr	29	V		
	Total Stems	**334**			

Relatório dos Totais das Palavras de cada classe gramatical extraídas pelo Sinapse

Substantivos	Verbos	Advérbios	Indeterminados	Desconhecidos
581606	229872	14143	389976	821015

Total Geral **2036612**

APÊNDICE B

Detalhamento da Ontologia

Este apêndice apresenta uma visão sucinta sobre taxonomias e conceitos relevantes no domínio de grades computacionais contidos na ontologia proposta nesta dissertação. No Capítulo 5, foi apresentada uma visão geral da ontologia para grades computacionais construída a partir da metodologia proposta. Aqui serão ilustradas as classes, suas hierarquias e propriedades e, mostrado o código da Ontologia.

B.1 Código da Ontologia

Abaixo segue o código da Ontologia em RDF/OWL gerado pela ferramenta Protégé.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1244934941.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
  xml:base="http://www.owl-ontologies.com/grade_computacional.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
    <owl:imports rdf:resource=""/>
  </owl:Ontology>
  <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Computador">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Maquina"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Metadados">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#RAM">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Memoria"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Componente_de_software">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#GRM">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Gerenciador"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Nó">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Computador"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Disco_rigido">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#OGSA">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquitetura"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Tudo que pode ser utilizado dentro de um Ambiente Computacional.</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório_de_aplicação">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Algoritmo">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Codigo"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Nucleo">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Componente_de_software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Conhecimento">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Buffer">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#RAM"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Espaço_de_armazenamento"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Memoria">
```

```
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#TCP">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Protocolo"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#LRM"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Software">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Artigo">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Texto"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Gerenciador">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Módulos do ambiente de grade computacional.</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Maquina">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Equipamento"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Corba">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#JVM">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Camada"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Middleware">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Comunidade">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Organização"/>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#BSP">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Algoritmo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Roteador">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Equipamento"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Instituição"/>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Checkpoint">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Serviço"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Documento">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquivo"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#PVM">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#OGSI">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquitetura"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquivo">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Grade_Computacional">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Administrador">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Indivíduo"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Codificacao"/>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Laboratório">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Aglomerado">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Biblioteca_de_funções">
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Proxy">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Componente_de_software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Mensagem">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Cache">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Memoria"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#LRM">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Gerenciador"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Equipamento">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Espaço_de_armazenamento">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Banco_de_dados">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Organização">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Instituição"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Globus_toolkit">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >É caracterizado como um conjunto de serviços para a construção de sistemas e aplicações de Grade.</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Cpu">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Processador">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
```

```
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Agente">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede_p2p">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#MPI">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquitetura"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Marca"/>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Impressora">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Equipamento"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Pilha">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Codigo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Vetor">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Codigo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Especialista">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Indivíduo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Usuário">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Indivíduo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Pacote">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Mensagem"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dicionario">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Plataforma">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Texto">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Documento"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Informação"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Conteudo"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema_operacional">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Padrões para construção de arquiteturas computacionais.</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Windows">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema_operacional"/>
</owl:Class>
```

```
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Protocolo">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Empresa">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Organização"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Resource_Broker">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Negociadores de Recursos que permitem que os usuários façam solicitações de alto nível, as quais serão traduzidas pe
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#ASCT">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Application Submission and Control Tool permite que um usuário submeta aplicações para serem executadas na Grade.</r
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Informação">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Imagem">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquivo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Internet">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Linguagem">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Codificacao"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Linux">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema_operacional"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Servidor">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Nó"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Serviço">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Conteudo">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Lista">
  <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Codigo"/>
</owl:Class>
<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Network"/>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_armazenado_por">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório_de_aplicação"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#armazena"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#disponibiliza">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
      <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#disponibiliza"/>
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#pode_ser">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Usuário"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Especialista"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Administrador"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#cria">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Especialista"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Serviço"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#recebe">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#LRM"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#realiza">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Usuário"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#utiliza">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#pertence_a">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Equipamento"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Instituição"/>
          <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#pertence_a"/>
        </owl:Restriction>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Instituição"/>
      </owl:unionOf>
    </owl:Class>
```

```
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#tem-">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Maquina"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquitetura"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#interconecta_por">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#recebe_dedicação_de">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#recebe_dedicação_de"/>
          <owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema"/>
        </owl:Restriction>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#faz_parte_de">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Instituição"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Usuário"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#permite">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#ASCT"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#define">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquitetura"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Corba"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#podeSer">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Laboratório"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#possuiUm">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema_operacional"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Nucleo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_feito_por">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Codigo"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_usado_por">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Usuário"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#armazena">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Disco_rigido"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Biblioteca_de_funções"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#é_armazenado_por"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#distribue">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Grade_Computacional"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#tem">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Equipamento"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Marca"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#tem_">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Metadados"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Dicionario"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#localiza">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_executado_em">
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#é_executado_em"/>
      <owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Maquina"/>
    </owl:Restriction>
  </rdfs:range>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#compartilha">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Aplicação"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#adiciona">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Grade_Computacional"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#possui">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Cpu"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Disco_rigido"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Memoria"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Processador"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_possuído_por"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Computador"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#deriva-de">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Conhecimento"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Informação"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#atua_como">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Serviço"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Network"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#oferece">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
        <owl:Restriction>
          <owl:allValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
          <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#oferece"/>
        </owl:Restriction>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Grade_Computacional"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_representado_por">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Algoritmo"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Codigo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#expressa_em">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Algoritmo"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Linguagem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#constitui-se-de">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Informação"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#possui_vários"/>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_alocado_por">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Sistema"/>
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_contido_por">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Disco_rigido"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Espaço_de_armazenamento"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#contém"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_armazenado_em">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Banco_de_dados"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#ativa">
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Mensagem"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Serviço"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#ocupa">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Repositório"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Espaço_de_armazenamento"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#pode_ser_um">
  <rdfs:subPropertyOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#pode_ser"/>
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Arquivo"/>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Software"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#contém">
  <owl:inverseOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#é_contido_por"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Banco_de_dados"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Hardware"/>
          <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#contém"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >1</owl:minCardinality>
          <owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#contém"/>
          <owl:valuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
        </owl:Restriction>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#gerencia">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
```

```
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#GRM"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#LRM"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#envia">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#GRM"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#LRM"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temEstado">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temTipo">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Rede"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temCaracterística">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Grade_Computacional"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Dado"/>
        <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temDesempenho">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temConfiguração">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Requisição"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temNatureza">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Grade_Computacional"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temLocal">
  <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Recurso"/>
</owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temPolíticaDeAcesso">
    <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Ambiente_computacional"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#temNome">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Marca"/>
          <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Instituição"/>
          <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Indivíduo"/>
          <owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Padrão_de_Middlware"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:InverseFunctionalProperty rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#é_possuído_por">
    <owl:inverseOf rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#possui"/>
    <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Middleware"/>
    <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1244934941.owl#Proxy"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#RDF"/>
  <Middleware rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#MobGrid"/>
  <Marca rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#IBM"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#RSL"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#C_plusplus"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#XML"/>
  <Marca rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Intel"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#RDQL"/>
  <Middleware rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#MyGrid"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#OIL"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#JAVA"/>
  <Marca rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Epson"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#IDL"/>
  <Middleware rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#OurGrid"/>
  <Marca rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#HP"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#UML"/>
  <Middleware rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#OppStore"/>
  <Linguagem rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#OWL"/>
  <Marca rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Xerox"/>
  <Marca rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#AMD"/>
  <Middleware rdf:about="http://www.owl-ontologies.com/Ontology1244934941.owl#Integrade"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.4, Build 533)  http://protege.stanford.edu -->
```

B.2 Ilustrações das Classes, suas SubClasses, Instâncias e Propriedades

Nesta Seção, serão ilustradas, detalhadamente, as classes que compõem a ontologia proposta. Foi utilizado o *plug-in* Jambalaya, que permite apresentar a ontologia de

maneira similar a um diagrama de entidade relacionamento, para criar os diagramas das classes e SubClasses detalhadamente.

A Figura B.1 apresenta uma visão de todas as super-classes.

Figura B.1: *Visão de todas as super classes da ontologia*

A Figura B.2 ilustra a classe Ambiente Computacional, suas SubClasses e Propriedades.

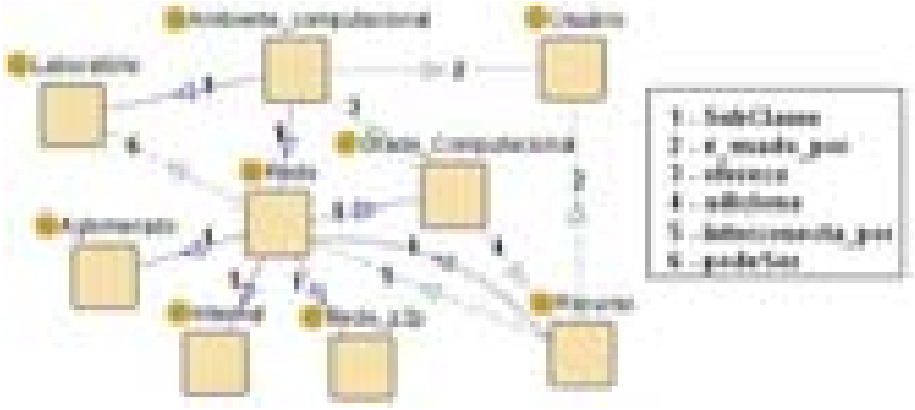

Figura B.2: *Classe Ambiente Computacional, suas SubClasses e Propriedades.*

A Figura B.3 ilustra as classes Padrão_de_Middlaware e Arquitetura, suas SubClasses e Propriedades.

A Figura B.4 ilustra as classes Conhecimento, Informação, Dado, Repositório, Espaço_de_armazenamento, suas SubClasses e Propriedades.

A Figura B.5 ilustra a classe Gerenciador, suas SubClasses e Propriedades.

A Figura B.6 ilustra as classes Hardware e Instituição, suas SubClasses e Propriedades.

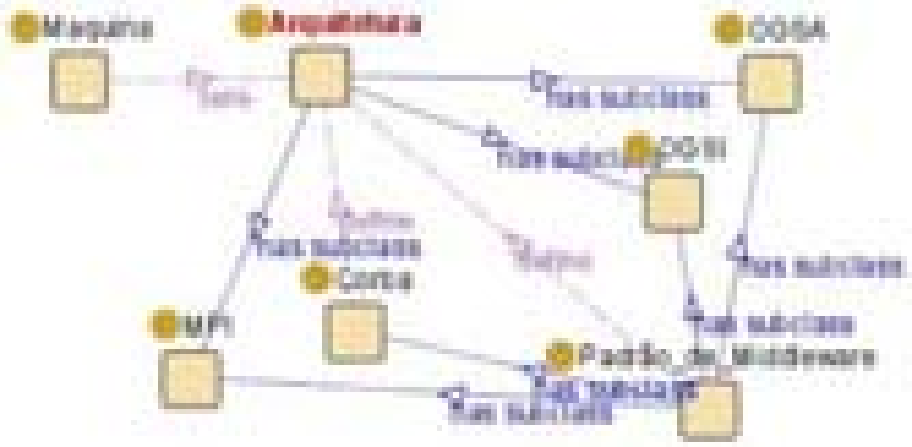

Figura B.3: *Classes Padrão_de_Middlaware e Arquitetura, suas SubClasses e Propriedades.*

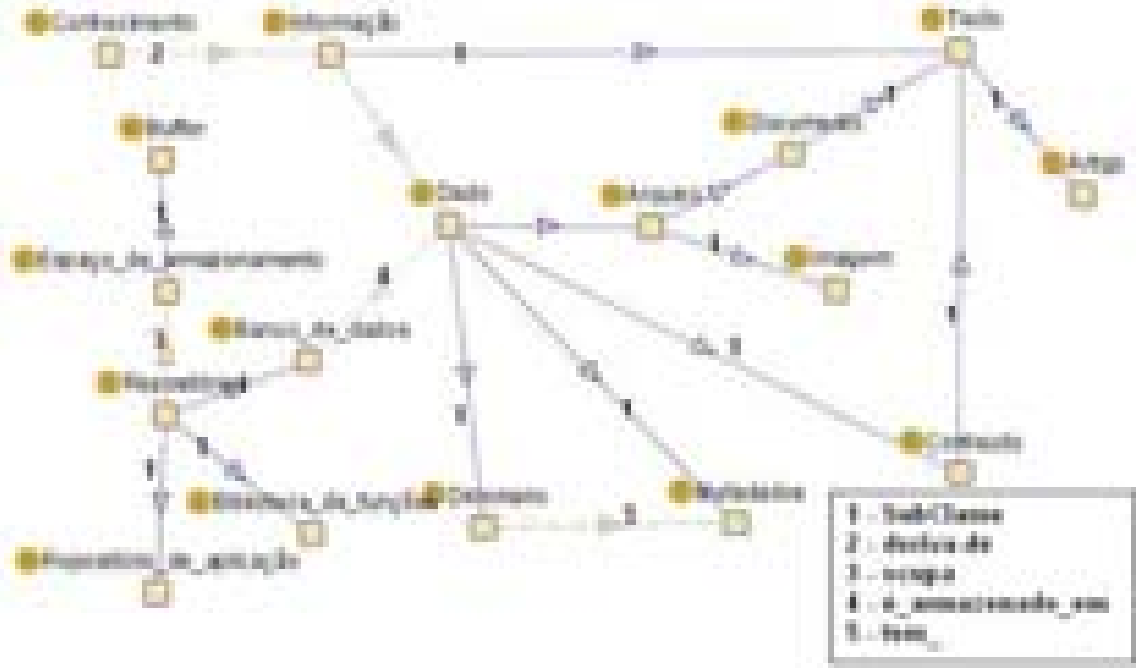

Figura B.4: *Classes Conhecimento, Informação, Dado, Repositório, Espaço_de_armazenamento, suas SubClasses e Propriedades.*

Figura B.5: *Classe Gerenciador, suas SubClasses e Propriedades.*

Figura B.6: *Classes Hardware e Instituição, suas SubClasses e Propriedades.*

A Figura B.7 ilustra as classes Indivíduo e Instituição, suas SubClasses e Propriedades.

A Figura B.8 ilustra as classes Codificação e Código, suas SubClasses, Instâncias e Propriedades.

A Figura B.9 ilustra a classe Marca, suas SubClasses, Instâncias e Propriedades.

A Figura B.10 ilustra a classe Recurso, suas SubClasses e Propriedades.

A Figura B.11 ilustra as classes Network, Mensagem, Serviço e Protocolo, suas SubClasses e Propriedades.

A Figura B.12 ilustra as classes Camada, Sistema, Componente_de_software e Sistema_operacional, suas SubClasses, Instâncias e Propriedades.

A Figura B.13 ilustra a classe Software, suas SubClasses e Propriedades.

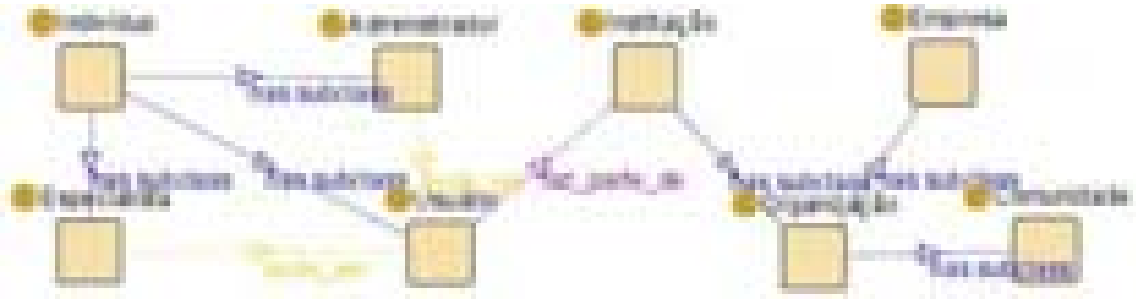

Figura B.7: *Classes Indivíduo e Instituição, suas SubClasses e Propriedades.*

Figura B.8: *Classes Codificação e Código, suas SubClasses, Instâncias e Propriedades.*

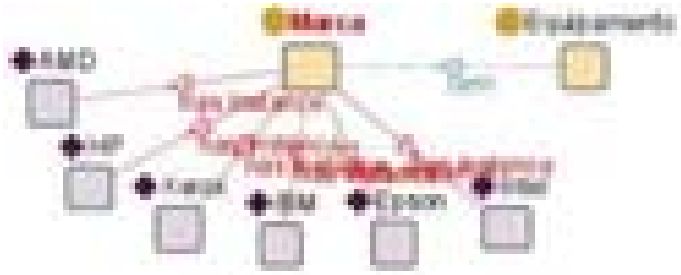

Figura B.9: *Classe Marca, suas SubClasses, Instâncias e Propriedades.*

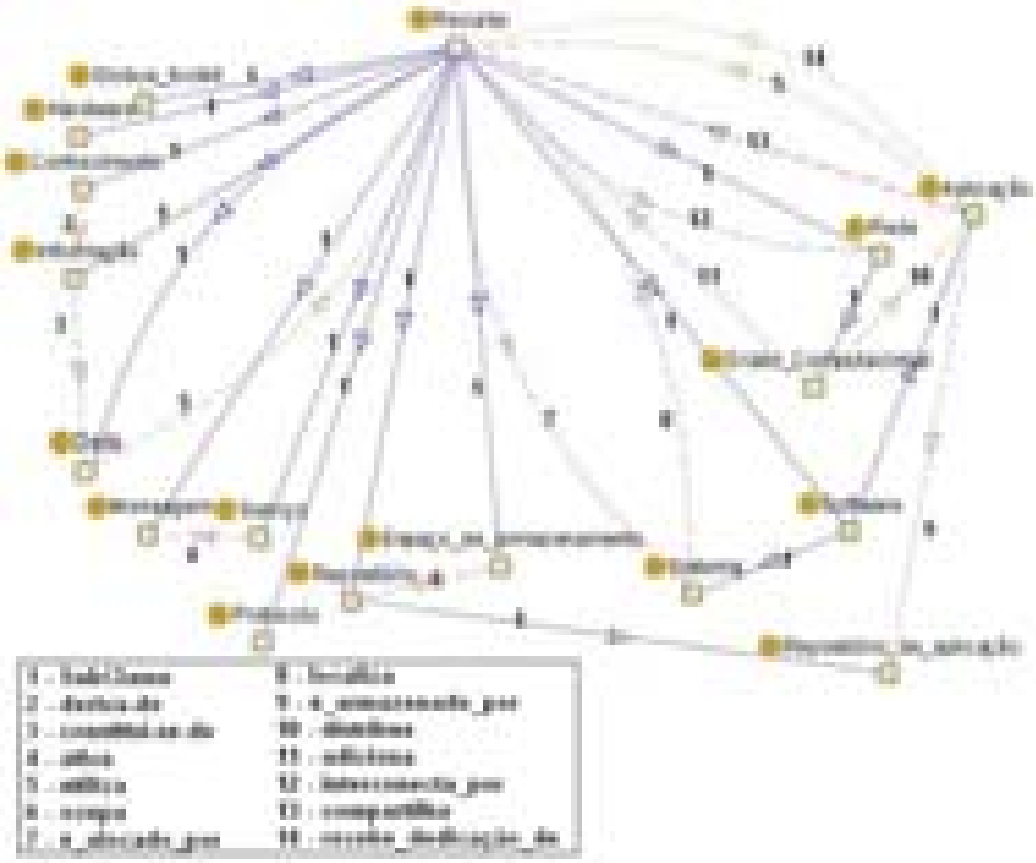

Figura B.10: *Classe Recurso, suas SubClasses e Propriedades.*

Figura B.11: *Classes Network, Mensagem, Serviço e Protocolo, suas SubClasses e Propriedades.*

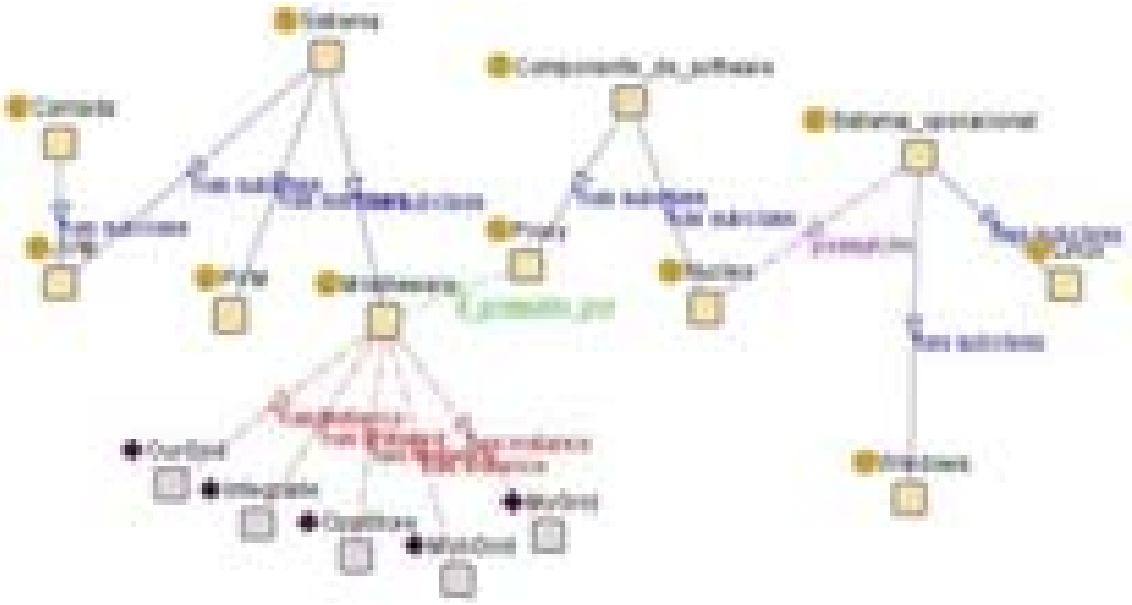

Figura B.12: *Classes Camada, Sistema, Componente_de_software e Sistema_operacional, suas SubClasses, Instâncias e Propriedades.*

Figura B.13: *Classe Software, suas SubClasses e Propriedades.*