Alexandre César Tavares Vidal

Abordagem Semântica Aplicada à
Integração e Gerenciamento de Recursos e
Aplicações em Grades Computacionais

Tese apresentada à Escola Politécnica
da Universidade de São Paulo para
obtenção do Título de Doutor em Enge-
nharia Elétrica.

São Paulo
2007

Alexandre César Tavares Vidal

Abordagem Semântica Aplicada à Integração e Gerenciamento de Recursos e Aplicações em Grades Computacionais

Tese apresentada à Escola Politécnica
da Universidade de São Paulo para
obtenção do Título de Doutor em Enge-
nharia Elétrica.

Área de concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. Sergio Takeo Kofuji

São Paulo
2007

FICHA CATALOGRÁFICA

César Tavares Vidal, Alexandre

Abordagem Semântica Aplicada à Integração e Gerenciamento de Recursos e Aplicações em Grades Computacionais. São Paulo, 2007. 155 p.

Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo. Departamento de Sistemas Eletrônicos.

1- Grades Semânticas 2- Gerenciamento de Recursos de Grades 3- Ontologias 4- Recuperação de Informação 5- Lógica de Descrição

I. Universidade de São Paulo. Escola Politécnica. Departamento de Sistemas Eletrônicos.

Departamento de Sistemas Eletrônicos. II. T.

Dedico esse trabalho à Maria Eduarda (Duda), Igor, minha irmã Rossana (em memória), Tia Socorro (em memória), à minha mãe, minha família e a todos os meus amigos.

Agradecimentos

“...I get high with a little help
from my friends.”

Lennon/McCartney

Agradeço profundamente ao meu orientador, Prof. Dr. Sergio Takeo Kofuji, pela confiança e tranquilidade.

Muito obrigado, Prof. Dr. Fabio Kon, pelas opiniões e apoio sinceros.

Agradeço aos amigos da USP que me acompanharam, tão de perto, nesses anos: Ricardo, Raphael, Celina, Giuliano, Arlindo, Marco e Priscila. Obrigado, pelo apoio e pela alegria da convivência com vocês.

Obrigado aos colegas do DEINF/UFMA, especialmente aos professores Francisco Silva, Maria Auxiliadora, Rosario Girardi, e Anselmo. O apoio desses amigos tem sido inestimável.

Alguns amigos estiveram particularmente próximos em momentos especiais e por vezes difíceis que vivi ao longo dessa jornada. Muito obrigado aos queridos amigos: Clodis, Cláudia, Jaqueline, Wendel, Pollyanna, Luis Paulo, Braga, Sandra e Rogério.

À minha família, pelo afeto e pelo apoio em tudo o que tenho feito e vivido.

Obrigado a Deus, por poder expressar minha gratidão.

Resumo

O domínio de grades computacionais é fortemente associado ao compartilhamento de recursos para a resolução de problemas em ambientes dinâmicos e heterogêneos. Reutilizar aplicações armazenadas na grade e alocar recursos para atender os requisitos das aplicações são tarefas essenciais na grade e dependem da qualidade das informações disponíveis. O presente trabalho propõe uma abordagem semântica para a construção de um serviço de metadados sobre grades e sobre as aplicações nelas disponibilizadas. A abordagem proposta explora a realização de inferência sobre ontologias de aplicações e recursos computacionais da grade e a realização de consultas sobre a base inferida para promover a realização de tarefas, tais como escalonamento de recursos para execução de aplicações na grade, de modo eficiente. O middleware para Grades oportunistas, InteGrade, foi a base para o desenvolvimento desse projeto.

Abstract

The grid computing domain is strongly related to resource sharing for problem resolution in dynamic and heterogeneous environments. The reuse of applications stored on grids and the allocation of resources to meet application requirements are essential grid tasks. These tasks depend on the quality of available information. This work proposes a semantic approach to build a metadata service about grid models, resources and applications. This approach explores inference and queries on ontologies about applications and grid computational resources aiming to enhance the execution of grid tasks, such as resource scheduling on grids. We used InteGrade, an opportunistic grid middleware, as the ground for the development of our project. The metadata integration will contribute to promote the efficient usage of grid resources and applications.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

1	Introdução	1
1.1	Motivação	2
1.1.1	Contexto	4
1.2	Definição do Problema	6
1.2.1	Visão geral do problema	6
1.2.2	Detalhamento do problema	7
1.3	Abordagem proposta	7
1.3.1	Desafios	9
1.3.2	Metodologia de pesquisa	10
1.4	Objetivos	12
1.4.1	Objetivos Específicos	13
1.4.2	Contribuições esperadas	13
1.5	Organização da Tese	14
2	Trabalhos Relacionados	17
2.1	Escopo do domínio	18

2.1.1	Core Grid Ontology (CGO)	18
2.1.2	DAta Mining ONtology (DAMON)	19
2.2	Escalonamento de recursos	19
2.2.1	GRIP broker	20
2.2.2	Semantic-Based Grid Architecture	20
2.2.3	Ontology-based Matchmaker	23
2.2.4	Virtual Grid (VG)	24
2.3	Reutilização de aplicações	25
2.3.1	SOFA/DCUP	26
2.3.2	KOntoR	27
2.4	Projetos em e-science	28
2.4.1	ICENI	28
2.4.2	Proteus	30
2.5	Conclusão	31
3	Grades Semânticas	32
3.1	Grades Semânticas	32
3.2	Grades baseadas em conhecimentos	34
3.3	Ontologias e linguagens de modelagem	36
3.3.1	A linguagem RDF	37
3.3.2	A linguagem OWL de modelagem semântica	41
3.4	O paradigma clássico de modelagem	48
3.5	Mecanismos de raciocínio	50
3.5.1	A ferramenta de raciocínio Pellet	51
3.6	Recuperação de informações	53

3.6.1	OWL-QL	54
3.6.2	A Linguagem SPARQL	57
3.7	Conclusão	59
4	Ontologias para o Gerenciamento de Grades Computacionais	60
4.1	Metodologias para o desenvolvimento de ontologias	63
4.2	Desenvolvendo ontologias para grades computacionais	64
4.2.1	Domínio e escopo da ontologia	65
4.2.2	Finalidade da ontologia	65
4.2.3	Questões tratadas pela ontologia	66
4.2.4	Atores e casos de uso	68
4.3	Estrutura e conteúdo das ontologias	74
4.3.1	Versão inicial	74
4.3.2	Versão atual	78
4.4	Conclusão	85
5	Integração e Gerenciamento Semântico de Recursos e Aplicações	87
5.1	Uso de ontologias como taxonomias	87
5.2	Extensão semântica dos metadados	89
5.3	Recuperação de informações	90
5.3.1	Características das ontologias	91
5.3.2	Inferência antecipada	92
5.3.3	Inferência e consultas	93
5.4	Cenários para abordagem semântica	94
5.4.1	Busca de Aplicações	94
5.4.2	Políticas de escalonamento	96

5.4.3	Heurística para uso dos mecanismos semânticos	99
5.5	Estudo de Caso	100
5.5.1	Descrição do contexto	101
5.5.2	Descrição da avaliação	102
5.5.3	Resultados obtidos	104
5.6	Conclusão	108
6	Implementação e Validação da Proposta	109
6.1	Componentes da Arquitetura	110
6.2	Implementação	113
6.3	Validação da proposta	118
6.3.1	Desempenho das consultas sobre a base de metadados . . .	119
6.3.2	Efetividade das consultas	121
6.4	Conclusão	125
7	Contribuições, Conclusão e Trabalhos Futuros	126
7.1	Contribuições	127
7.1.1	Contribuições técnicas	128
7.1.2	Contribuições científicas	128
7.2	Trabalhos futuros	129
Apêndice A – Gerenciamento de Aplicações e Recursos em Grades		131
A.1	Conceitos e Taxonomias	131
Apêndice B – Consultas SPARQL para avaliação de desempenho		144
Apêndice C – Consultas baseadas na API Protégé-OWL		146

Lista de Figuras

3.1	Posição da grade semântica no contexto de Grades	33
3.2	Tecnologias chave para grades	36
4.1	Casos de uso para o <i>Grid Knowledge Architect</i>	69
4.2	Desenvolvendo uma ontologia - Protégé-OWL GUI	70
4.3	Casos de uso para o <i>Grid Manager</i>	70
4.4	Casos de uso para o <i>Grid Application Developer</i>	71
4.5	Casos de uso para o <i>Grid User</i>	72
4.6	Casos de uso para o <i>Semantic Grid Integration Component</i>	73
4.7	Visão Geral da Ontologia de Grades	75
4.8	Conteúdo de software da grade	76
4.9	Conteúdo de dados da grade	76
4.10	Conceitos comuns ao domínio da grade	77
4.11	Sistemas de middleware de grade	78
4.12	Arquitetura da Base de Conhecimentos	80
4.13	Componentes da grade	81
4.14	Conceitos de software na grade	82
4.15	Grid Management Ontology - Hierarquia de Domínios	84
4.16	Grid Management Ontology - Hierarquias de Problemas e Abor- dagens	85
5.1	Hierarquia de classes - Pré-Inferência	89
5.2	Inferência de hierarquia de classes	90

5.3	Arquitetura do InteGrade	98
5.4	Hierarquia de classes - Softwares	107
5.5	Hierarquia inferida de classes - Softwares	107
6.1	Arquitetura para integração semântica da grade	110
6.2	Componentes da arquitetura	111
6.3	Distribuição dos componentes nos nós da grade	113
6.4	Componentes internos do SGIC	114
6.5	Diagrama de seqüência - Registrando um recurso da grade	115
6.6	Diagrama de seqüência - Registrando um domínio	116
6.7	Diagrama de seqüência - Registrando uma aplicação	117
6.8	Diagrama de seqüência - Preparando a execução de aplicação . . .	118
6.9	Definição da classe <i>PatternRecognitionAlgorithm</i>	122
6.10	Descrição da classe <i>DBSCAN_Algorithm</i>	122
6.11	Hierarquia inferida	123
A.1	Características das tarefa de escalonamento (CASAVANT; KUHL, 1988)	132
A.2	Taxonomia de sistemas de grade. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)	140
A.3	Taxonomia do modelo de organização de máquinas. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)	141
A.4	Taxonomia do modelo de recursos. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)	141
A.5	Taxonomia da organização do espaço de nomes. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)	142
A.6	Taxonomia de descoberta de recursos. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)	142

A.7	Taxonomia de disseminação de recursos. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)	142
-----	--	-----

Lista de Tabelas

5.1	Tipos de Plataformas	103
5.2	Plataformas equivalentes	103
5.3	Algoritmos ITA e artefatos de software relacionados disponíveis .	104
5.4	Abordagem de escalonamento do InteGrade	105
5.5	Estratégia Semântica de Pré-escalonamento - <i>seleção de recursos</i> .	106
5.6	Estratégia Semântica de Pré-escalonamento - <i>seleção de recursos</i> <i>e executáveis</i>	106
6.1	Desempenho das consultas em ms	120
6.2	Tempo para inferência de classes em segundos	120

Lista de Abreviaturas

BSP *Bulk Synchronous Parallelism*

CGO *Core Grid Ontology*

DAMON *Data Mining Ontology*

DIG *DIG Description Logic Interface*

DQL *DAML Query Language*

GBO *Grid Base Ontology*

GGF *Global Grid Forum*

GMO *Grid Management Ontology*

GRIP *GRid Interoperability Project*

GRM *Global Resource Manager*

ICENI *Imperial College e-Science Networked Infrastructure*

ITA *Image Texture Analysis*

JTP *Java Theorem Prover*

LRM *Local Resource Manager*

MPI *Message Passing Interface*

OGSA *Open Grid Service Architecture*

OWL *Web Ontology Language*

OWL-DL *OWL Description Language*

OWL-QL *OWL Query Language*

PSE *Problem Solving Environment*

PDKD *Parallel and Distributed Knowledge Discovery*

PO *Platform Ontology*

RDF *Resource Description Framework*

RDF-S *Resource Description Framework Schema*

S-OGSA *Semantic Open Grid Service Architecture*

SPARQL *SPARQL Protocol and RDF Query Language*

UML *The Unified Modelling Language*

UNICODE *The universal character encoding*

VG *Virtual Grid*

1 Introdução

“...na volta do barco é que sente o
quanto deixou de cumprir.”

Chico Buarque

O termo Grades Computacionais (*Computational Grids*) (FOSTER; KESSELMAN, 2004; BERMAN; FOX; HEY, 2003) denomina uma infra-estrutura que permite o compartilhamento de recursos heterogêneos em ambientes dinâmicos, dispersos geograficamente e em diferentes domínios administrativos para a execução de aplicações. Estas infra-estruturas normalmente oferecem serviços de descoberta e compartilhamento de recursos, segurança, escalonamento e tolerância a falhas, entre outros.

A natureza heterogênea dos recursos dispersos nas grades computacionais e a sinergia potencial desses recursos justificam o esforço para integrá-los de modo eficiente. A integração eficiente da grande diversidade de recursos disponíveis alavanca a utilização das grades em larga escala. Por outro lado, essas mesmas características de diversidade e comportamento dinâmico dos recursos estão na base de alguns dos principais desafios encontrados atualmente na pesquisa em grades computacionais.

O problema de como prover recursos apropriados para uma aplicação considerando que algumas aplicações só podem ser executadas sob restrições de arquiteturas de computadores, sistemas operacionais ou máquinas específicas é um

exemplo desse tipo de desafio. Russel et al. (RUSSEL et al., 2003) sugerem a associação de metadados às aplicações para descrever restrições sobre plataformas e requisitos sobre bibliotecas e ferramentas. Esses metadados poderiam ser usados para alocar recursos computacionais apropriados para uma determinada aplicação. Tal estrutura de metadados também deve ser útil para descrever as demais entidades que compõem as grades computacionais, tais como dados, usuários, políticas, etc.

Numa perspectiva mais ampla, o uso de metadados para descrever aplicações, recursos e outras entidades em grades computacionais, pode ser estendido para otimizar a utilização das grades facilitando a integração de seus conteúdos e recursos.

De um modo concreto, essa abordagem pode ser realizada através da implementação de um modelo de metadados que possibilite representar aspectos centrais relacionados a grades computacionais. O modelo deve ser extensível para acompanhar a evolução e a dinâmica das grades, resultantes da integração constante de novos recursos, serviços e funcionalidades.

1.1 Motivação

O conceito de grades computacionais, sua criação e evolução são largamente motivados pela necessidade de reutilização e compartilhamento do conjunto de recursos que as constituem. Para esse fim, tais recursos precisam ser integrados de forma eficiente. Aplicações, dados, estruturas de rede, middleware e equipamentos compõem o universo dos recursos disponibilizados no escopo das grades.

A regência organizada desse conjunto promove o uso da grade computacional como um meio efetivo sobre o qual aplicações complexas e com grande demanda de recursos podem ser eficientemente executadas. Essa orquestração se verifica

em torno de diferentes camadas e domínios na grade, como por exemplo, escalonamento de máquinas para execução de aplicações e armazenamento de dados, identificação de variantes de aplicações de usuários para execução na grade, descoberta de componentes e bibliotecas de software reutilizáveis através de diferentes domínios na grade, entre outros.

A infra-estrutura da grade computacional é organizada principalmente para atender requisitos gerados pela demanda de execução de aplicações e do armazenamento e recuperação dos dados produzidos e consumidos por essas mesmas aplicações. De acordo com deRoure (ROURE et al., 2003), esses requisitos têm sua origem principalmente no domínio de aplicações complexas como, por exemplo, aplicações científicas executando na grade (*e-science*) e continuam válidos no contexto de aplicações para grades de um modo geral.

Requisitos computacionais das aplicações incluem, entre outros: ciclo do 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.

Segundo (RUSSEL et al., 2003), considera-se desejável que as aplicações sejam capazes de descrever suficientemente seus requisitos. Por exemplo, a comparação dos requisitos necessários para a execução das aplicações com as características dos dispositivos disponíveis na grade (espaço de memória, ciclo do processador, espaço de armazenamento em disco, etc.) pode se apoiar na descrição das aplicações e deve influenciar positivamente a otimização do gerenciamento e alocação dos recursos computacionais. Em um segundo exemplo, a descrição dos requisitos computacionais e dos requisitos sobre os dados associados às aplicações pode ser empregada para otimizar a localização dinâmica de recursos para computação de alto desempenho, combinando acesso a conjuntos de dados com capacidade de

replicação.

O uso de metadados em grades favorece o compartilhamento e reutilização de aplicações e dados gerando integração de conteúdo das grades, acesso a visões compartilhadas de problemas e soluções e construção de base comum para integração de ferramentas. Adicionalmente, essa abordagem pode ajudar a promover a redução de esforço e tempo empregados nas atividades de desenvolvimento de aplicações, previsibilidade e confiança sobre os resultados gerados pelas execuções das aplicações, confiança sobre a origem dos dados, incremento da produtividade através de atividades de cooperação e propagação de conhecimento.

Uma base de metadados viabiliza ainda a implementação e execução de uma grande diversidade de aplicações na grade. Usuários da grade podem realizar variados tipos de consultas sobre aplicações e dados armazenados na grade, seus domínios e versões. Por exemplo, identificar aplicações e dados associados a domínios e usuários específicos, ou ainda, identificar versões disponíveis para uma determinada aplicação.

Os argumentos expostos acima sugerem que o desempenho eficiente de tarefas essenciais na grade depende, em uma extensão considerável, de uma base bem definida de metadados e informações sobre aplicações e recursos da grade.

1.1.1 Contexto

O InteGrade (GOLDCHLEGER et al., 2004) é um middleware para grades oportunistas, isto é, voltado para o compartilhamento de recursos computacionais ociosos.

Inicialmente, essa participação no projeto InteGrade incluía o desenvolvimento de um repositório de aplicações, com o objetivo de promover a reutilização de aplicações registradas e executadas na grade. O Repositório de aplicações

do InteGrade armazena aplicações registradas pelos desenvolvedores e metadados relacionados a essas aplicações. O modelo de metadados para descrição de aplicações do repositório de aplicações do InteGrade é um modelo de atributos simples, voltado principalmente para permitir a reutilização das aplicações registradas na grade. A simplicidade do modelo de metadados do repositório de aplicações possibilitou, contudo, uma oportunidade de explorar seu potencial de extensão e de incremento de uso.

O potencial de extensão do repositório de aplicações do InteGrade suscitou, desse modo, o aprofundamento da nossa pesquisa para explorar de modo mais eficiente os metadados sobre aplicações armazenadas no repositório de aplicações e aumentar o seu escopo de utilização. Percebeu-se que diferentes cenários e atividades na grade poderiam ser melhor trabalhados a partir da idéia de reutilização das aplicações, com base sobre o seu conjunto de metadados.

Mais especificamente, considerou-se a possibilidade de usar o modelo de metadados para incrementar a eficiência de algumas tarefas importantes realizadas na grade. O conjunto de tarefas potencialmente beneficiadas poderia incluir, por exemplo, a busca de aplicações na grade realizada por usuários interessados em reutilização de software e o apoio para a promoção de escalonamento eficiente de aplicações na grade.

Essas novas possibilidades apontaram, necessariamente, para um modelo de metadados não limitado apenas a aplicações, mas que levasse em conta outras características das grades computacionais, tais como recursos e usuários, considerando inclusive, a heterogeneidade dessas características.

Adicionalmente, considerou-se que a utilidade de tal modelo seria proporcional, no nosso ponto de vista, à sua independência em relação a um ambiente específico de grade. Tal independência permitiria a reutilização do próprio modelo

em diferentes ambientes de grade.

As reflexões descritas nesta subseção foram a motivação inicial para a investigação acerca do uso de uma base de metadados independente e extensível, sobre aplicações, recursos e atividades realizadas em ambientes de grades computacionais.

1.2 Definição do Problema

A Seção 1.1 identifica um conjunto de razões e de cenários propícios à integração de recursos em grades computacionais. Contudo, o problema de como integrar, reutilizar e compartilhar aplicações e recursos computacionais no ambiente de grades, de modo eficiente, ainda permanece uma questão complexa. Nesta seção, analisamos as principais questões associadas ao problema. A condução dessas questões devem orientar a abordagem proposta na Seção 1.3.

1.2.1 Visão geral do problema

Uma estratégia bem sucedida para realizar as atividades descritas na Seção 1.1, depende, entre outras coisas, da qualidade da informação disponível sobre artefatos de software e recursos da grade. Tarefas como a definição de estratégias para escalonamento de aplicações e algoritmos de escalonamento são baseadas nas descrições dos atributos dos recursos e dos requisitos e preferências das aplicações. Portanto, o modo como tais informações são capturadas, armazenadas, organizadas, disponibilizadas e utilizadas representam um papel relevante nas principais tarefas executadas na grade.

1.2.2 Detalhamento do problema

Alguns aspectos adicionais, mas também considerados bastante significativos, devem ser tratados nesse contexto. Em primeiro lugar, à medida que a estrutura da grade cresce pelo incremento de aplicações e recursos instalados, a dificuldade de se encontrar e reusar uma aplicação desejada ou um conjunto específico de recursos também cresce proporcionalmente. Isso se deve em princípio ao grande volume de informações distribuído e a dificuldade de armazenamento dessas informações.

Adicionalmente, aplicações e recursos são geralmente descritos em uma infinidade de maneiras diferentes, sob diferentes perspectivas. As causas envolvidas, incluem entre outras, a ausência de vocabulários comuns, a produção de informação em diferentes formatos e por diferentes fontes, e a natureza incompleta da informação produzida. A heterogeneidade de tal base de informações aumenta consideravelmente as chances de que essas informações, tomadas em seu conjunto, sejam pouco confiáveis.

1.3 Abordagem proposta

A presente abordagem é construída em torno do conceito de *grades semânticas* definido em (ROURE; JENNINGS; SHADBOLT, 2005). Grades computacionais semânticas (*Semantic Grids*) são extensões de grades computacionais nas quais informações e serviços possuem um significado bem definido, propiciando o trabalho cooperativo entre computadores e pessoas.

Com o objetivo de lidar com os problemas descritos na Seção 1.2, propomos uma abordagem semântica, baseada em ontologias. Ontologias podem promover, pela submissão a mecanismos de inferência, a integração de informações inicialmente não conectadas explicitamente; por esse motivo, espera-se que essa

abordagem possa contribuir para atenuar os efeitos da captura de informações advindas de fontes heterogêneas e de forma incompleta, como descrito na seção anterior.

A partir da demanda originada pela comunidade de *e-science* por ambientes de grades computacionais, identificou-se a importância da abordagem semântica, como uma alternativa para lidar com a heterogeneidade dos recursos computacionais das grades, do middleware das grades e das aplicações que nelas são executadas.

Na referência (CHANDRASEKARAN; JOSEPHSON; BENJAMINS, 1999), ontologias são definidas como teorias do conhecimento sobre tipos de objetos, propriedades dos objetos e relações entre objetos que são possíveis em um domínio de conhecimento especificado. Argumenta-se que uma vez disponibilizada uma boa teoria do conhecimento, muitos diferentes mecanismos para manipulação de conhecimento poderiam ser usados satisfatoriamente na implementação de sistemas efetivos sobre o mesmo conteúdo.

Nossa abordagem parte dessa idéia e propõe inicialmente o uso de metadados e ontologias para a descrição de aplicações e recursos de grades computacionais, e definição de mecanismos para promover integração, compartilhamento e reutilização eficiente desses recursos. O modelo de metadados deve contribuir para o uso flexível e uniforme das diferentes aplicações e recursos continuamente integrados à grade. O caráter extensível da abordagem proposta deve garantir a integração consistente de novas formas de conteúdo.

Metadados e ontologias serão a base de integração dos recursos e conteúdo das grades. Associados à essa base de informações, mecanismos de inferência permitirão a descoberta de relacionamentos semânticos e a produção de novas informações à partir das informações declaradas e armazenadas na base. Esses

novos conhecimentos serão continuamente incorporados à base de informações.

Desse modo, a base de informações semanticamente estendida deverá ser usada, de forma integrada, para a exploração de cenários relevantes do gerenciamento de recursos em grades.

1.3.1 Desafios

Atualmente, o middleware InteGrade é dirigido principalmente para o aproveitamento de recursos computacionais ociosos em grades oportunistas. Em consequência, a relativa escassez de aplicações executando sobre o InteGrade não oferece um conjunto suficiente de cenários variados e bem definidos que possam servir de base para a elaboração de uma base de metadados genérica e extensível. Para promover a integração e reutilização do conteúdo da grade, a base de metadados deve ser facilmente extensível para descrever situações específicas.

Conseqüentemente, o primeiro desafio encontrado é a identificação de cenários e taxonomias suficientemente genéricos e potencialmente extensíveis para descrever situações representativas para diferentes ambientes de grade. O cerne desse desafio é alcançar a construção de um modelo genérico adequado para ser simultaneamente reutilizável e escalável.

A complexidade desse desafio deve-se à necessidade de se realizar um levantamento detalhado dos conceitos considerados relevantes no domínio. É necessário investigar a natureza das grades, as aplicações e dados que integram seu conteúdo, suas políticas de utilização, tipos de usuários e infra-estrutura, entre outros aspectos importantes.

De acordo com a proposta descrita na Seção 1.3, essa investigação deve servir de base conceitual para a criação de um modelo de metadados semântico. A descrição semântica de conceitos, escopo e uso dos elementos do domínio é

inerentemente complexa. Isso nos conduz ao segundo desafio: a definição de uma estratégia para a representação semântica da grade. Essa definição envolve escolha de tecnologias maduras e apropriadas, decisões equilibradas e soluções criativas associadas ao emprego coerente de padrões arquiteturais e exige uma aproximação flexível, extensível, clara e não ambígua.

A tecnologia para a representação semântica de recursos em grades computacionais ainda está evoluindo, em busca de maturidade, estabilidade e padronização. Por essa razão, assim como os recursos da grade, a representação semântica desses recursos deve ser pensada de modo a ser ela própria também compartilhada, extensível e reutilizável de modo flexível, para acompanhar a evolução das tecnologias e vir a ser, de fato, um meio de integração entre diferentes estruturas de grade.

1.3.2 Metodologia de pesquisa

A atividade inicial da metodologia assume o caráter de revisão das tecnologias consideradas relevantes relacionadas ao contexto. A principal tarefa nessa fase é a investigação do estado da arte através da pesquisa e revisão bibliográfica. Esse processo inclui necessariamente os dois temas centrais da pesquisa, ou seja, grades semânticas e gerenciamento integrado de recursos e aplicações em grades.

Esse é também um processo iterativo, no qual diversas tecnologias e modelos são investigados e eventualmente descartados por se mostrarem inadequadas às finalidades do trabalho.

A tarefa seguinte consiste na identificação de cenários genéricos em grades computacionais para adoção da abordagem semântica, desenvolvida paralelamente às atividades de definição da estratégia para a representação semântica da grade e a construção mesma das primeiras versões das ontologias. Essa ativi-

dade também é conduzida necessariamente de modo iterativo e incremental.

O primeiro passo dessa fase é a elaboração de uma ontologia básica para representar os conceitos fundamentais relacionados a grades computacionais, seus componentes, recursos, entidades, usuários, mecanismos, políticas, etc. A ontologia deve tornar explícito o conhecimento sobre o mecanismo de uma grade computacional e definir uma taxonomia sobre conceitos comuns ao domínio de grades computacionais. Sua função mais relevante, contudo, é prover uma base semântica extensível para o conjunto de metadados em torno de recursos e aplicações de grades visando promover a reutilização desses itens, a interoperabilidade entre diferentes organizações virtuais e entre diferentes sistemas de middleware de grades.

A atividade subsequente é a extensão da ontologia básica para permitir a definição de um modelo semântico voltado para a descrição de aplicações na grade e conceitos relacionados à essas aplicações. O conjunto de conceitos inclui requisitos computacionais, características gerais do domínio, dependências de software tais como bibliotecas e componentes, controle de versões, modelos de configuração e dados de entrada e saída.

A definição de uma estratégia adequada para a representação semântica da grade envolve fatores tais como a escolha de linguagens de modelagem e de recuperação de informação, identificação de mecanismos e ferramentas de inferência, elaboração de arquitetura flexível, e integração sinérgica desses requisitos. Ao longo do trabalho, a análise cuidadosa dessas características e de suas alternativas de implementação deve justificar o conjunto de decisões que vai culminar na proposta arquitetural. Contudo, algumas direções e decisões podem ser antecipadas neste contexto introdutório.

O potencial de extensão da base semântica apóia-se na adoção da *Web On-*

ology Language-OWL (W3C, 2007d), linguagem para definição de ontologias recomendada pelo consórcio W3C. A construção de um protótipo de uma base de conhecimento em OWL será facilitada pela utilização da ferramenta Protégé-OWL (KNUBLAUCH et al., 2004). Protégé-OWL é uma ferramenta que permite a criação de ontologias OWL extensíveis e a realização de consultas e raciocínios sobre a base de conhecimento, para inferência, por exemplo, de hierarquia de classes. A ferramenta possibilita a conexão de motores de inferência, especificamente na ferramenta Pellet (SIRIN et al., 2007). As razões e a cadeia de implicações associadas a essas decisões serão discutidas ao longo do trabalho.

A metodologia adotada na validação da pesquisa é dirigida à implementação. Os produtos de software são implementados iterativamente, a partir dos princípios definidos nas motivações, problemas, abordagem introdutória e desafios definidos. Os produtos de software implementados são, basicamente, um conjunto de ontologias centrais, ontologias específicas derivadas das primeiras e um conjunto de mecanismos para explorar o uso dessa base de informações em cenários relevantes identificados ao longo do processo.

1.4 Objetivos

O principal objetivo deste trabalho é contribuir para a integração e gerenciamento eficiente de recursos em grades computacionais. Espera-se que essa integração propicie um incremento da reutilização de software e recursos, facilite a descoberta de aplicações na grade através de mecanismos de consulta eficientes e ofereça alternativas de utilização de recursos e aplicações disponíveis na grade.

1.4.1 Objetivos Específicos

Objetivos específicos incluem mitigar os efeitos da heterogeneidade das grades e otimizar o compartilhamento e uso de recursos na grade, através do uso de um modelo semântico de metadados. O modelo semântico de metadados propõe a extensão da base de informações sobre recursos e aplicações das grades através da realização de inferência sobre informações capturadas, cuja natureza é comumente incompleta.

Os objetivos específicos enumerados a seguir estão intrinsicamente relacionados em uma cadeia de causa e efeito com a extensão semântica da base de informações.

1. Propor um conjunto de ontologias para representar aplicações e recursos na grade;
2. Propor uma arquitetura de sistema de software, baseada em uma abordagem semântica para exploração de metadados, que permita promover o gerenciamento eficiente de recursos e aplicações na grade;
3. Implementar a arquitetura proposta;
4. Validar a implementação da arquitetura proposta.

1.4.2 Contribuições esperadas

Nossa proposta apóia-se no conceito de grades semânticas, buscando, contudo, aplicações originais para esse conceito em cenários específicos no contexto de grades. Nosso ponto de partida foi o reconhecimento da importância da abordagem semântica na integração e compartilhamento de recursos em grades, particularmente, em domínios de aplicações específicos, tais como *e-science*.

Essa generalização da abordagem semântica apresenta evidentemente o risco de super-exposição e aplicação exagerada, possivelmente em áreas inadequadas.

Desse modo, entendemos que a nossa contribuição vai se definir pontualmente, pelo emprego da abordagem semântica no contexto de dois cenários específicos relacionados às atividades desenvolvidas em grades computacionais. Esses cenários tratam especificamente das atividades de busca de aplicações na grade e do escalonamento de recursos para a execução das aplicações e foram esboçados nos parágrafos iniciais deste capítulo e na Seção 1.1. O Capítulo 3 discute conceitos fundamentais em torno desses cenários. O Apêndice A apresenta taxonomias relacionadas a tais conceitos.

De um modo geral, o presente trabalho espera contribuir, através da utilização da abordagem semântica sobre metadados na grade, para a exploração da sinergia desses cenários, especialmente sobre os aspectos relacionados aos ganhos de produtividade e desempenho. O Capítulo 5 trata dos detalhes dessas contribuições.

Espera-se que a utilização de uma base semântica extensível e consistente de metadados sobre grades computacionais, seus componentes, recursos e conteúdo promova a integração, compartilhamento e reutilização eficientes da grade como um todo. A avaliação dessa abordagem é realizada inicialmente no middleware InteGrade. Aspectos relevantes dessa avaliação são detalhados no Capítulo 5.

1.5 Organização da Tese

O corpo da tese está organizado nos seguintes capítulos e apêndices. O Capítulo 2 apresenta trabalhos relacionados considerados relevantes, com ênfase na análise comparativa com a presente proposta. Nesse capítulo apresentamos dez trabalhos relacionados ao escopo da nossa tese, organizados em quatro seções, as quais

são divididas em razão do foco principal dos trabalhos, a saber: trabalhos que tratam da criação de ontologias para grades, trabalhos que descrevem estratégias de escalonamento de recursos para execução de aplicações, trabalhos cujo foco é a reutilização de aplicações em grades computacionais e, finalmente, trabalhos voltados para projetos de *e-science*. No Capítulo 3, definem-se os conceitos de grades semântica e ontologias; descrevem-se linguagens de representação em ontologias, com ênfase sobre a linguagem OWL; discute-se o paradigma clássico de modelagem, mecanismos e políticas para exploração da base de conhecimentos baseados na inferência de informações; e, finalmente, apresentam-se estratégias de recuperação de informações em ontologias OWL. O Capítulo 4 apresenta um conjunto de ontologias para grades computacionais e descreve os mecanismos de raciocínio e estratégias de inferência usadas para aproveitamento eficiente da estrutura semântica no contexto da grade. O Capítulo 5 apresenta uma abordagem integrada de cenários relevantes para o gerenciamento semântico de recursos em grades computacionais e descreve mecanismos para exploração dos cenários semanticamente integrados e que deverão ser incorporados à arquitetura proposta. O Capítulo 6 descreve aspectos da arquitetura proposta para atender os requisitos do modelo e as necessidades delineadas nos capítulos anteriores, além de detalhes da implementação e experimentos relacionados à validação do modelo. O Capítulo 7 apresenta as conclusões obtidas e relaciona as contribuições resultantes da realização do presente trabalho. A parte final da nossa tese inclui três pequenos apêndices, inseridos com o objetivo de complementar o corpo da tese através do detalhamento de alguns aspectos específicos do trabalho. O Apêndice A apresenta uma descrição das estratégias definidas para a resolução dos problemas identificados, com ênfase aos aspectos inerentes à sua integração, dentro do escopo do trabalho. O Apêndice B apresenta um conjunto incompleto das consultas, em linguagem SPARQL, realizadas sobre ontologias estendidas a partir das ontologias base propostas. Finalmente, o Apêndice C apresenta exemplos

de métodos alternativos de consultas, as quais são realizadas sobre as ontologias através do emprego de métodos definidos na biblioteca Protégé-OWL API.

2 Trabalhos Relacionados

Os trabalhos descritos no presente capítulo foram selecionado em razão de sua relevância e da intersecção de seu escopo com o problema tratado pela nossa pesquisa. Conforme descrito no capítulo introdutório, nosso trabalho investiga uma abordagem para o problema de como integrar, reutilizar e compartilhar aplicações e recursos no ambiente de grades computacionais de modo eficiente.

Também no capítulo introdutório foi definida uma abordagem semântica para o problema e foram discutidos os principais desafios envolvidos no seu emprego. Resumidamente, estes desafios envolvem a identificação de cenários e taxonomias genéricas relacionados à atividades de gerenciamento de recursos na grade e a definição de estratégias para a representação do modelo semântico de metadados.

Em razão da abordagem proposta, este capítulo descreve trabalhos relacionados que empregam diferentes tratamentos para a descrição semântica da estrutura e funcionamento de grades computacionais e para a utilização dessa estrutura na integração e gerenciamento de recursos na grade.

As seções que se seguem apresentam uma análise comparativa entre o nosso trabalho e os trabalhos relacionados. As seções são organizadas com o objetivo de apresentar um trabalho, ou grupo de trabalhos, que compartilham aspectos semelhantes entre si, identificando-se, simultaneamente, as diferenças relevantes entre o nosso trabalho e cada uma das propostas.

2.1 Escopo do domínio

Os trabalhos desta seção propõem criação de ontologias para grades. As propostas apresentadas se distinguem em razão do escopo do domínio tratado pelas ontologias.

2.1.1 Core Grid Ontology (CGO)

A maioria dos esforços recentes em sistemas de grades baseados em ontologias tratam de subdomínios específicos, tais como mineração de dados e escalonamento de recursos para execução de aplicações na grade. Identificamos apenas um trabalho relacionado que propõe uma representação ampla e genérica de uma ontologia de grades, *Core Grid Ontology* (CGO), apresentado em (XING et al., 2005). CGO é um sistema extensível de ontologia de grades que expressa conceitos e relacionamentos específicos e fundamentais para a grade. O objetivo do CGO é prover uma base de conhecimento comum sobre sistemas de grade, incluindo recursos, middleware, serviços, aplicações e usuários, como descrito em (XING; DIKAIKOS; SAKELLARIOU, 2006). Uma das principais metas da CGO é ser geral e extensível o suficiente para ser usada por diferentes arquiteturas de grade. Apesar das metas anunciadas em trabalhos mais recentes, não foram definidas estratégias de inferência sobre a ontologia proposta no contexto de sistemas e aplicações de grade.

Nossa proposta é semelhante à CGO no que diz respeito à construção de uma ontologia genérica. Em comparação, o foco do nosso trabalho é sobre a identificação e descrição de cenários e mecanismos extensíveis para explorar a capacidade de inferência do modelo semântico. Esperamos que o modelo resultante possa ser utilizado por diferentes sistemas de grades.

2.1.2 DAta Mining ONtology (DAMON)

O projeto DAMON, sobre mineração de dados para programação em grade é descrito em (CANNATARO; COMITO, 2003) e define uma ontologia sobre mineração de dados, escrita na linguagem DAML+OIL (PAGELS, 2005) para representação de ontologias.

A ontologia DAMON explora conceitos, tais como *Task*, *Method*, *Algorithm* e *Software*, relacionados ao domínio de mineração de dados. O principal objetivo desse trabalho é promover a reutilização de software para mineração de dados, usando as ontologias e mecanismos de inferência para apoiar a escolha de ferramentas adequadas para mineração de dados. Espera-se que essa abordagem contribua com a otimização de projetos de aplicações distribuídas de descoberta de conhecimentos através da mineração de dados.

DAMON explora especificamente o domínio de mineração de dados. Nosso trabalho, comparativamente, propõe uma abordagem mais geral para o gerenciamento do conteúdo da grade. Espera-se alcançar um nível maior de generalidade e, conseqüentemente, um nível maior de reutilização. Adicionalmente, integramos nossa abordagem para descoberta de informações com outros cenários relevantes no contexto de grades.

2.2 Escalonamento de recursos

Os trabalhos nesta seção descrevem estratégias de escalonamento de recursos para execução de aplicações; um cenário relevante no contexto de grades computacionais. Com apenas uma exceção, todos os trabalhos exploram uma abordagem semântica para apoiar estas estratégias.

2.2.1 GRIP broker

GRIP é o acrônimo para *G*Rid *I*nteroperability *P*roject (PALLAS et al., 2007), um projeto para promover a interoperabilidade dos projetos Globus (GLOBUS, 2007) e UNICORE (UNICORE, 2007), e simultaneamente buscar padrões de interoperabilidade no contexto do *Global Grid Forum (GGF)*. Em 2006, o *Global Grid Forum* uniu-se com a *Enterprise Grid Alliance* formando o *Open Grid Forum* (OGF, 2007).

O foco do trabalho descrito em (BROOKE et al., 2004) é o mapeamento semântico de diferentes ontologias associadas a diferentes sistemas de middleware de grade. Esse trabalho propõe a construção de um protótipo para a realização da combinação semântica da descrição de recursos de grade para o projeto GRIP, o *GRIP broker*.

A similaridade na descrição de recursos da grade e requisitos de aplicações é um domínio largamente explorado por aplicações semânticas na grade. O objetivo do *GRIP broker* é facilitar o acesso aos recursos da grade descritos para diferentes sistemas de middleware, através de uma abordagem semântica.

Enquanto o *GRIP broker* promove o mapeamento semântico entre ontologias diferentes, porém relacionadas, nossa proposta, em comparação, tem seu foco na criação de uma estrutura semântica baseada em ontologias para integrar recursos e aplicações em grades.

2.2.2 Semantic-Based Grid Architecture

O trabalho descrito em (SOMASUNDARAM et al., 2006) propõe uma arquitetura para grades semânticas baseada na introdução de uma camada de metadados para representação de conhecimento no topo de uma arquitetura de grade. O objetivo da camada de metadados é possibilitar a descoberta de recursos na grade através

de uma abordagem semântica. O componente semântico introduz uma ontologia base para viabilizar a descrição dos recursos da grade. A arquitetura proposta é implementada com base em um conjunto de ferramentas e bibliotecas, tais como Protégé-OWL (KNUBLAUCH et al., 2004; HORRIDGE, 2004), Protégé-OWL API (PROTÉGÉ, 2007), o sistema de descoberta e monitoramento (*Monitoring and Discovery System - MDS*) (CZAJKOWSKI et al., 2001) apud (SOMASUNDARAM et al., 2006), do Globus toolkit (GLOBUS, 2007), e o mecanismo de inferência Algernon¹. O mecanismo de inferência interage com a base de conhecimentos para descobrir recursos adequados à execução das aplicações na grade.

A arquitetura proposta é organizada em cinco camadas: *Fabric Layer*, *Core Middleware Layer*, *High Level Middleware Layer*, *Knowledge Layer* e *Application Layer*. O cerne da proposta é a camada de conhecimento (*Knowledge Layer*), cuja responsabilidade é implementar vários módulos para descrição semântica e descoberta de recursos de grade.

A descrição dos recursos da grade é baseada na idéia de uma ontologia *template* ou *gabarito*. Uma ontologia gabarito é definida como “uma ontologia específica de domínio que define uma hierarquia de conceitos, associada a um conjunto de propriedades para definir suas características”. Os recursos são modelados como instâncias de conceitos específicos, desde que possam ser descritas usando as propriedades associadas a tais conceitos. Após a criação da ontologia gabarito, recursos registrados na grade, e tratados por outras camadas de mais baixo nível da arquitetura, são associadas dinamicamente à instâncias dos conceitos da ontologia gabarito.

A descoberta dos recursos é baseada no uso do mecanismo de inferência Algernon. As consultas dos usuários são mapeadas em consultas para o Algernon e podem incluir operadores relacionais e manipulação de restrições. Para redu-

¹<http://algernon-j.sourceforge.net/tutorial>

zir a complexidade da consulta, uma base de dados é mantida, para associar os conceitos da ontologia gabarito com os sinônimos identificados. Dentro do mesmo propósito, uma interface de usuário é disponibilizada para a inserção de novos sinônimos e a conseqüente atualização da base de dados pelos usuários. A ferramenta Algernon permite descobrir não apenas os recursos que combinam exatamente com o requisito do usuário, mas também possibilita a recuperação de recursos que exibem relacionamento de subclasse na hierarquia de conceitos da ontologia.

A nossa proposta possui algumas semelhanças em relação ao trabalho analisado na presente seção. As semelhanças mais relevantes são associadas ao emprego de ontologias para representação de recursos da grade, ao uso da linguagem OWL para representação das ontologias, ao uso de mecanismos de inferência para descoberta de recursos e ao emprego de sistemas de middleware para execução de tarefas básicas da grade. Contudo, existem também diferenças fundamentais entre as duas propostas; as mais relevantes são descritas abaixo:

1. Nossa proposta desenvolve um conjunto de ontologias básicas que podem ser estendidas, além de instanciadas;
2. O conjunto de conceitos contemplados pelas ontologias inclui também conceitos de software, além dos conceitos para descrição de recursos computacionais da grade, possibilitando o registro de diferentes domínios verticais de software;
3. Os mecanismos de inferência em nossa proposta são usados com propósito semelhante, também mediados pela biblioteca Protégé OWL-API para redução dos custos da realização de inferência. Entretanto, exploramos uma ferramenta diferente para a realização de inferência e esta é realizada em momento distinto da recuperação de informações através de consultas, com

o objetivo de minimizar o custo das consultas.

2.2.3 Ontology-based Matchmaker

A proposta apresentada por Tangmunarunkit et al. (TANGMUNARUNKIT; DECKER; KESSELMAN, 2003) tem semelhanças com o trabalho mostrado na Seção 2.2.1 no sentido de buscar a combinação de recursos e aplicações na grade. Contudo, o foco da estratégia proposta é baseado em linguagem de regras para a descrição de políticas de gerenciamento de recursos baseados em ontologias.

Em (TANGMUNARUNKIT; DECKER; KESSELMAN, 2003), a descrição de recursos e da solicitação de execução de tarefas na grade são definidas declarativamente através de conceitos e regras contidos em ontologias criadas separadamente. A combinação de recursos e requisições é baseada nos termos definidos nessas ontologias e é orientada, principalmente, para atender as seguintes características:

- Descrição assimétrica de recursos e requisições, os quais são descritos separadamente e cuja combinação é baseada na semântica definida nas ontologias.
- Facilidade de compartilhamento e manutenção em comparação com abordagens baseadas em listas simples de atributos.
- Possibilidade de definição de restrições tanto na requisição de recursos, para especificar os atributos desejados dos recursos, quanto na expressão de políticas de uso por parte dos recursos.
- Possibilidade de descrição de critérios de preferência para facilitar a seleção de recursos quando existirem múltiplas combinações possíveis.
- Verificação de integridade na descrição dos recursos e requisitos.
- Expressividade, flexibilidade e facilidade de extensão.

A arquitetura proposta nesse trabalho é baseada em três principais componentes:

- Ontologias - as ontologias *Resource Ontology*, *Resource Request Ontology* e *Policy Ontology* foram desenvolvidas usando a linguagem RDF-Schema (W3C, 2007a) e a ferramenta Protégé² ;
- Conhecimento do domínio - este componente captura conhecimento em termos de regras, escritas usando a linguagem TRIPLE (SINTEK; DECKER, 2002), para expressar axiomas que não podem ser definidos nas ontologias;
- Combinação de recursos e requisitos baseados em regras - estas regras definem restrições para a realizar as combinações entre recursos e requisitos e envolvem os dados ontologias e as regras de conhecimento do domínio.

Comparativamente, nosso trabalho propõe o uso da linguagem de representação OWL e o mecanismo de inferência baseado em classificação hierárquica e inferência de tipos, através de um mecanismo de inferência externa que será discutido nos próximos capítulos, sem considerar o uso de regras. Adicionalmente, o foco das ontologias em nosso trabalho inclui, além da descrição de recursos da grade, a descrição de aplicações. A nossa estratégia de exploração semântica busca integrar, através das ontologias e mecanismos de inferência, tarefas relevantes no contexto de grades, tais como, recuperação de aplicações e escalonamento de recursos para execução dessas aplicações.

2.2.4 Virtual Grid (VG)

O trabalho apresentado em (ZHANG et al., 2006) descreve uma estratégia de escalonamento de aplicações em grades, baseada na realização da tarefa de seleção dos recursos computacionais antes da execução do algoritmo de escalonamento.

²[urlhttp://protege.stanford.edu](http://protege.stanford.edu)

A *Virtual Grid* não utiliza uma abordagem semântica. Contudo, os critérios de seleção de recursos são baseados em uma taxonomia para classificar aplicações e recursos em grupos. A taxonomia define categorias para classificar recursos baseadas na maior ou menor heterogeneidade de um grupo de recursos e no maior ou menor poder de comunicação entre os recursos. De modo semelhante, as aplicações são classificadas em categorias de acordo com seus requisitos de maior ou menor heterogeneidade ou de recursos de comunicação. A classificação concreta dos recursos e aplicações se dá segundo critérios quantitativos.

Apesar de não empregar uma abordagem semântica, a hierarquia da taxonomia definida no trabalho, sugere uma interessante possibilidade de sua exploração para a pré-seleção de recursos no escalonamento de aplicações na grade. Esta possibilidade será explorada e avaliada em nosso trabalho.

2.3 Reutilização de aplicações

A reutilização de aplicações em grades computacionais pode ser uma tarefa desafiadora em razão da possibilidade de existir um grande número de versões de aplicações disponíveis, as quais podem se diferenciar sob diferentes aspectos.

O gerenciamento de configurações de software, conforme (CONRADI; WESTFECHTEL, 1998), atende às necessidades de controle e gerenciamento de software, por parte dos gerenciadores de projeto, e de desenvolvimento de software, por parte dos desenvolvedores, propondo, particularmente, modelos para controle de versões de software no contexto dessas necessidades. Nossa abordagem se distingue desses contextos porque pretende ser usada sob uma perspectiva mais geral, visando atender as necessidades de um usuário indistinto da grade que deseje identificar relações no espaço de produtos e versões das aplicações registradas na grade. Por exemplo, um usuário pode estar buscando uma aplicação associada

a um determinado domínio, ou a problemas e abordagens relacionados, dentro de um domínio específico. Nesse contexto, uma ontologia que descreve conceitos sobre versões de aplicações pode ser útil para a descoberta de variantes alternativas de software adequadas às necessidades, restrições e preferências definidas pelo usuário.

2.3.1 SOFA/DCUP

O trabalho descrito em (GERGIC, 2003) propõe um modelo de controle de versões para construção de software baseado em componentes, no contexto de um repositório de versões dentro do ambiente de componentes SOFA/DCUP (PLASIL; BALEK; JANECEK, 1998) apud (GERGIC, 2003). O modelo proposto contempla tanto o desenvolvimento de componentes quanto a composição de aplicações baseada em componentes.

Apesar de não adotar uma abordagem semântica nem ter sido pensado para o contexto de grades, o trabalho apresenta uma bem definida hierarquia de conceitos relacionados ao controle de versões de aplicações. Vários conceitos presentes na hierarquia de modelagem, tais como propriedades relacionais reflexivas, transitivas e simétricas, também vão ser encontradas na linguagem OWL de modelagem semântica, apresentada em detalhes no próximo capítulo.

Essa hierarquia classifica os componentes e versões com base em múltiplos critérios. Uma linguagem de consulta específica, *M-Cube* foi desenvolvida para a realização de consultas sobre o conteúdo do repositório de versões.

Na perspectiva de controle de versões, nosso trabalho se diferencia da proposta em (GERGIC, 2003) por propor um modelo mais geral para o controle de versões de aplicações que pode ser usado para além do contexto de aplicações baseadas em componentes.

2.3.2 KOntoR

KOntoR, descrita em (HAPPEL et al., 2006), é uma abordagem para reutilização de software apoiada em ontologias. As ontologias em *KOntoR* tratam de diversos aspectos relacionados à reutilização de software, tais como licença de uso, desenvolvedores e descrição de atributos de componentes. O modelo de reutilização se apóia em três dimensões conceituais: artefato, utilidade e representação. A representação significando os metadados que vão maximizar a utilidade (e o reuso) dos artefatos.

KOntoR explora de forma criativa o emprego de semântica para potencializar o uso de metadados disponíveis sobre componentes, no sentido de promover a integração desses metadados. Essa abordagem pode ser aplicada tanto em ambiente de desenvolvimento quanto em atividades em tempo de execução, como por exemplo na descoberta e carga automática de bibliotecas interdependentes.

A abordagem semântica em *KOntoR* se apóia em uma arquitetura com dois elementos chave:

- Um repositório de metadados baseado em XML, para descrever artefatos de software de modo independente de um formato específico;
- Um componente de conhecimento que compreende uma infraestrutura de ontologia e inferência para explorar o conhecimento sobre os artefatos de software. Para permitir o gerenciamento de metadados de modo arbitrário, o componente de metadados é baseado sobre um metaesquema particular. Diferentes formatos de metadados podem ser aplicados para descrever diferentes aspectos dos artefatos e podem ser mapeados através do metaesquema proposto.

Nosso trabalho, em comparação, difere em vários aspectos, tais como, a lin-

guagem de representação e mecanismos de inferência usados na implementação. Particularmente, nossa proposta tem o foco voltado para situações de melhoria da eficiência de uso da grade através da integração semântica de atividades na grade, como busca de aplicações e escalonamento de recursos.

2.4 Projetos em e-science

Projetos voltados para atender à grande demanda de recursos computacionais em aplicações científicas estão na origem da grade semântica. Um aspecto interessante dos projetos de *e-science* é que esses mesmos projetos se constituem também nos alvos da próxima geração de grades, justamente pela demanda continuamente crescente de novos e sofisticados recursos computacionais.

Consideramos importante a descrição de trabalhos desse porte neste capítulo que trata de trabalhos relacionados para atender o objetivo de esboçar uma visão mais ampla do espectro de grades semânticas. Para efeito de comparação com o nosso trabalho, serão considerados, evidentemente, apenas aspectos isolados de cada projeto.

2.4.1 ICENI

A plataforma de pesquisa ICENI (ICENI, 2007; FURMENTO W. LEE; DARLINGTON, 2002) é voltada para o apoio à comunidade científica britânica e é desenvolvida pelo *London e-Science Centre* (LESC, 2007). A arquitetura ICENI define protocolos para captura de políticas e restrições de usuários, construção de aplicações baseadas na descrição de comportamento e desempenho esperados sobre recursos específicos, descrição de características e políticas de uso de recursos e controle distribuído e cooperativo de aplicações (NEWHOUSE et al., 2002).

O *middleware* de grade do ICENI foi implementado inicialmente em Jini (SUN,

2007). Uma revisão da arquitetura (ICENI II), apresentada em (MCGOUGH; LEE; DARLINGTON, 2005), propõe o desenvolvimento de ICENI sobre serviços Web.

A plataforma ICENI emprega mecanismos semânticos baseados em ontologias e metadados, orientados a serviços e voltados para atender principalmente as necessidades de recursos computacionais de usuários da grade e de suas aplicações. Esses mecanismos permitem a composição automática de serviços para atender às necessidades de usuários e provedores. Um exemplo de implementação desse tipo de mecanismo de grade semântica no ICENI é a validação semântica do modelo de programação do ICENI. O modelo de programação proposto pelo ICENI é baseado em componentes cujo ciclo de vida é dividido em estágios bem definidos: especificação, otimização e execução. A validação semântica é realizada na fase de otimização e é baseada em conceitos de orquestração de *workflows* (MCGOUGH; LEE; DARLINGTON, 2005).

Na fase de otimização, o conhecimento científico sobre as ações executadas por um componente e sobre os significados de entradas e saídas desse componente pode ser usado para determinar se a conexão de dois componentes faz sentido. Nessa fase, o ICENI seleciona o melhor uso de recursos e implementações disponíveis na grade para tentar atender os critérios definidos pelos cientistas para uma aplicação. Esses critérios definem geralmente restrições de tempo de execução ou de confiabilidade da execução ou do nível de confiança exigido dos recursos e componentes usados. No exemplo em (MCGOUGH; LEE; DARLINGTON, 2005), argumenta-se que, embora seja sintaticamente correto fornecer uma matriz como entrada para um código que implementa um algoritmo de diferença finita, essa entrada fará pouco sentido se a matriz for uma solução diagonal para um conjunto de equações lineares.

A abordagem semântica do ICENI tem um dos seus focos na orquestração de *workflows*. Comparativamente, o nosso trabalho busca integrar vários aspectos

das atividades de grade através de ontologias e mecanismos semânticos.

2.4.2 Proteus

Proteus (CANNATARO et al., 2004) é um “Ambiente para Resolução de Problemas” (*Problem Solving Environment, PSE*). Problemas relacionados à bioinformática e seus requisitos constituem o domínio do sistema. Proteus é baseado em grades computacionais e usa ontologias e metadados para melhorar a composição de aplicações no domínio de bioinformática.

Ontologias e metadados são empregados no sistema Proteus para simplificar o uso de ferramentas de software disponíveis e a modelagem de aplicações na grade. Ontologias são usadas para construir a base de conhecimento do Proteus através da modelagem de recursos e processos no domínio de bioinformática. Dois tipos de ontologia compõem a sua base semântica. Uma ontologia de domínio descreve e classifica conceitos biológicos além de recursos de bioinformática, tais como, ferramentas de software e bases de dados. Uma ontologia de aplicações descreve e classifica as principais aplicações em bioinformática, descritas como *workflows*.

Especificamente, Proteus pode ser usado para auxiliar os usuários nas seguintes tarefas:

- escolher entre as aplicações disponíveis na grade através da comparação das descrições dessas aplicações com base na formulação de problemas; novas aplicações podem também ser definidas pela integração de componentes de software;
- potencializar os serviços de escalonamento e balanceamento de carga na grade através da escolha eficiente dos recursos disponíveis em um dado momento;

- interferir sobre tarefas computacionais através da visualização e análise de resultados e de histórico de execuções usando ferramentas gráficas e de controle de alto nível.

Atualmente, Proteus é implementado com base no projeto KNOWLEDGE GRID (CANNATARO; TALIA, 2003), uma arquitetura de software dirigida para sistemas PDKD (*Parallel and Distributed Knowledge Discovery*) geograficamente distribuídos. Proteus estende KNOWLEDGE GRID pela adição dos módulos para gerenciamento e armazenamento de ontologias.

O foco da abordagem semântica do Proteus sobre bioinformática constitui o principal diferencial em relação à nossa proposta. Nosso trabalho propõe uma abordagem mais geral, definida sobre um conjunto extensível de ontologias e metadados para o gerenciamento e execução de aplicações relacionadas a diferentes domínios na grade computacional.

2.5 Conclusão

No presente capítulo, descrevemos um conjunto de trabalhos que, de diferentes maneiras, têm intersecção com o problema tratado nesta tese e com a abordagem proposta para a sua solução. Alguns desses trabalhos empregam diferentes tratamentos para a descrição semântica da estrutura e funcionamento de grades computacionais e para a utilização dessa estrutura na integração e gerenciamento de recursos na grade. Outros trabalhos não adotam a abordagem semântica como estratégia, mas descrevem soluções criativas e interessantes associadas ao problema tratado.

O foco do próximo capítulo é a fundamentação teórica da tese, com o objetivo de tratar, de forma ampla, o conceito de grades semânticas.

3 Grades Semânticas

O presente capítulo trata da fundamentação teórica relativa a Grades Semânticas, um dos temas centrais da tese. O detalhamento do assunto inclui os seguintes tópicos:

- grades semânticas e ontologias;
- linguagens atualmente relevantes para a modelagem semântica, com ênfase sobre a linguagem OWL (W3C, 2007d);
- o paradigma clássico de modelagem;
- mecanismos e políticas para exploração da base de conhecimentos baseados na inferência de informações;
- estratégias de recuperação de informações em ontologias OWL.

3.1 Grades Semânticas

A Grade Semântica é uma extensão da Grade computacional “na qual são dados significados bem definidos à informação e aos serviços, capacitando computadores e pessoas a trabalharem cooperativamente” (ROURE; JENNINGS; SHADBOLT, 2005). O uso da palavra “semântica”, nesse contexto, não implica, evidentemente, na captura do significado das informações por parte das máquinas, mas

que conteúdo com significado lógico pode ser mecanicamente manipulado pelos computadores com finalidades úteis para os seres humanos.

Visitando a definição de Web Semântica, constatamos que 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 computadores e pessoas a trabalharem cooperativamente” (BERNERS-LEE; HENDLER; LASSILA, 2001). 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 (CANNATARO; TALIA, 2004).

Em sua origem, o conceito de Grade Semântica desempenha o papel de uma conexão entre os domínios de grades computacionais e de Web semântica, como representado na Figura 3.1, extraída de (ROURE, 2005).

Figura 3.1: Posição da grade semântica no contexto de Grades

O conhecimento em grades semânticas pode ser pensado sob duas visões complementares e interdependentes como “conhecimento ou semântica na grade” e “conhecimento ou semântica para a grade” (GOBLE et al., 2004).

A primeira visão, que também pode ser entendida como uma visão de “co-

nhecimento 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.

3.2 Grades baseadas em conhecimentos

Grades baseadas em conhecimento são consideradas a próxima geração de grades computacionais.

Segundo Moore (MOORE, 2001), conhecimento corresponde a todo relacionamento mantido entre dois itens de informação. No mesmo trabalho, grades baseadas em conhecimentos são definidas como extensões de grades de dados que provêem mecanismos para integrar o gerenciamento de relacionamentos com o gerenciamento de informações e dados.

Grades de dados constroem uma infra-estrutura de gerenciamento de dados sobre ambientes de grade para prover acesso a um conjunto potencialmente grande de objetos digitais. Sua maior capacidade é o apoio à descoberta de dados.

Desse modo, grades baseadas em conhecimento são construídas sobre o gerenciamento de informações provido por grades de dados e sobre mecanismos de interoperabilidade providos pela grade global.

São exemplos de integração de gerenciamento de relacionamentos com gerenciamento de informações e dados (MOORE, 2001):

- Relacionamentos lógico/semântico são mantidos por bibliotecas digitais, através de *crosswalk*, para gerenciar interoperabilidade semântica entre duas coleções. O termo *crosswalk* identifica uma técnica de mapeamento entre diferentes arcabouços de metadados.
- Relacionamentos temporal/procedimental são mantidos por sistemas de *workflow* que acompanham o processamento de documentos.
- Relacionamentos espacial/estrutural são mantidos por sistemas de informação gráfica para imagens co-registradas.
- Relacionamentos funcionais são expressos em algoritmos científicos usados para extração de características.

De acordo com (CANNATARO; TALIA, 2004), três classes de serviços são consideradas essenciais para grades baseadas em conhecimento:

- Serviços baseados em ontologias para gerenciamento de conhecimento são necessários para construir, manipular e interoperar a base de conhecimento da grade de modo homogêneo;
- Serviços de descoberta de conhecimento são responsáveis pela extração de conhecimentos armazenados na base de conhecimentos da grade;
- Adaptação e descoberta dinâmicas de recursos.

A Figura 3.2, extraída de (CANNATARO; TALIA, 2004), mostra as tecnologias que deverão ser integradas para compor a próxima geração de grades. O eixo das abscissas representa o crescimento da distribuição, mobilidade e ubiquidade,

enquanto o eixo das ordenadas representa o crescimento da expressividade dos modelos.

Figura 3.2: Tecnologias chave para grades

As questões discutidas no restante do capítulo envolvem conceitos originalmente relacionados à Web semântica. Tais questões, contudo, podem ser livremente associadas ao contexto de grades semânticas sem perda de coerência.

3.3 Ontologias e linguagens de modelagem

Ontologias, conforme definido em (PATEL-SCHNEIDER; HORROCKS, 2006), são vocabulários de termos que podem ser usados em anotações semânticas e cujo significado é formalmente especificado e pode ser acessado e processado automaticamente.

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

mento podem ser construídas (SWARTOUT; TATE, 1999). Uma ontologia define o vocabulário do domínio de um problema e restrições sobre as possíveis combinações de termos na modelagem do domínio. 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 (DEVEDZIC, 2002).

Trabalhos pioneiros para realização da Web semântica foram voltados para o desenvolvimento de linguagens tais como *Resource Description Language* (RDF) (W3C, 2007b) e *Web Ontology Language* (OWL). OWL é uma linguagem padrão recente para representação de ontologias, desenvolvida pelo *World Wide Web Consortium* (W3C). Essas linguagens podem ser usadas tanto para estender o conteúdo da Web com anotações semânticas quanto para a definição de ontologias.

3.3.1 A linguagem RDF

Resource Description Framework (RDF) é o padrão W3C para representação de conhecimento na web semântica. A padronização visa facilitar a construção de aplicações, particularmente em sistemas descentralizados, permitindo expressar informações de forma flexível, atribuir identificadores únicos de modo descentralizado para eliminar referências ambíguas, e compartilhar vocabulários distintos, porém semanticamente relacionados. RDF provê um padrão para esses requisitos (TAUBERER, 2006).

RDF provê um arcabouço comum para expressar informações que precisam ser processadas por aplicações, além de permitir compartilhar essas informações entre as aplicações sem perda de significado (W3C, 2006; WALSH, 1998). A linguagem RDF é voltada para a representação de metadados sobre recursos da Web, incluindo coisas que podem ser identificadas, ainda que não possam ser di-

retamente recuperadas. Os recursos são descritos, em termos de propriedades e valores de propriedades, como um grafo cujos nós e arcos representam os recursos, suas propriedades e valores, em declarações na forma de triplas contendo sujeito, predicado e objeto. São dois os formatos de arquivos que possibilitam o compartilhamento de dados em RDF, os formatos RDF/XML e N3/Turtle (TAUBERER, 2006).

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 exemplo abaixo, extraído de (TAUBERER, 2006), o nó *Description* é o único do tipo *resource*. Os nós *resource* contêm apenas nós *property*. Estes últimos são predicados sobre os nós *resource*.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:edu="http://www.example.org/">
  <rdf:Description rdf:about="http://www.princeton.edu">
    <geo:lat>40.35</geo:lat>
    <geo:long>-74.66</geo:long>
    <edu:hasDept rdf:resource="http://www.cs.princeton.edu"
      dc:title="Department of Computer Science"/>
  </rdf:Description>
</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 abaixo, extraído de (TAUBERER, 2006), mostra a mesma

informação do exemplo anterior, descrita na notação N3/Turtle.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84\_pos\#> .
@prefix edu: <http://www.example.org/> .

<http://www.princeton.edu> geo:lat "40.35" ; geo:long "-74.66"
<http://www.cs.princeton.edu> dc:title "Dept. of Computer Science" .
<http://www.princeton.edu> edu:hasDept <http://www.cs.princeton.edu> .
```

A notação Turtle (Terse RDF Triple Language) (BECKETT, 2006), é uma extensão de N3 (N-Triples) (BERNERS-LEE, 2006). Essas notações são comumente tratadas sem distinção, mas existem diferenças entre elas que podem afetar o resultado de analisadores sintáticos (*parsers*).

A linguagem RDF é baseada no conceito de identificadores Web chamados identificadores uniformes de recursos (URI)¹. Um identificador URI pode ser criado para referenciar desde qualquer coisa acessível pela rede, tais como documentos, imagens e serviços, até coisas não acessíveis pela rede, como empresas e pessoas, além de conceitos abstratos. Para obter tal generalidade na descrição de recursos, a linguagem RDF usa referências URI (ou URIsrefs). Uma URIsref é uma URI junto com um fragmento identificador, podendo conter caracteres definidos no padrão de codificação de caracteres UNICODE (UNICODE, 2007), como no exemplo mostrado no próximo parágrafo.

A URIsref `http://www.co-ode.org/onto/2005/06/15/meta.owl#ValuePartition` consiste do identificador URI “`http://www.co-ode.org/onto/2005/06/15/meta.owl`” e do

¹RFCs relacionados a URI: <http://gbiv.com/protocols/uri/rfc/>

fragmento *ValuePartition*, separados pelo caractere “#”. Um recurso é globalmente definido em RDF como algo identificável por uma URIref, de modo que o uso de URIrefs permite à linguagem RDF descrever praticamente qualquer coisa assim como estabelecer relacionamentos entre essas coisas. Desse modo, uma URI definida em um documento RDF pode ser usada em outro documento RDF com o mesmo significado com o qual foi criado, isto é, referenciando o mesmo recurso, pois a URIRef identifica univocamente o recurso e permite o compartilhamento desse significado.

3.3.1.1 RDF Schema

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 RDFS.

A linguagem de descrição de vocabulário RDFS (*RDF Schema*) (W3C, 2007a) é uma extensão semântica de RDF que define classes e propriedades que podem ser usadas para descrever classes, propriedades e outros recursos em domínios específicos.

Comparada aos sistemas orientados a objetos, enquanto nesse paradigma as classes são definidas em termos de propriedades que suas instâncias podem ter, na linguagem RDFS as propriedades podem ser descritas em termos das classes de recursos para as quais elas se aplicam, através dos conceitos *Domain* e *Range*, definidos em RDFS, entendidos, respectivamente, como domínio e valores da propriedade. Por exemplo, uma propriedade *ex:locador* pode ser definida para um domínio do tipo *ex:imóvel* e valores do tipo *ex:pessoa*, enquanto que um sistema orientado a objetos definiria uma classe *Imóvel* com um atributo chamado *Locador* do tipo *Pessoa*. Na abordagem RDFS, centrada no conceito de propriedade, novas propriedades podem ser adicionadas sem precisar que a

classe seja redefinida.

3.3.2 A linguagem OWL de modelagem semântica

A linguagem OWL é uma extensão da linguagem RDF (MANOLA; MILLER, 2004) e é derivada da linguagem DAML+OIL (PAGELS, 2005) para representação de ontologias.

O foco da linguagem OWL (MCGUINNESS; HARMELEN, 2004) são as aplicações que precisam processar conteúdo da informação, não se limitando apenas a apresentar informações às pessoas. OWL pode ser usada para descrever ontologias contendo o significado explícito de termos em vocabulários e o relacionamento explícito entre esses termos.

A linguagem OWL oferece recursos adicionais para interpretação de conteúdo que não estão presentes em linguagens como XML, RDF e RDF Schema (RDFS), tais como relacionamento entre classes (por exemplo, disjunção), cardinalidade, igualdade, características de propriedade (por exemplo, transitividade), e classes enumeradas, também conhecidas como *nominals*. Esses recursos facilitam o processamento e integração automáticos de informação disponível na Web e em grades computacionais.

Três sublinguagens provêem diferentes níveis crescentes de expressividade da linguagem OWL, atendendo a diferentes requisitos. A sublinguagem OWL *Lite* atende principalmente a requisitos de classificação hierárquica e restrições simples. OWL *Lite* estende a linguagem RDF, mas não usa todas as características da linguagem OWL, enquanto OWL-DL e OWL *full* usam o vocabulário OWL completo. Contudo, OWL-DL está sujeita a algumas restrições. OWL-DL visa obter o máximo de expressividade sem comprometer completude e decidibilidade. Isso significa, respectivamente, garantir que todas as conclusões são computáveis e que são realizadas em um tempo finito. OWL *full* oferece expressividade máxima somada à liberdade sintática da linguagem RDF, sem oferecer qualquer garantia computacional. OWL *full* pode ser entendida como uma extensão irrestrita de RDF, enquanto OWL-DL pode ser vista como uma extensão restrita de RDF com expressividade maximizada dentro dessas restrições.

Uma ontologia OWL consiste de indivíduos (também chamados de instâncias), propriedades e classes. Os próximos parágrafos descrevem esses conceitos e outras características relevantes das sublinguagens de OWL.

- OWL *Lite* - Características herdadas de RDF-S

Class: uma classe define um grupo de indivíduos que compartilham um conjunto de propriedades. Cada indivíduo de uma classe é chamado uma instância da classe e o conjunto de indivíduos associados a uma classe é chamado extensão da classe. O significado intensional, isto é, a lista de características de uma classe, é relacionado, mas não igual à extensão da classe. Desse modo, duas classes diferentes podem ter a mesma extensão. Classes em OWL podem ser descritas através da definição de um nome para a classe e pela adição de restrições sobre a extensão da classe, ou simplesmente declarando restrições sobre a extensão de uma classe anônima. A classe *Thing* é a classe raiz da hierarquia de classes OWL. Todo indivíduo em uma base de conhecimento é uma instância da classe *Thing*.

Individual: indivíduos são instâncias de classes e relacionam-se entre si através de propriedades.

rdfs:subClassOf: é usada para declarar que uma classe é subclasse de uma outra classe e criar uma hierarquia de classes.

rdf:Property: é usada para estabelecer relacionamentos entre indivíduos (*ObjectProperty*) ou de indivíduos para valores de dados (*DatatypeProperty*).

rdfs:subPropertyOf: declara que uma propriedade é subpropriedade de outra, criando uma hierarquia de propriedades e possibilitando inferir que um indivíduo relacionado através de uma propriedade também o será através de outra propriedade da qual a primeira foi declarada subpropriedade.

rdfs:domain: o domínio de uma propriedade limita os indivíduos para o quais a propriedade pode ser aplicada. Declarar uma classe como domínio de uma propriedade implica que um indivíduo que se relaciona com outro através dessa propriedade pertence à classe que foi declarada domínio dessa mesma propriedade.

rdfs:range: o valor (*range*) de uma propriedade limita os indivíduos que a propriedade pode ter como valores. Se uma classe é declarada valor de uma propriedade e um indivíduo se relaciona com outro através dessa propriedade então o outro indivíduo pertence à classe que foi declarada valor dessa mesma propriedade.

- OWL *Lite* - Igualdades e desigualdades

equivalentClass: é usada para declarar classes sinônimas, implicando que classes equivalentes têm as mesmas instâncias.

equivalentProperty: é usada para declarar propriedades sinônimas. Isso implica que indivíduos relacionados através de uma propriedade também o são através de propriedades sinônimas. Propriedades equivalentes também são subpropriedades umas das outras.

sameAs: usado para criar diferentes nomes para um mesmo indivíduo.

differentFrom: estabelece que um indivíduo é diferente de um outro indivíduo. Usada principalmente para explicitar que dois nomes não se aplicam a um mesmo indivíduo, reforçando a hipótese de “nome único” (ver *Seção 3.4*).

allDifferent: pode ser entendida como uma extensão da característica anterior para explicitar que indivíduos em um grupo são mutuamente distintos.

- OWL *Lite* - Características de propriedades

inverseOf: estabelece que uma propriedade é inversa de outra. Se P1 e P2 são propriedades inversas e um indivíduo X se relaciona a um indivíduo Y através de P1 então Y se relaciona a X através de P2. Considerando por exemplo, uma ontologia na qual as propriedades *implements* e *isImplementedBy* são declaradas inversas e a declaração E1 *implements* A1 expressa o fato de que um arquivo executável E1 (instância de uma classe *Executable*) implementa uma aplicação A1 (instância de uma classe *Application*), então pode-se inferir que A1 *isImplementedBy* E1.

TransitiveProperty: declara que uma propriedade é transitiva. Por exemplo, se em uma ontologia a propriedade `isReplaceableBy` é declarada transitiva e as instâncias M1, M2 e M3 da classe `Method` se relacionam de acordo com as declarações M1 `isReplaceableBy` M2 e M2 `isReplaceableBy` M3, pode-se então inferir que M1 `isReplaceableBy` M3.

SymmetricProperty: declara que uma propriedade é simétrica. Tomando o exemplo anterior, se a ontologia declara que a propriedade `isReplaceableBy` é simétrica, se M1 `isReplaceableBy` M2, então M2 `isReplaceableBy` M1.

FunctionalProperty: declarar que uma propriedade é funcional significa que ela só pode assumir um único valor. Isso equivale a declarar que a cardinalidade mínima da propriedade é zero e a cardinalidade máxima da propriedade é 1. Se uma ontologia declara, por exemplo, que a propriedade `implements` é funcional, pode-se então inferir que uma instância da classe `Executable` implementa no máximo uma instância da classe `Application`.

InverseFunctionalProperty: declarar que uma propriedade é funcional inversa implica que a inversa da propriedade é funcional. No exemplo anterior, poderíamos declarar a propriedade `isImplementedBy` como funcional inversa da propriedade `implements`.

- OWL *Lite* - Restrições de quantificadores sobre propriedades

allValuesFrom: o quantificador universal restringe o valor (*range*) de uma propriedade sobre uma determinada classe. Por exemplo, a classe `Method` pode ter a propriedade `isReplaceableBy` restrita pelo quantificador universal para `Method`. Isso implica que se um indivíduo M1, da classe `Method`, é relacionado a um indivíduo X, então pode-se inferir que X é uma instância da classe `Method`. Essa restrição permite reusar a propriedade com outras classes declarando um valor de restrição adequado a cada situação.

someValuesFrom: o quantificador existencial restringe uma propriedade em relação a uma classe para que pelo menos um valor para aquela propriedade seja

de um certo tipo. Por exemplo, a propriedade `implements` relacionando a classe `Application` à classe `Executable` pode ser restrita pelo quantificador existencial. Essa restrição garante que pelo menos um indivíduo relacionado a uma instância da classe `Application` através da propriedade `implements` seja uma instância da classe `Executable`.

- *OWL Lite* - Restrições de Cardinalidade - as restrições de cardinalidade em *OWL Lite* são limitadas a 0 ou 1 e são atribuídas a uma propriedade em relação a uma determinada classe.

minCardinality: definir cardinalidade mínima igual a 1 para uma propriedade em relação a uma classe equivale a exigir um valor para essa propriedade para todas as instâncias dessa classe; definir cardinalidade mínima zero implica, na ausência de informação mais específica, que a propriedade é opcional em relação à classe.

maxCardinality: definir cardinalidade máxima igual a 1 para uma propriedade em relação a uma classe implica que toda instância dessa classe se relaciona a no máximo um indivíduo através dessa propriedade; definir cardinalidade máxima zero estabelece que a classe não possui valor para aquela propriedade.

cardinality: serve para definir cardinalidades mínimas e máximas iguais a zero, ou iguais a 1, para uma propriedade em relação a uma classe.

- *OWL Lite* - Intersecção de Classes

intersectionOf: usada para declarar intersecções de classes com nome e restrições. Por exemplo, em uma possível ontologia sobre aplicações paralelas, uma classe nomeada `BSPLibrary` poderia ser descrita como uma intersecção de uma classe `Library` com uma classe `BSPApplication`. Esta, por sua vez, poderia ser definida como instâncias da classe `Application` que implementam (`implementsModel`) instâncias da classe `BSPModel`.

- *OWL Lite* - *Datatypes*: os tipos de dados OWL são emprestados de RDF e XML Schema. O nosso trabalho vai se limitar à descrição do emprego correto da sintaxe.

Mais detalhes sobre *Datatypes* podem ser encontrados em (BECHHOFFER et al., 2004).

- *OWL Lite - headers*: os cabeçalhos das ontologias aceitam inclusão de ontologias, relacionamentos e informações gerais sobre a ontologia.
- *OWL Lite - annotations*: indivíduos, propriedades, classes e cabeçalhos das ontologias podem receber anotações; contudo, essas anotações sofrem algumas restrições, não podendo por exemplo ser usadas em axiomas de propriedades, tais como subpropriedades e restrições de domínio e valor.
- *OWL Lite - versions*: As declarações do tipo: *owl:versionInfo*, *owl:priorVersion*, *owl:incompatibleWith*, *owl:deprecatedClass*, *owl:backwardCompatibleWith* e *owl:deprecated:Property* permitem monitorar a evolução das ontologias OWL. Informações detalhadas sobre as características declaradas acima são encontradas em (BECHHOFFER et al., 2004).
- Características OWL-DL e OWL-full

oneOf: usado para descrever classes através da enumeração de indivíduos; os membros da classe são exatamente o conjunto de indivíduos enumerados. Por exemplo, uma classe **Season** pode ser definida simplesmente pela enumeração dos indivíduos **Summer**, **Winter**, **Autumm** e **Spring**.

hasValue: usado em uma propriedade para descrever uma classe restrita a todos os indivíduos para os quais a propriedade tem o valor definido através de *owl:hasValue*. Por exemplo, uma classe de aplicações MPI poderia ser descrita através da propriedade *implementsModel* restrita por *owl:hasValue* = "MPI".

disjointWith: declara que classes são disjuntas entre si. Nesse caso, pode-se inferir estados inconsistentes se indivíduos forem declarados instâncias de um conjunto de classes disjuntas, ou ainda que um indivíduo de uma certa classe não é instância de classes declaradas disjuntas da sua classe.

unionOf, *complementOf*, *intersectionOf*: usadas para descrever combinações

booleanas de classes e restrições.

minCardinality, *maxCardinality*, *cardinality*: estende as restrições de cardinalidade de OWL *Lite* permitindo declarações de cardinalidade para quaisquer valores inteiros não negativos.

classes complexas: extensão OWL *full* para declaração arbitrária de classes complexas em diversas estruturas não permitidas em OWL *Lite*. Também em OWL *full* classes podem ser tratadas como indivíduos, isto é, instâncias de outras (meta)classes.

Em OWL, propriedades, *Properties*, são usadas para criar *restrições* e estas são associadas aos quantificadores para restringir os indivíduos que pertencem a uma classe. As restrições definem as instâncias que podem pertencer a uma classe através de propriedades que as relacionam com instâncias de outras classes. Por exemplo, a declaração *hasBase some PizzaBase*, associada à classe *Pizza*, expressa que instâncias da classe *Pizza* se relacionam com instâncias da classe *PizzaBase* através da propriedade *hasBase*. O termo *some*, usado na restrição, identifica o quantificador existencial *someValuesFrom*, descrito no parágrafo anterior, que restringe uma propriedade para uma classe.

As classes sobre as quais são aplicadas restrições podem ser de dois tipos, em razão do tipo de condições associadas a essas restrições (HORRIDGE, 2004):

- Classes Primitivas - Uma classe é *Primitiva* se as restrições sobre seus indivíduos são apenas condições do tipo *Necessária*. Condições *necessárias* podem ser entendidas como: “se alguma coisa é membro desta classe então é *necessário* que satisfaça estas condições”. Diz-se que uma classe primitiva é apenas *descrita*. Assim, descrever uma classe (*Primitiva*) é declarar restrições usando condições necessárias. No exemplo do parágrafo anterior, se a restrição é descrita como necessária, isso implica que, se algo é instância da classe *Pizza*, então é *necessário* para ele ter, pelo menos, uma base do tipo *PizzaBase*.
- Classes Definidas - Uma classe é *Definida* se as restrições sobre seus indivíduos são

condições do tipo *Necessária e Suficiente*. No exemplo do parágrafo anterior, se a restrição é descrita como necessária e suficiente, isso implica que se um indivíduo qualquer é relacionado com a classe *PizzaBase* através da propriedade *hasBase*, então ele é uma instância da classe *Pizza*. Ou seja, essa condição é suficiente para inferir o seu tipo.

Os requisitos e implicações descritos acima são relacionados ao uso de características específicas da linguagem OWL. Esses requisitos serão tratados através de exemplos concretos nos capítulos que abordam a definição e a exploração da ontologia de grades computacionais proposta no presente trabalho.

3.4 O paradigma clássico de modelagem

Esta seção introduz aspectos relevantes do chamado *paradigma clássico* de modelagem associado às linguagens OWL e RDF, discute sua relação com lógica de descrição (BAADER et al., 2003) (*Description Logic - DL*) e o papel do paradigma clássico e das linguagens de modelagem no ambiente de grades semânticas.

O *Paradigma Clássico*, baseado em noções de lógica proposicional, lógica de primeira ordem (FIESER; DOWDEN, 2007; SILVA; FINGER; MELO, 2006) e lógica de descrição, é incorporado em linguagens, tais como OWL, RDF e RDF-Schema. OWL-DL tem correspondência direta com lógica de descrição. Além de sua aplicação no raciocínio sobre valores verdade, a lógica proposicional clássica também pode ser usada como uma linguagem para descrição de conjuntos. A Lógica de descrição e a lógica modal, uma vez que ambas introduzem estruturas e operadores à essa interpretação teórica de conjuntos (OHLBACH; KOEHLER, 1999).

Em lógica de descrição, o conhecimento de um domínio (o “mundo”) é representado, em bases de conhecimento, pela definição dos conceitos relevantes do domínio, ou seja, sua terminologia; e pelo uso dessa terminologia para especificar propriedades de objetos e indivíduos que ocorrem no domínio (BAADER et al., 2003).

Bases de conhecimento apresentam uma clara distinção entre conhecimento intensional e conhecimento extensional. O primeiro é o conhecimento geral sobre o problema do domínio, enquanto o último é específico de um problema particular. De modo análogo, uma base de conhecimento em lógica descritiva é tipicamente organizada em dois componentes: *TBox* e *ABox*. O componente *TBox* contém conhecimento intensional na forma de terminologia, ou taxonomia, de onde deriva o caractere *T* de *TBox*, e costuma não ser considerado sujeito a mudanças. O componente *ABox* contém conhecimento extensional, ou assertivo, de onde deriva o caractere *A* de *ABox*, e costuma ser considerado contingente, isto é, dependente de circunstâncias e, portanto, sujeito a mudanças ocasionais ou mesmo constantes.

Na linguagem OWL, o componente terminológico *TBox* contém axiomas sobre classes, subclasses, classes equivalentes, classes disjuntas e alguns padrões de projeto em ontologias, como os descritos em (GONGG, 2007; RECTOR et al., 2004). O componente assertivo *ABox* contém declarações sobre indivíduos, como tipo, valores de propriedades, igualdade e desigualdade. Uma ontologia OWL completa corresponde a uma combinação de um componente *TBox* e um componente *ABox*.

O *Paradigma Clássico* é baseado em um conjunto de características apropriadas ao ambiente aberto da Web semântica (PATEL-SCHNEIDER; HORROCKS, 2006). Nesse paradigma, um conjunto de objetos e seus relacionamentos representam um domínio sendo modelado. Interpretações ou modelos, descrevendo os estados possíveis desses relacionamentos, são potencialmente infinitos na ausência de outras informações restritivas.

Ontologias no paradigma clássico consistem de um conjunto de declarações, frequentemente referenciados como axiomas, sobre um determinado domínio e admitem uma multiplicidade de interpretações “razoáveis” em relação ao domínio tratado. Ontologias são separadas das interpretações. O conjunto de interpretações disponíveis que são consistentes com a informação disponível (ontologia) dão o significado da informação. A recuperação de informação através de consultas nesse paradigma implica em verificar

se uma situação específica se ajusta a todas as interpretações que são consistentes com a informação disponível.

Outras características do paradigma clássico, tais como a não adoção a priori da hipótese de *nome único*, também conhecida como *Unique Name Assumption (UNA)*; a hipótese de *mundo aberto*, conhecida como *Open World Assumption (OWA)* e o suporte à inferência serão discutidos nos próximos parágrafos como parte da fundamentação teórica relacionada à exploração da nossa proposta de grade semântica.

O paradigma clássico não assume, a priori, a hipótese de *nome único*; isso torna possível identificar um mesmo objeto através de diferentes nomes, além da criação de indivíduos anônimos apenas pela descrição de propriedades desses indivíduos. Por outro lado, esse fato não impede expressar explicitamente nesse paradigma que um determinado conjunto de nomes sejam considerados “únicos”.

A hipótese de *mundo aberto* atende uma característica da Web semântica, para a qual não existe obrigação de que uma fonte de informação seja completa sob nenhum aspecto. Em tais casos, falta de informação não implica negação, como no paradigma de *mundo fechado*. Uma consulta nesse paradigma pergunta sobre “o que é conhecido”. Nesse contexto, uma resposta negativa implica impossibilidade de responder. O modelo de banco de dados relacional e banco de dados orientado a objetos são exemplos do paradigma de mundo fechado.

Contudo, se necessário, recuperação fechada de informações pode ser adicionada através de uma restrição permitindo inferir informação completa sob a hipótese de mundo aberto. A linguagem SPARQL é uma linguagem de consulta para RDF que pode ser usada produtivamente nesse contexto.

3.5 Mecanismos de raciocínio

Mecanismos de raciocínio 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.

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, desse modo, ser processadas por um mecanismo de inferência (*reasoner*) para alcançar diferentes metas.

Uma importante tarefa de inferência, conhecida como *class subsumption*, implica em testar se uma classe é ou não subclasse de outra, baseada na descrição das classes. Por exemplo, uma hierarquia declarada manualmente (*asserted hierarchy*) serve como base para que um mecanismo de raciocínio possa inferir e manter uma hierarquia de herança múltipla. Tarefas dessa natureza são relacionadas ao conhecimento da definição do domínio. Portanto, em geral, é útil executar tais inferências antecipadamente, uma vez que estas envolvem o domínio como um todo, e não apenas um conjunto limitado de indivíduos específicos.

O mecanismo de raciocínio também pode executar verificação de consistência. Baseado na descrição de uma classe, esse mecanismo pode inferir se essa classe pode ou não ser instanciada, como exposto por Horridge (HORRIDGE, 2004). Adicionalmente, de acordo com Roure (ROURE, 2005), uma vez que ontologias codificam relacionamento entre classes de objetos, inferências entre instâncias dessas classes podem ser realizadas. A Seção seguinte descreve a ferramenta padrão *Pellet*, de código aberto, largamente utilizada no desenvolvimento de aplicações baseadas em ontologias OWL-DL.

3.5.1 A ferramenta de raciocínio 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 (LLC, 2007). *Pellet* atende as recomendações W3C de casos de testes definidas em

(CARROL; ROO, 2004), apud. (SIRIN et al., 2007), para verificadores de documentos OWL: 1) verificação de sintaxe OWL, 2) verificação de consistência OWL para OWL *Lite*, OWL-DL e OWL *full* e 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.

Pellet é construída em Java e amplamente utilizada pela comunidade de Web semântica. *Pellet* integra a verificação de consistência ao seguinte conjunto padrão de serviços de inferência lógica, os quais podem ser acessados através de consultas:

- verificação de consistência - assegura que uma ontologia não contém fatos contraditórios. Na terminologia DL, essa operação verifica a consistência do componente *ABox* da ontologia OWL-DL com o componente *TBox*;
- satisfazibilidade de conceito - verifica se é possível a uma classe (conceito) possuir instâncias;
- classificação - calcula as relações de subclasse existentes entre cada classe nomeada para criar a hierarquia de classes;
- 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, sobre o componente *ABox*, usando linguagens como SPARQL. Uma consulta sobre o componente *ABox*², expressa em SPARQL, é transformada em um formato apropriado para permitir a sua otimização. *Pellet* também realiza verificação sintática automática através de heurísticas que tentarão transformar documentos OWL *full* em OWL-DL e então processá-los normalmente. Na realização de detalhamento e depuração de ontologias, *Pellet* considera a totalidade de características OWL-DL (propriedades inversas

²ver requisitos sobre uma consulta *ABox* em (SIRIN et al., 2007)

e transitivas, raciocínio sobre tipos de dados, restrições de cardinalidade, classes enumeradas e declarações de instâncias).

A arquitetura de *Pellet* é centrada em torno de um pequeno motor de inferência adaptado para sofrer extensões. Esse modelo permitiu o desenvolvimento de interfaces para diferentes ferramentas OWL/RDF, tais como Jena (CARROLL et al., 2003) e *WonderWeb OWL API* (BECHHOFFER; LORD; VOLZ, 2003), ou para apoiar aplicações que se comunicam através da interface DIG (BECHHOFFER; MÖLLER; CROWTHER, 2003). *Pellet* provê diferentes interfaces para carga de ontologias e também implementa interfaces de raciocínio, definidas pelas ferramentas de extensão, para responder consultas.

3.6 Recuperação de informações

As bases de conhecimento em grades computacionais semânticas e na Web semântica existem e 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 no paradigma clássico envolve, como descrito na Seção 3.4, a realização de consultas baseadas em mecanismos de inferência.

Nesta seção, discutiremos 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 as bases de conhecimento. Também, ambas desempenham papel relevante no cenário atual da Web semântica e de grades semânticas em razão de sua larga adoção nesses contextos. Nosso objetivo é avaliar as abordagens apresentadas e identificar soluções eficientes para a definição de uma arquitetura de metadados. Essa arquitetura, proposta no Capítulo 6, deve ser adequada às necessidades de recuperação de informações sobre a ontologia OWL descrita no Capítulo 4, no contexto de grades semânticas. 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, descrevemos características relevantes da linguagem de consulta SPARQL (DODDS, 2006; MCCARTHY, 2005). Essa linguagem foi adotada na implementação da nossa arquitetura e na realização de consultas nos experimentos apresentados no Capítulo 6.

3.6.1 OWL-QL

OWL-QL (FIKES; HAYES; HORROCKS, 2003) é uma linguagem formal e também um protocolo usado por agentes de consulta (clientes) e agentes de resposta (servidores) na condução de um diálogo do tipo “consulta e resposta” sobre uma base de conhecimento representados em OWL. OWL-QL é uma atualização da linguagem de consulta *DAML Query Language* (DQL) (FIKES et al., 2003).

Nos diálogos de consulta e resposta em OWL-QL, o servidor pode usar métodos de raciocínio automáticos para derivar respostas e cenários nos quais o conhecimento usado para responder a 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.

Uma base K de conhecimentos em OWL é considerada uma coleção de sentenças lógicas, KS , que representa uma teoria lógica na qual uma coleção de sentenças vinculadas, KES , são verdadeiras tais que $KS \subseteq KES$. Uma consulta em OWL-QL busca identificar sentenças em KES que satisfaçam um dado esquema de sentenças e conexões para variáveis nessas sentenças como respostas para a consulta.

Um diálogo consulta/resposta em OWL-QL é iniciado quando o cliente envia uma consulta para um servidor. Uma consulta OWL-QL contém, necessariamente, um *padrão de consulta* no qual algumas URIs (cujo conceito é definido na Seção 3.3.2) são consideradas variáveis. O *padrão de consulta* pode ser representado com uma série de triplas da forma ($\langle \text{propriedade} \rangle \langle \text{sujeito} \rangle \langle \text{objeto} \rangle$), onde qualquer item da tripla pode ser uma variável. Nos exemplos e ferramentas disponíveis, as variáveis são indicadas por nomes começando com o caractere “?”. O *padrão de consulta* também inclui uma referência para a base de conhecimento da resposta, no agente servidor, usada na realização da consulta; e, opcionalmente, listas mutuamente exclusivas designando as variáveis como *Must-Bind*, *May-Bind*, e *Don't-Bind*. O significado dessas listas é explicado nos próximos parágrafos.

Uma consulta pode ter zero ou mais respostas, cada uma das quais provê conexões (*bindings*) de URIs ou literais para algumas das variáveis no *padrão de consulta*. Cada conexão em uma resposta a uma consulta é uma URI ou um literal que ou ocorre explicitamente na base de conhecimento da resposta, ou é um termo em OWL. Em outras palavras, OWL-QL é projetada para responder consultas na forma “*Quais URIs e literais da base de conhecimento de resposta e OWL denotam objetos que tornam verdadeiro o padrão de consulta?*”. Diz-se de uma variável que tem conexão em uma resposta a uma consulta que ela é identificada nessa resposta.

OWL-QL suporta respostas existencialmente quantificadas para permitir ao cliente designar quais variáveis da consulta serão aceitas com ou sem conexões, dependendo de qual lista (*Must-Bind*, *May-Bind*, ou *Don't-Bind*) contém uma variável na consulta. As

respostas devem prover conexões para todas as variáveis da lista *Must-Bind-variables*, podem prover conexões para qualquer variável da lista *May-Bind-variables*, e não prover conexões para qualquer variável da lista *Don't-Bind-variables*.

O exemplo abaixo, retirado de (FIKES; HAYES; HORROCKS, 2003), esclarece a semântica do uso das listas de variáveis.

- Considerando uma base de conhecimento contendo as seguintes proposições:

cada Pessoa tem exatamente um Pai;

cada um de um grande número de itens da base, C_i s, é uma Pessoa;

F_k é o Pai de C_k em um pequeno subconjunto de C_i s.

- Considerando uma consulta para solicitar a relação de pessoas e seus pais encontrados na base de conhecimento, que utiliza o seguinte *padrão de consulta* (**hasFather ?p ?f**), a base de conhecimentos deve gerar respostas diferentes para cada um dos casos abaixo:

Se **?f** é uma variável *Don't-Bind*, então o conjunto completo de respostas contém C_i respostas, ou seja, uma para cada pessoa conhecida, mas não vai identificar os pais dessas pessoas;

Se **?f** é uma variável *Must-Bind*, então o conjunto completo de respostas contém apenas C_k respostas, ou seja, uma para cada pai conhecido, e cada resposta identifica a pessoa e o nome do seu pai;

Se **?f** é uma variável *May-Bind*, então o conjunto completo de respostas não redundantes contém C_i respostas, ou seja, uma para cada pessoa conhecida, e cada resposta identifica uma pessoa, e o seu pai, nos casos em que o pai é conhecido. Se o pai não é conhecido, o nome da pessoa aparece sem o nome do pai. No caso anterior, para o quantificador *Must-Bind*, pessoas cujo pai não fossem conhecidos, não apareceriam na resposta.

Uma interessante característica de OWL-QL é a possibilidade de realização de consultas do tipo “se-então” através da inclusão opcional de premissas na consulta. O

uso de premissas na consulta permite, por exemplo, realizar consultas com base em pré-condições dinâmicas que poderiam ser atualizadas para diferentes contextos. Essa característica pode se mostrar útil em ambientes dinâmicos e heterogêneos como é o caso das grades computacionais.

O exemplo descrito anteriormente nesta seção explora diretamente o poder expressivo da linguagem de representação OWL. Outros padrões de consulta sugeridos em (FIKES; HAYES; HORROCKS, 2003) envolvem a realização de consulta no estilo SQL; “otimização iterativa” para obter respostas a consultas envolvendo conceitos, tais como “subclasse mais geral” ou “tipo mais específico”, os quais não são expressos em OWL; e consultas envolvendo questões referentes a “quantos” (por exemplo, quantos computadores possuem uma plataforma específica), que podem gerar respostas inconclusas.

Mais detalhes sobre especificação, exemplos de uso, *download* e outras informações relevantes sobre OWL-QL são encontrados no sítio <http://ksl.stanford.edu/projects/owl-ql>.

3.6.2 A Linguagem SPARQL

A linguagem de consulta SPARQL (W3C, 2007c) é baseada na semelhança de padrões de grafos e é definida em termos do modelo RDF definido pelo consórcio W3C, funcionando para qualquer fonte de dados que possa ser mapeada em RDF. SPARQL é, ao mesmo tempo, uma linguagem de consulta e um protocolo de acesso.

Diversas ferramentas e APIs são disponibilizadas para prover funcionalidade baseada em SPARQL. Uma pequena lista disponível em (DODDS, 2006) inclui, entre outros, ARQ, um processador SPARQL para Jena; KAON2, um mecanismo de inferência com suporte parcial a consultas em SPARQL e Pellet, o mecanismo Java de código aberto, descrito na Seção 3.5.1.

SPARQL, assim como RDF, é construída sobre uma tripla consistindo de sujeito, predicado e objeto. Uma tripla padrão SPARQL pode incluir variáveis e estas 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 abaixo, extraído de (DODDS, 2006), usa variáveis para sujeito e objeto, respectivamente, *element* e *name*.

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

A palavra chave *PREFIX* equivale em sua essência à definição de um espaço de nomes XML. O rótulo atribuído a uma URI pode ser usado em qualquer lugar em uma consulta em substituição à própria URI. A cláusula *SELECT* é usada para definir os itens de dados que serão retornados pela consulta. A palavra chave *FROM* identifica os dados sobre os quais a consulta vai ser executada. Finalmente, a cláusula *WHERE* 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. Essa consulta seleciona nome e número dos elementos do grupo 18 da tabela periódica.

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.

3.7 Conclusão

As discussões em torno dos paradigmas, linguagens e ferramentas que compõem a Web semântica e, por analogia, grades semânticas revelam um domínio em evolução, ainda não totalmente maduro. A definição de padrões para linguagens e ferramentas permite a exploração de projetos concretos voltados para a construção e utilização de bases de conhecimentos eficientes. Diferentes abordagens e ferramentas relacionadas apontam para diferentes soluções.

O próximo capítulo define uma ontologia para grades computacionais, com base em uma metodologia simples. Os resultados obtidos na elaboração dessa ontologia e a discussão sobre ferramentas, linguagens e paradigmas, iniciada no presente capítulo, devem se refletir na arquitetura proposta no Capítulo 6. A arquitetura proposta descreve um modelo que deverá integrar, em uma estrutura colaborativa, as diferentes soluções e ferramentas apresentadas, para atender os requisitos emergentes do contexto de grades computacionais semânticas.

4 Ontologias para o Gerenciamento de Grades Computacionais

A necessidade de descrição do conteúdo do repositório de aplicações do InteGrade foi o ponto de partida da investigação em torno de linguagens para representação de metadados e de mecanismos para sua recuperação e utilização eficientes.

No passo seguinte, essa experiência foi enriquecida pela realização de uma extensa investigação sobre o estado da arte das linguagens de representação semântica, das ferramentas e tecnologias relacionadas e do crescente emprego desse conjunto no contexto de grades, da web e do gerenciamento de recursos e aplicações. O esforço de realização da pesquisa resultou na aquisição de uma visão ampla sobre as possibilidades de exploração da abordagem semântica, suas linguagens, tecnologias e ferramentas para tratar do problema de integração e compartilhamento de aplicações e recursos em grades computacionais. O resultado concreto dessa pesquisa está contida no Capítulo 3.

A segunda constatação importante resultante da pesquisa teórica aponta para as deficiências e os problemas em potencial associados à falta de um vocabulário comum para ser compartilhado pelos diversos usuários da grade, causados pelo grande volume de informação distribuída e pela heterogeneidade de suas fontes, formatos e concepções. Adicionalmente, a ausência de uma estrutura para compartilhamento de significados também afeta as informações disponíveis sobre a própria estrutura e funcionamento da grade.

Diante do exposto, vislumbramos possibilidades de incremento da qualidade e eficiência de algumas tarefas executadas na grade e na utilização do conteúdo nela armazenado, particularmente em relação às seguintes características:

- adoção de significado comum entre os diferentes tipos de conteúdos armazenados na grade através do uso compartilhado de taxonomias e axiomas;
- facilidade de descoberta de conteúdo armazenado;
- reutilização do conteúdo (em razão do incremento da facilidade de descoberta); o que deve afetar positivamente a realização de algumas atividades comuns ao domínio de grades. tais como:
 - descoberta de aplicações registradas na grade para executar uma tarefa na grade;
 - descoberta de componentes registrados na grade para composição de aplicações para a grade;
 - descoberta de versões diferentes, intercambiáveis ou não, de aplicações registradas na grade;
- reutilização dos metadados permitindo a integração de conteúdo;
- eficiência em estratégias de escalonamento de recursos (vide Capítulo 5).

A revisão bibliográfica sobre os trabalhos relacionados e análise crítica desse material foi essencial para que pudéssemos obter uma visão geral dos principais problemas relacionados ao domínio tratado e das soluções propostas no bojo dos trabalhos investigados. Alguns trabalhos relacionados foram particularmente inspiradores, em especial, os trabalhos voltados para a aplicação prática dos metadados semânticos, como por exemplo os trabalhos descritos na Seção 2.2 do Capítulo 2.

Os argumentos apresentados anteriormente suscitaram importantes questões que contribuíram de modo objetivo para orientar o roteiro da nossa investigação. Essa questões estão enumeradas abaixo:

1. Que vocabulários comuns poderiam se definir para descrever as grades, sua estrutura, recursos, conteúdo, em especial aplicações que nela executam, atividades realizadas e políticas de gerenciamento?
2. Que mecanismos seriam necessários para permitir aos usuários e aplicações acesso transparente às informações desejadas ou o funcionamento e interação eficientes na grade?
3. Que princípios, tecnologias, linguagens de representação e ferramentas de manipulação seriam mais apropriadas para exploração eficiente do domínio?
4. Como utilizar mecanismos de inferência disponíveis para explorar individualmente cada cenário de modo eficiente?
5. Como utilizar mecanismos de inferência disponíveis para explorar a integração de cenários de modo eficiente?

Esperamos que as respostas para algumas dessas perguntas possam estar satisfatoriamente respondidas no corpo dos próximos capítulos, particularmente no presente capítulo 4 e no Capítulo 5.

Este capítulo trata da concepção e definição do conjunto de ontologias que vão compor a estrutura semântica para o gerenciamento de grades computacionais. Essas são as primeiras atividades relacionadas à implementação efetiva da proposta da tese.

Os conceitos, propriedades e axiomas definidos nas ontologias devem refletir, de maneiras diferentes, os conceitos tratados nos capítulos de fundamentação teórica. O Capítulo 3 contribui com os conceitos, linguagens e ferramentas usados na construção das próprias ontologias. Por sua vez, o Apêndice A será a fonte de grande parte dos conceitos e relacionamentos descritos pelas ontologias, fornecendo, portanto, aporte teórico sobre grades computacionais e seu conteúdo.

Este capítulo é dividido informalmente em dois blocos principais. O primeiro bloco inclui as Seções 4.1 e 4.2, discute o uso de metodologias para desenvolvimento de ontologias, e emprega uma metodologia específica na concepção e elaboração da base de

ontologias proposta. O segundo bloco descreve, em níveis de detalhamento adequados ao perfil da tese, o conjunto de ontologias da base semântica proposta. Essa descrição inclui classes, propriedades e axiomas que representam os domínios tratados. Decisões e definições implícitas no modelo também são discutidas. A parte final do segundo bloco, elabora uma série de argumentos em torno de cenários genéricos de utilização das ontologias.

4.1 Metodologias para o desenvolvimento de ontologias

Existem atualmente um conjunto considerável de metodologias propostas para o desenvolvimento de ontologias. A Referência (CORCHO; FERNANDEZ-LOPEZ; GOMEZ-PEREZ, 2003) apresenta uma ampla revisão e comparação das principais metodologias, ferramentas e linguagens para desenvolvimento de ontologias. Esse trabalho investiga propostas de metodologias em um período entre 1990 e 2003, cada um seguindo diferentes abordagens.

A maioria das metodologias analisadas propõem 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.

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.

Essas abordagens também podem ser classificadas em relação ao grau de dependência entre a ontologia sendo desenvolvida e sua aplicação final. Sob essa perspectiva, as abordagens são classificadas em dependentes, semi-dependentes e independentes. Metodologias também são classificadas em razão de orientarem a construção de ontologias do zero ou a partir da reutilização de outras ontologias.

Finalmente, nenhuma das abordagens apresentadas na referência é considerada totalmente madura em comparação com metodologias da área de engenharia de software ou da engenharia de conhecimento. Constatou-se adicionalmente a ausência de qualquer iniciativa voltada para a unificação das abordagens. Cada grupo define e aplica sua própria abordagem.

4.2 Desenvolvendo ontologias para grades computacionais

Nossa escolha pela metodologia proposta em (NOY; MCGUINNESS, 2001) foi definida no contexto de um conjunto de decisões tomadas a partir da escolha da linguagem OWL como linguagem para representação das ontologias que integram o nosso trabalho. A linguagem OWL, descrita no Capítulo 3, é uma linguagem padrão para representação de ontologias desenvolvida pelo *World Wide Web Consortium* (W3C) e largamente usada no contexto da Web semântica e de Grades semânticas. A escolha da linguagem para representação das ontologias determinou a adoção da ferramenta Protégé-OWL (KNUBLAUCH et al., 2004) para o desenvolvimento das ontologias. Essa ferramenta permite o desenvolvimento de ontologias de modo incremental e confiável, principalmente graças a uma interface gráfica amigável e à conexão de motores de inferência para verificação de consistência das ontologias. A metodologia descrita em (NOY; MCGUINNESS, 2001) é disponibilizada no conjunto de ferramentas e documentação associadas ao Protégé-OWL e à linguagem OWL, pelos seus desenvolvedores. Esse aspecto de confiabilidade

e simplicidade da metodologia, foram os dois fatores determinantes para a sua adoção.

A metodologia propõe a realização do seguinte conjunto de tarefas:

1. definição do domínio e do escopo cobertos pela ontologia;
2. definição da finalidade da ontologia;
3. identificação dos tipos de questões para os quais a informação na ontologia deve prover respostas;
4. identificação dos atores responsáveis pelo uso e manutenção da ontologia.

A aplicação dessa metodologia possibilita aos desenvolvedores da ontologia refletir sobre as necessidades reais de utilização da ontologia em um contexto. Em nosso trabalho, adotamos a metodologia como ponto de partida para o desenvolvimento do conjunto de ontologias para grades computacionais. As próximas subseções investigam, apresentam e analisam as definições e as identificações relacionadas nas quatro tarefas descritas acima, propostas pela metodologia adotada para o desenvolvimento das ontologias.

4.2.1 Domínio e escopo da ontologia

As ontologias para grades computacionais devem cobrir o domínio referente à integração de recursos de grades computacionais. As classes da ontologia relacionam-se a conceitos como nós de computação da grade, plataformas de hardware e software, aplicações que podem ser executadas na grade, dados produzidos e consumidos por essas aplicações, domínio das aplicações e versões alternativas de aplicações.

4.2.2 Finalidade da ontologia

A principal finalidade da ontologia para grades é apoiar o gerenciamento e integração de diferentes recursos no ambiente heterogêneo das grades. Mais especificamente, a

ontologia deverá facilitar a busca de conteúdo da grade, tais como aplicações e dados e a seleção de recursos adequados para execução das aplicações na grade.

4.2.3 Questões tratadas pela ontologia

Seguindo a metodologia, as definições do escopo e finalidade das ontologias para grades computacionais podem ser melhor compreendidos com o apoio de um conjunto de questões que a base de conhecimento centrada na ontologia deve ser capaz de responder. No domínio analisado, a base de conhecimento deve poder responder a questões como as relacionadas nos exemplos a seguir.

- Existe um executável de uma determinada aplicação adequado para uma certa plataforma?
- Quais plataformas podem executar um binário específico de uma aplicação?
- Quais computadores na grade executam um binário específico de uma aplicação?
- Quais executáveis implementam determinada aplicação?
- Quais computadores na grade executam uma determinada aplicação?
- Quais aplicações codificam determinado algoritmo?
- Quais computadores na grade executam aplicações que codificam determinado algoritmo?
- Quais problemas são relacionados a determinados domínios?
- Quais abordagens propõem soluções para um determinado problema?
- Quais algoritmos descrevem uma determinada abordagem para a resolução de um problema?
- quais executáveis implementam uma solução para um determinado problema?

As questões definidas a seguir envolvem um conjunto de conceitos que ultrapassa o escopo das ontologias propostas em nosso trabalho. Contudo, os conceitos abordados podem ser integrados através da agregação de novas ontologias, para tratar de domínios específicos, como segurança e QoS.

- Qual é o conjunto de máquinas adequado para executar uma versão de uma aplicação com uma funcionalidade específica e com um desempenho mínimo esperado?
- Qual é o conjunto de máquinas adequado para executar uma versão de uma aplicação com uma funcionalidade específica e com um desempenho mínimo esperado?
- Qual é a origem, propriedade e grau de confiança de uma determinada base de dados?
- Qual é a ação recomendada, segundo critérios definidos em políticas explícitas, para decidir sobre transferência de arquivos na execução de uma aplicação? Por exemplo, o middleware poderia decidir entre uma política de transferência de arquivos por antecipação, ou sob demanda, de acordo com a natureza da aplicação e dos dados, descrita pelos metadados, e com o estado da rede;
- Qual é a ação recomendada, de acordo com critérios explicitamente definidos, para decidir sobre escalonamento de máquinas para execução de aplicações? Como no exemplo anterior, metadados sobre aplicações e dados podem estar associados através de propriedades a determinadas políticas para escolha de recursos;
- É possível executar uma aplicação sobre determinados dados, conforme políticas de acesso, propriedade e confiança, explicitamente definidas? Também como nos dois últimos exemplos, os relacionamentos entre recursos computacionais, aplicações, dados e outros conceitos definidos na ontologia, tais como relações de confiança, podem ser estabelecidos através das classes e instâncias de uma ontologia e da natureza das propriedades que as associa.

O formato e a linguagem para expressar as questões não são necessariamente os expostos nos exemplos abaixo; podendo inclusive serem transparentes para os usuários,

através da utilização de interfaces gráficas, por exemplo, para a realização de consultas e promoção de interatividade entre usuários e sistema. Nosso objetivo é prover meios básicos reutilizáveis, tais como interfaces, que permitam a obtenção de respostas para algumas dessas questões.

A construção da ontologia e da base de conhecimento associada realiza-se em torno de questões dessa natureza. As perguntas acima são exemplos de exploração da semântica de utilização do *middleware* de grade. A capacidade da base de conhecimento de respondê-las de modo eficiente deve legitimar sua utilidade.

4.2.4 Atores e casos de uso

De acordo com o roteiro proposto na metodologia, a próxima atividade investiga os atores, papéis e casos de uso relacionados à manutenção e ao uso da base de semântica. Analisamos apenas alguns atores e casos de usos considerados mais relevantes identificados no contexto de gerenciamento e uso da grade e de seu conteúdo. O conjunto de atores considerados inclui: o *Grid Knowledge Architect*, o *Grid Manager*, o *Grid Application Developer*, o *Grid User* e o *Semantic Grid Integration Component (SGIC)*. Os modelos de casos de uso, mostrados em diagramas UML (OMG, 2006), e uma descrição sucinta dos principais casos de uso para cada um desses atores são mostrados a seguir.

4.2.4.1 Grid Knowledge Architect - Casos de uso

Este ator representa uma abstração criada para descrever as atividades desenvolvidas na realização da nossa pesquisa e a rigor não integra o corpo de atores envolvidos nas atividades normais do ciclo de vida das ontologias.

A ontologia de grade é criada e mantida pelo *Grid Knowledge Architect*. Os casos de uso desse ator são concentrados na fase de concepção da abordagem. Modificações das classes e propriedades e de sua estrutura hierárquica têm custo elevado em fases posteriores do ciclo de vida da base de ontologias, em razão dos seus reflexos em todas as situações, incluindo aplicações, em que são utilizadas.

A Figura 4.1 mostra o diagrama UML com os casos de uso para esse ator.

Figura 4.1: Casos de uso para o *Grid Knowledge Architect*

- **Create Grid ontology** - o *Grid Knowledge Architect* usa uma ferramenta gráfica (ex: Protégé-OWL) para criar um conjunto de ontologias sobre o domínio de grades computacionais. Classes, propriedades, restrições e instâncias de classe são criados de modo incremental. A consistência do modelo deve ser verificada interativamente por alguma ferramenta de inferência, como por exemplo Racer (KG, 2007) Pellet. Essa tarefa é definida no caso de uso do sistema *Make inference*.
- **Support ontologies** - ontologias são construídas para reuso, através da sua extensão, da instanciação de indivíduos da ontologia, ou da realização de atividades de inferência sobre elas. O *Grid Knowledge Architect* executa tarefas relacionadas mais especificamente à extensão, criação e eliminação de classes, propriedades e indivíduos, nas etapas finais do ciclo de desenvolvimento das ontologias manutenção da ontologia. A realização dessas tarefas deve preservar a coerência semântica do modelo. Para esse caso de uso, a consistência base de ontologias também é definida no caso de uso do sistema *Make inference*.

A Figura 4.2 apresenta a interface gráfica da ferramenta Protégé-OWL para desenvolvimento de ontologias.

4.2.4.2 Grid Manager - Casos de uso

O Grid Manager é um ator humano responsável pelo controle e monitoramento da grade. Como a maior parte das tarefas de controle da grade são realizadas pelo próprio

Figura 4.2: Desenvolvendo uma ontologia - Protégé-OWL GUI

Figura 4.3: Casos de uso para o *Grid Manager*

middleware de grade, reduzimos o papel desse ator para a participação em um único caso de uso. A Figura 4.3 mostra o diagrama UML com o caso de uso para esse ator.

- **Register resource** - registrar recursos computacionais da grade, definindo valores para seus atributos e classificando-os sob uma ou mais categorias pertinentes, definidas nas ontologias. O caso de uso do sistema *Make inference* é usado para associar o recurso a outras classes na hierarquia.

Figura 4.4: Casos de uso para o *Grid Application Developer*

4.2.4.3 Grid Application Developer - Casos de uso

O Grid Application Developer é um ator humano responsável pelo desenvolvimento de aplicações para execução na grade. Seus casos de usos são os seguintes:

- **Register Application** - O conceito de aplicação adotado na ontologia identifica um artefato de software. Uma aplicação corresponde à implementação de uma abordagem para tratar um problema, empregando possivelmente um ou mais algoritmos ou heurísticas. Uma aplicação pode ter um ou mais *releases* executáveis para diferentes plataformas de hardware e software.

Toda aplicação deve ser registrada para que possa ser armazenada em um repositório, como por exemplo, o repositório de aplicações do InteGrade, descrito em (BRAGA; VIDAL; KON, 2005), e executada na grade. Os metadados associados às aplicações incluem, entre outros atributos, o nome da aplicação, os componentes que fazem parte da aplicação e as relações de dependência entre esses componentes. O sistema armazena esses dados em um repositório de metadados associado à base de conhecimento. Essas informações serão disponibilizadas para serem exibidas, por exemplo, por interfaces gráficas de usuário, e para serem usadas por outras ferramentas, tais como, ferramentas de busca e de inferência.

Este caso de uso pode ser especializado para tratar a criação de diferentes tipos de executáveis, tais como binários, *workflows*, componentes e bibliotecas.

Figura 4.5: Casos de uso para o *Grid User*

4.2.4.4 Grid User - Casos de uso

Os casos de uso mostrados na Figura 4.5 são responsabilidade do *Grid User* (usuário da grade). A seguir, cada caso de uso é descrito de forma resumida.

- **Request resource** - Nesse caso de uso, o usuário da grade busca identificar os recursos de uma grade e seu potencial. O ator realiza uma consulta por diferentes parâmetros, tais como plataforma, heterogeneidade, capacidade de memória, etc.
- **Request application** - O usuário da grade busca identificar aplicações registradas na grade com base em atributos tais como Domínio, Algoritmo implementado, Desempenho, Plataforma, etc. O usuário pode realizar essas consultas sobre a base de conhecimento para atender diferentes finalidades, como por exemplo, identificar versões variantes de aplicações, compatibilidade entre componentes, dados relacionados a execuções realizadas, entre outros.
- **Execute application** - ao submeter uma aplicação para execução, o usuário poderá selecionar um executável (ou executáveis) especificamente, ou pode realizar uma busca menos precisa com base em conceitos como domínio, algoritmos, etc. Os mecanismos de otimização do componente de integração semântica, SGIC, pode auxiliar a seleção de recursos e componentes executáveis adequados no sen-

Figura 4.6: Casos de uso para o *Semantic Grid Integration Component*

tido de facilitar alguma heurística de escalonamento.

4.2.4.5 Semantic Grid Integration Component (SGIC) - Casos de uso

Este ator é um componente da arquitetura de software associado às ontologias e responsável pela realização de alguns casos de usos importantes incluídos nos diagramas de casos de uso dos atores humanos descritos acima. Esses casos de uso têm relação com tarefas de manutenção das ontologias e do uso das ontologias em diferentes cenários do ambiente de grades computacionais. A Figura 4.6 mostra o diagrama UML de casos de uso para esse ator.

- **Make inference** - o SGIC realiza inferências a partir da inserção de novas informações para atualizar a base de dados semântica;
- **Select suitable applications** - o SGIC seleciona, da hierarquia inferida das ontologias de aplicações, um conjunto de aplicações adequadas aos requisitos definidos pelo usuário;
- **Select suitable resources** - o SGIC seleciona, da hierarquia inferida das ontologias de recursos, um conjunto de recursos adequados aos requisitos definidos

pelo usuário;

- **Prepare suitable input for scheduling** - uma vez definidos os requisitos e restrições sobre aplicações e sobre características e limitações dos recursos computacionais da grade, o SGIC vai executar inferências sobre a base de conhecimento para identificar e classificar as combinações adequadas de recursos e aplicações para otimizar o escalonamento de recursos.

4.3 Estrutura e conteúdo das ontologias

Todo o processo de desenvolvimento das ontologias foi realizado em etapas sucessivas e de modo incremental. A cada nova iteração, as ontologias existentes foram revistas e sofreram acréscimos, eliminações e outras alterações semânticas em sua estrutura.

4.3.1 Versão inicial

Na elaboração da primeira versão da base de ontologias, lançamos mão dos resultados dos passos iniciais da metodologia para obter um esboço preliminar. Iniciamos a definição das ontologias pela identificação dos elementos mais básicos presentes no domínio de grades computacionais tais como recursos de hardware (por exemplo, CPU, memória, disco), recursos de software (por exemplo, middleware, aplicações de usuários, bibliotecas, ferramentas), dados de entrada e saída de aplicações, usuários, etc. Essa tarefa foi baseada em um trabalho de investigação preliminar e na discussão com especialistas no domínio. Uma descrição resumida dessa ontologia preliminar e da abordagem de desenvolvimento empregada em seu desenvolvimento foi publicada em (VIDAL et al., 2006).

Nessa etapa do desenvolvimento das ontologias, os conceitos relacionados à estrutura de grades adotados na ontologia foram inspirados pelo padrão arquitetural de grades descrito em (CAMARGO et al., 2006). A reunião de conceitos importantes no formato de padrão arquitetural de grades facilitou a concepção da ontologia para a

grade. O padrão foi especialmente relevante na descrição de conceitos e relacionamentos comuns ao domínio de grades como um todo, promovendo o uso da ontologia como base para integração de diferentes sistemas de *middleware* de grade.

O primeiro refinamento dos conceitos resultou em três sub-ramos principais da hierarquia de classes: conceitos relacionados ao conteúdo de software da grade, conceitos relacionados aos recursos computacionais da grade e conceitos relacionados aos dados armazenados na grade. Um esquema com esse três sub-ramos é mostrado na Figura 4.7. Os conceitos mostrados nesta seção não serão descritos em maiores detalhes. Apenas os conceitos mais relevantes, presentes na versão atual das ontologias descritas na Seção 4.3.2, serão discutidos mais detidamente.

Figura 4.7: Visão Geral da Ontologia de Grades

A Figura 4.8 mostra o sub-ramo da ontologia relacionado aos conceitos de software (*GridSoftwareContent*) na grade. Nessa versão da fase de esboço da ontologia, o sub-ramo *GridSoftwareContent* define os conceitos, tais como, métodos, aplicações, componentes e executáveis.

O sub-ramo relacionado ao conteúdo de dados (*GridDataContent*) da grade é mostrado na Figura 4.9. Os conceitos definidos nesse sub-ramo descrevem organização, formato e tamanho de conjuntos de dados armazenados na grade. Os dados são usados principalmente como entrada e/ou saída na execução de aplicações na grade. A descrição dos dados com base nesses conceitos pode ser usada, por exemplo, como parâmetro na escolha de recursos para armazenamento, promovendo a alocação eficiente desses recursos na grade. Outro caso comum de uso de metadados sobre dados é a identificação de compatibilidade de formato de dados de entrada e os requisitos definidos por uma

Figura 4.8: Conteúdo de software da grade

versão específica de uma aplicação. Metadados específicos sobre o domínio de origem e utilização dos dados podem ser acrescentados aos conceitos genéricos definidos inicialmente.

Figura 4.9: Conteúdo de dados da grade

Conceitos comuns ao domínio de grades computacionais compõem o sub-ramo da ontologia mostrado na Figura 4.10. Esses foram inspirados no padrão de grades descrito em (CAMARGO et al., 2006) e podem ser particularmente úteis para estabelecer uma conexão semântica no mapeamento entre diferentes sistemas de middleware de grade.

Figura 4.10: Conceitos comuns ao domínio da grade

A Figura 4.11 mostra o sub-ramo que descreve ocorrências de sistemas de middleware de grade conhecidos (*GridMiddlewareCase*). Os conceitos presentes neste sub-ramo da ontologia são fortemente relacionados aos componentes estruturais da grade apresentados no sub-ramo da Figura 4.10. Os dois sub-ramos podem ser usados em conjunto como base para ferramentas de mapeamento entre middleware de grades distintos.

Figura 4.11: Sistemas de middleware de grade

4.3.2 Versão atual

Alcançamos a conformação atual da arquitetura e conteúdo da base de ontologias a partir da ontologia de grade descrita em (VIDAL et al., 2006). Essa versão resultou de uma abordagem exploratória que consistia em definir uma taxonomia inicial abrangente e investigar as possibilidades de utilização de mecanismos de inferência no domínio de grades. A versão original foi revisitada na subseção anterior deste capítulo e revisada no trabalho (VIDAL et al., 2007).

Contudo, a estrutura da ontologia tornou-se excessivamente ampla tanto em relação à variedade, quanto em relação à quantidade, potencialmente crescente, de diferentes conceitos, propriedades e axiomas. Assim, com o objetivo de promover facilidades de manutenção e escalabilidade, além de reduzirmos o foco, excessivamente amplo, da primeira versão, reduzimos o escopo e separamos a ontologia original em um conjunto de ontologias conectadas.

O refatoramento da ontologia original resultou em um conjunto de três tipos distintos de ontologias, aqui denominadas: *upper-level ontologies* (ou *top-level ontology*), *ancillary ontologies* e *concrete ontologies*.

Uma ontologia do tipo *upper-level ontology*, é definida em (CHANDRASEKARAN; JOSEPHSON; BENJAMINS, 1999), como uma “ontologia que descreve conhecimento em um nível mais alto de generalidade”. Em geral, ontologias do tipo *upper-level* são usadas para a descrição de conceitos básicos que podem ser estendidos em ontologias mais específicas. Neste texto, esse tipo de ontologia também será denominado *Top ontologies* ou ainda ontologias de topo.

Ontologias do tipo *Ancillary* (ou ontologias auxiliares), descrevem conceitos específicos de um determinado domínio. Consideramos auxiliares as ontologias que apenas são referenciadas por outras ontologias, geralmente buscando facilitar o reuso dos conceitos que descreve. O uso do termo *Ancillary* pareceu-nos o mais apropriado para denotar o papel secundário desse tipo de ontologia no contexto de um conjunto de ontologias que referenciam os conceitos descritos nesse tipo de ontologia auxiliar.

A categoria *Concrete ontologies*, aqui também chamada de ontologias concretas, aglutina um grupo de ontologias que estendem ontologias de topo para descrever conceitos mais específicos, reusando conceitos genéricos definidos nas ontologia de topo.

Diferentes ontologias semanticamente relacionadas podem se conectar através de diferentes mecanismos. O mecanismo *import*, definido na linguagem OWL, permite a uma ontologia “importar” uma ou mais ontologias. Este mecanismo possibilita, à ontologia que importou, referenciar e estender todos os conceitos e propriedades da ontologia importada, melhorando as características de compartilhamento e reuso do conjunto de ontologias. Conceitos e propriedades da ontologia importada recebem um prefixo referente ao espaço de nomes relacionado.

Inicialmente, nossa base de ontologias para grades computacionais define uma ontologia de topo, chamada *Grid Base Ontology*; uma ontologia auxiliar, chamada *Platform Ontology*; e uma ontologia concreta, mais específica, chamada genericamente de *Grid*

Software Management Ontology. A Figura 4.12 mostra os três tipos de ontologias citadas acima, anotadas com seus respectivos estereótipos, as quais compõem o núcleo da arquitetura da nossa base de conhecimentos sobre grades computacionais.

Figura 4.12: Arquitetura da Base de Conhecimentos

A Figura 4.13 mostra os principais conceitos de uma ontologia auxiliar (*Ancillary*), chamada *Platform Ontology*. Essa ontologia contém uma pequena hierarquia de classes e propriedades relacionadas ao domínio de plataformas computacionais. Isso inclui, entre outros, conceitos tais como sistemas operacionais, arquitetura e processadores. As ontologias concretas usarão esses conceitos como base para extensão e instanciação. Nessa hierarquia, as classes que descrevem famílias de processadores são classes *Primitivas*, e são usadas para instanciar diferentes tipos de processadores. Em contraponto, as classes que representam tipos de processadores são classes *Definidas*.

As classes, na hierarquia de classes gerada pela ferramenta Protégé-OWL, são representadas em cores distintas. Classes *Primitivas*¹ são apresentadas em branco (visualizadas, através da ferramenta, na cor amarelo claro) enquanto classes definidas são apresentadas em cinza (visualizadas, através da ferramenta, na cor laranja).

A ontologia de plataformas é dividida em três ramos principais: *Processor*, *Opera-*

¹ver Capítulo 3

Figura 4.13: Componentes da grade

tingSystem e *Computer*, descritos a seguir.

- *Processor* - As classes dessa hierarquia descrevem famílias de processadores (ex: AMD, Intel).
- *OperatingSystem* - As classes dessa hierarquia descrevem famílias de sistemas operacionais (ex: MacOS, Unix). *Computer* - As instâncias dessa classe são descritas através das propriedades *hasOS* e *hasProcessor*. Os predicados dessas declarações são, respectivamente, instâncias das classes *OperatingSystem* e *Processor*. Essas descrições serão usadas em consultas e inferências em tarefas relacionadas à execução de aplicações na grade.

Essa ontologia, como as demais do conjunto proposto, não pretende ser exaustiva, uma vez que pode ser estendida a qualquer tempo para inclusão de novas classes representando novos conceitos.

Em nossa base de ontologias, a ontologia *Grid Base Ontology* contém conceitos, propriedades e axiomas que são comuns a aplicações e domínios em grades. A Figura 4.14 mostra uma hierarquia de classes dessa ontologia, cuja raiz é classe *GridSoftwareConcept*. As classes dessa hierarquia *definem* conceitos como domínio, problema, abordagem, algoritmo, aplicação, e componentes executáveis, anotados respectivamente como: *Domain*, *Problem*, *Approach*, *Algorithm*, *Application* e *Executable*). Tais conceitos têm um nível suficientemente alto de abstração para serem estendidos, por exemplo, pelo ator *Grid Application Developer* para descrever os conceitos de seu próprio domínio de desenvolvimento, seus problemas, abordagens e algoritmos, a partir de uma terminologia comum. Esse ramo da ontologia será explorado no Capítulo 5, no contexto de integração de cenários de utilização do arquitetura proposta.

Figura 4.14: Conceitos de software na grade

As classes da hierarquia baseada na classe *GridSoftwareConcept* são semanticamente relacionadas através das seguintes propriedades: *isRelatedTo*, *dealsWith*, *describes*, *codifies*, *aggregates* e *implements*. A seguir, apresentamos os significados dessas classes e como são descritas as restrições que as definem. As restrições definem as instâncias que podem pertencer a uma classe através de propriedades que as relacio-

nam com instâncias de outras classes. O termo *some*, conforme definido no Capítulo 3, identifica o quantificador existencial que restringe uma propriedade para uma classe.

- *Domain* - É a classe base para as classes que descrevem domínios de software
- *Problem* - É a classe base para as classes que descrevem problemas relacionados a um domínio específico.

isRelatedTo some Domain - uma instância da classe *Problem* relaciona-se com pelo menos um indivíduo da classe *Domain*, através da propriedade *isRelatedTo*.

- *Approach* - É a classe base para as classes que descrevem abordagens computacionais para solucionar um problema específico.

dealsWith some Problem - uma instância da classe *Approach* relaciona-se com pelo menos um indivíduo da classe *Problem*, através da propriedade *dealsWith*.

- *Algorithm* - É a classe base para as classes que descrevem algoritmos computacionais para uma abordagem específica.

describes some Approach - uma instância da classe *Algorithm* relaciona-se com pelo menos um indivíduo da classe *Approach*, através da propriedade *describes*.

- *Application* - É a classe base para as classes que descrevem aplicações de software para uma algoritmo específico.

codifies some Algorithm - uma instância da classe *Application* relaciona-se com pelo menos um indivíduo da classe *Algorithm*, através da propriedade *codifies*.

- *Executable* - É a classe base para as classes que descrevem componentes de software executáveis para uma aplicação específica.

implements some Application - uma instância da classe *Executable* relaciona-se com pelo menos um indivíduo da classe *Application*, através da propriedade *implements*.

A ontologia *Grid Management Ontology* é um exemplo de ontologia que denominamos *concrete ontology*. Essa ontologia importa a ontologia *Grid Base Ontology*, de alto nível, e também importa a ontologia auxiliar *Platform Ontology*.

A ontologia *Grid Management Ontology* usa as classes definidas nas ontologias auxiliares e de alto nível de dois modos distintos: para derivar novas classes e para instanciar indivíduos. Implementações dessa ontologia, para aglomerados organizados em diferentes domínios administrativos da grade, devem conter um volume maior de instâncias de classes em relação às ontologias de mais alto nível. As instâncias das classes dessa ontologia, são, predominantemente, indivíduos das classes de componentes executáveis registrados nos domínios e de computadores que compõem os aglomerados.

A Figura 4.15 mostra um sub-ramo contendo diferentes domínios de problemas que podem explorar uma grade como plataforma computacional. Os domínios foram concebidos por inspiração em aplicações de *e-science* e representam uma estratégia simples de organização de atividades na grade. Em nossa ontologia usamos como exemplo o domínio de mineração de dados, denominado na ontologia como *Data Mining Domain* e o domínio de reconhecimento de padrões através da análise de textura de imagens (MIR; HANMANDLU; TANDON, 1995). Conceitos como *Data Mining Problems*, *Classification Approaches* e *Texture Analysis Algorithm*, definidos como extensões de conceitos da ontologia *Grid Base Ontology*, são parte dessa ontologia.

Figura 4.15: Grid Management Ontology - Hierarquia de Domínios

A Figura 4.16 mostra o ramo da ontologia concreta que descreve problemas relacionados a um domínio específico e algumas abordagens relacionadas à resolução desses

problemas esse problemas. Nos estudos de caso, apresentados no Capítulo 5, descreveremos as classes que estendem essas ontologias para domínios específicos e as classes que descrevem algoritmos, aplicações e executáveis relacionados.

Figura 4.16: Grid Management Ontology - Hierarquias de Problemas e Abordagens

O conjunto das ontologias definidas nesta seção estão disponíveis para download em <http://www.deinf.ufma.br/~vidal/GBOS/>, em formato XML.

4.4 Conclusão

Neste capítulo discorremos sobre metodologias para desenvolvimento de ontologias e empregamos uma metodologia específica para o desenvolvimento de uma ontologia seminal sobre recursos e aplicações em grades computacionais.

Exploramos um mecanismo da linguagem OWL para conexão de ontologias que permite promover reutilização e compartilhamento de conceitos, propriedades e axiomas definidos em uma determinada ontologia. O mecanismo *import* propicia um modo prático de estender e reutilizar ontologias em um domínio.

Nossa base de conhecimentos inclui exemplos de conceitos em diferentes níveis de abstração. Por exemplo, a classe *Problem* é um conceito genérico, potencialmente extensível encontrado na *Grid Base Ontology*. Enquanto a classe *GLDM_Algorithm* descreve

uma algoritmo específico de um determinado domínio.

Nos próximos capítulos serão descritos cenários para exploração da base de conhecimentos no domínio de grades computacionais, definida no presente capítulo, com o objetivo de promover a integração de recursos e aplicações na grade.

5 Integração e Gerenciamento Semântico de Recursos e Aplicações

As ontologias descritas no Capítulo 4 foram organizadas em torno de dois tópicos dominantes: os recursos requisitados pelas aplicações e a integração de recursos e requisitos de aplicações no contexto das grades computacionais.

Neste capítulo, descrevemos um conjunto de estratégias, mecanismos e cenários para a integração e gerenciamento de recursos e aplicações em grades computacionais, baseados no uso de ontologias. Ao longo do capítulo, utilizamos os cenários descritos para discutirmos e avaliarmos a abordagem proposta. Nas duas seções iniciais, abordamos o uso de ontologias como taxonomias para recursos e aplicações na grade e a possibilidade de extensão dessas ontologias, baseada em mecanismos de inferência. Nas seções finais, discutimos detalhadamente a exploração das taxonomias e mecanismos de inferência no contexto dos cenários tratados, resumidas em uma heurística, definida na Seção 5.4.3, e apresentamos um estudo de caso ilustrativo da discussão.

5.1 Uso de ontologias como taxonomias

Um importante papel das ontologias é atuar como uma taxonomia integrando os principais conceitos relacionados a um domínio particular.

Ambientes de grade integram uma grande quantidade de recursos heterogêneos para permitir a execução de diferentes tipos de aplicações em diferentes modos. Além dos

seus aspectos arquiteturais, aplicações de software podem ser classificadas com base em algumas características relevantes, tais como, o domínio do problema tratado, a abordagem de resolução adotada pela aplicação e o algoritmo ou o conjunto de algoritmos que traduzem uma abordagem específica em termos concretos. Cada aplicação implementada também pode variar em razão de outros fatores, como a linguagem de programação usada, suas dependências em relação a determinadas bibliotecas, componentes ou arcabouços, a plataforma, ou plataformas de execução para as quais foi compilada, assim como em relação às diferentes versões dessas mesmas linguagens, bibliotecas e plataformas.

Como as aplicações, recursos computacionais de grades também podem ser classificados sob uma grande variedade de aspectos e características. O conjunto de atributos pode incluir, entre outros, arquitetura de processador, sistema operacional, espaço de memória e espaço de armazenamento em disco. Finalmente, os recursos computacionais da grade, tomados em conjunto, também podem ser classificados sob diferentes perspectivas, tais como grau de heterogeneidade e taxas de comunicação em rede.

Essas classificações podem ser úteis como base de uma estratégia para utilização eficiente da grade em diferentes cenários.

Os conceitos comentados acima, ainda que não em sua totalidade, serviram de base para uma reflexão que resultou no conjunto de ontologias proposto no Capítulo 4. Aquelas ontologias foram produzidas como resultado do nosso esforço exploratório para criar uma base genérica de conhecimento sobre grades.

Como foi descrito, essa base de conhecimento foi pensada inicialmente para atender os requisitos de metadados do middleware InteGrade. O processo de desenvolvimento das ontologias conduziu posteriormente à concepção de uma base de conhecimentos com maior nível de generalidade e conseqüentemente maior potencial de reutilização.

Além de desempenhar o papel de taxonomia de um domínio, ontologias permitem que novos conhecimentos possam ser gerados a partir dos conceitos e axiomas nela definidos. A próxima seção descreve mecanismos e contexto para a extensão das ontologias

propostas.

5.2 Extensão semântica dos metadados

Um atraente apelo do modelo semântico é a possibilidade de inferir-se conhecimento a partir da prévia definição de axiomas e de informações parcialmente completas. Analisemos, por exemplo, as inferências realizadas pelo mecanismo de inferência conectado à ferramenta Protégé-OWL, sobre uma extensão da ontologia *Grid Base Ontology*, parcialmente apresentada no Capítulo 4. A Figura 5.1, mostra a hierarquia de classes antes da classificação da taxonomia.

Figura 5.1: Hierarquia de classes - Pré-Inferência

A Figura 5.2 mostra as classes que foram adicionadas pela realização da inferência sobre a hierarquia de classes pela ferramenta Protégé-OWL. As classes *KNN_Algorithm*, *SVM_Algorithm* e *NaiveBayesAlgorithm* foram adicionadas como subclasses da classe *ClassificationAlgorithm*.

Figura 5.2: Inferência de hierarquia de classes

Nas próximas seções, discutimos cenários para a exploração semântica dos metadados, no contexto das atividades realizadas na grade, incluindo mecanismos para consulta e inferência no contexto de *Busca de aplicações* e *Políticas de escalonamento*.

5.3 Recuperação de informações

Além de atuar como taxonomia, um conjunto de ontologias deve funcionar principalmente como base para recuperação de informação no domínio de grades computacionais. Através do uso de linguagens de consulta e de mecanismos de inferência sobre a base de conhecimento, um usuário da grade pode propor diferentes questões como as definidas na Seção 4.2.3 do Capítulo 4.

De um modo geral, os objetivos do mecanismo são:

1. Permitir a inferência de conhecimentos sobre um conjunto informações declaradas que podem ser obtidas a partir de diferentes fontes na grade.
2. Permitir a realização de consultas, com resultados relevantes, sobre informações explícitas e possivelmente incompletas.

3. Possibilitar a realização eficiente dessas operações sobre a base de conhecimento.

Nas seções a seguir, descrevemos e justificamos a estratégia proposta para recuperação de informações em nosso trabalho. A principal característica da proposta é a combinação adequada de mecanismos para a realização eficiente das consultas.

5.3.1 Características das ontologias

Os domínios centrais, das ontologias propostas no Capítulo 4 são relacionados especificamente à descrição de software e recursos de hardware em grades computacionais. Essas ontologias são usadas como bases que podem ser estendidas para diferentes domínios de aplicações, no caso da ontologia de software, e para diferentes famílias de plataformas, no caso da ontologia de recursos de hardware.

Nessas ontologias, a maior parte das classes na base de conhecimentos representam a taxonomia de um determinado domínio. Assim, a ontologia de software é estendida para descrever problemas e soluções de cada domínio específico de software. Do mesmo modo, diferentes plataformas de hardware são descritas através das classes e relacionamentos da ontologia de hardware, para permitir que os recursos computacionais da grade possam ser manipulados de modo flexível.

Aplicações e máquinas são elementos concretos que constituem o ambiente de grade. Por essa razão, em nossas ontologias, as classes que descrevem esses elementos são necessariamente instanciadas. O principal papel das demais classes é a descrição do conhecimento que organiza as relações entre essas instâncias.

Essa abordagem determina uma separação efetiva do mecanismo de manipulação das ontologias em partes distintas que implementam diferentes soluções para diferentes componentes da base de conhecimento. Objetivamente, trataremos de modo distinto as classes que descrevem aplicações e máquinas, das demais classes da base de conhecimento.

5.3.2 Inferência antecipada

Conforme a argumentação apresentada desde os capítulos introdutórios, a heterogeneidade semântica das fontes de informações em grades influenciam na produção de informações incompletas. Uma estratégia possível para mitigar os efeitos dessa situação é o emprego de mecanismos de inferência sobre ontologias, como descrito no Capítulo 3, e na Seção 5.2, para a produção de novas informações.

Desse modo, nossa abordagem para obtenção de melhores resultados nas consultas é baseada sobre uma política de realização de inferência antecipada, que determina a realização de inferência sobre a base de conhecimentos, à medida que novos fatos são declarados e adicionados a essa mesma base. A realização de inferência sobre a base como um todo, e com a menor periodicidade possível, deve-se ao alto custo desse tipo de operação (NEXP-TIME no pior caso) (THOMAS; PAN; SLEEMAN, 2007). A idéia é adotar o padrão recomendado em (PROTÉGÉ, 2007), para obter informação inferida e inserir essa informação no modelo de metadados que representa a ontologia.

A atualização dos metadados com base na realização de inferências, deve antecipar-se em relação ao uso dos metadados na realização de atividades da grade e sobre a grade, tais como busca e execução de aplicações. A realização dessa atualização pode ser controlada por políticas baseadas em padrões de gerenciamento de recursos, como os descritos em (KIRCHER; JAIN, 2004) para reduzir o custo computacional da realização da inferência.

A base da nossa estratégia é o uso intensivo de hierarquia de classes, como descrito no Capítulo 4, da inferência de hierarquia de classes através da realização de *subsumption* e da inferência de tipos de instâncias da ontologia. O papel da inferência na recuperação de informações em ontologias será tratado na próxima seção. As motivações para a adoção dessa estratégia são as seguintes:

1. A maior parte das informações contidas nas ontologias são conceitos pertinentes a áreas específicas de gerenciamento de grades, de gerenciamento de aplicações,

ou de domínio de aplicações. Sendo, portanto, informações relativamente mais estáticas em comparação, por exemplo, com informações sobre os estados dos recursos. Nesses casos, as ontologias funcionam como taxonomias sobre as quais se pode inferir conhecimento.

2. Aproveitando a estrutura hierárquica predominante nas ontologias, a inferência para classificação taxonômica pode ser usada para a descoberta de relacionamentos implícitos entre os conceitos da taxonomia.
3. Adicionalmente, a classificação inferida pode ser associada a outros mecanismos e recursos (ex: linguagem SPARQL) para a realização de consultas sobre a base de conhecimentos.
4. A inferência de tipos de instâncias da ontologia também pode ser usada, alternativamente, na realização de consultas.

5.3.3 Inferência e consultas

Nosso trabalho propõe um conjunto de ações para recuperação de informações, baseadas no uso combinado da linguagem de consulta SPARQL, da realização de inferência antecipada através do mecanismo de inferência *Pellet*¹ e da biblioteca de classes *Protégé-OWL API*, apresentada no Capítulo 6.

A linguagem SPARQL é uma linguagem de consulta RDF, usada para recuperar informações em OWL, porque, em geral, conceitos OWL são armazenados em triplas RDF. Como tal, SPARQL é boa para achar coisas que são diretamente declaradas na ontologia OWL, mas não para descobrir informações implícitas. SPARQL não conhece as restrições da semântica OWL e não consegue recuperar informações inferidas. Por outro lado, mecanismos de inferência, como o *Pellet*, produzem novas informações sobre conceitos e axiomas declarados na ontologias.

A política para recuperação de informações é baseada nos seguintes mecanismos:

¹ambos apresentados no Capítulo 3

1. Realização de inferência antecipada sobre as ontologias;
2. Uso direto de SPARQL para realizar consultas, com base em informações explicitamente declaradas nas ontologias. Por exemplo, a consulta abaixo recupera todos as instâncias que se relacionam com instâncias da classe *Ubuntu* através da propriedade *hasOS*:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns #type>

SELECT ?x WHERE {

    ?y < rdf #type> <#Ubuntu> .

    ?x < #hasOS > ?y

}
```

3. Uso de métodos da biblioteca de classes *Protégé-OWL API* para percorrer a hierarquia inferida e descobrir tipos inferidos de instâncias. Por exemplo, os métodos *getInferredClass()* e *getInferredInstance()*, da classe *OWLNamedClass*, recuperam, respectivamente, classes e instâncias inferidas no modelo de metadados.
4. Uso de SPARQL, combinado com métodos da API, para realizar consultas com base em informações inferidas nas ontologias. Os métodos da API podem são usados para percorrer uma hierarquia de classes inferidas, e realizar consultas SPARQL sobre instâncias dessas classes.

5.4 Cenários para abordagem semântica

Nesta seção, apresentamos alguns cenários específicos do domínio de grades e descrevemos estratégias implementadas para o emprego da abordagem semântica em grades computacionais.

5.4.1 Busca de Aplicações

Nesse cenário, a recuperação de informações sobre aplicações pode possibilitar o uso da grade como um repositório distribuído e integrado de aplicações.

O crescimento de um ambiente de grade implica também no crescimento da quantidade de aplicações e recursos instalados e na conseqüente dificuldade de se encontrar uma aplicação, ou conjunto de aplicações ou de recursos para reutilização. Adicionalmente, recursos e aplicações podem ser descritos em uma plethora de diferentes maneiras, sob variadas perspectivas e por diferentes fontes de informação. Tamanha heterogeneidade semântica na base de metadados aumenta as chances de se produzir informação incompleta. A abordagem semântica deve contribuir para superar problemas causados pela diversidade dos metadados sobre as aplicações e para a integração dessas aplicações.

Ao registrar uma aplicação, o desenvolvedor deve inserir informações na base de conhecimentos através da extensão de conceitos na ontologia de software e da criação de uma instância da classe representando um componente executável. Tão logo quanto possível, a base de conhecimentos deve ser atualizada pela realização de alguma tarefa de inferência. Essa abordagem foi escolhida em razão da natureza relativamente estática dos metadados que compõem a base de conhecimentos sobre a grade.

Uma aplicação pode ser buscada em uma grade por muitas razões diferentes. Por exemplo, um usuário pode estar interessado em usar a grade para resolver um problema em um domínio específico, talvez de acordo com uma determinada abordagem. Um usuário também pode escolher explicitamente um binário específico de uma aplicação para ser executado na grade. Sob certas condições, arquivos executáveis de uma aplicação em um domínio podem ser intercambiáveis. Do mesmo modo, componentes de software relacionados a um domínio específico podem ser reutilizados em diferentes domínios. Um usuário pode buscar componentes na fase de desenvolvimento de uma aplicação, ou componentes que possam ser integrados dinamicamente durante a execução de uma aplicação.

Um usuário da grade, ou o próprio middleware da grade, pode submeter consultas à base de conhecimentos atualizada, através da interface que compõe a arquitetura descrita no Capítulo 6. Espera-se que os resultados obtidos pelas consultas sobre as hierarquias inferidas sejam mais significativas e ofereçam um conjunto maior de alter-

nativas explícitas para atender às necessidades dos usuários, considerando um critério de avaliação definido.

5.4.2 Políticas de escalonamento

Uma questão central no gerenciamento de grades é o escalonamento de recursos para execução de aplicações e o conseqüente mapeamento entre recursos da grade e requisitos de aplicações. Comumente, um mecanismo de busca toma como entrada uma solicitação de recursos para uma aplicação, e tenta encontrar um recurso ou conjunto de recursos adequados sobre os quais a aplicação pode ser executada.

Tarefas de escalonamento de recursos dependem, entre outras coisas, da qualidade da informação disponível acerca da aplicação que se deseja executar e dos recursos computacionais da grade sobre a qual se deseja executar a aplicação.

Estratégias, heurísticas e algoritmos para escalonamento de aplicações devem ser baseados nas descrições dos recursos disponíveis e nos requisitos e preferências das aplicações. Sob essa perspectiva, o modo como essa informação é capturada, armazenada, organizada e disponibilizada é determinante para atender aos critérios de eficiência e desempenho das atividades de mapeamento de aplicações e recursos. Metadados sobre recursos e aplicações permeiam e conectam essas atividades e, conseqüentemente, podem afetar sua realização em diferentes níveis de importância e influência.

A maioria dos sistemas de gerenciamento de recursos de grades computacionais têm suas abordagens de mapeamento de recursos e aplicações focadas apenas na descrição de recursos baseadas em atributos. Alguns mecanismos para descoberta de recursos, como os descritos em (CONDOR, 2007), executam um mapeamento exato de requisitos da aplicação com a descrição de recursos baseada em atributos. Mecanismos mais flexíveis como o descrito em (SIDDIQUI et al., 2005) permitem um mapeamento assimétrico de requisitos de aplicações incompletos ou inexatos com uma descrição de recursos baseada em ontologias.

Segundo (ZHANG et al., 2006), um problema de escalonamento consiste em definir

um escalonamento que otimize uma métrica determinada. Apoiados nesse conceito, definimos uma proposta para tratar do problema de escalonamento no contexto de grades computacionais e usamos o middleware de grades InteGrade como base para avaliação da proposta.

Um middleware de grade deve prover condições adequadas para que as aplicações sejam executadas de modo eficiente, de acordo com seu perfil. Por exemplo, aplicações seqüenciais comumente não exigem qualquer preparação prévia para serem executadas em grades. Por sua vez, aplicações paralelas, tais como BSP, MPI e aplicações de *workflow* vão exigir características adicionais de comunicação, que devem ser providas pelo middleware de grade.

O papel dos metadados sobre recursos e aplicações é crucial para a realização de uma tarefa de execução de aplicação bem sucedida na grade. Nesse cenário, nós empregamos a mesma política de “inferência antecipada”, descrita anteriormente, para o conjunto completo de ontologias. Como explicado na Seção 5.4.1, a inferência deve atualizar a base de conhecimentos com uma frequência adequada ao funcionamento eficiente do sistema.

A base de conhecimentos inferida deverá conter novos relacionamentos hierárquicos estabelecidos entre os conceitos declarados, acrescentando novas informações e significados aos metadados disponíveis. Nos próximos parágrafos, descrevemos como esse resultado vem ao encontro da abordagem de escalonamento empregada no middleware InteGrade.

A unidade arquitetural básica do InteGrade é o *cluster*, uma coleção de máquinas normalmente conectadas através de uma rede local. Os *clusters* podem ser organizados em uma hierarquia, permitindo agregar um grande número de máquinas. Um cluster típico do InteGrade é mostrado na Figura 5.3.

Cada *cluster* contém um nó, chamado *cluster Manager*, que executa componentes do InteGrade responsáveis pelo gerenciamento dos recursos de computação do *cluster*

Figura 5.3: Arquitetura do InteGrade

e pela comunicação inter-cluster. O componente ASCT (*Application Submission and Control Tool*) é responsável pela submissão das aplicações na grade, o componente LRM (*Local Resource Manager*) gerencia os recursos em um nó da grade e executa aplicações submetidas via o ASCT. O componente LUPA (*Local Usage Pattern Analyser*) captura informações sobre padrões de uso de uma nó e tenta prever a utilização futura do recurso baseado em técnicas de aprendizado de máquina.

O GRM (*Global Resource Manager*) é o componente responsável pelo escalonamento das aplicações na grade. Nossa proposta de procedimento de inferência pode ser usada para descoberta de recursos equivalentes ou intercambiáveis, promovendo um mapeamento eficiente entre recursos e aplicações que são executadas na grade. Um roteiro possível para realização dessa estratégia é descrito a seguir. Esse roteiro é baseado no caso de uso *Execute application* definido no Capítulo 4.

Nosso roteiro inicia quando um usuário seleciona uma aplicação para executar na grade, interagindo em um diálogo para informar, possivelmente através de uma interface gráfica, as preferências e restrições associadas à execução. Em consequência, uma consulta é executada sobre a base de metadados estendida. A consulta sobre a base estendida tem mais chances de alcançar um mapeamento mais amplo entre recursos disponíveis na grade que possuem atributos consistentes em relação aos requisitos e

preferências definidos na solicitação de execução da aplicação.

A hierarquia inferida pode melhorar as possibilidades de o total de recursos necessários ser encontrado localmente no *cluster* onde foi feita a solicitação de execução ou, pelo menos, provavelmente em quantidade maior do que a quantidade encontrada antes do procedimento de inferência ser realizado.

5.4.3 Heurística para uso dos mecanismos semânticos

Os cenários descritos nas Seções 5.4.1 e 5.4.2 exploram o conjunto de ontologias e mecanismos de inferência e consulta permitindo incrementar o uso dos recursos e o reúso de aplicações registrados na grade. De um modo geral, essa abordagem segue a heurística descrita abaixo, baseada nos papéis desempenhados pelos atores descritos no Capítulo 4.

1. O *Grid Knowledge Architect* descreve e mantém as ontologias contendo conceitos e propriedades fundamentais para a descrição de conteúdo e recursos das grades computacionais.
2. O *Grid Manager* e o *Grid Application Developer* podem usar, dentre os conceitos e propriedades que compõem as ontologias, apenas aqueles que considerem suficientes para descrever um item de software ou hardware de acordo com o seu conhecimento. Essa descrição pode ser incompleta em razão da multiplicidade das fontes de informação e das possibilidades de descrição.
3. Componentes do *Semantic Grid Integration Component (SGIC)* são usados para atualizar a base de conhecimentos através da realização de inferências que resultam na identificação de novos relacionamentos semânticos entre classes e instâncias das ontologias. Em razão dos custos dos algoritmos de inferência sobre ontologias OWL, as tarefas de inferência são realizadas, sobre o conjunto das ontologias, de acordo com uma política que possa aproveitar a ociosidade dos recursos da grade.

4. O *Grid User* utiliza mecanismos de busca do *Semantic Grid Integration Component (SGIC)* para a realização de consultas sobre a base de metadados. O objetivo do *Grid User* é selecionar aplicações e recursos para a realização de atividades como as descritas nos cenários apresentados acima. As consultas sobre a base de dados, estendida através dos mecanismos de inferência, possibilitam a obtenção de resultados mais ricos em termos de opções intercambiáveis de recursos e aplicações.

5.5 Estudo de Caso

Na Seção 5.4.1, descrevemos um mecanismo de busca flexível que permite obter um conjunto mais completo de resultados consistentes em resposta a uma consulta.

A combinação do conjunto de resultados de uma consulta sobre aplicações, em uma base de metadados semanticamente estendida, com um mecanismo flexível de busca, como descrito na Seção 5.4.2, deve resultar em um conjunto mais completo de pares apropriados de recursos e aplicações mapeados.

Em conseqüência, propomos que uma abordagem eficiente de gerenciamento de recursos em grades deve ser baseada em um mecanismo flexível de mapeamento, empregado com o objetivo de integrar os resultados das buscas de recursos e aplicações.

Por exemplo, se um usuário procura por uma aplicação que implementa um algoritmo específico, o componente arquitetural *Semantic Grid Integration Component*, identificará toda aplicação equivalente que implementa tal algoritmo, e seus diferentes componentes executáveis. A variedade resultante de executáveis disponíveis deve ser maior que aquela resultante da busca baseada apenas na descrição e comparação exata dos atributos, basicamente porque considera alternativas equivalentes inferidas. Conseqüentemente, o conjunto de recursos de grades adequados aos requisitos das aplicações também deve crescer, uma vez que o surgimento de novas variações das aplicações implica no crescimento do espectro de recursos adequados.

De um modo geral, a maioria das propostas do nosso conhecimento não exploram a relação implícita entre aplicações alternativas e busca de recursos na grade, visando um aumento da eficiência do escalonamento das aplicações.

O objetivo desta seção é analisar a utilização do mecanismo proposto para exploração do modelo de base de dados semântica para gerenciamento de grades computacionais. A avaliação dos diferentes aspectos da mecanismo foi organizada em torno da integração de cenários e atividades realizadas na grade com suporte de uma estrutura semântica, conforme descrito na Seção 5.4.

5.5.1 Descrição do contexto

O contexto do caso em estudo envolve a descrição da estrutura de recursos da grade explorada e o povoamento da ontologia com dados suficientes para a finalidade de avaliação. Cabem aqui algumas considerações adicionais:

- As ontologias descritas no Capítulo 4 foram parcialmente povoadas apenas para efeito do presente estudo de caso.
- A descrição das aplicações, considerando os aspectos definidos nas ontologias relacionadas, acontece durante a realização do caso de uso *Register Application*, no qual são informados dados estáticos sobre as aplicações, tais como tipo de aplicação, domínio, variantes de algoritmos, executáveis disponibilizados, etc. Para a realização do experimento, por razões de eficiência, os resultados correspondentes à realização desse caso de uso foram gerados previamente.
- A descrição de um conjunto de *solicitações de execuções*, considerando os aspectos definidos nas ontologias, acontecem na realização do caso de uso *Execute application*, quando são informados requisitos dinâmicos relacionados à execução das aplicações, tais como:
 - número de nós necessários;
 - recursos selecionados: seleção de recursos de software em diferentes possíveis

graus de generalidade (abordagens, algoritmos e artefatos de software: aplicações e executáveis). Também para efeito de realização do experimento, e por razões de eficiência, os resultados correspondentes à realização desse caso de uso serão gerados previamente.

5.5.2 Descrição da avaliação

A avaliação do modelo proposto vai ser centrada unicamente no cenário integrado de busca de aplicações e de escalonamento de recursos para execução das aplicações. Queremos mostrar que a abordagem semântica consegue resultados satisfatórios, em situações práticas de recuperação e uso de metadados na grade, se comparada a abordagens de gerenciamento de metadados convencionais, nas quais as comparações são feitas sobre os valores explícitos dos atributos descritos na base de metadados. Em tais situações, as informações disponíveis são comumente incompletas ou implícitas.

Nosso objetivo é avaliar a eficácia da abordagem, no sentido de incrementar a disponibilização da maior quantidade possível de máquinas para execução de aplicações. Esse resultado é obtido através de dois caminhos distintos: a descoberta de recursos equivalentes e a descoberta de artefatos de software equivalentes. Os dois caminhos produzem, ao cabo da extração semântica de informações, uma quantidade maior de recursos passíveis de utilização para execução das aplicações. Esse resultado é considerado positivo porque propicia um aumento na oferta de alternativas para execução das aplicações em caso de se encontrarem recursos ocupados.

Os conceitos mostrados nas tabelas abaixo estão definidos no conjunto de ontologias descritos no Capítulo 4, como classes ou indivíduos.

A Tabela 5.1 mostra os tipos de plataformas de computadores descritos na ontologia *Platform Ontology*. O conceito de *plataforma* não é explícito na ontologia. Nesta seção, utilizamos o termo para denotar a combinação dos conceitos de *Sistema Operacional* e *Processador*, simplificando a montagem das tabelas.

Tipos de Plataformas			
Tipo	Plataformas		Nós
P1	Intel(R) Pentium(R) 4 CPU 3.00GHz	i686 GNU/Linux	8
P2	AMD Athlon(tm) XP 2800+	i686 GNU/Linux	6
P3	Intel(R) Xeon(TM) CPU 3.00GHz	i686 GNU/Linux	1
P4	AMD Athlon(tm) 64 Processor 3200+	x86_64 GNU/Linux	1
P5	AMD Athlon(tm) 64 Processor 3000+	x86_64 GNU/Linux	1
P6	Intel(R) Pentium(R) 4 CPU 2.80GHz	i686 GNU/Linux	3
P7	Intel(R) Pentium(R) D CPU 2.80GHz	i686 GNU/Linux	1
P8	Power PC G4	Power PC G4	1

Tabela 5.1: Tipos de Plataformas

A Tabela 5.2 mostra os tipos de plataformas que são equivalentes na perspectiva de execução das aplicações. Esses dados também são parte da ontologia *Platform Ontology*.

Plataformas equivalentes	Nós disponíveis
P1, P2, P3, P6 e P7	19
P4 e P5	2
P8	1

Tabela 5.2: Plataformas equivalentes

“Análise de textura de imagens” denomina uma família de algoritmos pertinentes ao domínio de reconhecimento de padrões. Esses algoritmos são empregados em diferentes domínios que vão desde a análise de fotos aéreas para estudos geográficos e geológicos, até a análise de imagens tomográficas para diagnóstico de diferentes tipos de câncer. Os métodos descritos pelos diferentes algoritmos da família variam em razão de fatores como eficiência no diagnóstico ou avaliação, desempenho previsto e ênfase em alguma variável particular. Contudo, dependendo do contexto esses algoritmos poderiam ser usados de modo intercambiável.

A escolha em particular por esse grupo de algoritmos para compor um domínio da nossa ontologia de recursos de software deveu-se principalmente à existência da implementação de três desses algoritmos como aplicações paramétricas para a plataforma

MAG, um middleware distribuído baseado no InteGrade, que também explora a tecnologia de agentes móveis (SOUSA et al., 2006).

A Tabela 5.3 mostra uma relação de algoritmos e aplicações para o domínio de análise de textura de imagens (*Image Texture Analysis - ITA*) e para quais plataformas na grade existem binários disponíveis para cada aplicação na tabela. Esses dados estão descritos na ontologia *Grid Management Ontology*.

Algoritmos	Aplicações	Plataformas válidas	Nós disponíveis
SGLDM	SGLDM_App	P1	8
GLRLM	GLRLM_App	P5 e P8	2
GLDM	GLDM_App	P4	1
PSM	PSM_App	P4	1

Tabela 5.3: Algoritmos ITA e artefatos de software relacionados disponíveis

5.5.3 Resultados obtidos

Nesta subseção comparamos duas abordagens distintas para o escalonamento de aplicações paramétricas na grade: a abordagem empregada pelo InteGrade e a abordagem baseada em metadados semânticos.

Em nosso exemplo, temos um conjunto de aplicações paramétricas que implementam um conjunto de algoritmos equivalentes. A escolha de uma aplicação paramétrica atende ao nosso objetivo de usar um modelo simples para o exercício de comparação das duas estratégias. Aplicações paramétricas, em geral executam tarefas diferentes em diferentes nós da grade, não exigindo normalmente mecanismos de comunicação para sincronização de tarefas executadas paralelamente.

A abordagem de escalonamento adotada pelo middleware do InteGrade permite que o usuário selecione um executável (binário) de uma aplicação, ou selecione uma aplicação. No primeiro caso, o middleware tentará identificar os nós do *cluster* local cuja plataforma seja descrita de modo idêntico à descrição da plataforma requisitada

pela aplicação. O segundo caso permite um pouco mais de flexibilidade na alocação dos recursos. Neste caso, o middleware primeiro vai identificar todos os executáveis disponíveis no repositório de aplicações do InteGrade, para a aplicação que se deseja executar. Esse tratamento vai aumentar o leque de plataformas sobre as quais é possível executar a aplicação e potencialmente vai aumentar a oferta de máquinas para a execução da aplicação. Em nosso exemplo, a abordagem de escalonamento vai permitir um número potencial de no máximo oito máquinas alocadas, caso seja escolhida a aplicação SGLDM_App, para o algoritmo em questão (ITA).

Abordagem de escalonamento do InteGrade	
Aplicação selecionada	Nós selecionados
SGLDM_App	8
GLRLM_App	2
GLDM_App	1
PSM_App	1

Tabela 5.4: Abordagem de escalonamento do InteGrade

As duas tabelas a seguir, mostram os resultados obtidos pela nossa abordagem de escalonamento baseada em semântica.

A Tabela 5.5 mostra o número potencial de máquinas disponibilizadas para execução de cada um dos algoritmos ITA isoladamente. Esses resultados foram obtidos porque foi possível ao mecanismo de inferência descobrir os relacionamentos de equivalência entre as plataformas descritas na ontologia. Esses relacionamentos foram definidos, a priori, na ontologia *Platform Ontology* através de subclassificação declarada. Contudo, poderíamos criar propriedades transitivas por exemplo que permitissem a classificação indireta de novas plataformas aos grupos definidos na ontologia através de mecanismos de inferência. O total de nós obtido deve-se ao fato de que toda plataforma que pertence a um mesmo grupo poder ser intercambiada para execução das aplicações desejadas. O melhor caso corresponde ao grupo de plataformas com maior número de máquinas.

Abordagem de escalonamento semântico - 1º cenário	
Aplicação selecionada	Nós selecionados
SGLDM_App	19
GLRLM_App	3
GLDM_App	2
PSM_App	2

Tabela 5.5: Estratégia Semântica de Pré-escalonamento - *seleção de recursos*

A Tabela 5.6 apresenta porém o melhor resultado em termos de número potencial de máquinas para execução da aplicação. Essa tabela integra a possibilidade de uso de plataformas alternativas, pertencentes á um mesmo grupo de plataformas equivalentes, à possibilidade de uso alternativo de aplicações diferentes, caso a seleção do recurso do software seja em nível de algoritmo.

Abordagem de escalonamento semântico - 2º cenário	
Aplicação selecionada	#Nós selecionados
SGLDM_App	22
GLRLM_App	22
GLDM_App	22
PSM_App	22

Tabela 5.6: Estratégia Semântica de Pré-escalonamento - *seleção de recursos e executáveis*

Como os algoritmos são potencialmente equivalentes, o usuário pode escolher qualquer dos algoritmos e pode decidir que é razoável executar diferentes aplicações (e seus diferentes executáveis) em diferentes plataformas adequadas e disponíveis. Nesse caso, qualquer das aplicações eleva ao máximo o potencial de uso das plataformas, vinte e duas máquinas em nosso exemplo.

Esses algoritmos são *descritos* e/ou *definidos* no ramo da ontologia *Grid Management Ontology*, mostrado na Figura 5.4

Figura 5.4: Hierarquia de classes - Softwares

Figura 5.5: Hierarquia inferida de classes - Softwares

A Figura 5.5 mostra que a inferência da hierarquia de classes resulta na inserção da classe *PSM_Algorithm* (*Pattern Similarity Matching*) como subclasse da classe *TextureAnalysisAlgorithm*. O mecanismo de inferência é usado para “descobrir” que os algoritmos que são *descritos* através da propriedade *gbo:describes some TextureAnalysisApproach*, são também subclasses da classe *TextureAnalysisApproach*. Isso é possível

porque a classe *TextureAnalysisApproach* também é subclasse da classe *Algorithm* e é definida através do relacionamento *gbo:describes only TextureAnalysisApproach*.

5.6 Conclusão

Este capítulo descreve alguns aspectos e cenários relevantes em grades computacionais. Também explora o potencial desses aspectos e cenários para o emprego de uma abordagem de integração semântica da estrutura e conteúdo das grades. A abordagem proposta é baseada em ontologias e mecanismos de inferência e consulta, e mostrou-se útil nos cenários de recuperação de informações sobre aplicações e na pré-seleção de recursos e aplicações visando o escalonamento eficiente de recursos para execução de aplicações na grade.

6 Implementação e Validação da Proposta

Este capítulo descreve a arquitetura proposta para a efetivação do sistema e alguns detalhes mais relevantes sobre a sua implementação. A base da arquitetura são os metadados sobre aplicações e recursos em grades computacionais. A arquitetura foi planejada visando facilitar o uso desses metadados para a integração, o compartilhamento e a reutilização de aplicações e recursos da grade.

As principais características que buscamos agregar à estrutura e funcionalidade da arquitetura são:

- **Extensibilidade da base de conhecimento:** a arquitetura é construída em torno dos conceitos definidos nas ontologias e permite a manipulação dessas ontologias e de suas instâncias. A arquitetura promove a extensão dos conceitos tratados no domínio através de dois mecanismos, cuja utilização depende do ator responsável e da natureza do caso de uso. O ator *Grid Knowledge Architect* é o único que deve usar ferramentas auxiliares que não são parte da arquitetura, como o Protégé-OWL. Os atores foram apresentados no Capítulo 4.
- **Exploração incremental da base de conhecimento:** a base de conhecimento pode ser explorada para atender as principais funcionalidades e requisitos relacionados no Capítulo 4. Essas funcionalidades são associadas aos casos de uso dos atores *Grid Knowledge Architect*, *Grid Application Developer*, *Grid User* e *Grid Application Agent*. De um modo geral, tais funcionalidades envolvem manutenção da base de conhecimento, recuperação de informações sobre aplicações armaze-

nadas na grade e uso eficiente dos recursos da grade para atender a demanda das aplicações que executam na grade;

- **Separação e independência do sistema de middleware de grade:** a arquitetura proposta não depende de um sistema de middleware particular para seu funcionamento. No entanto, o middleware pode usar a arquitetura para gerenciar informações sobre aplicações e recursos.
- **Baseada em padrões:** a adoção de linguagens e protocolos considerados modelos na comunidade de usuários e desenvolvedores, para a definição da arquitetura, visa mitigar os problemas relacionados ao ambiente heterogêneo intrínseco dos sistemas de grade.

6.1 Componentes da Arquitetura

A Figura 6.1 mostra o componente arquitetural *Semantic Grid Integration Architecture* conectando a base de conhecimento e o middleware de grade e auxiliando na integração desses componentes.

Figura 6.1: Arquitetura para integração semântica da grade

A Figura 6.2 mostra os principais componentes da arquitetura do sistema desenvolvido no presente trabalho. Apresentamos em seguida uma descrição sucinta desses componentes, aspectos relevantes de sua integração e uso em grades computacionais, e algumas considerações sobre sua implementação.

Figura 6.2: Componentes da arquitetura

O componente agregador da arquitetura proposta é o *Semantic Grid Integration Component*. Esse componente corresponde ao padrão *Facade* descrito em (GAMMA et al., 1995) e fornece uma interface para os requisitos funcionais identificados. O escopo desses requisitos é descrito a seguir em um nível maior de detalhes:

- **instanciação:** criação e manutenção de instâncias dos conceitos definidos na ontologia e dos relacionamentos entre essas instâncias, com a finalidade de representar aplicações e recursos da grade computacional;
- **verificação de consistência:** funcionalidade exercida de modo constante e transparente sobre a base de conhecimento, para contornar ou evitar inconsis-

tências e contradições, conforme descrito no Capítulo 3;

- **recuperação de informações:** exercida por diferentes atores em diferentes casos de uso, para atender objetivos variados, visando o uso eficiente da grade computacional. O conjunto de casos de uso atendidos pode crescer à medida que novas aplicações sejam desenvolvidas em torno da exploração da base de conhecimento. A recuperação de informações é baseada na realização de consultas e inferências sobre a base de conhecimento. Nesse contexto, o ator *Grid User* pode usar uma interface de usuário para realização de consulta, e esta pode então chamar um método do componente *Semantic Grid Integration Component* para identificar uma aplicação armazenada que atenda de algum modo à consulta. Em outra situação, um ator *Grid Application Agent* pode estar envolvido na identificação de recursos computacionais que possam ser escalonados de modo eficiente para a execução de uma determinada versão de uma aplicação na grade computacional. Por exemplo, um serviço de *resource brokering* deve poder achar informações sobre como obter e instalar o código executável de uma aplicação sobre recursos alvo e sobre como essas atividades afetarão a demanda por recursos (SCHOPF, 2003).

Enquanto o *Semantic Grid Integration Component* é central na arquitetura, o componente *Grid Knowledge Base* representa a base de conhecimento formada pela ontologia OWL-DL e suas instâncias. Essa base pode ser considerada o alicerce da arquitetura e é descrita detalhadamente no Capítulo 4 e nos artigos (VIDAL et al., 2006) e (VIDAL; KOFUJI, 2007).

Uma característica importante da nossa arquitetura é o seu potencial de reutilização para diferentes sistemas de middleware de grade. O *Grid Middleware* mostrado na Figura 6.2 representa um componente genérico. A implementação inicial da arquitetura é baseada no sistema InteGrade, o qual é uma instância específica desse componente genérico. Os cenários mostrados no Capítulo 5 exploram aplicações que executam no MAG (SOUSA et al., 2006), um middleware distribuído baseado no InteGrade.

Os componentes *Knowledge Base Manager* e *Information Explorer* são ambos base-

ados em bibliotecas e ferramentas Java desenvolvidas em código aberto e dirigidas para a manipulação de linguagens, protocolos e plataformas de desenvolvimento considerados padrões, tais como OWL e Pellet. O componente *Knowledge Base Manager* trata da manutenção da base de conhecimento, em particular da sua consistência semântica. O componente *Information Explorer* apóia a integração de consultas e inferências de modo eficiente sobre a base de conhecimento.

A Figura 6.3 mostra a distribuição dos componentes da arquitetura de metadados nos nós da grade. Cada componente do conjunto de componentes reunidos no *Global Manager Node* é logicamente único em cada agrupamento de nós da grade. Enquanto o componente mostrado no *Local Manager Node* deve estar presente em todos os nós que precisam acessar a base de conhecimento.

Figura 6.3: Distribuição dos componentes nos nós da grade

6.2 Implementação

A arquitetura esboçada na seção anterior foi parcialmente implementada através da utilização da API Protégé-OWL, da linguagem de consulta SPARQL e da ferramenta Pellet para realização de inferência sobre a base de conhecimentos representada em OWL-DL. A IDE Eclipse, foi usada como apoio às atividades de implementação.

A prioridade para implementação dos componentes da arquitetura foi definida principalmente em razão da necessidade de avaliação da proposta. Em consequência a

implementação dos componentes *Semantic Grid Integration Component* e *Information Explorer* foi privilegiada.

A figura 6.4 mostra os componentes internos do componente SGCI, descritos resumidamente a seguir.

Figura 6.4: Componentes internos do SGIC

- *Semantic Grid Manager GUI (SGMGui)*: A finalidade desse componente é ser usado para submissão de consultas e submissão de execução de aplicações pelos *Grid User*, para o registro de aplicações e executáveis pelo *Grid Application Developer* e para o registro de máquinas na grade pelo *Grid Manager*.
- *Semantic Grid Resource Manager (SGRM)*: Esse componente integra e coordena os demais componentes, descrevendo a lógica envolvida na realização das funcionalidades disponibilizadas através das interfaces de usuário, tais como, registrar aplicações e recursos e submeter aplicações.
- *Semantic Metadata Manager (SMM)*: A responsabilidade desse componente é a manutenção da base de conhecimentos, através de métodos que permitem a

manipulação, armazenamento e recuperação dos metadados sobre as classes e instâncias das ontologias.

- *Semantic Services Manager (SSM)*: Esse componente é responsável pelos serviços de inferência sobre as ontologias, tais como, inferência da hierarquia de classes, checagem de consistência das classes e inferência de tipos das instâncias.

Os próximos parágrafos descrevem a realização de alguns casos de usos relevantes na utilização da arquitetura proposta no contexto de grades computacionais.

O diagrama de seqüência, mostrado na Figura 6.5, apresenta a interação entre o sistema de middleware InteGrade e os componentes da arquitetura proposta, para a realização do caso de uso *Register resource*. O ator *Grid Manager* registra um recurso computacional na grade através da interface de usuário do middleware de grade.

Figura 6.5: Diagrama de seqüência - Registrando um recurso da grade

No middleware InteGrade, o componente LRM se registra no componente GRM de um *cluster*. O componente GRM acessa a interface do componente SGRM da arquitetura semântica. Este componente coordena as ações das classes da arquitetura semântica. O componente SGRM chama o método *registerResource* do componente SMM, que é executado em colaboração com o componente *SSM*, para classificar a hierarquia de classes, criar e armazenar a instância de metadados referente ao recurso registrado.

A tarefa de inferência pode ser adiada, sob controle de uma política definida, que permita a sua realização sobre o modelo como um todo, em uma frequência adequada para mitigar os custos dessa tarefa.

A Figura 6.6 mostra o diagrama de sequência *registerDomain*. No Capítulo 4, limitamos as descrições dos casos usos aos exemplos mais representativos. Por essa razão, este caso de uso não foi explicitamente descrito anteriormente.

Figura 6.6: Diagrama de sequência - Registrando um domínio

Ao registrar um domínio, o ator *Grid Application Developer* utiliza a interface gráfica de usuário implementada pelo componente SGMGui, da arquitetura semântica. Novamente, o componente SGRM coordena a ação dos demais componentes da arquitetura semântica; ele chama o método *registerDomainData* do componente SMM, e este é executado em colaboração com o componente SSM, para inferir a nova hierarquia de classes, criar e armazenar as instâncias de metadados referentes ao domínio registrado e aos conceitos relacionados à esse domínio. O conjunto de abstrações associadas ao domínio de software inclui os conceitos de problemas, abordagens e algoritmos, além das propriedades associadas a esses conceitos básicos. Essa tarefa é independente da natureza do middleware da grade e seu objetivo é atualizar a base de metadados para ser utilizada por uma determinada comunidade de de usuários.

O diagrama de sequência mostrado na Figura 6.7 descreve a interação dos com-

ponentes que participam da realização do caso de uso **Register Application**, para efetivar o registro de uma aplicação em uma grade. O Ator *GridApplicationDeveloper* acessa a interface gráfica de usuário para informar o nome do artefato de software (executável ou aplicação) que deseja registrar. O método *registerApplication* do componente SMM, colabora com o componente SMM para inferir a classificação hierárquica dos conceitos da ontologia, criar e armazenar instâncias correspondentes ao artefato de software registrado.

Figura 6.7: Diagrama de sequência - Registrando uma aplicação

A Figura 6.8 mostra um diagrama de sequência genérico com os componentes envolvidos na pré-seleção de recursos para a execução de uma aplicação. O Ator *GridUser* interage com a interface gráfica para selecionar um executável ou aplicação desejada. A seleção de recursos é feita com base na seleção do software. Uma lista de pares (executáveis/recursos) é disponibilizada e o ator pode confirmar a execução da aplicação. Esse diagrama de sequência pode variar, por exemplo, no sentido de o ator poder realizar várias interações na realização consultas, através de diferentes parâmetros, para a escolha dos executáveis.

Figura 6.8: Diagrama de seqüência - Preparando a execução de aplicação

6.3 Validação da proposta

De um modo geral, as principais abordagens para avaliação de ontologias se enquadram nas seguintes categorias (BRANK; GROBELNIK; MLADENI, 2005):

- abordagens baseadas na comparação com um determinado padrão;
- abordagens baseadas na utilização da ontologia em uma aplicação e a avaliação dos resultados dessa utilização;
- abordagens envolvendo comparações com uma base de dados sobre o domínio tratado pela ontologia;
- abordagens nas quais atores humanos avaliam a ontologia segundo critérios pré-definidos.

O cerne do nosso trabalho é o modelo semântico para a integração de recursos e aplicações em grades, que compreende as ontologias descritas no Capítulo 4 e os mecanismos descritos no Capítulo 5 e nas seções iniciais deste capítulo. Nesse modelo, ontologias interagem com componentes de um mecanismo para possibilitar a realização de tarefas relacionadas à execução de aplicações na grade. Por essa razão, o modelo

proposto (ontologias e mecanismo) é avaliado com base nos resultados obtidos em sua utilização na realização de algumas tarefas relevantes nesse contexto.

A avaliação foi planejada sob dois enfoques, que se distinguem pelos seus objetivos. O primeiro enfoque é sobre a realização de experimentos para avaliação do tempo de submissão das consultas sobre a base de metadados. O segundo enfoque avalia a efetividade das consultas, analisando os resultados de algumas consultas específicas.

6.3.1 Desempenho das consultas sobre a base de metadados

Nesse experimento, comparamos o desempenho da realização de consultas sobre bases de diferentes tamanhos. O principal objetivo desse tipo de experimento é avaliar o impacto do crescimento das bases de conhecimentos sobre o tempo requerido para a realização de consultas.

Foram realizadas 100 consultas, definidas em torno de variações sobre o conjunto de consultas, parcialmente mostrado no Apêndice B. Esse conjunto de consultas foi elaborado considerando-se a perspectiva do ator *Grid User* para a seleção de aplicações, executáveis e computadores na grade. As consultas são expressas na linguagem SPARQL, submetidas através de modelos da API *Protégé-OWL*, associada à API Jena (CARROLL et al., 2003; PROGRAMME, 2007).

As consultas foram executadas sobre bases de metadados com diferentes quantidades de instâncias. O primeiro experimento executou as consultas com a base contendo 1.000 instâncias, o segundo executou sobre uma base com 5.000 instâncias e o terceiro experimento, sobre uma base com 8.000 instâncias armazenada. Usamos um laptop com processador Intel(R) Pentium(R) M, 1.70 GHz, 496 MB de RAM e sistema operacional Linux, distribuição Ubuntu 7.04 (feisty).

Para cada experimento, sobre bases com diferentes volumes de instâncias, foram aferidos os tempos de execução máximo, o tempo de execução mínimo, a média dos tempos de execução e o desvio padrão do tempo de execução. Os resultados obtidos são apresentados na tabela a seguir.

Consultas sobre a base de plataformas				
Instâncias	Tmax	Tmin	Média	Desvio padrão
1000	2,39	0,35	0,55	0,39
5000	3,15	0,34	0,57	0,50
8000	3,48	0,36	0,59	0,48

Tabela 6.1: Desempenho das consultas em ms

Os resultados obtidos mostram desempenhos satisfatórios nos três casos, com tempos médios de processamento da consulta abaixo de 1 ms e valores pequenos de desvio-padrão. As razões para esses resultados devem-se, principalmente, ao fato de que as classes e instâncias são lidas das bases de dados para os modelos que as representam na memória. Uma segunda razão é que as consultas SPARQL não realizam inferências sobre os dados, o que reduz consideravelmente o tempo de execução dessas consultas. O mecanismo proposto em nosso trabalho prevê a realização de inferência em bloco e antecipada, e as consultas, em geral, são realizadas sem custos adicionais de inferência a cada consulta.

A tabela 6.2 mostra o tempo, em segundos, despendido na realização de inferência para classificação da hierarquia de conceitos. Os tempos foram medidos para os diferentes tamanhos da mesma base de metadados utilizada para medir o tempo gasto na realização de consultas. Os resultados obtidos impõem a adoção de uma política de inferência que equilibre a necessidade de atualização dos metadados, gerada pela dinâmica da grade, com o alto custo computacional da realização de inferência.

Inferência sobre a base de plataformas	
Instâncias	Tempo
1000	4,64
5000	4,77
8000	4,94

Tabela 6.2: Tempo para inferência de classes em segundos

6.3.2 Efetividade das consultas

O foco dessa análise é sobre a efetividade dos resultados obtidos pelas consultas, sob dois contextos distintos, a base de conhecimentos declarada e a base de conhecimentos inferida, isto é, após ser submetida à classificação taxonômica e à inferência dos tipos das instâncias.

Na abordagem semântica, uma informação possivelmente incompleta, cadastrada, por exemplo, pelo ator *Grid Application Developer*, pode ser relacionada a outras classes e instâncias existentes na base de metadados, através de inferência sobre os axiomas pré-definidos. Desse modo, uma consulta pode obter informações inferidas, em acréscimo às informações explicitamente declaradas.

Esse é o caso exemplificado na análise das figuras mostradas a seguir. Esses exemplos foram inspirados em duas fontes principais: ontologias de *Data Mining*, como o projeto DAMON, citado no Capítulo 2, que descreve trabalhos relacionados; e um interessante e acessível livro eletrônico disponível em (STATSOFT, 2007).

Várias bibliotecas de componentes de software, arquivos binários e código fonte, sob diversas licenças de uso, são encontrados para esse domínio (ver o sítio (KDNUGETS, 2007)). A utilização de algumas dessas suites de software, como encontradas em (HUNTSVILLE, 2007) e (LJUBLJANA, 2007), são baseadas em *scripts*, permitindo a criação de configurações e composição de aplicações de modo flexível. Essa política de uso dos componentes vem ao encontro da nossa proposta de seleção de componentes para otimizar o uso dos recursos da grade.

A Figura 6.9 mostra a *definição* da classe *PatterRecognitionAlgorithm*, uma classe que representa algoritmos para reconhecimento de padrões. Uma condição necessária e suficiente para inferir-se que um indivíduo é membro dessa classe é que ele se relacione com uma instância da classe *PatterRecognitionApproach* através da propriedade *usedFor*.

A Figura 6.10 mostra a *descrição* da classe *DBSCAN_Algorithm*, que representa um tipo de algoritmo de *clustering*. Na descrição, declara-se que instâncias dessa classe

relacionam-se com instâncias da classe *PatternRecognitionApproach* através da propriedade *usedFor*. Isso é suficiente para inferir-se que essa classe é subclasse de *PatternRecognitionAlgorithm*, como mostrado na Figura 6.11.

Figura 6.9: Definição da classe *PatternRecognitionAlgorithm*

Figura 6.10: Descrição da classe *DBSCAN_Algorithm*

Figura 6.11: Hierarquia inferida

De acordo com os mecanismos definidos na política de recuperação de informações do nosso modelo, descrita na Seção 5.3.3 do Capítulo 5, podemos tirar vantagem desse tipo de inferência, adotando, particularmente, o terceiro e o quarto mecanismos, na realização de consultas, como as dos exemplos abaixo:

- Selecione Executáveis de aplicações para reconhecimento de padrões;
- Selecione Executáveis para reconhecimento de padrões para a plataforma Windows;

A realização dessas consultas recuperou a totalidade de informações esperadas, porque elas já estavam disponibilizadas na base de metadados, como resultado da inferência. Sem o recurso da inferência, esse resultado não seria possível.

A estratégia de resolução da primeira consulta segue o seguinte roteiro:

1. identifica subclasses declaradas e inferidas de algoritmos para reconhecimento de padrões;

2. em seguida, identifica as aplicações que codificam esses algoritmos (incluindo subclasses declaradas e inferidas dessas aplicações);
3. finalmente, recupera instâncias de executáveis que implementam essas aplicações.

A estratégia de resolução da segunda consulta apenas impõe uma restrição sobre a estratégia descrita acima, para a primeira consulta.

Essa flexibilidade pode levar a consultas com resultados inadequados em termos de relevância. As medidas comumente usadas para medir a efetividade das consultas são *precisão* e *cobertura* (RIJSBERGEN, 1979). Precisão é a razão entre o número de itens recuperados pelo sistema que são relevantes para a consulta e o número total de itens recuperados. Cobertura é a razão do número de itens relevantes recuperados sobre o total de itens relevantes em um conjunto.

Em outras palavras, consultas que acessam bases de conhecimentos inferidas podem recuperar resultados com diferentes graus de precisão e cobertura. Uma vantagem desse tipo de consultas é que a precisão dos resultados das consultas pode ser controlada, interativamente, pelo usuário.

Nossa proposta, particularmente, é baseada em um pequeno conjunto de classes e propriedades com significado bem definido em contexto simples de aplicações e recursos, que podem ser estendidos e usados em diferentes domínios verticais na grade. As classes relacionadas abaixo podem ser estendidas para quaisquer domínios que envolvem aplicações e recursos:

- Classes: *Domain*, *Problem*, *Approach*, *Algorithm*, *Application*, *Executable*, *OperatingSystem*, *Processor*;
- Propriedades: *hasOS*, *hasProcessor*, *isRelated*, *dealsWith*, *describes*, *codifies*, *implements*, *usedFor*.

Isso permite definir um conjunto de modelos de consultas que podem ser refinadas em um escopo restrito. Por exemplo, a restrição *implements ONLY Algorithm*, sobre a

classe *Executable*, estabelece que um membro da classe *Executable* se relaciona, através da propriedade *implements*, apenas com membros da classe *Algorithm*. Essas restrições podem ser usadas para orientar as interfaces de usuários para a realização de consultas sobre a base de conhecimento.

6.4 Conclusão

Este capítulo descreveu uma arquitetura para exploração de base semântica de conhecimentos sobre grades computacionais, com ênfase sobre aspectos relevantes de sua implementação, apresentou experimentos para validação da arquitetura proposta e analisou os resultados obtidos. A arquitetura é baseada em um conjunto de componentes cuja principal finalidade é permitir a utilização da base de metadados para descrição de grades computacionais e seus conteúdos. Através da arquitetura proposta, os metadados podem ser utilizados em tarefas relevantes no contexto de grades, tais como, busca de aplicações e escalonamento de recursos para execução na grade. Uma característica fundamental da arquitetura é a independência do middleware de grade, o que permite a reutilização dos metadados para diferentes sistemas de middleware, além de promover a integração desses sistemas.

7 Contribuições, Conclusão e Trabalhos Futuros

Conforme definido no capítulo introdutório, o principal objetivo deste trabalho foi contribuir para a integração e gerenciamento eficiente de recursos em grades computacionais.

Especificamente, a nossa proposta contribui para suavizar os efeitos da heterogeneidade das grades e otimizar o compartilhamento e uso de recursos internamente nas grades e entre grades diferentes, através do uso de um modelo semântico de metadados.

O modelo semântico de metadados propõe a extensão da base de informações sobre recursos e aplicações das grades através da realização de inferência sobre informações capturadas, cuja natureza é comumente incompleta.

Para concretizar esse objetivo, definimos e implementamos uma arquitetura em torno dos metadados sobre aplicações e recursos em grades computacionais, apresentada no Capítulo 6.

A motivação inicial para o projeto partiu da nossa experiência de desenvolvimento junto ao projeto InteGrade. O middleware InteGrade, base inicial da implementação, possui um repositório de aplicações e dados desenvolvido como parte de nossas atividades acadêmicas no projeto de doutorado.

O repositório de aplicações do InteGrade está associado atualmente a uma base incipiente de metadados para descrição das aplicações. Os passos iniciais da presente abordagem buscaram estender essa base, conforme a nova arquitetura proposta e validá-

la através da realização de casos de uso em torno da reutilização do conteúdo do repositório e de sua integração com demais recursos da grade, como por exemplo na seleção de aplicações e no escalonamento de nós para execução de tarefas.

A versatilidade do modelo proposto foi demonstrada através de seu uso em diferentes estruturas de grade, como o MAG (SOUSA et al., 2006). Nossa meta foi usar o modelo para representar diferentes middleware e conteúdos de grades.

O foco da presente tese foi dirigido para a definição de um modelo de recuperação de informações baseado em estratégias combinadas de consulta e inferência que permitiu a exploração flexível da base de conhecimento em diferentes domínios verticais.

7.1 Contribuições

Entendemos que a nossa contribuição se definiu pontualmente, pelo emprego da abordagem semântica no contexto de dois cenários específicos relacionados às atividades desenvolvidas em grades computacionais. Esses cenários tratam especificamente das atividades de busca de aplicações na grade e do escalonamento de recursos para a execução das aplicações. O uso da nossa abordagem na exploração desses cenários foi discutido no Capítulo 3.

De um modo geral, o presente trabalho contribui, através da utilização da abordagem semântica sobre metadados na grade, para a exploração da sinergia desses cenários, especialmente sobre os aspectos relacionados à reutilização de aplicações e recursos. O Capítulo 5 trata dos detalhes dessas contribuições. Mostramos naquele capítulo que o uso da abordagem semântica promoveu a reutilização eficiente de aplicações e recursos da grade.

Consideramos que o valor da nossa contribuição deriva particularmente da natureza aberta da grade. Assim como as próprias grades, suas estruturas e recursos, as aplicações desenvolvidas, registradas e executadas na grade, também são naturalmente heterogêneas, em relação a inúmeros aspectos, tais como origem, domínio, problema,

abordagem, algoritmo, linguagens e ferramentas, desempenho, precisão, plataformas de implementação e assim por diante. Em razão dessa diversidade, não é razoável esperar que os metadados sobre essas aplicações sejam a priori homogêneos ou sigam um processo linear de construção. A estrutura semântica em nossa proposta permite a descrição incremental de recursos e aplicações e possibilita, posteriormente, a descoberta de relacionamentos e conexões entre essas entidades a partir de axiomas definidos na base de conhecimentos. Principalmente em razão dessas características, nossa abordagem de metadados se adequa sobremaneira aos ambientes de grades computacionais.

Com este trabalho esperamos contribuir para a integração, compartilhamento e reutilização de recursos, aplicações e dados para grades computacionais, possibilitando em consequência a integração e interoperabilidade de diferentes estruturas de grades. As contribuições estão presentes no seguintes pontos.

7.1.1 Contribuições técnicas

O nosso trabalho incorpora as seguintes contribuições técnicas:

- Definição de uma taxonomia concreta e reutilizável para representar conceitos sobre aplicações e recursos em grades de computadores;
- Projeto e implementação de um modelo de metadados para descrever aplicações e recursos para grades computacionais e seus relacionamentos;
- Definição de uma heurística, baseada em semântica, para exploração das taxonomias e mecanismos de inferência no contexto dos cenários de busca e descoberta de aplicações na grade e de escalonamento de aplicações e alocação de recursos na grade.

7.1.2 Contribuições científicas

Consideramos que este trabalho produziu as seguintes contribuições científicas:

- Definição de mecanismos inovadores para incrementar o potencial de integração, reutilização e compartilhamento de aplicações e outros recursos em grades computacionais através da definição e implementação de um modelo de metadados sobre esses recursos;
- Proposição de um modelo de utilização de metadados, baseado em uma estrutura de metadados extensível para diferentes domínios verticais. As classes base das ontologias propostas se constituem em um metamodelo a partir do qual diferentes domínios concretos podem ser descritos;
- Publicação de artigos científicos:
 - *Semantics-Based Grid Resource Management* (VIDAL et al., 2007)
 - *Enhancing Grid Usage through Semantic Metadata (Short Paper)* (VIDAL; KOFUJI, 2007)
 - *Defining and exploring a grid system ontology (Short Paper and Poster)* (VIDAL et al., 2006)
 - *Repositório Seguro de Aplicações baseado em GSS (Artigo Curto)* (BRAGA; VIDAL; KON, 2005)

7.2 Trabalhos futuros

Um dos principais aspectos que pretendemos explorar de imediato é a definição de mecanismos que permitam reduzir o tempo de realização de inferência sobre a base de metadados. A abordagem que pretendemos explorar inicialmente é baseada na separação e mapeamento do conjunto de metadados, buscando limitar o volume de dados submetidos à inferência.

Consideramos também que a extensão da base de conhecimentos e da arquitetura para atender o domínio de composição de aplicações baseadas em componentes é um campo promissor a ser explorado por trabalhos futuros elaborados sobre a nossa proposta.

Os aspectos relacionados a dados e suas conexões semânticas com aplicações e recursos na grade, não foram explorados nesse trabalho. A nossa meta, no grupo de pesquisa de sistemas e mobilidade do Departamento de Informática da UFMA, é explorarmos esses aspectos no escopo do middleware MAG (SOUSA et al., 2006), referido no Capítulo 5.

Apêndice A – Gerenciamento de Aplicações e Recursos em Grades

Este apêndice apresenta uma visão sucinta sobre taxonomias e conceitos relevantes no domínio do gerenciamento de recursos e aplicações em grade e tem como principal objetivo compor um painel de visões diferentes e complementares acerca do gerenciamento de recursos em grades. Alguns aspectos da organização e dos conceitos presentes nas ontologias do Capítulo 4 foram influenciados pelos resultados deste levantamento.

A.1 Conceitos e Taxonomias

A ordem de apresentação das taxonomias nesta seção segue um padrão no qual os conceitos evoluem do contexto de sistemas distribuídos para o contexto de grades computacionais.

A taxonomia apresentada em (CASAVANT; KUHL, 1988) volta-se especificamente para o problema de escalonamento distribuído e de um conjunto de algoritmos relacionados. Contudo, esse trabalho busca tratar o problema do escalonamento sob uma visão mais geral, como um problema de gerenciamento de recursos; em outras palavras, como um mecanismo ou política usada para gerenciar de modo efetivo e eficiente o acesso e o uso dos recursos pelos consumidores. Nesse sentido, o trabalho sugere que a taxonomia proposta ultrapassa o escopo do escalonamento de recursos, podendo portanto ser empregada de modo proveitoso para classificar qualquer conjunto de sistemas de gerenciamento de recursos. No contexto do nosso trabalho, essa taxonomia introduz uma visão geral dos conceitos relacionados ao gerenciamento de recursos.

A taxonomia é descrita de uma forma híbrida, com alguns conceitos organizados hierarquicamente enquanto outros não se adequam a essa estrutura. A Figura A.1 é parte de uma figura encontrada em (CASAVANT; KUHL, 1988) e contém uma hierarquia de características relacionadas à tarefa de escalonamento. Alguns desses termos são rapidamente descritos a seguir.

Figura A.1: Características das tarefa de escalonamento (CASAVANT; KUHL, 1988)

As características abaixo são parte da hierarquia mostrada na Figura A.1.

- *Local vs. Global* - Escalonamento local é relacionado com a atribuição de processos a fatias de tempo em uma única máquina. Escalonamento Global trata do problema de decidir em qual máquina executar o processo.
- *Static* - Indica o momento no qual o escalonamento ou decisões de atribuição são feitas. No escalonamento estático espera-se que as informações sobre os processos no sistema estejam disponíveis em tempo de ligação dos módulos objetos e módulos de carga. Isso resulta em uma atribuição estática dos executáveis a um processador ou configuração de hardware específica.

Optimal vs. Sub-Optimal - Havendo disponibilidade de informação (solução estática), uma atribuição ótima pode ser realizada com base em algum critério, como por exemplo, minimização do tempo total de conclusão de um processo, utilização máxima de recursos, ou maximização da *vazão*, ou taxa de transferência efetiva do sistema.

Approximate vs. Heuristic (categorias definidas no domínio da solução sub-ótima)

Approximate: Esta solução usa o modelo computacional formal do algoritmo, mas não espera pela solução ótima, contentando-se com uma solução que considera satisfatória;

Heuristic: Assume uma hipótese considerada a mais realista sob a ótica de um conhecimento disponível, a priori, acerca das características do processo e da carga do sistema. Em geral, essa abordagem consome uma maior quantidade de tempo e outros recursos do sistema na realização de suas funções.

A característica mais notória do escalonamento heurístico é o uso de parâmetros especiais que afetam o sistema de modo indireto. É comum existir uma correlação indireta entre o desempenho do sistema e um parâmetro sendo monitorado. O agrupamento de processos que se comunicam intensamente em um mesmo processador e, de modo contrário, a separação de processos que se beneficiam de paralelismo, influenciam diretamente na redução do custo relacionado à passagem de informação entre processadores, reduzindo ao mesmo tempo a interferência entre processos que prescindem de sincronização entre si. Essa influência não pode, contudo, ser aferida quantitativamente. Intuitivamente, associamos as ações de agrupamento/separação dos processos aos resultados percebidos sobre a melhoria do desempenho do sistema. Contudo, não somos capazes de provar a existência de um relacionamento de primeira ordem entre mecanismos empregados e resultados obtidos.

- *Soluções Dinâmicas* - Usadas quando se dispõe de muito pouca informação a

priori sobre os recursos requisitados por um processo;

- *Distributed vs. Non-Distributed* - Esta definição está relacionada à responsabilidade sobre a decisão de escalonamento: se a tarefa de escalonamento dinâmico global deve ser de um processador único (fisicamente não distribuído) ou se o trabalho envolvido nessas decisões deve ser fisicamente distribuída entre diferentes processadores; o conceito em questão é definição da autoridade lógica responsável pelas tomadas de decisões na tarefa de escalonamento;
- *Cooperative vs. Non-Cooperative* - O escalonamento dinâmico pode ser classificado como cooperativo e não-cooperativo. Essa classificação diz respeito ao grau de autonomia que tem cada processador na determinação de como seus recursos devem ser usados. No escalonamento cooperativo cada processador assume a responsabilidade sobre sua parte na tarefa de escalonamento, mas todos os processadores trabalham em razão de um objetivo comum. No escalonamento não-cooperativo cada processador decide sobre seus próprios recursos, “indiferente” aos efeitos de suas decisões sobre o resto do sistema.

As características relacionadas abaixo são classificadas de modo não-hierárquico:

- *Adaptive vs. Non-Adaptive* - Na solução adaptativa para o problema do escalonamento, algoritmos e parâmetros usados para implementar a política de escalonamento mudam dinamicamente em resposta ao comportamento anterior e atual do sistema.
- *Load Balancing* - Essa solução tenta balancear a distribuição de tarefas entre os processadores. Ela se mostra mais efetiva em ambientes homogêneos, uma vez que todos os nós conhecem, em grande extensão, sobre a estrutura dos demais nós.
- *Bidding* - Essa solução define um arcabouço básico para protocolo que descreve como os processos são designados para os processadores. Nesse arcabouço cada nó exerce simultaneamente dois papéis: contratante e fornecedor. Como contratante,

o nó anuncia uma tarefa que precisa ser executada, recebe propostas dos nós fornecedores, e tem autonomia para fazer a escolha do nó que vai executar a tarefa.

- *Probabilistic* - Nessa solução, realiza-se uma escolha aleatória do próximo processo a ser atribuído a um processador, com base em alguma distribuição conhecida. Essa estratégia é motivada porque a quantidade de tempo requerida para uma tomada de decisão analítica poderia ser proibitiva em razão do número potencialmente excessivo de mapeamentos de processos.
- *One-time Assignment vs. Dynamic Reassignment* - Nesse caso, as atribuições dos processos são feitas uma única vez ou alteradas dinamicamente em razão do tipo de entidades escalonadas. A reatribuição dinâmica tenta melhorar uma decisão anterior através do uso de informações para modificar atribuições de unidades menores de computação, tais como subtarefas.

De acordo com o exposto em (CASAVANT; KUHL, 1988), do ponto de vista do consumidor, a métrica comumente envolvida na avaliação de desempenho de estratégias de escalonamento é o tempo de finalização de um programa, enquanto que do ponto de vista dos recursos a taxa de execução dos processos é considerada no escopo da taxa de vazão total do sistema.

Ainda de acordo com (CASAVANT; KUHL, 1988), essas métricas são difíceis de avaliar durante a operação normal do sistema e também na realização de análises ou simulações dos sistemas. A razão seria a necessária interconexão entre dois aspectos importantes do escalonamento: desempenho e eficiência. Desempenho é a parte do comportamento do sistema que diz respeito à quão bem o recurso gerenciado está sendo usado em benefício de todos os usuários do sistema. Eficiência é relacionada ao custo adicional do gerenciamento do próprio recurso.

O comportamento ideal para um sistema deveria elevar ao máximo o seu nível de desempenho, incorrendo, ao mesmo tempo, no menor custo adicional possível, isto é, de modo eficiente.

O trabalho apresentado em (BRAUN et al., 1998) propõe uma taxonomia voltada para ambientes de computação heterogêneos e discute problemas relacionados a tais ambientes. A heterogeneidade é, também, como vimos no Capítulo 1, uma característica de grades computacionais.

De acordo com (BRAUN et al., 1998), o problema de mapeamento de tarefas e comunicação para múltiplas máquinas e redes em ambientes de computação heterogêneos, definido como mapeamento e escalonamento, é NP-completo, exigindo o desenvolvimento de abordagens heurísticas.

A taxonomia proposta aborda essas questões através dos conceitos tratados e organizados em 3 modelos principais: o modelo de requisitos de aplicações e comunicação, o modelo de plataformas de hardware e o modelo de heurísticas de mapeamento. Esses modelos são sucintamente descritos nos próximos parágrafos:

No modelo de requisitos de aplicações e comunicação, as aplicações são classificadas por suas características computacionais que podem influenciar decisões de mapeamento. As características consideradas das aplicações incluem entre outras:

- O tamanho da aplicação, isto é, o número de tarefas e subtarefas da aplicação;
- O tipo da aplicação: as tarefas e subtarefas são independentes ou dependentes?
- Padrões de comunicação: quais as tarefas de origem e destino dos itens de dados a serem transferidos;
- A disponibilidade de dados:
 - Os dados de saída estão disponíveis antes de se completar uma sub tarefa?
 - Uma sub tarefa pode começar a execução sem receber todos os dados de entrada?
- Modelo do tempo de execução: estimativa do tempo de execução de cada tarefa em cada máquina;

- Heterogeneidade de multitarefa: para cada máquina, em que extensão e sob quais condições, o tempo de execução de cada tarefa varia para diferentes tarefas?
- Múltiplas versões: a aplicação tem múltiplas versões que podem ser executadas? Por exemplo, uma aplicação que requer uma FFT (*Fast Fourier Transform*) (CORMEN et al., 2001) poderia ser capaz de executar a FFT com uma de duas diferentes implementações disponíveis, com diferentes precisões, tempos de execução e requisitos de recursos; essa característica envolve a identificação dos diferentes “valores” relativos das versões do ponto de vista dos usuários;
- Prioridades: podem ser atribuídas prioridades para a execução das aplicações?
- Requisitos QoS precisam ser considerados para algumas aplicações?

O Modelo para plataformas de hardware trata dos padrões de plataformas de hardware disponíveis nos sistemas de computação heterogêneos. Os modelos considerados incluem:

- *Benchmarks* analíticos: provêm medidas de desempenho de cada máquina disponível na plataforma heterogênea, para diferentes tarefas computacionais;
- Tempo de comunicação: tempo de transferência de dados de uma máquina a outra que pode ser expresso como uma função do tempo de definição do caminho e da largura de banda;
- *Send/Receive concorrentes*: as máquinas podem executar operações concorrentes de *send* e *receive* para outras máquinas?
- Rede de interconexão: muitas características de rede podem afetar decisões de mapeamento e o desempenho do sistema; por exemplo: largura de banda, latência, capacidade para transferência de dados de modo concorrente, controle de roteamento e topologia;
- Arquiteturas de máquinas: vários atributos da arquitetura das máquinas, que podem ter impacto no desempenho, devem ser considerados, como por exemplo,

tipo do processador, velocidade do processador, tamanho da memória e número de processadores em máquinas paralelas;

- Heterogeneidade do conjunto de máquinas: para cada tarefa, em que extensão e sob quais condições, o tempo de execução de cada tarefa varia em diferentes máquinas?
- Número de conexões de uma máquina na rede ou diretamente com outras máquinas;
- Número de máquinas: finito ou infinito, fixo ou variável;
- Sobreposição de tarefas de computação e comunicação;
- Controle do sistema: a estratégia de mapeamento controla todos os recursos? Ou os recursos são compartilhados com usuários externos?
- Compatibilidade de tarefas: todas as máquinas podem executar todas as aplicações? Ou existem aplicações que exigem capacidades não disponíveis em todas as máquinas? Essas capacidades podem envolver questões como I/O, espaço de memória e segurança.

O modelo de estratégias de mapeamento descreve características para a descrição das estratégias de mapeamento, e inclui, entre outras:

- Aferição do tempo de comunicação: o tempo de comunicação entre subtarefas são considerados no mapeamento?
- Localização do controle da estratégia: a estratégia de mapeamento é centralizada ou distribuída? se distribuída, é cooperativa ou não-cooperativa?
- Dependências: mapeamento de meta-tarefas lida com um conjunto de tarefas independentes, enquanto mapeamento de subtarefas lida com subtarefas dependentes;

- Transferência de dados: a execução de uma subtarefa em uma máquina poderia receber dados de uma máquina intermediária de modo mais imediato que da fonte original?
- Possibilidade de duplicação das tarefas: uma dada subtarefa pode ser duplicada e executada em múltiplas máquinas para reduzir sobrecarga de comunicação?
- Mapeamento estático e dinâmico: técnicas de mapeamento dinâmico operam durante (e possivelmente antes) da execução da aplicação e usam informação em tempo real, podendo não conhecer, a priori, o total de tarefas a serem executadas, assim como uma tarefa pode ser redirecionada em razão de falhas de uma máquina; o mapeamento estático lida com conjuntos fixos de aplicações e máquinas;
- Localização da execução do mapeamento: a estratégia de mapeamento será executada por qualquer máquina de um conjunto? ou por uma máquina externa?
- Tempo de execução da estratégia de mapeamento: esse tempo pode ser previsto com precisão?
- Mecanismos para tolerância a falhas são considerados na estratégia de mapeamento?

Uma visão de taxonomias mais próxima do domínio da nossa investigação é apresentada em (KRAUTER; BUYYA; MAHESWARAN, 2002). Este trabalho descreve um conjunto de definições essenciais e problemas relevantes em gerenciamento de recursos em grades. No contexto de grades de computadores, *Recursos* são definidos como entidades reutilizáveis empregadas para atender à demanda de tarefas realizadas na grade. Esse conceito inclui desde itens tangíveis como máquinas e estruturas de redes, até serviços que empregam em sua realização uma combinação de máquinas, redes e software. O *provedor de recursos* é definido como uma agente responsável pelo controle dos recursos. Enquanto o *consumidor de recursos* é definido como um agente que representa e controla os consumidores.

O trabalho apresentado em (KRAUTER; BUYYA; MAHESWARAN, 2002) desenvolve um modelo abstrato e uma taxonomia que inclui uma totalidade suficiente de termos para a descrição de arquiteturas de gerenciamento de recursos. A argumentação sobre modelo e taxonomia é elaborada a partir dos conceitos de sistemas de computação em rede e grades. Um sistema de computação em rede é definido como um conjunto de máquinas heterogêneas conectadas em rede que concordam em compartilhar seus recursos locais entre si. Uma grade é definida como um sistema genérico de redes distribuído em grande escala, podendo alcançar ambientes em escala de internet, cujas máquinas estão distribuídas através de múltiplas organizações e domínios administrativos.

Um *sistema de gerenciamento de recursos* é definido como um serviço provido por um sistema de redes distribuído que gerencia um grupo de recursos nomeados disponíveis para a rede, com o objetivo de otimizar uma determinada medida de desempenho sobre uma tarefa ou sobre o sistema.

A Figura A.2 apresenta três categorias de sistemas de grades. A categoria de grade computacional, *Computational Grid*, denota sistemas com maior poder computacional agregado para execução de aplicações que qualquer das máquinas do sistema isoladamente. As grades de dados, *Data Grid*, incluem sistemas que provêem infra-estrutura para síntese de novas informações em repositórios de dados, tais como, bibliotecas digitais e depósitos de dados. A categoria de grades de serviços, *service Grid*, é adequada para sistemas que provêem serviços que não podem ser providos por uma única máquina.

Figura A.2: Taxonomia de sistemas de grade. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)

O modo como as máquinas são organizadas na grade afetam os padrões de comu-

nicação dos sistemas de gerenciamento de recursos e conseqüentemente determinam a escalabilidade da arquitetura resultante. A Figura A.3 mostra a taxonomia proposta em (KRAUTER; BUYYA; MAHESWARAN, 2002) para modelos de organização das máquinas. Essa organização descreve como as máquinas responsáveis pelo gerenciamento de recursos tomam decisões de escalonamento, a estrutura de comunicação entre essas máquinas e os diferentes papéis que essas máquinas desempenham nas decisões de escalonamento.

Figura A.3: Taxonomia do modelo de organização de máquinas. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)

O modelo de recursos determina como as aplicações e os sistemas de gerenciamento de recursos descrevem recursos na grade. O foco da taxonomia apresentada na Figura A.4 é voltado para a descrição dos recursos na grade e das operações sobre eles.

Figura A.4: Taxonomia do modelo de recursos. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)

Uma vez que um sistema de gerenciamento de recursos em grades operam sobre um conjunto de recursos nomeados globalmente, a organização de um espaço de nomes que possa ser criado e mantido por uma entidade externa ao sistema terá forte influência sobre o projeto dos protocolos de gerenciamento de recursos e também sobre métodos de descoberta. A Figura A.5 mostra a taxonomia sobre a organização do espaço de nomes.

Figura A.5: Taxonomia da organização do espaço de nomes. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)

A Figura A.6 apresenta a taxonomia para descoberta de Recursos e a Figura A.7 mostra a taxonomia para disseminação de recursos, definidas em (KRAUTER; BUYYA; MAHESWARAN, 2002). Segundo (KRAUTER; BUYYA; MAHESWARAN, 2002), a descoberta e disseminação de recursos podem ser consideradas funções complementares. Enquanto a descoberta é iniciada pela aplicação na rede para encontrar recursos adequados para a sua execução na grade, a disseminação é iniciada pelo recurso tentando encontrar uma aplicação adequada para usá-lo.

Figura A.6: Taxonomia de descoberta de recursos. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)

Figura A.7: Taxonomia de disseminação de recursos. Fonte: (KRAUTER; BUYYA; MAHESWARAN, 2002)

Neste apêndice tratamos os conceitos relacionados a sistemas de gerenciamento de recursos em grades computacionais. Optamos por uma perspectiva baseada em taxono-

mias em razão da forte relação entre os modelos de abordagem semântica de informações e as estruturas taxonômicas. As ontologias propostas não têm como objetivo incorporar todos os aspectos tratados nas taxonomias descritas neste capítulo, principalmente em razão da amplitude dessas taxonomias. Apenas um reduzido número dos conceitos apresentados foram abordados, com o objetivo de avaliar a pertinência da proposta.

Apêndice B – Consultas SPARQL para avaliação de desempenho

Este apêndice apresenta um conjunto incompleto das consultas, em linguagem SPARQL, realizadas sobre ontologias estendidas a partir das ontologias base propostas. Essas ontologias compreendem conceitos sobre recursos de hardware (plataformas e computadores) e software (domínios, problemas, algoritmos, aplicações e executáveis). O conjunto foi usado na realização do experimento para avaliação do desempenho de consultas sobre bases com diferentes tamanhos.

- “SELECT ?x ?y ?z WHERE ?x ?y ?z”;
- “SELECT ?x WHERE ?x <#hasProcessor> <#Pentium4CPU3.00> ”;
- “SELECT ?x WHERE ?x <#type> <#Windows> ”;
- “SELECT ?x WHERE ?x <#type> <#Ubuntu> ”;
- “SELECT ?x WHERE ?x <#subClassOf> <#Linux> ”;
- “SELECT ?x WHERE ?x <#hasOS> <#AMDAthlonXP2800> ”;
- “SELECT ?x WHERE ?x <#subClassOf> <#AMD> ”;
- “SELECT ?x WHERE ?x <#hasProcessor> <#PowerPC_G4> ”;
- “SELECT ?x WHERE ?x <#hasOS> <#Linux> ”;
- “SELECT ?x WHERE ?x <#type> <#BSD> ”;

- “SELECT ?x WHERE ?x <#subClassOf> <#Intel> ”;
- “SELECT ?x WHERE ?x <#codifies> <#KNN_Algorithm> ”;
- “SELECT ?x WHERE ?x <#dealsWith> <#TextureAnalysisProblem> ”;
- “SELECT ?x WHERE ?x <#isRelatedTo> <#DataMiningDomain> ”
- “SELECT ?x WHERE ?x <#implements> <#PSM_Application> ”

Apêndice C – Consultas baseadas na API Protégé-OWL

Este apêndice apresenta alguns exemplos de consultas realizadas sobre as ontologias através do emprego de métodos definidos na biblioteca Protégé-OWL API. Os métodos disponíveis na biblioteca constituem uma alternativa para recuperação de informações das ontologias, em comparação com as consultas baseadas em SPARQL. Tais métodos podem ser utilizados para percorrer as ontologias de modo iterativo e podem ser usados em conjunto com o mecanismo de inferência associado ao modelo de metadados.

- *public void subclassQuery(JenaOWLModel model, OWLNamedClass myclasse)*

Este método percorre uma ontologia para identificar as subclasses declaradas da ontologia que são subclasses de uma determinada classe;

- *public void inferredSubclassQuery(JenaOWLModel model, OWLNamedClass my-classe)*

Este método percorre uma ontologia para identificar as subclasses de uma determinada classe da ontologia, inferidas através do mecanismo de raciocínio associado ao modelo;

- *public void getInstancesQuery(JenaOWLModel model, String name)*

Este método permite recuperar instâncias declaradas de uma determinada classe;

- *public void getInferredInstancesQuery(JenaOWLModel model, String name)*

Como uma alternativa ao método anterior, este método recupera instâncias cujos tipos foram inferidos pelo mecanismo de inferência associado ao modelo de metadados;

- *public void classEquivalentQuery(JenaOWLModel model, OWLNamedClass my-classe)*

Este método identifica e recupera classes equivalentes identificadas pelo mecanismo de inferência.

Referências

- BAADER, F.; CALVANESE, D.; MCGUINNESS, D.; NARDI, D.; PATEL-SCHNEIDER, P. *The Description Logic Handbook: Theory, implementation, and application*. Cambridge, UK: Cambridge University Press, 2003.
- BECHHOFFER, S.; HARMELEN, F. V.; HENDLER, J.; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEINER, L. A. *OWL Web Ontology Language Reference*. February 2004. Disponível em: <<http://www.w3.org/TR/2004/REC-owl-ref-20040210/>>.
- BECHHOFFER, S.; LORD, P.; VOLZ, R. Cooking the semantic web with the OWL API. In: *Proceedings of the 2nd. Int. Semantic Web Conference (ISWC 2003)*. Sanibel Island, Florida, USA: Springer-Verlag, 2003.
- BECHHOFFER, S.; MÖLLER, R.; CROWTHER, P. The DIG description logic interface. In: *Proceedings of the Int. Description Logics Workshop, DL'2003*. Rome, Italy: CEUR-WS.org, 2003.
- BECKETT, D. *Turtle - Terse RDF Triple Language*. December 2006. Disponível em: <<http://www.dajobe.org/2004/01/turtle/>>.
- BERMAN, F.; FOX, G. C.; HEY, A. J. The grid: past, present, future. In: *Grid Computing - Making the Global Infrastructure a Reality*. New York, NY, USA: John Wiley & Sons, 2003. p. 9–50.
- BERNERS-LEE, T. *Notation 3*. March 2006. Disponível em: <<http://www.w3.org/DesignIssues/Notation3.html>>.
- BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, v. 279, n. 5, p. 33–43, March 2001.
- BRAGA, J. José de R. P.; VIDAL, A. C. T.; KON, F. Repositório Seguro de Aplicações baseado em GSS (Artigo Curto). In: *Anais do V Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Florianópolis, SC, Brasil: SBSEG, 2005. v. 1, p. 271–274.
- BRANK, J.; GROBELNIK, M.; MLADENI, D. A survey of ontology evaluation techniques. In: *8th Int. multi-conference Information Society IS-2005*. Ljubljana, Slovenia: Josef Stefan Institute, 2005.
- BRAUN, T. D.; SIEGEL, H. J.; BECK, N.; BNI, L.; MAHESWARAN, M.; REUTHER, A. I.; ROBERTSON, J. P.; THEYS, M. D.; YAO, B. A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems. In: *Symposium on Reliable Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 1998. p. 330–335.

BROOKE, J.; FELLOWS, D.; GARWOOD, K.; GOBLE, C. Semantic Matching of Grid Resource Descriptions. In: *Second European AcrossGrid Conference, AGrids 2004*. Nicosia, Cyprus: Springer Verlag, 2004. v. 3165, p. 240–249.

CAMARGO, R. Y. de; GOLDCHLEGER, A.; CARNEIRO, M.; KON, F. The Grid Architectural Pattern: Leveraging Distributed Processing Capabilities. In: *Pattern Languages of Program Design 5*. Boston, MA, USA: Addison-Wesley, 2006, (Software Pattern Series). p. 337–356.

CANNATARO, M.; COMITO, C. A Data Mining Ontology for Grid Programming. In: *Workshop on Semantics in Peer-to-Peer and Grid Computing (in conj. with WWW2003)*. Budapest, Hungary: ERCIM EEIG, 2003. p. 113–134.

CANNATARO, M.; COMITO, C.; SCHIAVO, F. L.; VELTRI, P. PROTEUS: a Grid Based Problem Solving Environment for Bionformatics. *IEEE Computational Intelligence Bulletin*, v. 3, n. 1, p. 7–18, 2004.

CANNATARO, M.; TALIA, D. The knowledge grid. *Commun. ACM*, ACM Press, New York, NY, USA, v. 46, n. 1, p. 89–93, 2003. ISSN 0001-0782.

CANNATARO, M.; TALIA, D. Semantics and knowledge grids: building the next-generation grid. *Intelligent Systems, IEEE*, IEEE Computer Society, v. 19, n. 1, p. 56–63, 2004.

CARROL, J. J.; ROO, J. D. *OWL Web Ontology Language Test Cases, W3C Recommendation*. 2004. Disponível em: <<http://www.w3.org/TR/owl-test/>>.

CARROLL, J.; DICKINSON, I.; DOLLIN, C.; REYNOLDS, D.; SEABORNE, A.; WILKINSON, K. *Jena: Implementing the semantic web recommendations*. 2003. Disponível em: <citeseer.ist.psu.edu/carroll04jena.html>.

CASAVANT, T.; KUHL, J. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, IEEE Computer Society, Los Alamitos, CA, USA, v. 14, n. 2, p. 141–154, 1988. ISSN 0098-5589.

CHANDRASEKARAN, B.; JOSEPHSON, J.; BENJAMINS, V. What are ontologies, and why do we need them? *Intelligent Systems and Their Applications, IEEE*, v. 14, p. 20–26, jan 1999.

CONDOR, T. condor team. *Condor, High Throughput Computing*. 2007. Disponível em: <<http://www.cs.wisc.edu/condor/>>.

CONRADI, R.; WESTFECHTEL, B. Version models for software configuration management. *ACM Comput. Surv.*, ACM Press, New York, NY, USA, v. 30, n. 2, p. 232–282, 1998. ISSN 0360-0300.

CORCHO, O.; FERNANDEZ-LOPEZ, M.; GOMEZ-PEREZ, A. Methodologies, tools and languages for building ontologies: where is their meeting point? *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 46, n. 1, p. 41–64, 2003. ISSN 0169-023X.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms*. Second. Cambridge, MA, USA: MIT Press and McGraw-Hill, 2001.

CZAJKOWSKI, K.; FITZGERALD, S.; FOSTER, I.; KESSELMAN, C. Grid Information Services for Distributed Resource Sharing. In: *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*. San Francisco, EUA: IEEE Computer Society Press, Los Alamitos, CA, USA, 2001.

DEVEDZIC, V. Understanding Ontological Engineering. *Commun. ACM*, ACM Press, New York, NY, USA, v. 45, n. 4, p. 136–144, 2002. ISSN 0001-0782.

DODDS, L. *Introducing SPARQL: Querying the Semantic Web*. November 2006. Disponível em: <<http://www.xml.com/pub/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html>>.

FIESER, J.; DOWDEN, B. *The Internet Encyclopedia of Philosophy*. 2007. Disponível em: <<http://www.iep.utm.edu/>>.

FIKES, R.; HAYES, P.; HORROCKS, I. *OWL-QL: A Language for Deductive Query Answering on the Semantic Web (Technical Report)*. Stanford, CA, 2003.

FIKES, R.; HAYES, P.; HORROCKS, I.; DEAN, H. B. M.; GROSOFF, B.; HARMELLEN, F. van; HAWKE, S.; HEFLIN, J.; MCGUINNESS, O. L. D.; PATEL-SCHNEIDER, P.; STEIN, L. A. *DAML Query Language*. April 2003. Disponível em: <<http://www.daml.org/2003/04/dql/>>.

FOSTER, I.; KESSELMAN, C. (Ed.). *The Grid 2: blueprint for a new computing infrastructure*. second. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN 1-55860-933-4.

FURMENTO W. LEE, A. M. S. N. N.; DARLINGTON, J. ICENI: An Open Grid Service Architecture Implemented with Jini. In: *SuperComputing 2002*. Baltimore, USA: IEEE Computer Society Press, 2002.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDE, J. *Design Patterns*. Reading, MA, USA: Addison-Wesley Pub Co, 1995.

GERGIC, J. Towards a Versioning Model for Component-based Software Assembly. In: *ICSM '03*. Washington, DC, USA: IEEE Computer Society, 2003. p. 138. ISBN 0-7695-1905-9.

GLOBUS. *Globus*. 2007. Disponível em: <<http://www.globus.org/>>.

GOBLE, C.; ROURE, D. D.; SHADBOLT, N.; FERNANDES, A. Enhancing services and applications with knowledge and semantics. In: *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, CA, USA: Morgan-Kaufmann, 2004. p. 431–458.

GOLDCHLEGER, A.; KON, F.; GOLDMAN, A.; FINGER, M.; BEZERRA, G. C. Integrate: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. *Concurrency and Computation: Practice and Experience*, v. 16, p. 449–59, March 2004.

GONGG. *Ontology Design Patterns*. May 2007. Disponível em: <<http://gongg.man.ac.uk/owl/ontologydesignpatterns>>.

- HAPPEL, H.-J.; KORTHAUS, A.; SEEDORF, S.; TOMCZYK, P. K_{OntoR}: An Ontology-enabled Approach to Software Reuse. In: *Software Engineering and Knowledge Engineering*. San Francisco, CA, USA: Knowledge System Institute Graduate School, 2006. p. 349–354.
- HORRIDGE, M. *A Practical Guide To Building OWL Ontologies With The Protégé-OWL Plugin*. Manchester, U.K: University of Manchester, 2004. Disponível em: <<http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf>>.
- HUNTSVILLE, U. of Alabama in. *Algorithm Development and Mining System*. 2007. Disponível em: <<http://datamining.itsc.uah.edu/adam/>>.
- ICENI. *Imperial College e-Science Networked Infrastructure*. 2007. Disponível em: <<http://www.lesc.imperial.ac.uk/iceni/>>.
- KDNUGGETS. *Software Suites for Data Mining and Knowledge Discovery*. 2007. Disponível em: <<http://www.kdnuggets.com/software/suites.html>>.
- KG, R. S. G. . C. *RacerPro*. 2007. Disponível em: <<http://www.racer-systems.com/>>.
- KIRCHER, M.; JAIN, P. *Pattern-Oriented Software Architecture, Volume 3, Patterns for Resource Management*. West Sussex, UK: Willey, 2004.
- KNUBLAUCH, H.; FERGERSON, R. W.; NOY, N. F.; MUSEN, M. A. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In: *ISWC 2004*. Hiroshima, Japan: Springer-Verlag Berlin and Heidelberg GmbH and Co. K, 2004.
- KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. *Software Practice and Experience*, v. 32, n. 2, p. 135–164, 2002. Disponível em: <citeseer.ist.psu.edu/article/krauter02taxonomy.html>.
- LESC. *The London e-Science Centre*. 2007. Disponível em: <<http://www.lesc.ic.ac.uk/>>.
- LJUBLJANA, U. of. *Data Mining Fruitful&Fun*. 2007. Disponível em: <<http://www.ailab.si/orange/>>.
- LLC, C. . P. *Pellet*. 2007. Disponível em: <<http://pellet.owldl.com/>>.
- MANOLA, F.; MILLER, E. *RDF-PRIMER, W3C Recommendation*. February 2004. Disponível em: <<http://www.w3.org/TR/REC-rdf-primer-20040210>>.
- MCCARTHY, P. *Search RDF data with SPARQL*. May 2005. Disponível em: <<http://www.ibm.com/developerworks/xml/library/j-sparql/>>.
- MCGOUGH, A. S.; LEE, W.; DARLINGTON, J. ICENI II Architecture. In: *UK e-Science All Hands Meeting*. Nottingham, UK: EPSRC, 2005.
- MCGUINNESS, D. L.; HARMELEN, F. V. *OWL Web Ontology Language Overview*. February 2004. Disponível em: <<http://www.w3.org/TR/owl-features>>.

- MIR, A.; HANMANDLU, M.; TANDON, S. Texture analysis of ct images. *Engineering in Medicine and Biology Magazine, IEEE*, Amsterdam, The Netherlands, The Netherlands, v. 14, p. 781– 786, 1995.
- MOORE, R. W. Knowledge-based grids. In: *MSS '01: Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies*. Washington, DC, USA: IEEE Computer Society, 2001. p. 29. ISBN 0-7695-9173-9.
- NEWHOUSE, S.; MAYER, A.; FURMENTO, N.; MCGOUGH, S.; STANTON, J.; DARLINGTON, J. Laying the foundations for the semantic grid. In: *AISB'02 Symposium AI and GRID Computing*. London, UK: Imperial College, 2002. ISBN 1902956248.
- NOY, N. F.; MCGUINNESS, D. L. *Ontology Development 101: A Guide to Creating Your First Ontology (Technical Report)*. March 2001.
- OGF. *Open Grid Forum*. 2007. Disponível em: <<http://www.ogf.org>>.
- OHLBACH, H. J.; KOEHLER, J. Modal logics, description logics and arithmetic reasoning. *Artificial Intelligence*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 109, n. 1-2, p. 1–31, 1999.
- OMG. *UML Resource Page*. april 2006. Disponível em: <<http://www.uml.org/>>.
- PAGELS, M. *The DARPA Agent Markup Language Homepage*. 2005. Disponível em: <<http://www.daml.org/>>.
- PALLAS; MANCHESTER, U. of; ICM; DWD; SOUTHAMPTON, U. of; FUJITSU; ARGONNE. *GRid Interoperability Project (GRIP)*. 2007. Disponível em: <<http://www.gloriad.org/gloriad/projects/project000032.html>>.
- PATEL-SCHNEIDER, P. F.; HORROCKS, I. Position paper: a comparison of two modelling paradigms in the semantic web. In: *15th International World Wide Web Conference*. Edinburgh, Scotland: ACM Press, New York, NY, 2006.
- PLASIL, F.; BALEK, D.; JANECEK, R. Sofa/dcup: Architecture for component trading and dynamic updating. In: *CDS '98: Proceedings of the International Conference on Configurable Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 1998. p. 43. ISBN 0-8186-8451-8.
- PROGRAMME, H. L. S. *Jena - A Semntic Web Framework dor Java*. 2007. Disponível em: <<http://jena.sourceforge.net>>.
- PROTÉGÉ. *Using the protégé-owl reasoner api*. Stanford, CA, USA: Stanford University, 2007. Disponível em: <<http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html>>.
- RECTOR, A.; DRUMOND, N.; HORRIDGE, M.; ROGERS, J.; KNUBLAUCH, H.; STEVENS, R.; WANG, H.; WROE, C. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: *Engineering Knowledge in the Age of the Semantic Web*. Whittleburg Hall, UK: Springer Berlin/Heidelberg, 2004, (Lecture Notes in Computer Science). p. 63–81.
- RIJSBERGEN, C. J. van. *Information Retrieval*. Glasgow, UK: Butterworths, 1979. Disponível em: <<http://www.dcs.gla.ac.uk/Keith/Preface.html>>.

- ROURE, D. D. A Brief History of the Semantic Grid. In: GOBLE, C.; KESSELMAN, C.; SURE, Y. (Ed.). *Semantic Grid: The Convergence of Technologies*. Schloss Dagstuhl, Germany: IBFI, 2005. (Dagstuhl Seminar Proceedings, 05271). ISSN 1862-4405. Disponível em: <<http://drops.dagstuhl.de/opus/volltexte/2005/398>>.
- ROURE, D. de; JENNINGS, N. R.; ; SHADBOLT, N. R. The Semantic Grid: A Future E-Science Infrastructure. In: *Grid Computing - Making the Global Infrastructure a Reality*. New York, NY, USA: John Wiley & Sons, 2003. p. 437–470.
- ROURE, D. de; JENNINGS, N. R.; SHADBOLT, N. R. The Semantic Grid: Present, Past, and Future. In: *Proceedings of IEEE*. Washington, DC, USA: IEEE Computer Society, 2005. v. 93, n. 3, p. 669–681.
- RUSSEL, M.; ALLEN, G.; GOODALE, T.; NABRZYSKI, J.; SEIDEL, E. Application Requirements for Resource Brokering in a Grid Environment. In: *Grid Resource Management. State of the Art and Future Trends*. Norwell, MA, USA: Kluwer Academic Publishers, 2003. p. 25–40.
- SCHOPF, J. M. Ten Actions When Grid Scheduling. In: *Grid Resource Management State of the Art and Future Trends*. Norwell, MA, USA: Kluwer Academic Publishers, 2003. p. 15–23.
- SIDDIQUI, M.; FAHRINGER, T.; HOFER, J.; TOMA, I. Grid Resource Ontologies and Asymmetric Resource-Correlation. In: INFORMATICS, G. S. of (Ed.). *2nd International Conference on Grid Service Engineering and Management (GSEM'05)*. Erfurt, Germany: Lecture Notes in Informatics, 2005.
- SILVA, F. S. C. da; FINGER, M.; MELO, A. C. V. de. *LÓGICA PARA COMPUTAÇÃO*. São Paulo, SP, Brasil: Thompson, 2006.
- SINTEK, M.; DECKER, S. Triple - an RDF query, inference, and transformation language. In: *International Semantic Web conference - ISWC 2002*. [S.l.]: Springer Verlag, 2002.
- SIRIN, E.; PARSIA, B.; GRAU, B. C.; KALYANPUR, A.; KATZ, Y. Pellet: A practical OWL-DL reasoner. *Web Semant.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 5, n. 2, p. 51–53, 2007. ISSN 1570-8268.
- SOMASUNDARAM, T. S.; R.A.BALACHANDAR; KANDASAMY, V.; BUYYA, R.; RAMAN, R.; N.MOHANRAM; S.VARUN. Semantic-based grid resource discovery and its integration with the grid service broker. *International Conference on Advanced Computing and Communications - ADCOM 2006*, Mangalore, India, p. 84–89, 2006.
- SOUSA, B. B. de; SILVA, F. J. da Silva e; TEIXEIRA, M. M.; FILHO, G. C. MagCat: An Agent-based Metadata Service for Data Grids. In: *AgentGrid2006: 4th International Workshop on Agent Based Grid Computing, IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'06)*. Singapore, Singapore: IEEE Computer Society, 2006.
- STATSOFT, I. *Electronic Statistics Textbook*. Tulsa, OK, EUA: StatSoft, 2007. Disponível em: <<http://www.statsoft.com/textbook/stathome.html>>.

SUN. *Jini Network Technology*. 2007. Disponível em: <<http://www.sun.com/software/jini/>>.

SWARTOUT, W.; TATE, A. Guest Editors' Introduction: Ontologies. *IEEE Intelligent Systems*, IEEE Computer Society, Los Alamitos, CA, USA, v. 14, n. 1, p. 18–19, 1999. ISSN 1094-7167.

TANGMUNARUNKIT, H.; DECKER, S.; KESSELMAN, C. Ontology-Based Resource Matching in the Grid - The Grid Meets the Semantic Web. In: *Second International Semantic Web Conference*. Whittleburg Hall, UK: Springer Berlin/Heidelberg, 2003. v. 2870, p. 706–721.

TAUBERER, J. *What is RDF*. July 2006. Disponível em: <<http://www.xml.com/pub/a/2001/01/24/rdf.html>>.

THOMAS, E.; PAN, J. Z.; SLEEMAN, D. ONTOSEARCH2: Searching Ontologies Semantically. In: *Proc. of OWL: Experiences and Directions (OWLED 2007)*. Murcia, Spain: International Association for Development of the Information Society (IADIS), 2007.

UNICODE, C. *Unicode Home Page*. may 2007. Disponível em: <<http://www.unicode.org/>>.

UNICORE. *The Unicore Forum*. 2007. Disponível em: <<http://www.unicore.org>>.

VIDAL, A. C. T.; BRAGA, J. José de R. P.; KON, F.; KOFUJI, S. T. Defining and exploring a grid system ontology (Short Paper). In: *MCG '06: Proceedings of the 4th International Workshop on Middleware for Grid Computing*. New York, NY, USA: ACM Press, 2006. p. 16. ISBN 1-59593-581-9.

VIDAL, A. C. T.; KOFUJI, S. T. Enhancing Grid Usage through Semantic Metadata (Short Paper). In: *ETNGRID '07: Proceedings of the international workshop on Emerging Technologies for Next-generation Grid*. Paris, France: GET/INT, 2007.

VIDAL, A. C. T.; SILVA, F. J. S.; ; KON, F.; KOFUJI, S. T. Semantics-Based Grid Resource Management. In: *MCG '07: Proceedings of the 6th International Workshop on Middleware for Grid Computing*. New York, NY, USA: ACM Press, 2007. ISBN 978-1-59593-944-9.

W3C, W. W. W. C. *Extensible Markup Language (XML) 1.0*. August 2006. Disponível em: <<http://www.w3.org/TR/owl-test/>>.

W3C, W. W. W. C. *RDF Vocabulary Description Language 1.0: RDF Schema*. april 2007. Disponível em: <<http://www.w3.org/TR/rdf-schema>>.

W3C, W. W. W. C. *Resource Description Framework (RDF)*. april 2007. Disponível em: <<http://www.w3.org/RDF/>>.

W3C, W. W. W. C. *SPARQL Query Language for RDF*. May 2007. Disponível em: <<http://www.w3.org/TR/rdf-sparql-query/>>.

W3C, W. W. W. C. *Web Ontology Language (OWL)*. april 2007. Disponível em: <<http://www.w3.org/2004/OWL>>.

WALSH, N. *A Technical Introduction to XML*. October 1998. Disponível em: <<http://www.xml.com/pub/a/98/10/guide0.html>>.

XING, W.; DIKAIKOS, M. D.; SAKELLARIOU, R. A Core Grid Ontology for the Semantic Grid. In: *CCGrid 2006*. Singapore, Singapore: IEEE Computer Society, 2006. p. 178–184.

XING, W.; DIKAIKOS, M. D.; SAKELLARIOU, R.; ORLANDO, S.; LAFORENZA, D. Design and Development of a Core Grid Ontology. In: *CoreGRID Integration Workshop 2005*. Pisa, Italy: S. Gorlatch and M. Danelutto (Eds.), 2005. p. 21–31.

ZHANG, Y.; MANDAL, A.; CASANOVA, H.; CHIEN, A. A.; KEE, Y.-S.; KENNEDY, K.; KOELBEL, C. Scalable Grid Application Scheduling via Decoupled Resource Selection and Scheduling. In: *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)*. Singapore: IEEE Computer Society, 2006. v. 1, p. 568 – 575.