Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Programa de Pós Graduação em Engenharia de Teleinformática

Framework Semântico Baseado em Ontologia para Descoberta e Seleção de Recursos

Francisca Aparecida Prado Pinto

Fortaleza – Ceará – Fevereiro 2010

Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Programa de Pos-Graduação em Engenharia de Teleinformática

Framework Semântico Baseado em Ontologia para Descoberta e Seleção de Recursos

Autor
Francisca Aparecida Prado Pinto

Orientador
Prof. Dr. Giovanni Cordeiro Barroso

Co-Orientador
Prof. Dr. Antônio de Barros Serra

Dissertação de Mestrado apresentada à Coordenação do Curso de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de ***Mestre em Engenharia de Teleinformática****.*

Fortaleza – Ceará – Fevereiro 2010

Resumo

Nesta dissertação é proposto um *framework* semântico para descoberta de recursos em ambiente de grade, fornecendo uma abordagem flexível e extensível, em que a correspondência (*matching*) de recursos é baseada em ontologia. Esse *framework* semântico consiste em quatro camadas: Fábrica, *Middleware*, Conhecimento e Aplicação. Na camada de Conhecimento, há um Repositório Semântico, que provê a descoberta dos recursos na grade. Esse repositório é constituído pela base de conhecimento e pela ontologia, composta, por sua vez, com os recursos da grade. Na camada *Middleware* utiliza-se o Sistema de Descoberta e Monitoramento da ferramenta Globus *Toolkit*. O sistema de descoberta coleta as informações dos recursos da grade juntamente com um provedor de informação, o qual é usado para coletar automaticamente as informações dos mesmos. Na camada de Aplicação foi desenvolvida uma interface em *Web*, utilizando uma linguagem de busca para prover aos usuários a consulta dos recursos da grade e possibilitando a seleção dos mesmos. Desenvolveu-se uma modelagem do *framework* em (Redes de Petri Coloridas) *Coloured Petri Nets*. A partir desse modelo, tornou-se possível fazer a análise e validação do mesmo. O *framework* foi, então, implementado no Laboratório de Instrumentação e Computação dos Departamentos de Física e Teleinformática da Universidade Federal do Ceará (UFC), sendo realizados estudos de caso, obtendo-se resultado satisfatório.

Palavras-chave: Grade Computacional; Ontologia; Semântica; Redes de Petri.

Abstract

This dissertation proposes a semantic framework for resource discovery in grid environment, providing a flexible and extensible, in that the matching resources is based on ontology. The proposed framework is comprised of four layers: Fabric, Middleware, Knowledge and Application. The Knowledge layer has a Semantic Repository (OWL) in order to provide semantic discovery of resources in a grid. This repository consists of the knowledge base and the template, composed, in turn, with the resources of the grid. In the Middleware layer uses the System Discovery and Monitoring Globus Toolkit. The System Discovery and Monitoring collects the information resources of the grid along with a provider of information, which is used to automatically collect information from them. In the application layer has developed a Web interface using a search language to provide users with the consulting resources of the grid and allowing the selection of them. Developed a modeling framework in Colored Petri Nets. From this model, it became possible to analyze and validate the same. The framework was then implemented in the Laboratory of Instrumentation and Computer Departments of Physics and Teleinformática Federal University of Ceará, and case studies, obtaining a satisfactory result.

Keywords: Grid Computing; Ontology; Semantic; Petri Nets.

Dedico este trabalho, especialmente,
ao meu pai Januário (Jeová) (*in memorian*),
o meu eterno amor. À minha mãe, Maria Prado,
por toda sua atenção, paciência e por ter estado
sempre do meu lado, com todo o seu amor e carinho.

Agradecimentos

Em primeiro lugar, eu gostaria de agradecer aos meus pais, Jeová (*in memorian*) e Maria Prado, por terem incentivado minha vocação para o estudo científico, desde quando eu ainda era criança até hoje, proporcionando-me os meios para o exercício da minha criatividade. Todas as minhas conquistas são conquistas de vocês dois também.

À Secretária de Educação do Ceará, pela oportunidade da realização deste curso.

Ao Professor Doutor Giovanni Cordeiro Barroso, orientador científico desta dissertação, agradeço o compromisso assumido, o empenho que colocou neste trabalho, os níveis de exigência dos desafios que me lançou e os suportes, formais e informais, que disponibilizou.

Ao Professor Doutor Antônio de Barros Serra, co-orientador desta dissertação, por sua ajuda e interesse, avaliação de lâminas e sábias ideias.

Aos Professores Doutores Cidcley Teixeira de Souza e Mário Fiallos Aguilar, pela participação da banca de defesa de dissertação.

A todos os professores da Pós-graduação de Engenharia de Teleinformática, pelos conhecimentos transmitidos.

Ao bolsista de Iniciação Científica, Vando Aderson, cujo auxílio foi muito importante na implementação desta dissertação.

A todos os colegas da Pós-graduação, em especial Daniela Cedro, Carla Katarina, Isabel, Lorena, Henrique Jorge, Carlos Henrique (Caíque), Eveline Sacramento e Fernanda Ligia, pela amizade e companheirismo sempre presentes.

Aos meus grandes amigos Ana Maria, Auri, Diana, Erotildes, Eugeniano, Giovana, Glória, Honório Brito, Joana Darc, Lucila, Luis José, Mary, Maria da Paz, Paulo Henrique e Sávio, pela amizade incondicional e momentos de lazer agradáveis.

Às minhas irmãs, Régia e Cláudia, aos meus irmãos Ricardo, Irailton (*in memorian*) e Ronaldo, bem como aos meus sobrinhos Marília, Ítalo e Mariana, por todo o apoio e carinho inesgotáveis. Um especial agradecimento ao meu cunhado, Beethoven Gondim, por ter revisado esta dissertação e pelas suas sugestões metodológicas.

Àqueles que por ventura não foram citados, mas que, direta ou indiretamente, contribuíram para realização deste trabalho, meus muito cordiais agradecimentos.

Por fim, agradeço a Deus, pois somente Ele me concede forças para enfrentar as dificuldades do dia-a-dia com a convicção de que irei vencer.

"É preciso sonhar, mas com a condição de crer em nosso sonho, de observar com atenção a vida real, de confrontar a observação com nosso sonho, de realizar escrupulosamente nossas fantasias. Sonhos, acredite neles."

Vladimir Ilitch Lenin

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AIFB	Institut für Angewandtel Informatik und Formale Beschreibungsverfahren
ASP	Active Server Pages
BoT	Bag-of-Tasks
CAS	Community Authorization Service
ClassAds	Classified Advertisements
DAML	DARPA Agent Markup Language
DDSA	Dynamic Distributed Services Architecture
DIG	Description Logics Implementors Group Interface
DNS	Domain Name System
DOM	Document Object Model
FLogic	Frame Logic
FTP	File Transfer Protocol
GGF	Global Grid Forum
GLUE	Grid Laboratory Uniform Environment
GRAM	Globus Resource Allocation Manager
GSI	Grid Security Infrastructure
GT4	Globus Toolkit na sua versão 4
ICMP	Internet Control Message Protocol
IP	Internet Protocol
JSP	JavaServer Pages
KIF	Knowledge Interchange Format
MDS	Monitoring and Discovery Service
MonALISA	Monitoring Agents using a Large Integrated Services Architecture
nRQL	new RACER Query Language

OASIS	Organization for the Advancement of Structured Information Standards
OCML	Operational Conceptual Modeling Language
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture
OGSI	Open Grid Services Infrastructure
OIL	Ontology Inference Layer
OSPF	Open Shortest Path First
OV	Organização Virtual
OWL	Web Ontology Language
OWL-QL	OWL Query Language
P2P	Peer-to-Peer
PERL	Practical Extraction And Report Language
RacerPro	Renamed Abox and Concept Expression Reasoner
RAM	Random Access Memory
RDF	Resource Description Framework
RDF(S)	RDF Schema
RP	Rede de Petri
RPC	Rede de Petri Colorida
RRDTool	Round Robin Database
RSVP	Resource ReserVation Protocol
SAML	Security Assertion Markup language
SGBD	Sistema de Gerenciamento de Banco de Dados
SO	Sistema Operacional
SPARQL	Semantic Protocol and RDF Query Language
SQWRL	Semantic Query-Enhanced Web Rule
SWRL	Semantic Web Rule Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UFC	Universidade Federal do Ceará
URI	Uniform Resource Identifier
URL	Uniform Resource Location
US CMS	U.S. Compact Muon Solenoid

W3C	World Wide Web Consortium
WS	Web Semântica
XDR	eXternal Data Representation
XML	eXtensible Markup Language

Capítulo 1

Introdução

1.1 Motivação

A tecnologia de computação em grade surgiu para tratar todos os recursos disponíveis como sendo parte de uma mesma infra-estrutura computacional, possibilitando a execução de tarefas muitas vezes complexas e que requerem alto poder computacional. Grades computacionais caracterizam uma infraestrutura que envolve o uso colaborativo e integrado de computadores, redes, bases de dados e instrumentos científicos pertencentes a diferentes organizações, visando à resolução de problemas pelo compartilhamento de recursos em organizações multi-institucionais (FOSTER; KELSSELMAN, 2004).

Para o sucesso de compartilhamento dos recursos entre as diversas organizações, é importante um mecanismo de descoberta eficiente, a fim de localizar os recursos desejados disponíveis no sistema de grade no momento da sua solicitação. No entanto, devido à natureza dinâmica, heterogênea e distribuída do ambiente de grade, a descoberta de recursos torna-se um trabalho desafiador.

Em grade computacional, os recursos são potencialmente em larga escala em relação à quantidade e variedade. Os recursos individuais não são controlados centralmente, e eles podem entrar e sair dos sistemas de grade a qualquer momento.

As grades são usadas para unir vários recursos computacionais e de dados geograficamente distribuídos, bem como para entregar esses recursos a comunidades de usuários heterogêneos. Esses recursos podem pertencer a diferentes instituições, ter diferentes políticas de uso e podem ser descritos de maneira diferente,

dependendo das Organizações Virtuais (OVs). Nesse ambiente geograficamente distribuído, heterogêneo e dinâmico, o mesmo tipo de recurso pode ser descrito de forma diferente por consumidores e provedores de recursos.

Na maior parte dos sistemas de construção de grade, a descoberta de recursos é baseada em simetria, na qual o processo de seleção dos recursos é feito através correspondência de recursos (*matching resources*)[1] (ZHANG; SONG, 2004) (VIDAL et al., 2009). Por meio desse processo, os provedores e os consumidores têm que concordar com o mesmo nome dos atributos de recursos e valores. Qualquer alteração no nome do atributo deve ser levada ao conhecimento do consumidor. Caso contrário, os consumidores serão incapazes de encontrar os recursos de forma eficiente. A correspondência de recursos exata entre fornecedor e consumidor torna os sistemas inflexíveis e difíceis de se expandir com novas características e conceitos. Dentre esses sistemas baseados em simetria, destaca-se o *Monitoring and Discovery System* (MDS) do conjunto de ferramentas do Globus (SOMASUNDARAM et al., 2006) (SAID; KOJIMA, 2009), definindo e implementando um mecanismo para descoberta e monitoramento de recursos em ambientes em grade.

Ao contrário da abordagem simétrica, propor-se-á uma semântica, onde o processo de busca de recursos é mais flexível e extensível, por se basear em ontologia, oferecendo mais resultados em suas consultas, já que a correspondência de recursos tem conceitos similares, e não apenas restritos ou limitados às palavras-chaves de correspondência.

1.2 Objetivo

Neste trabalho é proposto um *framework* semântico para descoberta de recursos em ambiente de grade, a partir de uma abordagem flexível e extensível para a realização da seleção de recursos, utilizando correspondência dos mesmos, baseada em ontologia.

Esse *framework* semântico está organizado em quatro camadas: Fábrica, *Middleware*, Conhecimento e Aplicação. Na camada de Conhecimento, há um Repositório Semântico, que provê a descoberta dos recursos na grade. Esse repositório é constituído pela base de conhecimento e ontologia. Para criação desta última, utilizam-se as informações dos recursos no ambiente de grade, por meio

[1]Neste trabalho, o termo *matching resources* será substituído pelo vocábulo 'correspondência de recursos'.

de uma ferramenta de edição. Na camada *Middleware*, faz-se uso do Sistema de Descoberta e Monitoramento da ferramenta Globus *Toolkit* (versão 4) (GT4), ou seja, o MDS4. Este último coleta as informações dos recursos da grade juntamente com um provedor de informação, o qual é usado para coletar automaticamente as informações dos mesmos. Para criar dinamicamente a base de conhecimento dos recursos da grade, utiliza-se um *framework*, o qual permite manipular a ontologia. Através de um mecanismo de inferência, interage-se com a base de conhecimento na descoberta de mais resultados desses recursos. Na camada de Aplicação foi desenvolvida uma interface em *Web portlet*,[2] fazendo-se uso de uma linguagem de busca dinâmica para prover aos usuários a consulta dos recursos da grade, possibilitando a seleção destes últimos de forma semântica.

O foco deste trabalho está na definição da arquitetura de grade semântica que viabiliza a correspondência entre os recursos, não abordando a execução e o escalonamento das aplicações, mas sim a descoberta e seleção de recursos. Em grade semântica, os recursos de computação e serviços são descritos, utilizando uma abordagem semântica, de modo que esses recursos são facilmente descobertos e ligados automaticamente para atender os requisitos das aplicações (ROURE; JENNINGS; SHADBOLT, 2003).

1.3 Estrutura da Dissertação

Quanto à estrutura desta dissertação, o trabalho está organizado da seguinte forma: No Capítulo 2, trata-se da computação em grade, da *Web* Semântica, da ontologia e das Redes de Petri Coloridas, fornecendo subsídios para a base teórica e compreensão desses ambientes e dessa ferramenta, bem como do trabalho descrito; No Capítulo 3, mostram-se alguns traballhos correlatos ao tema abordado nesta dissertação; No Capítulo 4, desenvolve-se uma modelagem do *framework* semântico para descoberta de recursos em ambiente de grade, sendo utilizadas as Redes de Petri Coloridas (ferramenta CPNTools), para fazer a análise e validação do *framework*; No Capítulo 5, trata-se da implementação do *framework* semântico, bem como do estudo de caso, obtendo-se resultados; No Capítulo 6 são apresentadas as conclusões e sugestões para trabalhos futuros.

[2] Componente visual independente, que pode ser utilizado para disponibilizar informações dentro de uma página *Web*.

Capítulo 2

Fundamentação Teórica: Grades Computacionais, Web Semântica, Ontologia e Redes de Petri

2.1 Grades Computacionais

Na última década, tem sido observada uma substancial modificação na maneira como os recursos e serviços vêm sendo utilizados na computação. Em décadas passadas, com a chegada das redes de comunicação e computadores com custo mais acessíveis, possibilitou-se a evolução da computação centralizada para a distribuída.

Como consequência, grupos de pesquisa têm implementado *middlewares*, bibliotecas e ferramentas que permitem o uso cooperativo de recursos geograficamente distribuídos e unificados, a fim de atuarem como uma poderosa plataforma de computação para a execução de um grande variedade de aplicações paralelas e distribuídas. Essa abordagem para a computação tem sido conhecida por diferentes nomes, tais como *metacomputing*, *scalable computing*, *global computing*, *grid computing* ou (computação em grade) (BAKER; BUYYA; LAFORENZA, 2000).

2.1.1 Definição de Grades

Segundo (FOSTER; KELSSELMAN; TUECKE, 2001), a expressão 'computação em grade' foi estabelecida em meados da década de 1990 para denotar uma proposta de infraestrutura de computação distribuída para a engenharia e ciências avançadas. Desde então, essa área vem apresentando novos ambientes de grades computacionais,

desenvolvidos em muitas partes do mundo e buscando alto poder computacional e alta disponibilidade de recursos e serviços.

Grade computacional é definida em (FOSTER; KELSSELMAN, 2004) como *um sistema que coordena recursos distribuídos, usando interfaces e protocolos padronizados, abertos e de propósito geral para entregar serviços de qualidade não trivial.* Os elementos principais dessa definição são assim analisados:

- Coordenação de recursos distribuídos:

  Uma grade computacional integra e coordena recursos e usuários que estão sob diferentes domínios e aborda temas de segurança, políticas, faturamento, afiliação, entre outros;

- Uso de interfaces e protocolos padronizados, abertos e de propósito geral:

  Uma grade computacional é construída a partir de protocolos e interfaces padronizados e abertos de propósitos múltiplos, relacionados com autenticação, autorização, descoberta de recursos e acesso a recursos;

- Entrega de serviços de qualidade não trivial:

  Uma grade computacional pode ser constituída de recursos para serem usados de forma coordenada, a fim de prover serviços com qualidades variadas. Uma grade computacional pode coalocar vários tipos de recursos que atendam a demandas complexas do usuário, de maneira que a utilidade combinada do sistema é significativamente maior que a soma das partes.

2.1.2 Organização Virtual

As grades computacionais são caracterizadas por uma infraestrutura que propicia o uso colaborativo de recursos computacionais pertencentes a uma Organização Virtual (OV), composta de diferentes entidades institucionais. Uma OV é composta por um conjunto de indivíduos e recursos multi-institucionais, regidos por regras comuns de compartilhamento de recursos (FOSTER; KELSSELMAN; TUECKE, 2001).

Diversas organizações estão utilizando a computação em grade para resolver problemas complexos, de alto custo computacional ou que demandam muita interação em atividades, tais como a pesquisa científica colaborativa, a descoberta de novos medicamentos, a análise de risco financeiro e a definição de produtos.

Ao superar as restrições de integração de dados e de computação, a tecnologia de grades computacionais permite a interação entre organizações, além de reduzir custos pela otimização da utilização dos recursos. Utilizando grade computacional, a colaboração entre entidades permite a resolução de problemas de computação e dados intensivos que estão além da capacidade individual de uma única organização.

Na Figura 2.1 são apresentadas três organizações, que estão distribuídas geograficamente e compartilham os seus recursos computacionais, formando duas OVs. Podem ser criadas diferentes OVs para determinados recursos computacionais, ou seja, podem ser feitos diferentes agrupamentos, representando as possíveis OVs. A existência da OV é limitada a um determinado período, correspondente ao tempo necessário à satisfação do seu propósito.

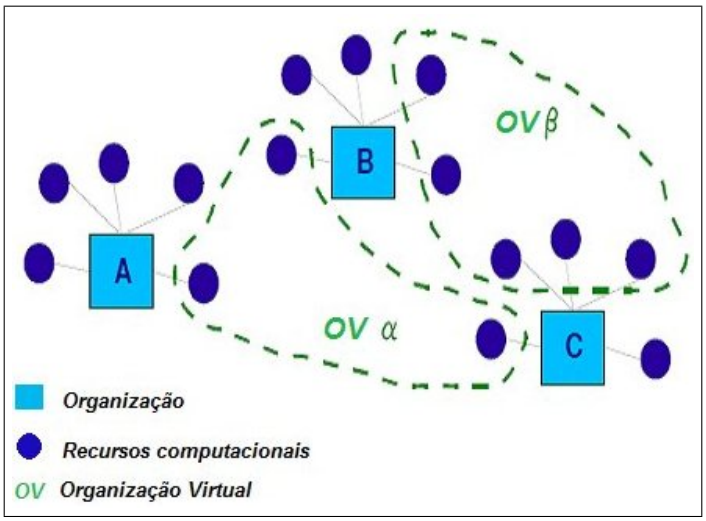


Figura 2.1: Exemplo de Organização Virtual (OV)

2.1.3 Arquitetura de Grade Computacional

A arquitetura das grades define os componentes fundamentais da tecnologia e especifica as funções e as interações entre os componentes. Essa arquitetura é apresentada por meio de uma estrutura em camadas, conforme mostrado na Figura 2.2 (FOSTER; KELSSELMAN, 2004). O foco principal desse ambiente é a interoperabilidade entre provedores de recursos e usuários, visando a estabelecer relacionamentos de compartilhamento. Essa interoperabilidade, por sua vez, necessita de protocolos comuns em cada camada do modelo de arquitetura, levando à definição dos mesmos na grade, como mostrado na referida figura.

A arquitetura de grades computacionais é representada na forma de uma ampulheta, como mostrado na Figura 2.2, que tem no gargalo (Serviços coletivos,

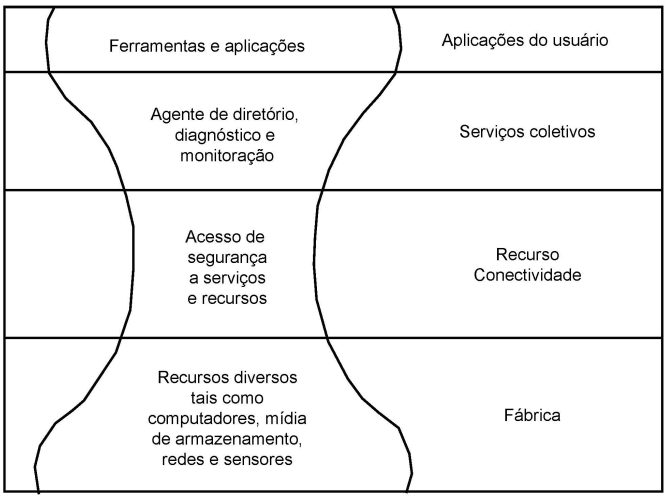


Figura 2.2: Modelo de arquitetura da computação em grade (FOSTER; KELSSELMAN, 2004)

Recursos e Conectividade) um número reduzido de protocolos e serviços. Acima do gargalo (Aplicações do usuário), existe um conjunto de comportamentos de alto nível, podendo ser mapeado para muitas e diferentes tecnologias na parte de baixo do gargalo (Fábrica). As camadas da arquitetura são descritas a seguir.

- **Camada de Fábrica**

  A camada de fábrica compreende os recursos para os quais os acessos compartilhados são mediados pelos protocolos da grade computacional. Exemplos dos elementos dessa camada são os diversos recursos que podem ser físicos ou lógicos, tais como sistemas de armazenamento, recursos de rede, sensores, computadores e *clusters*. Os componentes de fábrica executam as operações locais que são específicas dos recursos, como resultado das solicitações de compartilhamento das operações de mais alto nível.

  Os desafios nessa camada estão relacionados com a implementação de mecanismos internos nesses recursos que permitam, por um lado, a descoberta de sua estrutura, estado e capacidades, e, por outro, o controle da qualidade de serviço, como mostram os exemplos a seguir:

  – Recursos computacionais: é necessária a definição de novos mecanismos para iniciar, monitorar e controlar a execução de processos. Funções internas são necessárias para determinar as características de *hardware* e *software*, assim como informações de estado relevantes, tais como carga

de trabalho corrente e estado de filas, no caso de recursos gerenciados por escalonadores.

- Recursos de armazenamento: é necessário o desenvolvimento de novos mecanismos para enviar e recuperar arquivos. Geralmente são mecanismos para leitura e escrita em arquivos e execução remota de dados selecionados.
- Recursos de rede: é importante o desenvolvimento de mecanismos de gerenciamento que forneçam controle sobre os recursos alocados para as transferências na rede. Funções internas devem ser providas para determinar as características e a carga na rede.

- **Camada de Conectividade**

  A camada de conectividade define os protocolos de comunicação e de autenticação para transações específicas de grades computacionais. Os protocolos de comunicação permitem a troca de dados entre os recursos da camada de fábrica, e os protocolos de autenticação fornecem mecanismos de segurança e criptografia para verificar a identidade de usuários e recursos.

  Os protocolos de comunicação de grades computacionais incluem transporte, roteamento e nomeação. Tais protocolos são atualmente projetados a partir da pilha de protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP), tais como o IP, o *Internet Control Message Protocol* (ICMP), o TCP, o *User Datagram Protocol* (UDP), o *Domain Name System* (DNS) , o *Open Shortest Path First* (OSPF) e o *Resource ReserVation Protocol* (RSVP). Desafios futuros devem abordar o projeto de novos protocolos de comunicação específicos para grades computacionais que atendam a demandas da dinâmica de redes específicas, por exemplo, redes ópticas ou redes sem fio.

  Os protocolos de segurança também estão sendo baseados em padrões dentro do contexto da suíte de protocolos *Internet*. Os aspectos de segurança para grades computacionais que são mais importantes são (NASSIF; NOGUEIRA, 2007):

  - Autenticação única: um usuário necessita autenticar somente uma vez, dispensando autenticações repetidas para fazer acessos a recursos ou domínios administrativos diferentes.

- Delegação: um usuário pode delegar a execução de um programa para os recursos onde ele tem autorização de uso. O programa pode opcionalmente, ser capaz de delegar seus direitos para outro programa.
- Integração com soluções de segurança local: as soluções de grades computacionais devem interoperar com soluções de segurança local.
- Relacionamento de confiança baseado no usuário: caso o usuário tenha permissão para executar programas nos recursos A e B, não deve ser obrigatório que A e B interajam.

- **Camada de Recursos**

  A camada de recursos é construída sobre os protocolos de comunicação e de autenticação da camada de conectividade e seu papel é definir protocolos para negociação, inicialização, monitoração, controle, contabilização e faturamento de operações compartilhadas em recursos individuais. As implementações desses protocolos da camada de recursos são baseadas nas funções da camada de fábrica para acessar e controlar os recursos locais.

  Os protocolos da camada de recursos concentram-se nos recursos individuais e ignoram o estado global. São definidas duas classes principais de protocolos da camada de recursos:

  - a primeira consiste nos protocolos de informação, que são usados para obter informações sobre a estrutura e o estado de um recurso, como sua configuração, carga corrente e política de uso e custo;
  - a segunda classe consiste nos protocolos de gerenciamento, que são usados para negociar acesso a recursos compartilhados, especificando, por exemplo, os requisitos do recurso e as operações a serem executadas, tais como criação de processo e acesso a dados. No projeto desses protocolos, é preciso ter um ponto de aplicação de políticas, assegurando que as operações solicitadas sejam consistentes com as políticas do recurso a ser compartilhado.

- **Camada de Serviços Coletivos**

  A camada de serviços coletivos trata problemas de descoberta, seleção e alocação de recursos, segurança, política e contabilização. Ela contém protocolos e serviços que não são associados a um recurso específico e

pode implementar soluções para uma coleção de recursos. Pelo fato de os componentes coletivos serem construídos acima da camada de recurso (o gargalo do modelo ampulheta) e, portanto, por representarem uma camada mais larga do modelo, a camada de serviços coletivos pode implementar uma grande variedade de serviços sem adicionar novos requisitos aos recursos que estão sendo compartilhados. Como exemplo de serviços temos:

- os serviços de diretório, que permitem aos participantes de uma organização virtual descobrir a existência de recursos compartilhados;
- os serviços de coalocação, escalonamento e corretagem (*brokering*)[1], os quais permitem aos participantes de uma organização virtual requisitar a alocação de um ou mais recursos para uma proposta específica, bem como escalonar as tarefas nos recursos apropriados;
- os serviços de monitoração e diagnóstico, que monitoram recursos para detectar falhas, intrusões e sobrecargas;
- os serviços de réplica de dados, que gerenciam o armazenamento de recursos para maximizar o desempenho no acesso a dados.

- **Camada de Aplicações**

  A camada de aplicações na arquitetura de grades computacionais compreende as aplicações do usuário que operam dentro de um ambiente de uma OV. As aplicações são construídas nos termos dos serviços oferecidos por cada camada da arquitetura. As ferramentas da camada de aplicação devem oferecer mecanismos para que os usuários consigam escrever e rodar suas aplicações nos ambientes de grades computacionais. Isso porque os usuários não dispõem de habilidade, tempo e motivação para aprender os detalhes dos serviços de informação da grade e fazer desses serviços a base para a tomada de decisão sobre a seleção de recurso.

2.1.4 Tipos de Grades

Existem várias aplicações nas quais as grades podem oferecer grandes benefícios. De acordo com (SKILLICORN, 2002), existem quatro tipos de grades: grades computacionais, grades de acesso, grades de dados e grades datacêntricas. Cada um dos tipos oferece algumas vantagens, como será descrito a seguir:

[1] *Broker* é responsável por selecionar os melhores recursos para execução de cada tarefa.

- **As grades computacionais**: por meio de uma única interface, usuários utilizam o sistema como um grande computador paralelo, executando suas requisições com grande aumento de desempenho;

- **As grades de acesso**: oferece-se um ambiente no qual usuários de diferentes organizações interagem como se estivessem na mesma plataforma de *hardware*. Esse tipo de grade não possui aceleração dos cálculos como objetivo primário;

- **As grades de dados**: permite-se que grandes conjuntos de dados sejam armazenados em repositórios e manipulados com a mesma facilidade com que se manipulam pequenos arquivos atualmente. O objetivo principal desse tipo de grade é a disponibilidade de dados;

- **As grades datacêntricas**: diferentemente da grade de dados, ao invés de se moverem estes últimos para a computação, move-se a computação para os dados, nos casos em que os dados devem permanecer imóveis. Uma aplicação das grades datacêntricas é a mineração de dados distribuídos (*distributed data mining*).

Esses vários tipos de grades tornam mais difícil a criação de padrões que possam ser utilizados para todos eles, de modo a não prejudicar o desempenho dos seus funcionamentos. Existem atualmente muitos projetos de pesquisa, desenvolvimento de plataformas de suporte e fóruns que tratam dos problemas da computação em grade. São apresentados, a seguir, alguns exemplos de fóruns específicos e abertos sobre grades computacionais, tais como: O *Global Grid Forum* (GGF)[2], que agora forma o *Open Grid Forum* (OGF), constituído por uma comunidade de desenvolvedores, vendedores e usuários para definir padrões globais para a comunidade de grade computacional. O GGF engloba cerca de quatrocentas organizações atuantes em aproximadamente cinquenta países; e o *Organization for the Advancement of Structured Information Standards* (OASIS)[3], que é um consórcio sem fins lucrativos para o desenvolvimento, convergência e adoção de padrões negócios eletrônicos. O OASIS tem representações de mais de seiscentas organizações em aproximadamente cem países.

[2]http://www.gridforum.org./
[3]http://www.oasis-open.org/home/index.php

2.1.5 Padronização

No sentido de se estabelecer um padrão para a construção de grades, o GGF especificou a *Open Grid Services Architecture* (OGSA) e a *Open Grid Services Infrastructure* (OGSI) (GLOBUS, 2009). A arquitetura OGSA fornece um padrão aberto que serve de referência para o desenvolvimento de aplicações para grades computacionais. Ela é baseada nos padrões para os Serviços *Web* (*Web Services*) e considera um serviço na grade como um Serviço *Web* com algumas particularidades. Seu objetivo é padronizar praticamente todos os serviços e funcionalidades que podem ser encontrados em grades computacionais, como, por exemplo, serviços de submissão de tarefas, serviços de informação, serviços de transferência de dados, autenticação, autorização, entre outros. Já a especificação OGSI estabelece as interfaces e protocolos necessários aos serviços e define como construir, gerenciar e expandir um serviço. Assim, a OGSI complementa a OGSA, fornecendo a interoperabilidade necessária às grades computacionais.

A seguir, é feita uma breve descrição dos principais serviços utilizados para a construção de grades computacionais. Eles incluem serviços de segurança, monitoramento, gerenciamento de recursos, gerenciamento de dados e escalonamento.

- **Segurança**

  É um componente importante numa grade computacional. Para um usuário executar uma aplicação na grade, deve-se garantir que outras aplicações ou usuários não terão acesso aos dados de sua aplicação. Da mesma maneira, um recurso oferecido à aplicação do usuário não deve sofrer interferências em seus dados privados. Os requisitos mínimos para disponibilizar segurança em uma grade são: autenticação, autorização, confiabilidade e integridade dos dados e gerenciamento de chaves e certificação.

- **Monitoramento**

  É utilizado para manter informações consistentes sobre a disponibilidade de recursos na grade e sobre a situação das aplicações. O serviço de monitoramento é responsável por publicar e acessar dados dos sistemas e das aplicações, mantendo esses dados atualizados para que a gerência de recursos tenha conhecimento do estado geral da grade.

A informação de um recurso pode ser classificada em uma das duas cetegorias: estática ou dinâmica. A informação estática é alterada com pouca frequência, por exemplo, o nome do sistema operacional. Já a informação dinâmica possui uma alta frequência de modificação, sendo atribuída em função do uso do recurso, por exemplo, a carga do processador.

O monitoramento do estado e da utilização de recursos pode ser realizado não apenas para auxiliar as decisões de um escalonador de aplicações, mas também para calcular o quanto um usuário está utilizando a estrutura como um todo. Em ambientes que disponibilizam recursos não dedicados à grade, mas que podem se tornar ociosos, um sistema de monitoramento ajuda a garantir a prioridade de utilização ao dono do recurso. A seguir são apresentados os sistemas de monitoramento *Monitoring Agents using a Large Integrated Services Architecture* (MonALISA), *Ganglia* e *Hawkeye*:

- **MonALISA**[4] é um projeto desenvolvido pela Caltech e seus parceiros, com suporte do laboratório de pesquisa *U.S. Compact Muon Solenoid* (US CMS). O *framework* é baseado na *Dynamic Distributed Services Architecture* (DDSA) e está apto a prover o completo monitoramento, controle e serviços de otimização globais para sistemas complexos.
- ***Ganglia***[5] foi desenvolvido para ser executado em ambientes de computação de alto desempenho, como *clusters* e grades. O sistema é um *software* de código aberto, que teve sua origem na Universidade da Califórnia, *Campus* de Berkeley. É baseado num protocolo que faz uso de uma técnica de difusão *multicast*[6] para comunicação entre os nós do ambiente monitorado, utilizando uma estrutura de árvores para representar as conexões ponto a ponto da rede de alto desempenho.

  O Ganglia utiliza algumas tecnologias de grande representatividade, como o *eXtensible Markup Language* (XML), para fazer a representação dos dados, o *eXternal Data Representation* (XDR), para compactá-los, viabilizando o transporte em rede, e o *Round Robin Database* (RRDtool)[7], para armazenamento e visualização dos dados. Ele é um sistema robusto

[4] http://monalisa.caltech.edu/monalisa.htm
[5] http://ganglia.sourceforge.net/
[6] *Multicasting* é um método ou técnica de transmissão de um pacote de dados para múltiplos destinos ao mesmo tempo.
[7] http://oss.oetiker.ch/rrdtool/

e, devido à característica heterogênea das grades, portável para uma grande gama de sistemas operacionais, além de diversas tecnologias de processadores.

- ***Hawkeye***[8] utiliza as tecnologias já presentes no *Condor*[9] e *Classified Advertisements* (ClassAds)[10] (SMITH; BAKER, 2008) para prover mecanismos na coleta, armazenagem e utilização das informações sobre os computadores. Um sistema *Hawkeye* pode ser utilizado para monitorar vários atributos de uma coleção de sistemas. O mecanismo de monitoramento pode também ser utilizado para fazer o gerenciamento dos sistemas.

- **Gerenciamento de Recursos**

  Esse sistema de gerenciamento associa os recursos disponíveis às aplicações, atuando como uma interface abstrata aos mesmos, provendo facilidades para alocar o recurso, informando o usuário/ambiente e oferecendo meio de cancelar, interromper a execução e outros tipos de gerenciamento.

- **Gerenciamento de Dados**

  Para a construção de um modelo de gerenciamento de dados em grades, deve-se levar em consideração o tipo de grade em que o modelo será inserido. Por exemplo, numa grade de dados, que objetiva integrar dados distribuídos, as aplicações/sistema não requerem respostas em tempos próximos ao imediato. Assim, o sistema de gerenciamento de dados pode ser composto apenas de serviços essenciais, tais como de transferência, descobrimento, manipulação, criação e gerenciamento de cópias de dados.

  O gerenciamento de dados numa grade computacional apresenta diferentes tipos e formatos de dados que podem existir e precisam ser interoperados de forma comum no ambiente. Entre os tipos existentes, há arquivos convencionais, base de dados relacionais, de objetos e XML e dados virtualizados.

  O transporte de dados é um aspecto importante no gerenciamento destes últimos, o qual não envolve apenas a movimentação dos dados sobre os

[8] http://www.cs.wisc.edu/condor/hawkeye/
[9] http://www.cs.wisc.edu/condor/index.html
[10] http://www.cs.wisc.edu/condor/classad/index.html

recursos, mas também o aspecto de acesso àqueles, tais como de segurança, controle de acesso e gerenciamento da transferência dos dados. Protocolos que conduzem essas questões incluem o *File Transfer Protocol* (FTP) e o GridFTP.

- **Escalonamento**

  Uma das principais áreas de pesquisa em grades computacionais é o escalonamento, que corresponde, em termos gerais, à alocação de tarefas a recursos. No caso de tarefas, há uma grande diversidade: existem desde tarefas diretamente ligadas a processamento, que podem ser processos, *threads*, ou programas de até, por exemplo, armazenamento de dados. Em todos esses casos, procura-se uma melhor alocação das tarefas aos recursos, de forma a otimizar alguma função objetivo previamente determinada. Como mostrado na Figura 2.3, o escalonamento consiste, basicamente, em três fases: a descoberta de recursos, a seleção de recursos e a execução da aplicação (SCHOPF, 2004)(JAISWAL, 2007).

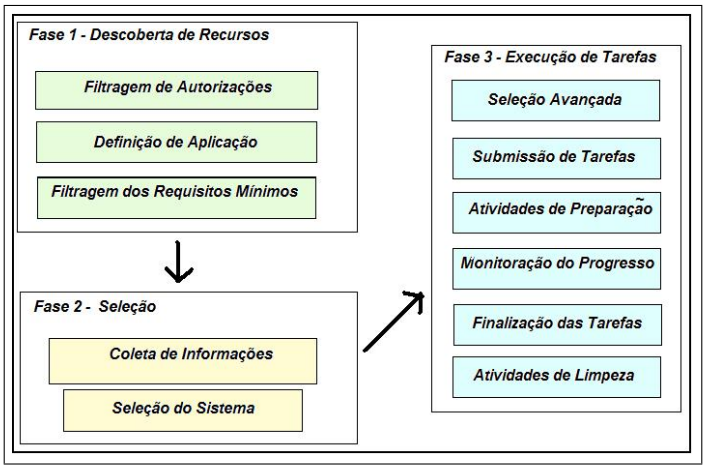


Figura 2.3: Arquitetura básica de um gerenciador de recursos (SCHOPF, 2004)(JAISWAL, 2007)

  – **A descoberta de recursos**: responsável por selecionar um conjunto de recursos para ser investigado mais profundamente na seleção dos mesmos. Ela se divide em três etapas: a autorização, a definição dos requisitos da aplicação e a filtragem dos requisitos mínimos. Na primeira etapa, apenas os recursos para os quais o usuário submissor da aplicação tem acesso são considerados. Na segunda, o usuário deve definir os requisitos básicos da aplicação tais como sistema operacional e arquitetura-alvo, além de

quantidades de recursos como, por exemplo, de memória e processamento a serem utilizados.

- **A seleção de recursos**: realizada em duas etapas: a da coleta de informações dinâmicas, detalhadas a respeito dos recursos, e a da seleção do melhor recurso para executar a aplicação-alvo. Na fase de coleta, a fonte das informações dinâmicas comumente consiste no Serviço de Monitoração, mas é possível consultar fontes de mais baixo nível como o próprio escalonador local de cada máquina.
- **A execução das aplicações**: para que a execução de uma aplicação seja realizada com sucesso, é importante que haja uma reserva antecipada dos recursos que essa aplicação irá utilizar. Essa etapa, muitas vezes inexistente em diversos gerenciadores de recursos, é seguida da submissão da aplicação ao nó executor. As opções para a submissão variam da execução de um simples comando à execução de diversos *scripts* para a configuração do ambiente. Em seguida, há a necessidade de transferir os arquivos de entrada e outras configurações necessárias para a execução remota da aplicação. Quando uma aplicação está sendo executada, é necessário realizar o monitoramento de sua execução, a fim de detectar possíveis comportamentos indesejáveis, tais como desempenho inferior ao esperado, ou mesmo uma falha no nó de execução. Em casos de falha, pode-se realizar um novo escalonamento. Ao término da execução, é preciso notificar o submissor da aplicação e transferir os arquivos de saída gerados. Os arquivos criados na máquina executora devem, então, ser removidos.

A realização dessas fases envolve diferentes entidades de um *middleware* para computação distribuída.

2.1.6 Middlewares para Grades

Um grande conjunto de pesquisadores desenvolve avanços na tecnologia e o mercado tem consolidado *middlewares* para grades computacionais, tais como o *OurGrid*, InteGrade e o Globus.

- ***OurGrid***[11] é um dos principais projetos brasileiros em grade computacional,

[11] http://www.ourgrid.org/

em produção desde dezembro de 2004, baseado em redes *Peer-to-Peer* (P2P). O objetivo do *OurGrid* é pesquisar e desenvolver soluções para uso e gerenciamento de grades computacionais. O projeto *OurGrid* é fundamentado no projeto *MyGrid*[12], desenvolvido também na Universidade Federal de Campina Grande, que propôs e desenvolveu uma implementação de um sistema de grade computacional. A comunidade de usuários oferece poder computacional a qualquer usuário interessado em se juntar ao grupo e executar suas aplicações paralelas. Seu poder computacional é obtido através dos recursos ociosos dos seus participantes e é compartilhado de tal forma que recebe mais recursos quem oferece mais recursos.

Atualmente, *OurGrid* continua seguindo o princípio básico do *MyGrid*, que é o suporte às aplicações *Bag-of-Tasks* (BoT), ou seja, aplicações paralelas cujas tarefas são independentes. Isso significa, porém, que as tarefas (partes da aplicação) executadas paralelamente na grade não se comunicam entre si. Aplicações que fazem simulação, mineração de dados e renderização de imagem são exemplos de aplicações desse tipo. O *OurGrid* possui código aberto.

- ***InteGrade***[13] também é um projeto nacional, consistindo de um *middleware* para grades computacionais que permite usar o poder de computação ocioso das estações de trabalho, sendo estruturado pelo agrupamento de *clusters*. O Projeto InteGrade é uma iniciativa de várias universidades brasileiras (USP, PUC-Rio, UFMS, UFG, UFMA e UPE) com o objetivo de construir um novo *middleware* de grade computacional, orientado a objetos. O InteGrade provê suporte para aplicações paralelas, segurança e ambiente de desenvolvimento integrado.

- ***Globus***[14] é atualmente o sistema de maior impacto para a construção de grades computacionais, pois oferece uma grande quantidade de serviços que são construídos com padrões aceitos e bem conhecidos pela comunidade científica. Globus é uma colaboração internacional que conduz pesquisa e desenvolvimento para criar tecnologias de grades computacionais.

  A versão 1.0 do Globus *Toolkit*, o *middleware* do sistema Globus foi lançada em 1998 e era utilizada principalmente pelos desenvolvedores do projeto. A versão

[12] http://mygrid.sourceforge.net/
[13] http://www.integrade.org.br
[14] http://www.globus.org

2.0, lançada em 2002, já trazia uma série de novas funcionalidades e facilidades de instalação e uso, o que disseminou o uso do Globus por instituições de pesquisa em todo o mundo. A partir da versão 3.0, lançada em 2003, houve uma mudança significativa: o sistema passou a se adequar a uma arquitetura e a um conjunto de interfaces-padrão, definidas por um comitê de padronização surgido a partir da equipe envolvida com o projeto Globus e seus colaboradores, o GGF. A arquitetura OGSA vem sendo refinada continuamente, desde a sua criação em 2002, e tem o sistema Globus como sua implementação de referência.

Atualmente, o Globus *Toolkit* na sua versão 4 (GT4)[15] é compatível com Serviços *Web*. O GT4 é um conjunto de ferramentas de código aberto como uma coleção de componentes de baixo acoplamento para o uso de recursos de uma grade computacional (GLOBUS, 2009). Esses componentes incluem serviços, bibliotecas de programação e ferramentas de desenvolvimento para a construção de aplicações.

Os componentes do GT4 estão inclusos em cinco áreas de domínio: Segurança (*Security*), Gerenciamento de Dados (*Data Management*), Gerenciamento de Execução (*Execution Management*), Serviços de Informação (*Information Services*) e Bibliotecas de Desenvolvimento (*Runtime Services*), conforme ilustrado na Figura 2.4. A seguir, são descritos alguns dos componentes *Grid Security Infrastructure* (GSI), quanto à Segurança, GridFTP, quanto a Gerenciamento de Dados, o *Globus Resource Allocation Manager* (GRAM), quanto a Gerenciamento de Execução, e o *Monitoring and Discovery Service* (MDS), quanto a Serviços de Informação.

- **Segurança**

  O GSI provê suporte a um conjunto de requisitos relacionados à segurança, incluindo a verificação das identidades de usuários e de recursos, a proteção à troca de mensagens, o controle de permissão de acesso a recursos e serviços e a implementação de políticas de acesso.

- **Gerenciamento de Dados**

  Para a transferência otimizada de grandes quantidades de dados entre localizações distantes, o GT4 oferece o componente GridFTP, que inclui

[15] http://www.globus.org/toolkit/

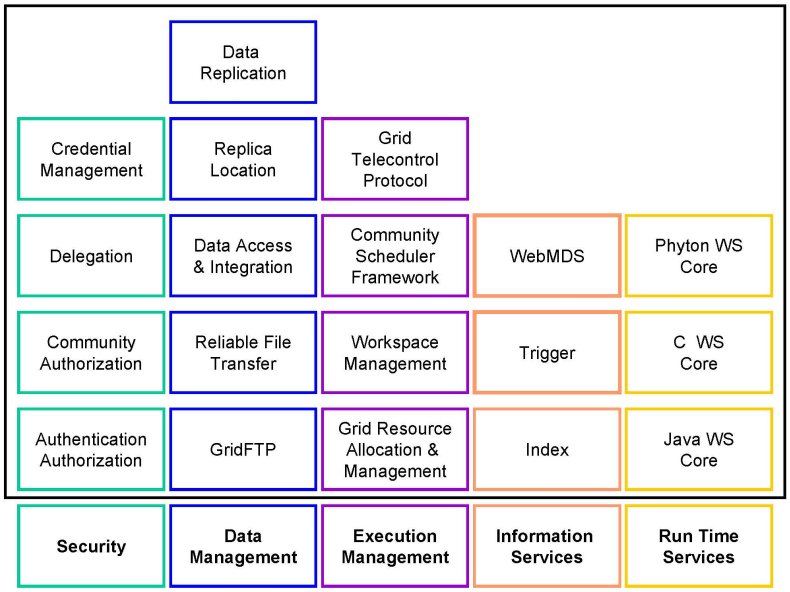


Figura 2.4: Componentes do GT4 (GLOBUS, 2009)

funcionalidades cliente/servidor. GridFTP disponibiliza mecanismos para a transferência confiável, segura e rápida de dados (de memória para memória, e de disco para disco), podendo interoperar com clientes e servidores FTP.

– **Gerenciamento de Execução**

Cada recurso é gerenciado por uma instância do GRAM (CZAJKOWSKI et al., 1998), o responsável por instanciar, monitorar e reportar o estado das tarefas alocadas para o recurso. As requisições do usuário são recebidas pelo *Gatekeeper*, responsável por autenticar e autorizar a execução, usando o GSI. Esse serviço permite uma autenticação única do usuário na grade. A partir dessa autenticação, o GRAM verifica se o usuário pode executar o recurso em questão. Caso o usuário tenha o acesso permitido, é criado um *Job Manager*, que é responsável por iniciar e monitorar a tarefa submetida. As informações sobre o estado da tarefa e do recurso são constantemente reportados ao MDS.

– **Serviços de Informação**

Esse serviço auxilia no gerenciamento de vários tipos de informação. O MDS permite a publicação e consulta de informações sobre a disponibilidade de recursos na grade.

Embora seja um conjunto de ferramentas consolidado no uso de grades computacionais, o Globus se aprimora por meio da integração de novos projetos em parceria com ele para atender demandas em diferentes atividades de uma grade, como de execução de tarefas, escalonamento, descoberta de recursos, ferramentas de programação, entre outras.

No Sistema de Descoberta e Monitoramento da ferramenta Globus *Toolkit*, a descoberta de recursos é baseada em simetria, na qual o processo de busca de recursos é feito através da correspondência dos mesmos. Por meio desse processo, os provedores e os consumidores têm que concordar com o mesmo nome dos atributos de recursos e valores. Para isso, urge solucionar os problemas relacionados à descoberta de recursos, a qual é responsável por selecionar um conjunto destes últimos, já que tal é o propósito deste trabalho.

2.2 Web Semântica

A *Web* atual possui dados espalhados que possuem significado apenas para os seres humanos, sendo difíceis de serem compreendidos por aplicações. Assim a *Web* Semântica (WS), idealizada por (BERNERS-LEE; HENDLER; LASSILA, 2001), tem como objetivo estender a *Web* atual, possibilitando a compreensão da informação pelas máquinas. Para isso ocorrer, é necessário que a informação na *Web* seja descrita através de anotações semânticas, isto é, por meio do conjunto de referências a conceitos e instâncias de uma ontologia.

A *Web* Semântica já produziu uma grande quantidade de conhecimento relacionado à modelagem de domínios e ao desenvolvimento de ferramentas e linguagens. Várias aplicações já foram desenvolvidas, especialmente na área de ferramentas inteligentes de busca. Serviços *Web* Semânticos já foram estruturados a ponto de existirem modelos de descrição que podem ser utilizados na melhoria da descoberta, composição, negociação e orquestração de serviços.

(BERNERS-LEE, 2009) propôs o modelo de uma arquitetura em camadas para *Web*. A ideia desse modelo é de construir em cima do que já existe, em vez de propor uma arquitetura totalmente nova e a consequente reestruturação da *Internet*. A proposta de Berners-Lee é que cada camada vai, gradativamente, trazendo uma nova contribuição, como, por exemplo, maior expressividade e possibilidade de se realizarem inferências e autenticação. Como mostrado na Figura 2.5, descrita nos parágrafos seguintes, o modelo em camadas de padrões e tecnologias sugere uma arquitetura para a *Web* Semântica.

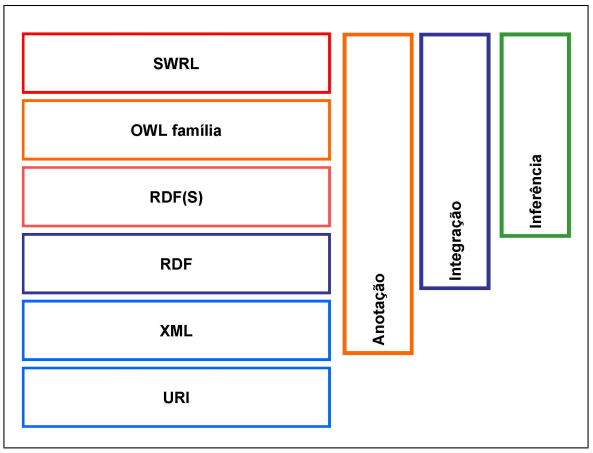


Figura 2.5: *Web* Semântica. Modelo em camada – Fonte adaptado (GOBLE, 2007)

- A camada *Uniform Resource Identifier* **(URI)** fornece a interoperabilidade

em relação à codificação de caracteres e ao endereçamento e nomeação de recursos da WS. Uma URI é um nome (*string*) curto que identifica recursos na *Web*, como documentos, imagens, endereços de arquivos, serviços, endereços eletrônicos, entre outros. A URI é uma forma genérica de *Uniform Resource Location* (URL), o endereço tradicional da *Web*, como por exemplo, www.w3c.org.

- As camadas *eXtensible Markup Language* **(XML)** e *Resource Description Framework* **(RDF)** representam os formatos de dados e informações usuais para a *Web*. A camada XML, junto com definições de tipos e esquemas, garante a integração das definições de WS com outros padrões baseados em XML. A RDF é um modelo universal de formato para dados na *Web*, permitindo que dados estruturados e semiestruturados sejam misturados, exportados e compartilhados por diferentes aplicações. A RDF descreve vários tipos de recursos, enquanto os esquemas XML, por exemplo, descrevem apenas documentos, possibilitando a interoperabilidade entre aplicações e permitindo que o significado dos termos e dos conceitos seja prontamente processado pelos computadores (BERNERS-LEE; HENDLER; LASSILA, 2001).

- A linguagem *RDF Schema* **RDF(S)** contém esquemas que facilitam a recuperação de informações no formato adequado para uma determinada aplicação, tornando possível aos programas de computador 'entender' as informações. A RDF(S) define classes e propriedades específicas às aplicações, sendo que as classes podem ser coleções de recursos.

- A camada *Web Ontology Language* **(OWL)** utiliza Lógica de Descrição (Lógica de primeira ordem) e pretende estender o significado das aplicações que utilizam a *Web* para programas que interpretam automaticamente essas informações. O objetivo é inserir a interoperabilidade de máquinas, programas automáticos, agentes etc. com sistemas *Web*. A linguagem OWL permite que se faça inferência sobre o conteúdo que ela representa.

- *Semantic Web Rule Language* **(SWRL)** é uma linguagem para escrever regras em lógica de primeira ordem. Utilizada em conjunto com OWL, forma uma dupla potente de linguagem de representação do conhecimento para *Web*. SWRL e OWL juntas permitem guardar informação categorizada e recuperá-la. Além disso, também permitem que um motor de inferência possa

usar essa base para acrescentar novos conhecimentos à mesma.

- A camada **Anotação** acrescenta dados à própria informação, permitindo distingui-la de outras, caracterizando-a melhor.

- A camada **Integração** leva a interoperabilidade para as aplicações, integrando fontes de informações; em termos semânticos, a integração traz à tona o significado exclusivo da informação, que extrapola aquela de uma simples palavra-chave ou de um tesauros[16] (*thesaurus*), aproximando-se de um catálogo do tipo 'páginas amarelas'.

- A camada **Inferência** raciocina com base na informação existente e cria novas instâncias da informação. A inferência é possível quando a linguagem de descrição é rica o suficiente para descrever axiomas e frases lógicas, permitindo que um programa do tipo sistema especialista ou uma máquina de regras interprete a informação, acrescentando-lhe nova informação. Esta última, embora ainda não estivesse escrita, obedece às regras existentes.

Em resumo, o desenvolvimento da WS concentra-se na definição de camadas de linguagens utilizadas no suporte à representação e utilização de metadados; as linguagens constituem o instrumental básico utilizado para acrescentar significado à informação necessária para a WS (GOBLE, 2007).

Na Seção 2.3.5 são apresentados mais detalhes da linguagem OWL, visto que a mesma será utilizada nesta dissertação para a definição dos componentes da ontologia, como as classes e suas propriedades.

2.3 Ontologia

Como ontologia tem sido utilizada por várias comunidades (Inteligência Artificial, Representação do Conhecimento, Processamento de Linguagem Natural, *Web* Semântica, Engenharia de *Software*, entre outras) é aceitável que exista uma grande diversidade de definições da mesma. A palavra ontologia tem a sua origem na *Metafísica* de Aristóteles. Dentre as dimensões da metafísica, a ontologia estuda o ser como ser. Várias definições sobre o significado de uma ontologia e sua evolução podem ser encontradas. (GRUBER, 1993a) definiu uma ontologia como

[16]Lista de palavras com significados semelhantes, dentro de um domínio específico de conhecimento.

uma especificação explícita de uma conceitualização, sendo essa definição a mais referenciada na literatura. Baseada na definição de Gruber, várias outras definições foram propostas. (BORST, 1997) modificou-a, dizendo que *ontologias são definidas como uma especificação formal de uma conceitualização compartilhada.*

As definições de Gruber e Borst foram agregadas e explicadas por (STUDER; BENJAMINS; FENSEL, 1998): *Conceitualização* se refere a um modelo abstrato de algum fenômeno do mundo pela identificação de conceitos relevantes daquele; *Explícito* significa que o tipo de conceitos usados e as restrições sobre seu uso são explicitamente definidos; *Formal* se refere ao fato de uma ontologia ser processável por máquina; *Compartilhado* reflete a noção que uma ontologia captura conhecimento consensual, isto é, não é particular a algum indivíduo, mas aceito por um grupo.

Em geral, esses autores tratam a respeito de ontologias de forma independente de domínio, propondo sempre o compartilhamento de informações e permitindo aos usuários a utilização de diferentes linguagens de representação e sistemas. (GUARINO, 1998) também se refere às ontologias de forma independente de domínio do conhecimento, defendendo a necessidade de se construírem ontologias que possam ser reutilizáveis e compartilhadas através de múltiplos serviços e métodos.

2.3.1 Classificação para Ontologias

Segundo (GUARINO, 1998), ontologia se trata de um sistema particular de categorias, de acordo com uma certa visão do mundo, referindo-se a um artefato de engenharia, constituído por um vocabulário específico, utilizado para descrever uma certa realidade. As ontologias podem ser classificadas segundo o seu nível de generalidade (GUARINO, 1998), conforme ilustrado na Figura 2.6:

- **Ontologias genéricas** ou ontologias de topo são reutilizáveis ou aplicáveis em diferentes domínios. Os exemplos mais representativos poderiam ser ontologias que estabelecem relações lógicas entre parte e todo, vocabulário relacionado a eventos, tempo, espaço, comportamento etc.

- **Ontologias de domínio** são reutilizáveis num dado domínio. Elas provêem vocabulários sobre os conceitos dentro de um domínio e seus relacionamentos, sobre as atividades que envolvem esse domínio e sobre as teorias e princípios elementares que governam aquele domínio. Por exemplo: domínio jurídico,

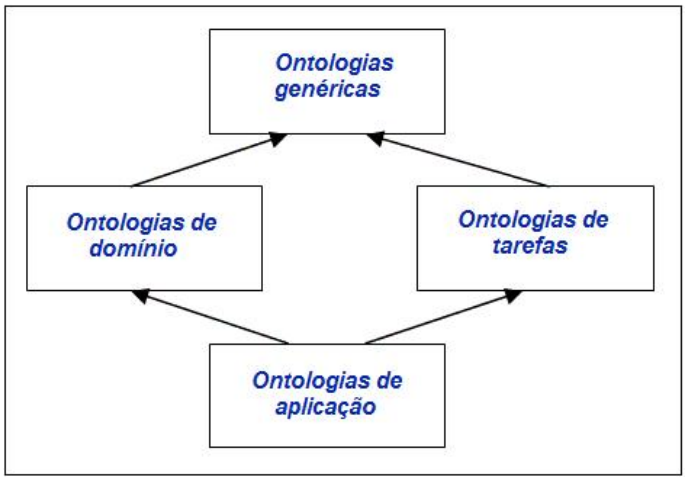


Figura 2.6: Classificação de ontologias (GUARINO, 1998)

direito tributário, microbiologia, recursos de grade computacional etc. Em razão disso, a ontologia do presente trabalho se classifica como uma ontologia de domínio;

- **Ontologias de tarefas** provêem um vocabulário sistematizado dos termos usados para resolver problemas associados com tarefas que podem ou não ser de um mesmo domínio. Essas ontologias fornecem um conjunto de termos para, genericamente, descrever a resolução de um tipo de problema. Incluem nomes genéricos (por exemplo, plano, objetivo, restrição), verbos genéricos (por exemplo, atribuir, classificar, selecionar), adjetivos genéricos (por exemplo, atribuído) e outros nas tarefas agendadas;

- **Ontologias de aplicações** contêm o conhecimento necessário para modelar situações específicas de uma tarefa num domínio particular.

Todos esses tipos de ontologias podem ser combinados para construir uma nova ontologia.

2.3.2 Princípios Básicos para Construção de Ontologias

Segundo (NOY; MCGUINESS, 2009), como todo exercício de modelagem, também não existe uma única forma ou método para moldar ontologias. Existem, porém, algumas regras fundamentais a serem seguidas durante um projeto de ontologias que podem, em muitos casos, ajudar durante as decisões de projeto:

- Não existe uma única forma de modelar um domínio, o que existe são alternativas viáveis ou não. A melhor solução quase sempre depende da

aplicação que se tem em mente e das extensões do domínio que se podem antecipar;

- O desenvolvimento de ontologias é necessariamente um processo interativo.
- Os conceitos na ontologia deveriam estar próximos de objetos (físicos ou lógicos) e seus relacionamentos no seu domínio de interesse. Assim, existe uma grande probabilidade de serem substantivos (objetos) ou verbos (relacionamentos) nas sentenças que descrevem seu domínio.

Conforme (PEREZ; BENJAMINS, 1999) e (GRUBER, 2009), para axiomatizar uma ontologia que venha a apresentar confiabilidade, seja reutilizável e que possibilite compartilhamento, é necessário respeitar alguns princípios a serem seguidos na construção de ontologias.

- Clareza e objetividade: os termos devem ser acompanhados de definições objetivas e também de documentação em linguagem natural;
- Completude: significa que uma definição é expressa por condições necessárias às circunstâncias de uma aplicação;
- Coerência: para permitir derivar inferências que sejam consistentes com as definições;
- Extensibilidade: uma ontologia deve ser projetada para antecipar a utilização de um vocabulário compartilhado, devendo fornecer uma representação que possa ser estendida e especializada, sendo possível a definição de novos termos para outros usos, baseados no vocabulário existente, de forma que não se exija a revisão das definições existentes;
- Mínimo compromisso ontológico: para permitir que sejam definidas tão poucas suposições quanto possíveis sobre o mundo a ser modelado, permitindo à ontologia a liberdade de especializar e instanciar a ontologia, conforme necessário;
- Princípio de distinção ontológica: indica que as classes numa ontologia devem ser disjuntas;
- Diversificação das hierarquias: para aproveitar ao máximo os mecanismos de herança múltipla;

- Modularidade: para minimizar o acoplamento entre os módulos;
- Minimização da distância semântica entre conceitos irmãos: conceitos similares são agrupados e representados como subclasses de uma classe e devem ser definidos usando as mesmas primitivas;
- Padronização: nomes padronizados, sempre que possível.

2.3.3 Componentes de uma Ontologia

Segundo (GRUBER, 1993a) e (PEREZ; BENJAMINS, 1999), o conhecimento nas ontologias é formalizado, usando componentes, como conceitos, relações, funções, axiomas e instâncias, descritos a seguir:

- **Conceitos** representam qualquer coisa sobre algo que é dito e, portanto, também poderiam ser a descrição de uma tarefa, função, ação, estratégia, raciocínio, etc;
- **Relações** representam os tipos de interações entre os conceitos do domínio. São definidas formalmente como qualquer subconjunto de um produto de n conjuntos, ou seja $R : C_1$ x C_2 x,. . . ,x C_n. São exemplos de relações binárias: *subclasse-de* e *conectada-a*;
- **Funções** são relações especiais onde o *n-ésimo* elemento da relação é único para os *n-1* elementos precedentes; formalmente as funções são definidas como $F : C_1$ x C_2 x,. . . ,x $C_{n-1} -> C_n$; são exemplos de funções *antecedente-de* e *causa*, indicando que o valor do segundo componente da relação depende do primeiro;
- **Axiomas** modelam sentenças que são sempre verdadeiras;
- **Instâncias** representam elementos específicos da ontologia, ou seja, os próprios dados.

2.3.4 As Vantagens do Uso de Ontologias

A seguir, é apresentada uma lista com as principais vantagens da utilização de ontologias na Ciência da Computação:

- Ontologias fornecem um vocabulário para representação do conhecimento. Esse vocabulário é baseado em conceitos, evitando desse modo interpretações ambíguas desse vocabulário;

- Ontologias permitem o compartilhamento de conhecimento. Sendo assim, caso exista uma ontologia que modele adequadamente certo domínio de conhecimento, essa pode ser compartilhada e usada por pessoas que desenvolvam aplicações dentro desse domínio. Para exemplificar, considere que exista uma ontologia para o domínio de livrarias. Uma vez que essa ontologia está disponível, várias livrarias podem construir seus catálogos, usando o vocabulário fornecido por essa ontologia, sem a necessidade de refazer uma análise do domínio de livraria;

- Ontologias fornecem uma descrição exata do conhecimento. Diferentemente da linguagem natural, em que as palavras podem ter semântica totalmente diferente conforme o seu contexto, a ontologia por ser escrita em linguagem formal, não deixa espaço para uma lacuna semântica que pode existir na linguagem natural. Por exemplo, quando uma pessoa fala para outra a palavra 'Globo' ela pode estar querendo falar a respeito de um corpo esférico, como também de um canal de televisão brasileiro. A interpretação da palavra pode ser atribuída a um conceito ou outro conforme o estado mental do indivíduo. Porém, se há uma conceitualização comum entre essas duas pessoas, a possibilidade de mal entendido diminui muito. Por exemplo, se essas pessoas concordam numa ontologia sobre o domínio de formas geométricas, possivelmente não haverá mal entendido;

- É possível fazer o mapeamento da linguagem da ontologia, sem que com isso seja alterada a sua conceitualização, ou seja, uma mesma conceitualização pode ser expressa em várias línguas;

- Pode ser possível estender o uso de uma ontologia genérica, de forma que ela se adeque a um domínio específico. Por exemplo, se alguém precisa de uma ontologia sobre bicicletas para construir uma aplicação e só encontra uma ontologia sobre o domínio genérico de veículos, pode utilizar essa ontologia, estendendo-a para o domínio específico da aplicação, no caso, bicicletas.

2.3.5 As linguagens para Representação de Ontologias

Segundo (CORCHO; FERNáNDEZ-LóPEZ; GóMEZ-PéREZ, 2003), no início da década de 1990, foi criado um conjunto de linguagens de implementação de ontologias, baseadas em Inteligência Artificial.

O crescimento da *internet* conduziu à criação das linguagens de ontologia que exploram características da *Web*. Tais linguagens são chamadas geralmente linguagens de ontologia baseadas em *Web* ou *ontology markup languages* (linguagens de marcação de ontologia).

Atualmente, as linguagens para a representação de ontologias são divididas em dois grupos: as baseadas na lógica de primeira ordem; e as baseadas em XML. Esses grupos de linguagens possuem diferentes expressividades e propriedades computacionais, sendo descritas a seguir.

Linguagens Baseadas em Lógica de Primeira Ordem

- ***Knowledge Interchange Format* (KIF)**[17] é baseada em lógica de primeira ordem e serviu de base para criação de uma segunda linguagem, Ontolingua. A linguagem KIF facilita a expressão dos fatos de um domínio de conhecimento, usando uma extensão da lógica de predicados, sendo comumente adotada na representação de ontologias.

- ***Ontolingua***[18] é baseada em KIF, tendo semântica declarativa, isto é, o significado das expressões existentes na representação pode ser compreendido sem recorrer a um interpretador para manipular as expressões.

- ***LOOM***[19] é descendente da família *Knwoledge Language One* (KL-ONE), baseado em lógica descritiva e regras de produção, permitindo a representação de conceitos, taxonomias, relações binárias, funções, axiomas e regras de produção.

- ***Operational Conceptual Modeling Language* (OCML)**[20] permite a especificação de funções, relações e classes, instâncias e regras. Utilizada em

[17] http://www-ksl.stanford.edu/knowledge-sharing/kif/
[18] http://www.ksl.stanford.edu/software/ontolingua/
[19] http://www.isi.edu/isd/LOOM/
[20] http://technologies.kmi.open.ac.uk/ocml//

aplicações de gerenciamento do conhecimento, desenvolvimento de ontologias, comércio eletrônico e sistemas baseados em conhecimento.

- ***Frame Logic* (FLogic)**[21] integra *frames* e lógica de primeira ordem. Trata de uma forma declarativa os aspectos estruturais das linguagens baseadas em *frames* e orientadas a objeto (identificação de objetos, herança, tipos polimórficos, métodos de consulta, encapsulamento etc), ensejando a representação de conceitos, taxonomias, relações binárias, funções, instâncias, axiomas e regras.

Linguagens Baseadas em XML

- ***Resource Description Framework* (RDF)**[22] foi desenvolvida pelo *World Wide Web Consortium* (W3C)[23] como uma linguagem baseada no princípios de redes semânticas, a fim de ser utilizada na descrição de recursos disponíveis na *Web*. A RDF é uma linguagem declarativa, a qual fornece uma maneira padronizada de utilizar o XML para representar metadados no formato de sentenças sobre propriedades e relacionamentos entre itens na *Web*. Esses itens podem ser virtualmente qualquer objeto (texto, figura, vídeo e outros), desde que possuam um endereço *Web*. Um dos objetivos dessa linguagem é tornar a semântica de recursos *Web* acessível a máquinas.

  Em RDF, a descrição dos dados e metadados é feita pelo esquema triplas (recurso-propriedade-valor), sendo respectivamente (sujeito-predicado-objeto):

  - recurso: pode ser qualquer entidade, desde que possua URI;
  - propriedade: recurso que possui um nome e pode ser utilizado para caracterizar um outro recurso, como, por exemplo, criador e título;
  - valor: é o único elemento em RDF que pode ser um recurso (descrito através de uma URI), ou literal, ou valor numérico, ou cadeia de caracteres (por exemplo, 'Aparecida Prado').

  A RDF provê as primitivas básicas para a criação de ontologias simples, incluindo relacionamento de generalização para classes e propriedades. No entanto, a RDF foi criticada por sua falta de expressividade, por não oferecer

[21] http://flora.sourceforge.net/aboutFlogic.php
[22] http://www.w3.org/RDF/
[23] http://www.w3.org/

conectivos lógicos para descrever negação, disjunção e conjunção, restringindo seu poder de comunicação.

- ***RDF Schema* (RDF(S))**[24] estende a especificação básica da RDF, permitindo a definição de vocabulários. Basicamente, essa linguagem fornece o suporte necessário para descrever classes e propriedades, e também para indicar quais propriedades são utilizadas para a descrição de uma classe.

- ***DARPA Agent Markup Language* (DAML)**[25] é uma linguagem que foi desenvolvida como uma extensão para XML e RDF. Provê uma infraestrutura básica que permite às máquinas fazerem a mesma classificação de inferências simples que os seres humanos fazem. Um sistema em DAML pode inferir conclusões que não estão explicitamente declaradas em DAML.

- ***Ontology Inference Layer* (OIL)**[26] é uma camada de inferência e representação baseada na *Web*, que combina a utilização de modelagens primitivas, provenientes das linguagens baseadas em *frames* com a semântica formal. A sintaxe de OIL é baseada em RDF(S) e provê definição de classes e relações, num conjunto limitado de axiomas.

- **DAML+OIL** é a combinação da duas linguagens DAML e OIL, que apresentavam características similares. A semântica formal DAML+OIL é fornecida através do mapeamento da linguagem para a KIF. DAML+OIL é dividida em duas partes: domínio dos objetos, que consiste nos objetos que são membros de classes definidas na ontologia DAML; e domínio dos tipos de dados, que consiste nos valores importados do modelo XML.

- ***Web Ontology Language* (OWL)**[27] foi recomendada pela W3C, sendo desenvolvida como uma extensão do vocabulário RDF e derivada das linguagens DAML+OIL. A OWL foi desenvolvida com base nas necessidades da *Web* Semântica. Ontologias desenvolvidas em OWL possuem as seguintes carcterísticas:

  - Podem ser distribuídas através de diferentes sistemas;
  - Escalabilidade conforme necessidades da *Web*;

[24] http://www.w3.org/TR/rdf-schema/
[25] http://www.daml.org/
[26] http://www.ontoknowledge.org/oil/
[27] http://www.w3.org/2004/OWL/

- Compatibilidade com padrões *Web* quanto à acessibilidade e internacionalização;
- Extensibilidade.

Segundo (SMITH; WELTY; MCGUINNESS, 2009), a OWL visa a prover uma linguagem que pode ser usada para descrever as classes e relacionamentos entre elas, que estão implícitos nos documentos *Web* e aplicações. Essa linguagem pode ser usada para formalizar um domínio através da definição de classes e suas propriedades, definir indivíduos e afirmar propriedades sobre eles e prover raciocínios lógicos sobre estas classes e indivíduos, de acordo com o grau permitido pela semântica formal da OWL.

OWL provê um conjunto de vocabulário mais rico do que o encontrado em RDF, a fim de melhor restringir o conjunto de triplas que podem ser representadas. Um documento OWL pode incluir um cabeçalho opcional de ontologia, classes, propriedades e descrições de indivíduos ou axiomas.

OWL possui três linguagens, em ordem crescente de expressividade:

- **OWL *Lite*** fornece suporte aos usuários que precisam de uma hierarquia de classificação e funcionalidades de restrições simples; por exemplo, dá suporte à cardinalidade, mas, no entanto, aceita apenas valores 0 e 1. Oferece suporte à migração de tesauros e taxonomias para o formato de ontologias.
- **OWL DL** é uma extensão da OWL *Lite*, suportando os usuários que necessitam de maior expressividade. Por exemplo, a sublinguagem permite definir uma classe através de união de outras duas classes, como também definir que a instância de uma classe não pode ser a instância de uma outra classe.
- **OWL *Full*** suporta o máximo de expressividade, além de maior liberdade sintática do RDF, permitindo também que uma ontologia aumente o significado de seu vocabulário, RDF ou OWL, predefinido. OWL *Full* pode ser vista como uma extensão da linguagem RDF. Deste modo, todo documento OWL pode ser visto como um documento RDF e todo documento RDF, como um OWL *Full*.

2.3.6 As ferramentas para a Construção de Ontologias

A formalização de ontologias para um conjunto de agentes não é uma tarefa trivial, já que significa tornar explícito algo que normalmente é implícito. As ferramentas de edição podem simplificar consideravelmente o processo de construção de ontologias, desde o início ou a partir de outras já existentes. Geralmente, estas ferramentas incluem documentação, importação e exportação de ontologias existentes (de diferentes formatos), visualização gráfica, bibliotecas e mecanismos de inferência.

Em função da diversidade de linguagens de construção de ontologias existentes, muitas ferramentas também foram propostas. Praticamente todas as linguagens possuem ao menos uma ferramenta para apoiar a construção de ontologias. Passaremos a expor sobre o Protégé, que foi a ferramenta utilizada neste trabalho, além de citarmos algumas outras ferramentas existentes, tais como o Altova SemanticWorks, TopBraid Composer, OntoEdit, Ontolingua, WebOnto e Chimaera.

- **Protégé**[28]: é uma plataforma desenvolvida pelo grupo de pesquisa *Stanford Medical Informatics* da escola de medicina da Universidade de Stanford. Através do Protégé, é possível descrever os conceitos pertencentes à ontologia, juntamente com seus atributos e relacionamentos. O Protégé possui uma arquitetura de metaclasses, documentos de formato padrão utilizados para definir novas classes numa ontologia, tornando-a facilmente extensível e permitindo o seu uso juntamente com outros modelos de conhecimento. Essa ferramenta apresenta interoperabilidade com outros sistemas de representação do conhecimento e, além disso, possui uma interface intuitiva que torna simples a sua utilização e configuração.

  Protégé é uma ferramenta que permite aos usuários construir domínios ontológicos e customizar formulários de entrada de dados. É também uma plataforma que pode facilmente ser estendida para incluir componentes gráficos como tabelas, arquivos de som, imagens e vídeos, e vários formatos de armazenamento como OWL, RDF, XML, além de tratar HTML.

  Percebendo o potencial de desenvolvimento de seus usuários, os desenvolvedores do Protégé tornaram o código da ferramenta aberto, permitindo a colaboração de toda a comunidade dos mesmos. A partir

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

daí, surgiu uma arquitetura integrável a diversas aplicações, através de componentes que podem ser conectados ao sistema. Essa arquitetura propiciou o desenvolvimento de *plugins* que acrescentam novas funcionalidades acopladas às já existentes. O Protégé permite que aplicações externas utilizem sua *Application Programming Interface* (API).

- **Altova SemanticWorks 2009**[29] é o editor perfeito de RDF/OWL para *Web* Semântica, fazendo documentos RDF graficamente, vocabulários RDF(S) e ontologias OWL, produzindo-os ou em formato RDF/XML, ou N-Triples. Esse editor não é uma ferramenta livre, sendo sua utilização grátis apenas por trinta dias.

- **TopBraid Composer**[30]: trata-se de uma das mais recentes plataformas destinadas ao desenvolvimento de ontologias na WS, bem como para a construção de aplicações semânticas. O TopBraid está em conformidade com as normas W3C, utilizando as linguagens RDF, RDF(S), OWL, SWRL e SPARQL, oferecendo suporte para o desenvolvimento, controle e teste dos modelos do conhecimento (ontologias) e das suas instâncias, fornecendo, por fim, uma estrutura flexível e extensível através de APIs para a construção de aplicações cliente/servidor semânticas. Além da definição de classes, subclasses e propriedades, as interfaces gráficas do Composer permitem a declaração de construtores e restrições OWL e regras SWRL. Esse editor não é uma ferramenta de domínio público.

- **OntoEdit**[31]: é uma ferramenta desenvolvida pelo *Institut für Angewandtel Informatik und Formale Beschreibungsverfahren* (AIFB) da Universidade de Karlsruhe, na Alemanha. A ferramenta concentra-se nos principais passos para o desenvolvimento de ontologias, contemplando as atividades de especificação, refinamento e avaliação;

- **Ontolingua**[32]: foi desenvolvida para dar suporte a projetos e especificações de ontologias com uma semântica lógica clara. É baseada na linguagem KIF, bem como numa ontologia de representação de conhecimento, a qual define termos de linguagens baseadas em quadros e orientadas a objetos.

[29] http://www.altova.com/semanticworks.html
[30] http://topquadrant.com/
[31] http://www.ontoknowledge.org/tools/ontoedit.shtml
[32] http://www.ksl.stanford.edu/software/ontolingua/

Provê suporte explícito para a construção de módulos ontológicos e faz distinção entre uma ontologia de representação e de aplicação, permitindo a construção de ontologias de três formas: usando expressões do KIF; usando apenas o vocabulário definido em *Frame Ontology*; ou usando as duas formas simultaneamente;

- **WebOnto**[33]: esta ferramenta possibilita a navegação, criação e edição de ontologias, representadas na linguagem de modelagem OCML. Permite o gerenciamento de ontologias por interface gráfica, inspeção de elementos, verificação da consistência da herança e trabalho cooperativo. Possui uma biblioteca com mais de cem ontologias.

- **Chimaera**[34]: é um *software* que permite aos usuários criar e manter ontologias distribuídas na *Web*. As duas maiores funções que ele suporta são mesclagem de múltiplas ontologias e diagnóstico de várias ontologias, coletiva ou individualmente. Também permite o carregamento de bases de conhecimento em diferentes formatos, reorganização de taxonomias, resolução de conflitos de nome, edição de termos, entre outras opções. O *Chimaera* está disponível para uso na *internet*, solicitando registro para acesso completo à sua versão funcional. Ele pode carregar e exportar arquivos no formato OWL e DAML.

2.3.7 Mecanismos de Inferência

Uma inferência permite extrair novos conhecimentos e novas conclusões válidas a partir de um conjunto de premissas, ou seja, da existência das regras e restrições das ontologias.

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

O Protégé fornece API para comunicação com os raciocinadores Pellet, RacerPro e o Fact++. Nesta dissertação, foi utilizado o Pellet, que será citado a seguir, juntamente com outros raciocinadores:

[33] http://projects.kmi.open.ac.uk/webonto/
[34] http://www.ksl.stanford.edu/software/chimaera/

- **Pellet**[35] é um raciocinador para OWL DL, desenvolvido em Java pela University of Maryland's Mindswap Lab, posteriormente se tornando uma linguagem de código aberto. É baseado nos algoritmos de *tableaux*, desenvolvidos para lógica de descrição e suporta toda a expressividade da OWL DL.

  O Pellet utiliza o protocolo conhecido como *Description Logics Implementors Group Interface* (DIG), o qual permite a interação do raciocinador com o Protégé. A única configuração necessária é a definição de uma porta para comunicação em ambos os aplicativos. O núcleo do Pellet consiste num quadro de algoritmos que inferem e verificam a consistência da ontologia. Pellet pode ser utilizado em conjunto com o Jena, como um motor de inferência externo, por ser mais completo.

  Jena[36] é um *framework* de código aberto, escrito em Java e desenvolvido pela Hewlett-Packard (HP). O Jena disponibiliza uma API que permite a uma aplicação manipular ontologias. Seu objetivo é proporcionar um *framework* na linguagem Java que dê suporte à utilização da WS por qualquer aplicativo capaz de utilizá-la. Esse suporte inclui recursos para manipulação de RDF, RDF(S), OWL e DAML+OIL.

- ***Renamed Abox and Concept Expression Reasoner*** **(RacerPro)**[37] é um sistema de representação de conhecimento baseado em DL. O sistema provê suporte para especificações de terminologias em geral, através do uso de axiomas, incluindo axiomas que descrevam a relação hierárquica entre conceitos, definições múltiplas ou cíclicas. O sistema também provê a verificação de consistência, além de várias funções e serviços de recuperação de informação pré-programados. Essa ferramenta é livre somente para organizações sem fins lucrativos, como as universidades.

- **Fact++**[38] é uma implementação do motor de inferência Fact, estendendo suas características e otimizações, além de buscar satisfazer de forma robusta e eficiente todas as características da linguagem OWL.

  As principais tarefas para as quais o Fact++ está habilitado:

[35] http://clarkparsia.com/pellet/
[36] http://jena.sourceforge.net/
[37] http://www.racer-systems.com/
[38] http://owl.man.ac.uk/factplusplus/

- verificação de consistência de uma ontologia;
- verificação da satisfação de um conceito ou de um grupo de conceitos;
- verificação de relacionamento entre conceitos e classificação da ontologia, criando taxonomias.

O Fact++ pode ser utilizado em conjunto com outras ferramentas através da interface DIG, desde que essas ferramentas também façam uso dessa interface. Porém, ele pode ser utilizado como um componente isolado, onde, dada uma ontologia como entrada, o Fact++ realizará as tarefas para as quais está habilitado.

Na Tabela 2.1, é apresentada uma comparação entre estes mecanismos de inferência.

Tabela 2.1: *Comparação entre Mecanismos de Inferência*

	Pellet	Racer Pro	FaCT++
Lógica	DL	DL	DL
Suporta	OWL-DL	OWL-DL	OWL-DL
Tipos Dados em	XML	XML	–
Baseada em	Java	Lisp	C++
Verificação(Consistência)	Sim	Sim	Sim
Interface	DIG,Java,Jena e Protégé	DIG e Protégé	DIG e Protégé
Linguagem(Consulta)	RDQL e SPARQL	nRQL,OWL-QL e SPARQL	–
Código Aberto	Sim	Não	Sim

2.3.8 Linguagens para Consultas em Ontologias

Neste trabalho, foi utilizada a linguagem de consulta SPARQL, sendo a mais indicada para consultas sobre ontologias, por seguir um padrão recomendado pelo W3C. A seguir, será feita uma breve descrição desta e de outras: *new RACER Query Language* (nRQL), *Semantic Query-Enhanced Web Rule Language* (SQWRL), *OWL Query Language* (OWL-QL) e *Semantic Protocol and RDF Query Language* (SPARQL).

- **nRQL**[39] faz parte do sistema RACER, um processador otimizado de consultas para OWL DL. Uma consulta nRQL é composta de um cabeçalho e um corpo. O corpo da consulta consiste de uma expressão, enquanto o cabeçalho corresponde às variáveis referenciadas no corpo da consulta, as quais serão exibidas no resultado.

[39] http://www.racer-systems.com/products/racerpro/preview/overview.phtml

- **SQWRL**[40] é uma linguagem de consulta baseada em SWRL, sendo a mais indicada para consultas sobre ontologias expressas em OWL, pois permite pesquisar documentos que utilizam toda e qualquer ferramenta disponível em OWL para essas descrições (GASSEN et al., 2008). A linguagem permite o uso de um motor de inferência, o qual facilita a descoberta de fatos a partir da representação de conhecimento de um determinado domínio expresso na ontologia.

- **OWL-QL**[41] é uma linguagem que especifica os relacionamentos semânticos entre uma consulta, uma resposta da consulta e uma base de conhecimento usada para produzir a resposta obtida (FIKES; HAYES; HORROCKS, 2003). OWL-QL provê um protocolo de comunicação entre agentes: o cliente, que faz a consulta, e o servidor, que responde a consulta, utilizando conhecimento representado em OWL.

- **SPARQL**[42] é um padrão recomendado pelo W3C para consulta em RDF. Essa linguagem possui também uma sintaxe semelhante à da linguagem SQL. As consultas em SPARQL são intuitivas e de mais fácil compreensão para profissionais com experiência em banco de dados.

  A SPARQL pode ser usada para expressar consultas entre diversas fontes de dados, podendo ser cada um destes armazenado ou visualizado como um documento RDF. A linguagem contém capacidades de consultar padrões opcionais ou obrigatórios de grafos com suas conjunções e disjunções, além de possibilitar expressar restrições nos termos RDF, que aparecerão no resultado da consulta, o qual pode ser um conjunto de grafos em RDF.

Na Tabela 2.2 são apresentadas algumas características das linguagens de consulta de ontologias descritas anteriormente.

Tabela 2.2: *Comparação entre Linguagens de Consulta*

	nRQL	SQWRL	OWL-QL	SPARQL
Consulta	OWL-DL	OWL	OWL	RDF
Motor de Inferência	Sim	Sim	Sim	Sim
Recomendação W3C	Não	Não	Não	Sim
Baseada em	Lisp	SWRL	XML	SQL

[40] http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL
[41] http://www.ksl.stanford.edu/projects/owl-ql/
[42] http://www.w3.org/TR/2009/WD-sparql-features-20090702/

2.4 Redes de Petri

Originalmente, o conceito de Redes de Petri (*Petri Nets*) (JENSEN, 1997) é uma técnica de descrição formal que faz uso de uma modelagem matemática e gráfica, desenvolvida por Carl Adam Petri, com a finalidade de representar sistemas concorrentes, controles, conflitos de sincronização e compartilhamento. Existem diversas extensões de Redes de Petri (RPs), dentre elas podem ser citadas: as coloridas, as temporizadas e as estocásticas.

O presente trabalho tem como objetivo abordar somente as Redes de Petri Coloridas (*Coloured Petri Nets*), as quais serão utilizadas para modelagem do trabalho proposto, em razão de as mesmas serem capazes de estruturar sistemas grandes e complexos, devido à quantidade de recursos dos quais dispõem, possibilitando uma redução nos tamanhos dos modelos.

2.4.1 Redes de Petri Coloridas

As Redes de Petri Coloridas (RPCs) consistem numa técnica formal com embasamento matemático para descrição de sistemas, especialmente apropriadas para modelagem de sistemas a eventos discretos. O principal objetivo das RPCs (JENSEN; KRISTENSEN, 2009) é a redução do tamanho do modelo, permitindo que fichas individualizadas (coloridas) representem diferentes processos ou recursos numa mesma subrede. Em RPC, as fichas são representadas por estruturas de dados complexas. Desse modo, as fichas podem conter informações. Além disso, cada lugar armazena fichas de um certo tipo definido e arcos realizam operações sobre elas. As transições determinam a dinâmica da RPC e podem apresentar expressões de guarda. Estas, por sua vez, indicam os tipos de fichas que possibilitam habilitar uma transição.

Uma RPC é composta por três partes: estrutura, inscrições e declarações.

- A estrutura é um grafo direcionado, com dois tipos de nós (lugares e transições), com arcos interconectando nós de tipos diferentes. Graficamente os lugares são representados, por círculos ou elipses, e transições, simbolizadas por retângulos, como mostrado na Figura 2.7.

- As inscrições são associadas aos lugares, transições e arcos.

- As declarações são tipos, funções, operações e variáveis. Quando a expressão do arco é avaliada, ela gera um multiconjunto de fichas coloridas.

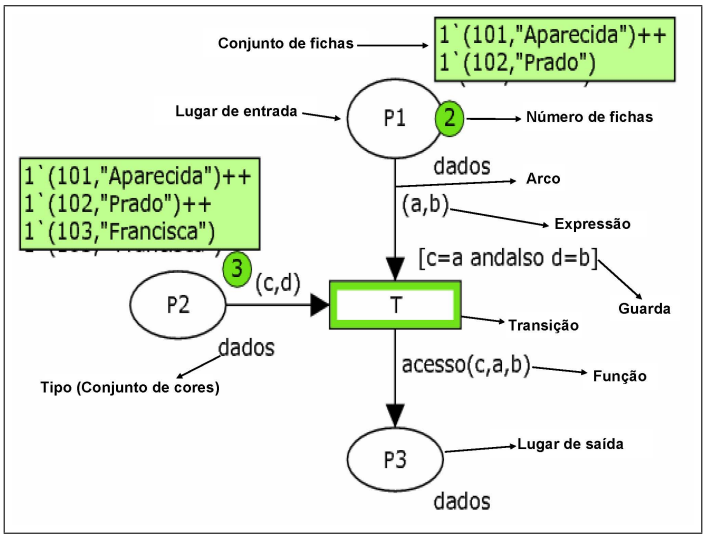


Figura 2.7: Representação de uma RPC

2.4.2 Definição Formal

Uma RPC é definida por:

(P, T, A, Σ, V, C, G, E, I), em que:

i. P é um conjunto finito de ***lugares***.

ii. T é um conjunto finito de ***transições***, de forma que $P \cap T = \phi$.

iii. A é um conjunto finito de ***arcos***, de forma que: $A \subseteq P$ x $T \cup T$ x P.

iv. Σ é um conjunto de tipos não-vazios chamados ***cores***.

v. V é um conjunto finito de ***variáveis***, de forma que $[\nu] \in \Sigma$ para todas as variáveis $\nu \in V$.

vi. $C{:}P \rightarrow \Sigma$ é uma função de ***coloração***, que atribui um conjunto de cores para cada lugar.

vii. G é uma função de ***guarda***, que associa a cada $t \subset T$ uma expressão do tipo booleano.

viii. E é uma função de ***expressão de arcos*** que associa uma expressão de arco para cada arco, tal que $[E(a)] = C(p)$, onde p é o lugar ligado ao arco a.

ix. I é uma função de ***inicialização*** que associa a cada $p \subset P$ uma expressão do tipo C(p), ou seja, é a marcação inicial da RPC.

2.4.3 As Vantagens do Uso de RPCs em Grade

Em (TRCKA et al., 2008) (BRATOSIN et al., 2008), listam-se os motivos para usar RPCs na modelagem e análise das arquiteturas de grade:

- RPC apresenta modelos gráficos, hierárquicos, e têm uma semântica formal. Uma modelagem em RPC de uma arquitetura de grade pode servir como um modelo descritivo de maneira inequívoca, mostrando claramente como as diferentes partes internamente são concebidas e estruturadas. Os modelos em RPC são executáveis, a fim de que eles forneçam os aspectos dinâmicos da rede, contribuindo para uma melhor compreensão de todo o mecanismo;

- O CPNTools é baseado em RPCs (JENSEN; KRISTENSEN, 2009), o qual se constitui numa ferramenta poderosa para a modelagem, verificação funcional (diversas técnicas) e análise de desempenho (simulação de eventos discretos). A ferramenta é especialmente adequada para a avaliação de desempenho, uma vez que fornece monitoramento das instalações utilizadas para extrair dados durante a simulação, dissociando totalmente a análise da modelagem;

- RPCs têm sido amplamente utilizadas em modelos de diferentes tipos de sistemas concorrentes. Existe uma infinidade de modelos, de várias áreas, especificando conceitos semelhantes aos presentes na grade.

Capítulo 3

Trabalhos Correlatos

A grade semântica é uma extensão da grade atual, em que é expresso um significado bem definido às informações dos recursos, permitindo um melhor trabalho de cooperação entre computadores e usuários (ROURE; JENNINGS; SHADBOLT, 2005). Como tecnologia Web Semântica, a grade permite a integração das aplicações, dados, recursos e usuários em domínios específicos, facilitando automação de tarefas, tais como a seleção de recursos para execução das mesmas, baseada em requisições (ROURE, 2005).

A produção bibliográfica existente trata do desafio da grade semântica através de diferentes abordagens. Na prática, trabalhar com grade semântica tem como significado principal a introdução de tecnologias da Web Semântica para a grade. O conhecimento e o vocabulário acumulados de um domínio podem ser capturados em ontologias, que são modelos de conceitos, seus inter-relacionamentos e suas restrições, que podendo ser processados pelas máquinas.

Existe um número crescente de projetos recentes que concentram seus esforços em ontologia relacionada à computação em grade, os quais podemos citar e comparar com nosso trabalho: (TANGMUNARUNKIT; DECKER; KESSELMAN, 2003); (PERNAS; DANTAS, 2005); (SOMASUNDARAM et al., 2006); (KAILASH et al., 2007); (VIDAL et al., 2009).

(TANGMUNARUNKIT; DECKER; KESSELMAN, 2003) implementaram um protótipo de um selecionador de recursos baseado em ontologia que explora ontologias, conhecimento e regras para a resolução de correspondência de recursos na grade. Os autores utilizam o editor Protégé para criar manualmente as instâncias da ontologia

que são salvas no formato RDF. Eles desenvolveram um protótipo *Ontology Resource Matchmaker* (OMM) (Figura 3.2) para o suporte a serviços de correspondência de recursos baseada em ontologia, a qual não está incluída no serviço de corretagem (*brokering*), pelo menos não nesse artigo. Vale ressaltar que utilizaram o sistema de banco de dados dedutivo TRIPLE/XSB para processar as consultas através das regras de correspondência, em combinação com o conhecimento e ontologias.

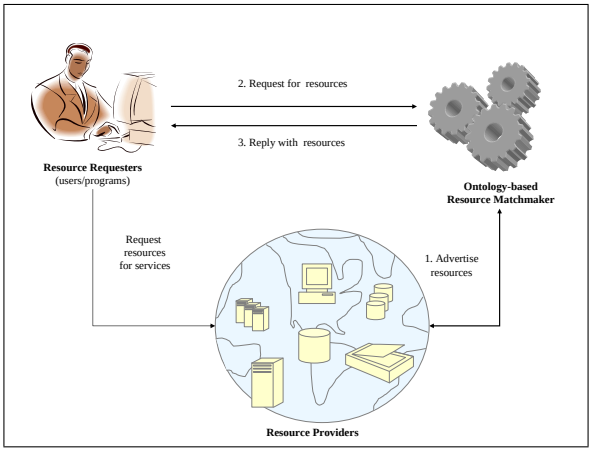


Figura 3.1: Proposta para seleção de recursos baseada em ontologia (TANGMUNARUNKIT; DECKER; KESSELMAN, 2003)

(PERNAS; DANTAS, 2005) desenvolveram uma ontologia na linguagem OWL, com o objetivo de facilitar a busca e seleção dos recursos. Essa ontologia atua diretamente nos serviços de diretório da grade, onde as consultas a respeito dos recursos são feitas sobre o vocabulário definido pela ontologia. Como mostrado na Figura 3.2, para se realizar a busca de recursos, fazendo uso de uma aplicação, primeiramente é feita uma consulta à ontologia, a qual utiliza, por sua vez, os metadados e as visões semânticas para obter maior informação acerca dos recursos.

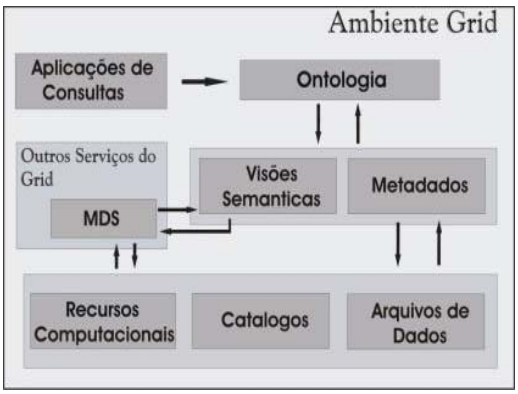


Figura 3.2: Arquitetura para busca e seleção de recursos da grade (PERNAS; DANTAS, 2005)

Em (SOMASUNDARAM et al., 2006) propõe-se uma arquitetura de grade semântica com cinco camadas (*Fabric*, *Knowledge*, *High Level Middleware*, *Middleware* e *Application*), como mostrado na Figura 3.3, introduzindo uma camada de *Knowledge* (Conhecimento) acima da de *High Level Middleware* (Alto Nível da *Middleware*) *Gridbus Broker*[1], possibilitando, assim, a descoberta semântica de recursos. A proposta deles concentra-se não apenas na descoberta semântica de recursos, mas também no suporte a escalonamento tanto computacional quanto de dados das aplicações de grade.

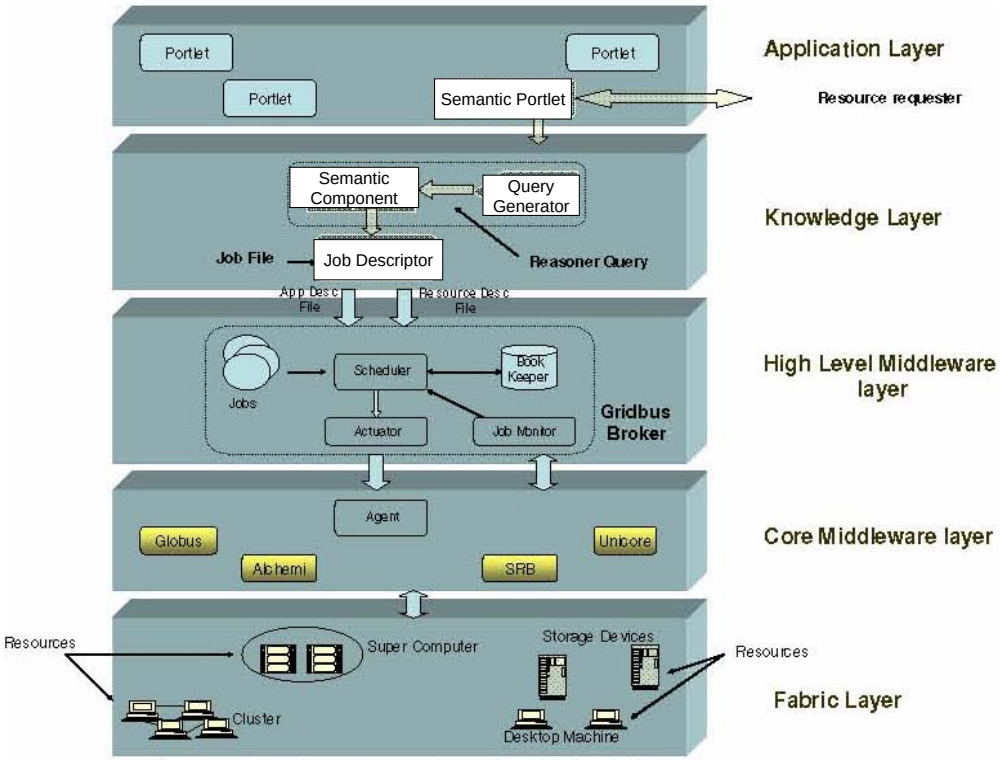


Figura 3.3: Arquitetura de grade semântica (SOMASUNDARAM et al., 2006)

(KAILASH et al., 2007) propõem uma ontologia com a descrição dos recursos da grade (Figura 3.4) para solucionar o problema de correspondência dos mesmos nesse ambiente. Essa ontologia foi desenvolvida na linguagem RDF *Schema* e para a sua edição foi utilizada a ferramenta Protégé. Porém, o trabalho não propõe um algoritmo de correspondência de recursos.

[1] http://www.gridbus.org/broker/

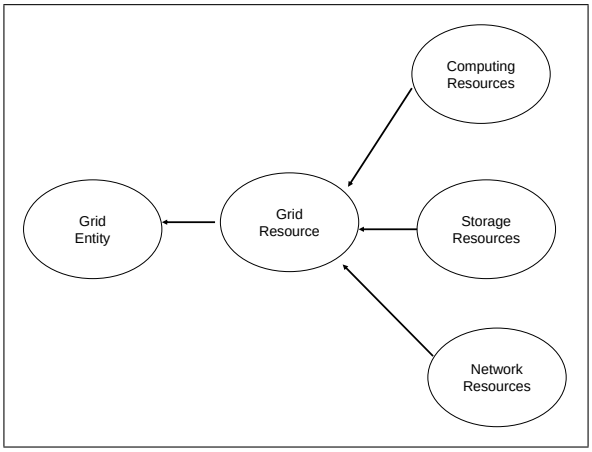


Figura 3.4: Proposta para descrição de recursos da grade baseada em ontologia (KAILASH et al., 2007)

Em (VIDAL et al., 2009) aponta-se uma abordagem semântica para seleção de recursos e *softwares* de artefatos equivalentes, a fim de melhorar a requisição dos recursos adequados para execução de um conjunto de aplicações. A implementação do protótipo utilizou o *middleware* InteGrade (Figura 3.5), uma grade de processamento genérico, com suporte a uma variada gama de aplicações.

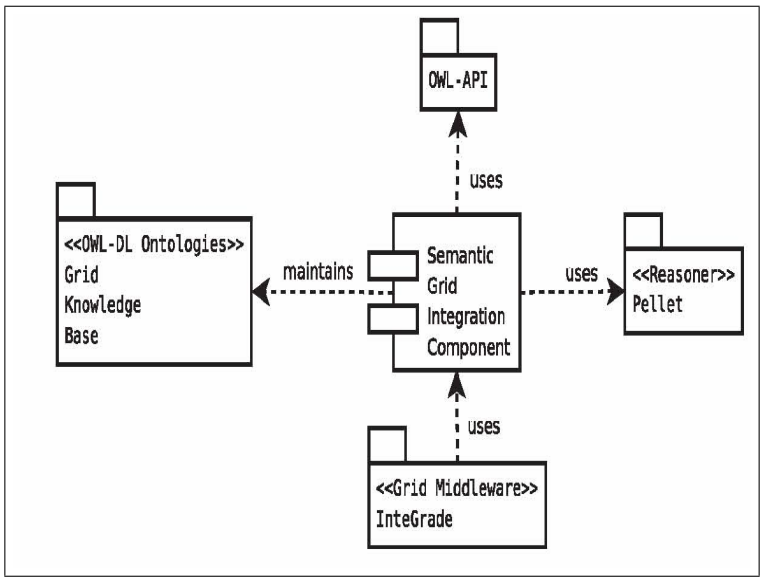


Figura 3.5: Arquitetura de integração baseada em semântica - *Middleware* InteGrade (VIDAL et al., 2009)

No Capítulo 4, mas precisamente na Seção 4.1, apresenta-se uma comparação entre o trabalho proposto e os trabalhos correlatos.

Capítulo 4

Modelagem do *Framework* Semântico Baseado em Ontologia para Descoberta e Seleção de Recursos

Neste trabalho é proposto um *framework* semântico para descoberta e seleção de recursos em ambiente de grade, fornecendo uma abordagem flexível e extensível, em que a correspondência de recursos é baseada em ontologia. A proposta do *framework* consiste em quatro camadas: Fábrica, *Middleware*, Conhecimento e Aplicação, mostradas na Figura 4.1. Na Tabela 4.1, é apresentada uma comparação entre o trabalho aqui proposto e os trabalhos correlatos, apresentados no Capítulo 3.

4.1 *Framework* Semântico

Como já foi mencionado anteriormente, esse *framework* apresenta quatro camadas (Fábrica, *Middleware*, Conhecimento e Aplicação) (Figura 4.1). Dentre essas, as camadas de Conhecimento e Aplicação são as contribuições do nosso trabalho, sendo apresentadas com mais detalhes nas Seções 4.2.3 e 4.2.4.

- Na camada ***Middleware*** utiliza-se o Sistema de Descoberta e Monitoramento da ferramenta GT4, ou seja, o MDS4. Este último coleta as informações dos recursos da grade juntamente com o Ganglia, o qual é usado para coletar automaticamente as informações dos mesmos;

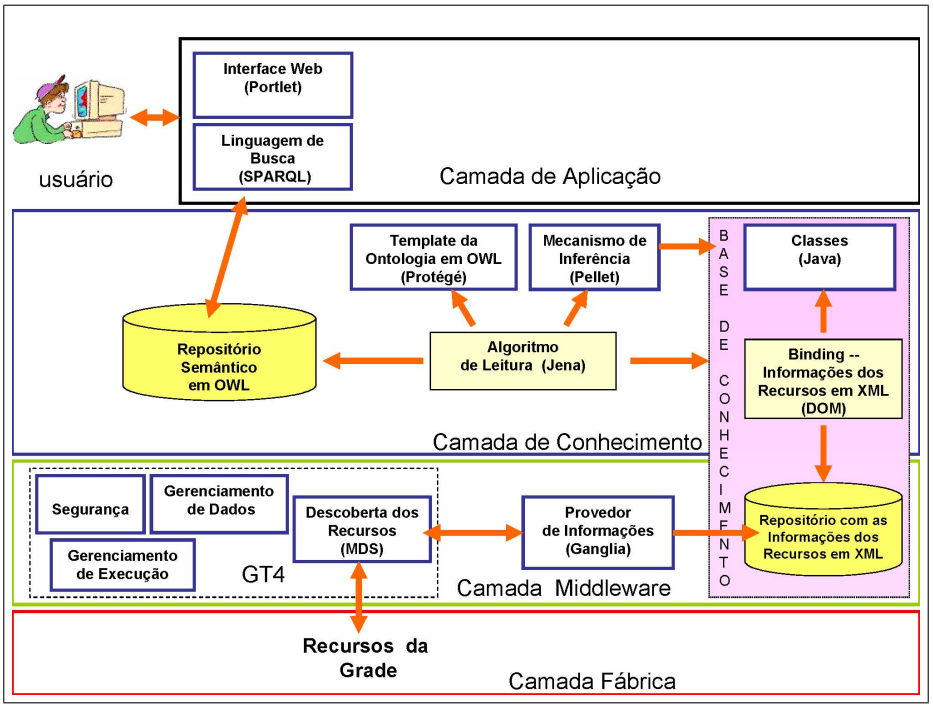


Figura 4.1: *Framework* Semântico para Descoberta e Seleção dos Recursos em Grade

- Na camada de **Conhecimento**, através de um Repositório Semântico (OWL), como ilustrado na Figura 4.1, que provê a descoberta dos recursos na grade. Esse repositório é constituído pela base de conhecimento juntamente com o *template* da ontologia. Para criação desta última, utilizam-se as informações dos recursos no ambiente de grade, por meio do editor Protégé-OWL. Para criar dinamicamente a base de conhecimento dos recursos da grade é utilizado o *framework* Jena. O mecanismo de inferência Pellet é usado para interagir com a base de conhecimento na descoberta de mais resultados desses recursos;

- Na camada de **Aplicação** é desenvolvida uma interface *Web portlet*[1], utilizando uma linguagem de busca dinâmica para prover aos usuários a consulta dos recursos da grade, possibilitando a seleção destes últimos de forma semântica.

[1] O *portlet* é um componente visual independente, que pode ser utilizado para disponibilizar informações dentro de uma página *Web*.

Tabela 4.1: *Comparação entre os trabalhos correlatos*

	Tangmunarunkit	Pernas	Somasundaram	Kailash	Vidal	Aparecida
Ano de Publicação	2003	2005	2006	2007	2009	2010
Middleware	Condor	Globus MDS2	Globus/GridBus Broker/MDS4	Globus	InteGrade	Globus MDS4/Ganglia
Mecanismo de Inferência	TRIPLE/XSD	Não	Algenon	Não	Pellet	Pellet
Mecanismo de consulta	Sim	Não	--------	Não	SPARQL	SPARQL
Linguagem de descrição da ontologia	RDF Schema	OWL	OWL	RDF Schema	OWL	OWL
Editor da ontologia	Protégé	Protégé	Protégé	Protégé	Protégé	Protégé 4.0
Aplicação para o usuário final	Não	Java	Web	Não	------	Web/Portlet
Objetivos	Selecionador de recursos	Busca e seleção de recursos	Descoberta de recursos (Broker)	Descrição dos recursos	Seleção de recursos	Descoberta e seleção de recursos

Foi projetado e implementado um *framework* semântico baseado em um template de ontologia e algoritmos para a correspondência de recursos em uma grade. No que diz respeito a esta dissertação, diferentemente de Tangmunarunkit, que incluiu um serviço de corretagem (*brokering*), tal não é o foco principal deste trabalho.

Já Pernas apresentou uma ontologia na linguagem OWL, com o objetivo de facilitar a busca e seleção dos recursos, através de uma aplicação em Java, enquanto que no presente trabalho a ontologia pode ser estendida posteriormente, havendo desenvolvido um algoritmo com um mecanismo de raciocínio, o qual permite consultar conhecimento semântico relacionado com ambientes de grade. Para o usuário fazer suas solicitações, utiliza-se uma aplicação em *Web*.

Em Somasundaram, propõe-se uma arquitetura de Grade Semântica introduzindo uma camada de conhecimento no topo da arquitetura do *broker* Gridbus, possibilitando assim a descoberta de recursos semanticamente através do *broker*. Eles propõem uma arquitetura de cinco camadas que implementa uma camada de conhecimento desenvolvida para suportar escalonamento tanto computacional quanto de dados das aplicações de grade, ao passo que esta dissertação traz a descoberta e seleção de recursos, facilitando a busca dos mesmos.

Kailash propõe uma ontologia com a descrição dos recursos da grade, o qual foi desenvolvida na linguagem RDF *Schema*. A linguagem apresenta menos

expressividade em relação a linguagem OWL. Porém, o trabalho recém-citado propõe um algoritmo de correspondência de recursos através de linguagem de busca dinâmica.

Quanto a Vidal, ele fez uso de um sistema (Integrade) para construção de grade, enquanto que o presente trabalho utilizou o Globus *Toolkit*, que é um sistema o qual oferece uma grande quantidade de serviços que são construídos com padrões aceitos e bem conhecidos.

4.2 Modelagem do *Framework*

Na modelagem do *framework*, descrito na Seção 4.1 e ilustrado na Figura 4.1, foram utilizadas as Redes de Petri Coloridas (RPCs) (JENSEN; KRISTENSEN, 2009), a ferramenta CPNTools[2], que se trata de um programa computacional, executado em ambiente Windows e Linux, para modelagem, análise e validação de RPCs.

Na Figura 4.2, é apresentada a página principal da modelagem do *framework*. Nela podem ser visualizadas as quatro camadas, Fábrica, *Middleware*, Conhecimento e Aplicação, as quais serão detalhadas a seguir, da base para o topo.

4.2.1 Camada Fábrica

Esta camada é constituída pelo conjunto de recursos que fazem parte da grade, onde o acesso distribuído é mediado pelos protocolos da mesma. Como mostrado na Figura 4.2, a camada Fábrica é modelada através do lugar *Recursos da Grade*. As fichas *1'(1,"M1")++ 1'(2,"M2")++1'(3,"M3")++1'(4,"M4")++ ...++1'(15,"M15")* nesse lugar representam as quinze máquinas, que são os recursos disponibilizados no ambiente modelado.

4.2.2 Camada *Middleware*

Esta camada provê o acesso unificado e seguro aos recursos remotos, podendo-se utilizar diferentes *middlewares* de grade. Nesta proposta, são utilizados os protocolos do GT4 e o provedor de informações Ganglia, o qual é usado para coletar automaticamente as informações sobre os recursos da grade.

Na Figura 4.2, a *Middleware* é representada pela subpágina que engloba o lugar *Repositório com as Informações dos Recursos em XML* e a transição de substituição

[2]http://wiki.daimi.au.dk/cpntools/cpntools.wiki

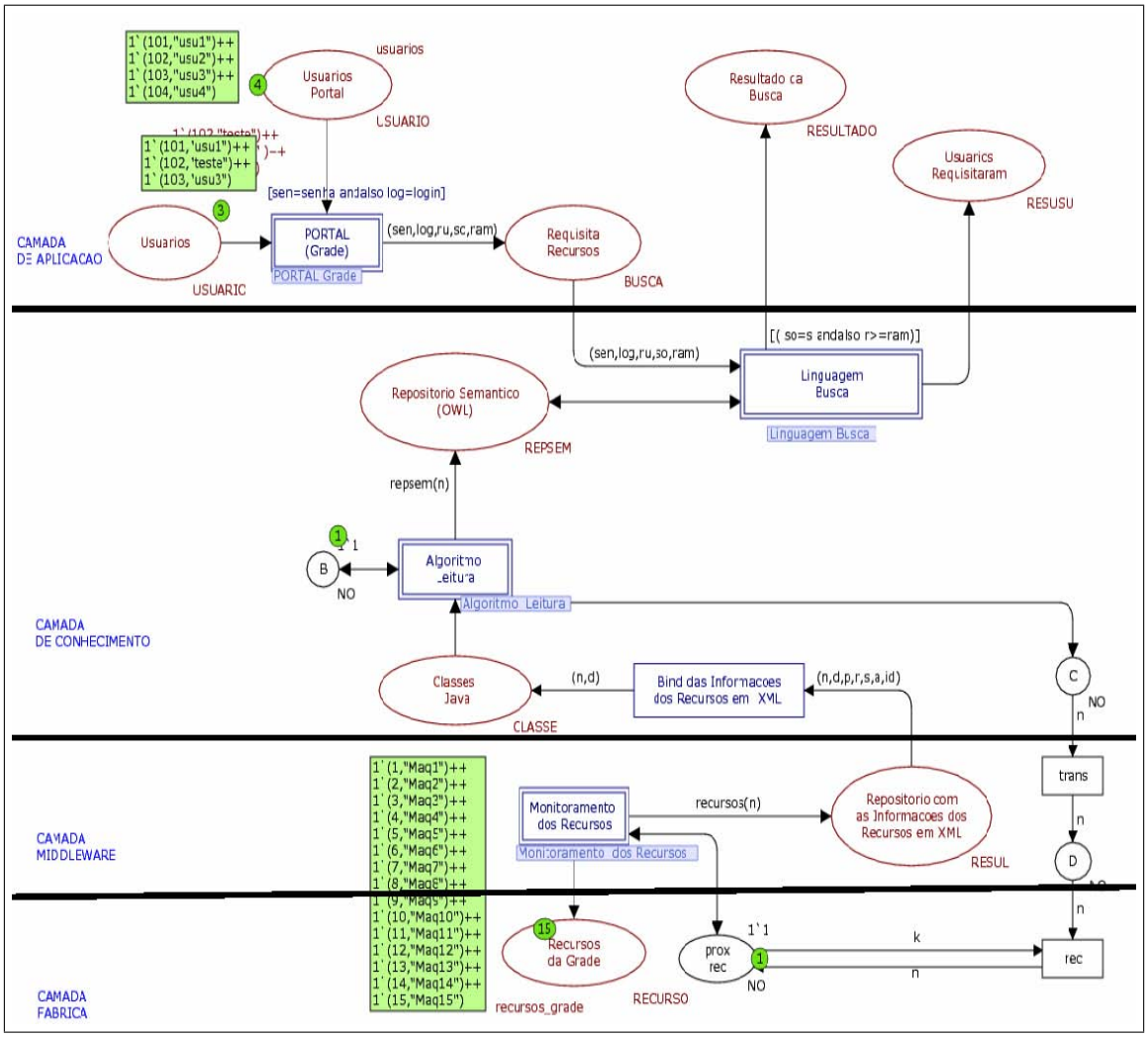


Figura 4.2: Modelagem em RPCs – *Framework* Semântico para Descoberta e Seleção dos Recursos em Grade

Monitoramento dos Recursos. Através da transição de substituição *Monitoramento dos Recursos* (detalhada na subpágina mostrada na Figura 4.3), o lugar *Provedor de Informações* captura as informações dos recursos do lugar *MDS*. Este último fornece as informações dos recursos, os quais constam na camada Fábrica. De acordo com as informações capturadas, o Ganglia as envia para o *Repositório com as Informações dos Recursos em XML*, a serem utilizados pela camada de Conhecimento.

4.2.3 Camada de Conhecimento

A camada de Conhecimento provê a descoberta e seleção de recursos computacionais de forma semântica. Esta camada (Figura 4.2) é representada pela subpágina que contém os lugares *Java Classes* e *Repositório Semântico* (*OWL*), a

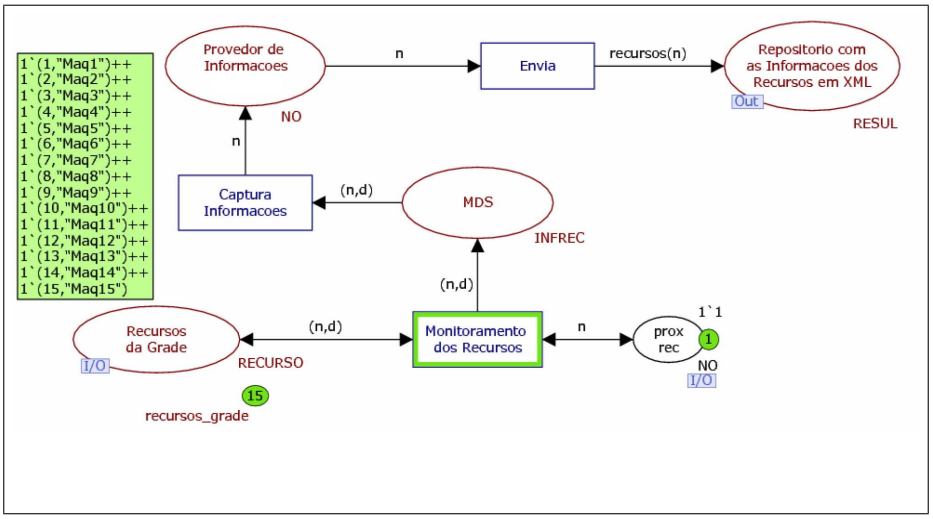


Figura 4.3: Monitoramento dos Recursos – Repositório com as Informações dos Recursos em XML

transição *Bind das Informações dos Recursos em XML* e as transições de substituição *Algoritmo de Leitura* e *Linguagem Busca*. Essa camada captura o *Repositório com as Informações dos Recursos da Grade em XML*(Figura 4.2), processando o mesmo numa linguagem mais expressiva, no formato OWL, para fornecer mais resultados na descoberta dos recursos.

A transição *Bind das Informações dos Recursos em XML* (Figura 4.2) vincula as informações dos recursos em XML para as *Classes Java*. Através da transição de substituição *Algoritmo Leitura*[3] (detalhada na Figura. 4.4), esse algoritmo provê a leitura do *Repositório com as Informações dos Recursos em XML* para instanciar com as *Classes Java*, que permitem salvar no Repositório Semântico o *Template da Ontologia* instanciado no formato OWL, utilizando-se o *Mecanismo de Inferência* para fornecer mais resultados ao usuário de suas buscas por recursos, através da correspondência dos últimos.

4.2.4 Camada de Aplicação

Na camada de Aplicação é desenvolvida uma interface em *Web portlet*, usando a linguagem de busca para prover aos usuários a busca dos recursos da grade e possibilitando a seleção dos mesmos. A busca por esses recursos se utiliza do *Repositório Semântico* (*OWL*) (Figura. 4.4), através da correspondência dos mesmos. Como mostrado na Figura 4.2, essa camada é composta pelos lugares

[3]Utiliza o *framework*-Jena
http://jena.sourceforge.net/index.html

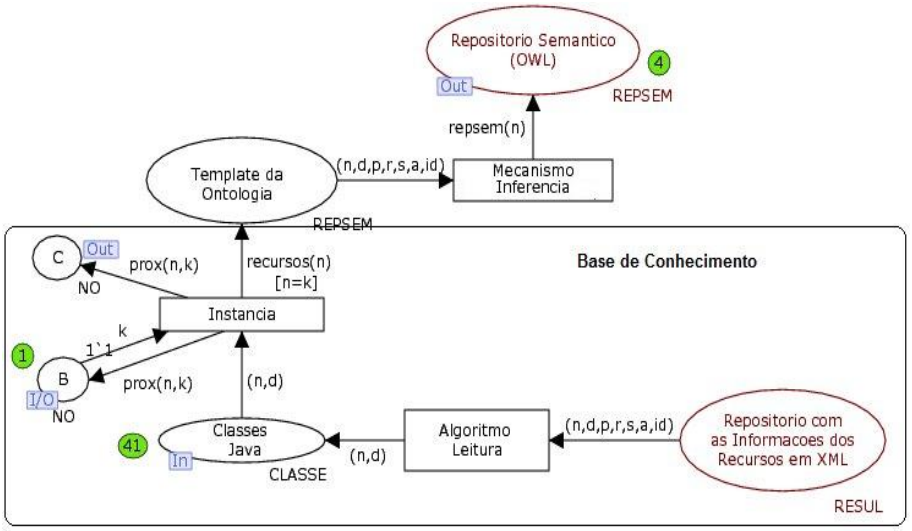


Figura 4.4: Algoritmo de leitura – A Construção do Repositório Semântico (OWL)

Usuários, Usuários Portal e *Requisita Recursos*, transição de substituição *Portal Grade*, bem como por fichas onde consta o cadastro dos usuários, as quais fazem parte do lugar *Usuários Portal*.

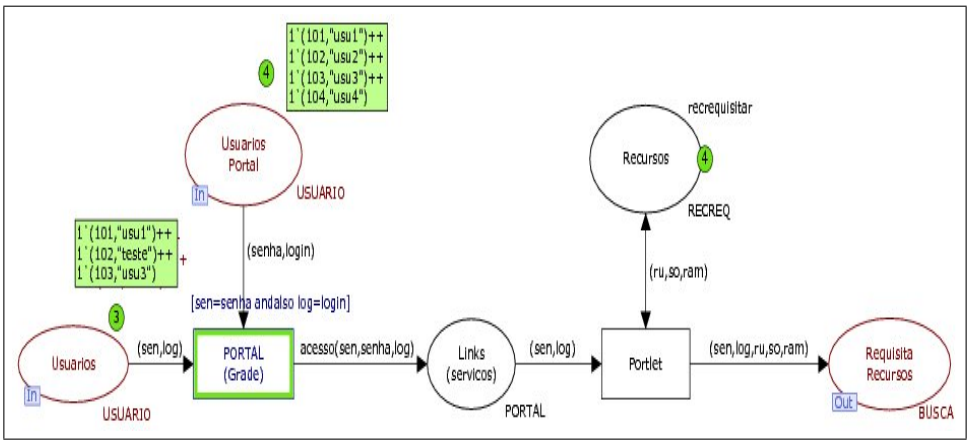


Figura 4.5: Transição de substituição *Portal Grade*– Acesso a interface *Web portlet*

Através da transição de substituição *Portal Grade* (Figura 4.2), detalhada na Figura 4.5, os usuários terão acesso ao *Portlet*, que oferece serviços de aplicação habilitados para *Web*, onde eles podem descobrir, invocar e selecionar os recursos necessários de forma semântica. O acesso a essas funcionalidades depende de o usuário realizar seu *login* junto à página principal do portal. Para que o *login* seja efetivado, o usuário deve estar previamente cadastrado no serviço, associado à grade, bem como ter sido aprovado pela administração do serviço. Após o *login* do usuário no portal, uma seção é mantida para o mesmo, com o objetivo de oferecer o acesso ao serviço.

4.3 Simulação e Análise do Modelo Proposto

Foram realizadas várias simulações com as seguintes características do modelo:

- **Consulta 1: usuário *101usu1***

  Sistema Operacional (SO)=AIX
  memória RAM (disponível) RAM>= 800(MB)

  Resultados: **usuário *101usu1***

  Máquina MQ3

  Sistema Operacional (SO)=AIX

  memória RAM= 1456MB

  Máquina MQ2

  Sistema Operacional (SO)=UNIX

  memória RAM= 1204MB

  O usuário *101usu1* obteve como resposta as máquinas MQ2 e MQ3 (marcação final do lugar *Resultado da Busca)*, como mostrado na Figura 4.6. A MQ3 possui as características requisitadas, e a máquina semelhante à solicitação, a MQ2, possui características equivalentes às mesmas.

- **Consulta 2: usuário *103usu3***

  Sistema Operacional (SO) = Linux
  Memória RAM (disponível) >= 512(MB)

  Resultados: **usuário *103usu3***

  Máquina MQ1

  Sistema Operacional (SO)=Linux

  memória RAM= 512MB

  Máquina MQ2

  Sistema Operacional (SO)=UNIX

  memória RAM= 1204MB

  Máquina MQ3

  Sistema Operacional (SO)=AIX

  memória RAM= 1456MB

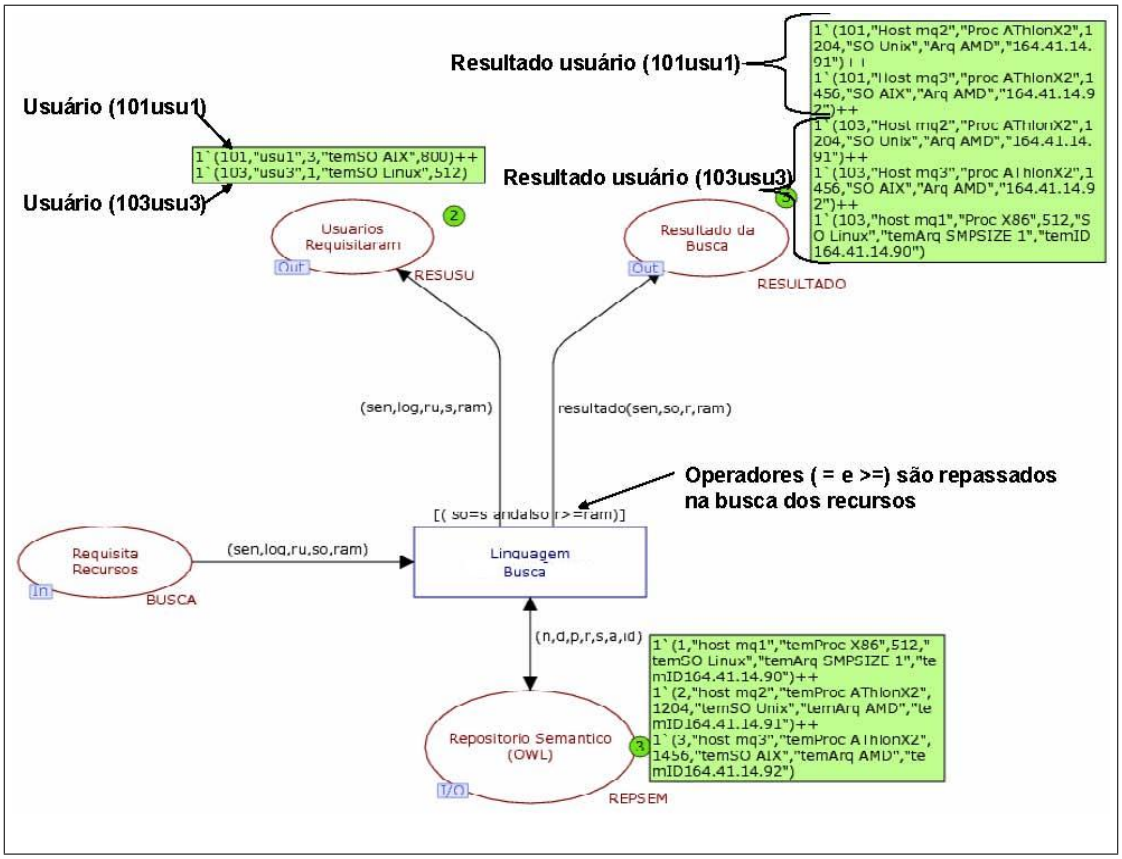


Figura 4.6: Subpágina relativa a transição de substituição *Linguagem Busca* – Resultados da descoberta dos recursos, baseada semântica)

O usuário *103usu3* obteve como resultado as máquinas MQ1, MQ2 e MQ3 (marcação final do lugar *Resultado da Busca)*, como mostrado na Figura 4.6: a máquina MQ1, que possui as características solicitadas, e as máquinas semelhantes às requisitadas MQ2 e MQ3.

Na Figura 4.6, é apresentado o resultado desta simulação. As marcações dos respectivos lugares *Usuários Requisitaram* e *Resultado da Busca* representam a solicitação de recursos e os resultados semânticos. Como ainda se pode ver na citada figura, na transição *Linguagem Busca* se faz o uso da linguagem de busca para consultar o *Repositório Semântico* (*OWL*), fornecendo mais resultados relativos aos recursos solicitados pelo usuário.

Das solicitações dos usuários (*101usu1* e *103usu3*) (Figura 4.6), obtiveram-se resultados semânticos, haja vista que as máquinas consideradas também podem estar aptas para utilização pelo usuário, pois têm as características semânticas equivalentes dos recursos solicitados.

4.4 Conclusão

Dado o modelo de grade semântica proposto (Figura 4.2), várias simulações foram realizadas no objetivo de se observarem a descoberta e a seleção dos recursos da grade. Após a realização das diversas simulações do modelo com os usuários, podemos comprovar a eficiência do tratamento semântico, uma vez que não só a quantidade de recursos retornada é maior, mas também a qualidade da consulta, já que as máquinas podem estar aptas para utilização pelo usuário, pois têm as mesmas características semânticas equivalentes da máquina que foi requerida.

Caso os usuários façam uma busca sintática, onde as mesmas não são baseadas em semântica, mas em simetria, com as seguintes características, ocorrerá isto:

- **Consulta 1: usuário *101usu1***

  Sistema Operacional (SO)=AIX
  memória RAM (disponível) RAM>= 800(MB)

  Resultado: **usuário *101usu1***

  Máquina MQ3

  Sistema Operacional (SO)=AIX

  memória RAM= 1456MB

  O usuário *101usu1* obtém somente como resposta a máquina MQ3 possui as características requisitadas.

- **Consulta 2: usuário *103usu3***

  Sistema Operacional (SO) = Linux
  Memória RAM (disponível) >= 512(MB)

  Resultado: **usuário *103usu3***

  Máquina MQ1

  Sistema Operacional (SO)=Linux

  memória RAM= 512MB

  O usuário *103usu3* obtém somente como resultado a máquina MQ1.

Já na busca semântica, como mostrado na Figura 4.6, são exibidos os recursos semanticamente semelhantes, ou seja, os mais parecidos com o que foi solicitado pelo

usuário. Para isso, é feita uma verificação no *Repositório Semântico* (*OWL*) com a finalidade de descobrir mais resultados dos recursos.

Dos resultados obtidos na simulação, conclui-se que é interessante oferecer a descoberta semântica no ambiente de grade computacional, de modo a simplificar o processo de correspondência de recursos, fornecendo resultados mais flexíveis e extensíveis.

Capítulo 5

Implementação do *Framework*

5.1 Aspectos da Implementação do *Framework*

Nesta seção, faz-se uma abordagem dos aspectos da implementação do *framework* semântico que já foram descritos na Seção 4.2 e ilustrados na Figura 4.2. É importante salientar que esse *framework* tem como foco a descoberta e seleção dos recursos, não abordando escalonamento e execução de tarefas. Como apresentado no Capítulo 4, o *framework* é composto de quatro camadas (Fábrica, *Middleware*, Conhecimento e Aplicação).

5.1.1 Camada *Middleware*

Nesta camada, é provido o acesso aos recursos remotos, podendo-se utilizar diferentes *middlewares* de grade. Neste trabalho, mais exatamente nessa camada, foi utilizado o Sistema de Monitoração e Descoberta (MDS) do conjunto de ferramentas do Globus *Toolkit* (Versão 4.2.0)[1], o qual foi instalado em quatro máquinas no Laboratório de Instrumentação e Computação dos departamentos de Física e Teleinformática da Universidade de Federal do Ceará (UFC).

Foi utilizado também o provedor de informação Ganglia (versão 3.1.1)[2], que é um *software* livre, sendo ainda descrito na linguagem de programação *Practical Extraction And Report Language* (PERL). Mais detalhes a respeito do provedor de informação foram apresentados na Seção 2.1.5. A informação de saída do Ganglia, que é publicada no MDS4, está determinada pelo *Grid Laboratory Uniform*

[1] http://www.globus.org/toolkit/downloads/
[2] http://ganglia.info/?tag=3-1-1

Environment (GLUE) *Schema* (um modelo abstrato para recursos e mapeamentos da grade para esquemas concretos que podem ser utilizados nos Serviços de Informação da Grade) (ANDREOZZI, 2009). O GLUE *Schema* é um padrão que vem sendo usado por ferramentas como Ganglia e o Globus (SCHOPF et al., 2006).

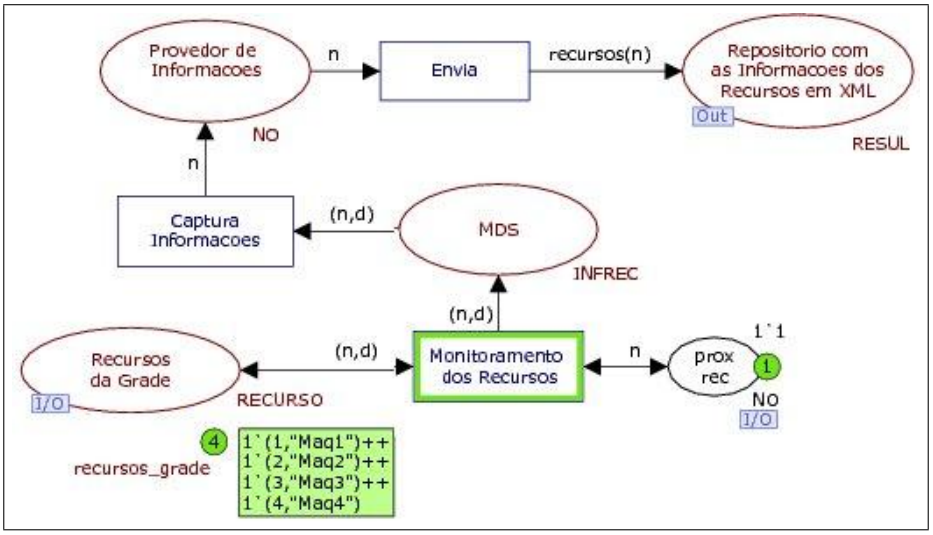


Figura 5.1: Monitoramento dos Recursos (MDS4/Ganglia)

Na camada *Middleware*, os dados sobre os recursos da grade capturados pelo provedor de informações Ganglia são armazenados num Repositório com as Informações dos Recursos em XML (Figura 5.1), como mostrado na Figura 5.2, que apresenta um trecho de código das informações dos recursos em XML, o qual mostra os *METRIC NAMES* do (1) *HOST NAME* aparecida.ufc.br: (2) *mem_free* (quantidade de memória RAM disponível); (3) *mem_total* (quantidade de memória RAM total); e (4) *os_release* (versão do sistema operacional), além de outros recursos disponíveis desse *host* e de outros *hosts*, estando todos armazenados no Repositório de Recursos, que foi instanciado pelo MDS4, utilizando o GLUE/Ganglia.

5.1.2 Camada Conhecimento

Na Figura 5.3 estão representados os módulos principais implementados na camada de Conhecimento, a qual está dividida em duas partes principais: *Template* da Ontologia e Base de Conhecimento, as quais formam o Repositório Semântico (OWL).

```
    <GANGLIA_XML VERSION="3.1.1" SOURCE="gmond">
    <CLUSTER NAME="aparecida" LOCALTIME="1256050830" OWNER="unspecified" LATLONG="unspecified" URL="unspecified">
(1) <HOST NAME="aparecida.ufc.br" IP="10.1.1.147" REPORTED="1256050824" TN="5" TMAX="20" DMAX="0" LOCATION="unspecifie
    GMOND_STARTED="1256048544">
(2) <METRIC NAME="mem_free" VAL="10168.000" TYPE="float" UNITS="KB" TN="5" TMAX="180" DMAX="0" SLOPE="both">
    <EXTRA_DATA>
    <EXTRA_ELEMENT NAME="GROUP" VAL="memory"/>
    <EXTRA_ELEMENT NAME="DESC" VAL="Amount of available memory"/>
    <EXTRA_ELEMENT NAME="TITLE" VAL="Free Memory"/>
    </EXTRA_DATA>
    </METRIC>
(3) <METRIC NAME="mem_total" VAL="508576.000" TYPE="float" UNITS="KB" TN="1085" TMAX="1200" DMAX="0" SLOPE="zero">
    <EXTRA_DATA>
    <EXTRA_ELEMENT NAME="GROUP" VAL="memory"/>
    <EXTRA_ELEMENT NAME="DESC" VAL="Total amount of memory displayed in KBs"/>
    <EXTRA_ELEMENT NAME="TITLE" VAL="Memory Total"/>
    </EXTRA_DATA>
    </METRIC>
(4) <METRIC NAME="os_release" VAL="2.6.28-15-generic" TYPE="string" UNITS="" TN="1085" TMAX="1200" DMAX="0" SLOPE="zero'
    <EXTRA_DATA>
    <EXTRA_ELEMENT NAME="GROUP" VAL="system"/>
    <EXTRA_ELEMENT NAME="DESC" VAL="Operating system release date"/>
    <EXTRA_ELEMENT NAME="TITLE" VAL="Operating System Release"/>
    </EXTRA_DATA>
    </METRIC>
    ←other machine resources →
    <HOST NAME="carcara.ufc.br" IP="10.1.1.147" REPORTED="1256050824" TN="5" TMAX="20" DMAX="0" LOCATION="unspecified"
    GMOND_STARTED="1256048544">
```

Figura 5.2: Trecho de código das informações dos recursos em XML

Template da Ontologia

O *template* é definido como a ontologia dos recursos da grade que prevêem a hierarquia de conceitos, nos quais algumas propriedades definem suas características. Para a construção da ontologia, seguiram-se alguns passos:

Num primeiro momento, avaliou-se o modelo de informação GLUE *Schema* (ANDREOZZI, 2009), que visa a padronizar a representação de informações sobre os recursos de uma grade computacional. Esse modelo representa sistemas computacionais, descrevendo seus elementos (*Computing Element*, *Cluster*, *SubCluster*, *Host*, etc) e as relações entre eles, como mostrado na Figura 5.4.

Porém, dos elementos do GLUE *Schema* serão utilizados somente aqueles que se conseguem obter com o Ganglia (versão 3.1.1). São apresentadas, a seguir, algumas das classes do GLUE *Schema* relacionadas com o Ganglia:

- *Cluster*: é um contêiner que agrupa *subclusters* ou nós. Os elementos do *subclusters* representam coleções 'homogêneas' de nós computacionais, enquanto os nós representam elos únicos, tais como os nós *head*, ou nós individuais de computação. Um *cluster* pode ser referenciado por mais de

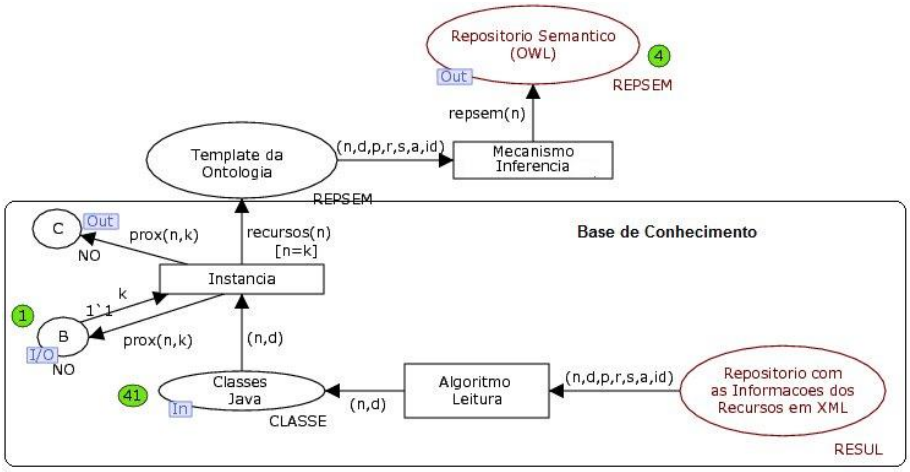


Figura 5.3: *Template* da Ontologia e a Base de Conhecimento, os quais formam o Repositório Semântico (OWL)

um elemento da computação;

- *Host*: representa um elemento físico de computação. Esse elemento caracteriza a configuração física de um nó de computação, como:
  - *ArchitectureType*: Tipo de arquitetura;
  - *Operating System*: Sistema Operacional;
  - *Processor*: Processador;
  - *MainMemory*: Memória Principal;
  - *Load*: Informações sobre a carga da máquina;
  - *NetworkAdapter*: Adaptador de Rede.

Num segundo momento, para axiomatização da ontologia, seguiram-se alguns critérios na sua construção, por exemplo clareza, completude, coerência, minimização da distância semântica, princípio de distinção ontológica e padronização, os quais foram descritos na Seção 2.3.2, para que a ontologia apresente confiabilidade, seja reutilizável e possibilite compartilhamento.

A ontologia proposta foi desenvolvida, utilizando-se a linguagem OWL, a qual foi descrita na Seção 2.3.5. Optou-se pelo seu uso, devido ao fato de apresentar todas as funcionalidades de outras linguagens, incluindo o fato de ser um padrão de linguagem para criação de ontologias reconhecido pela W3C.

Por último, para edição do *template* da ontologia dos recursos da grade,

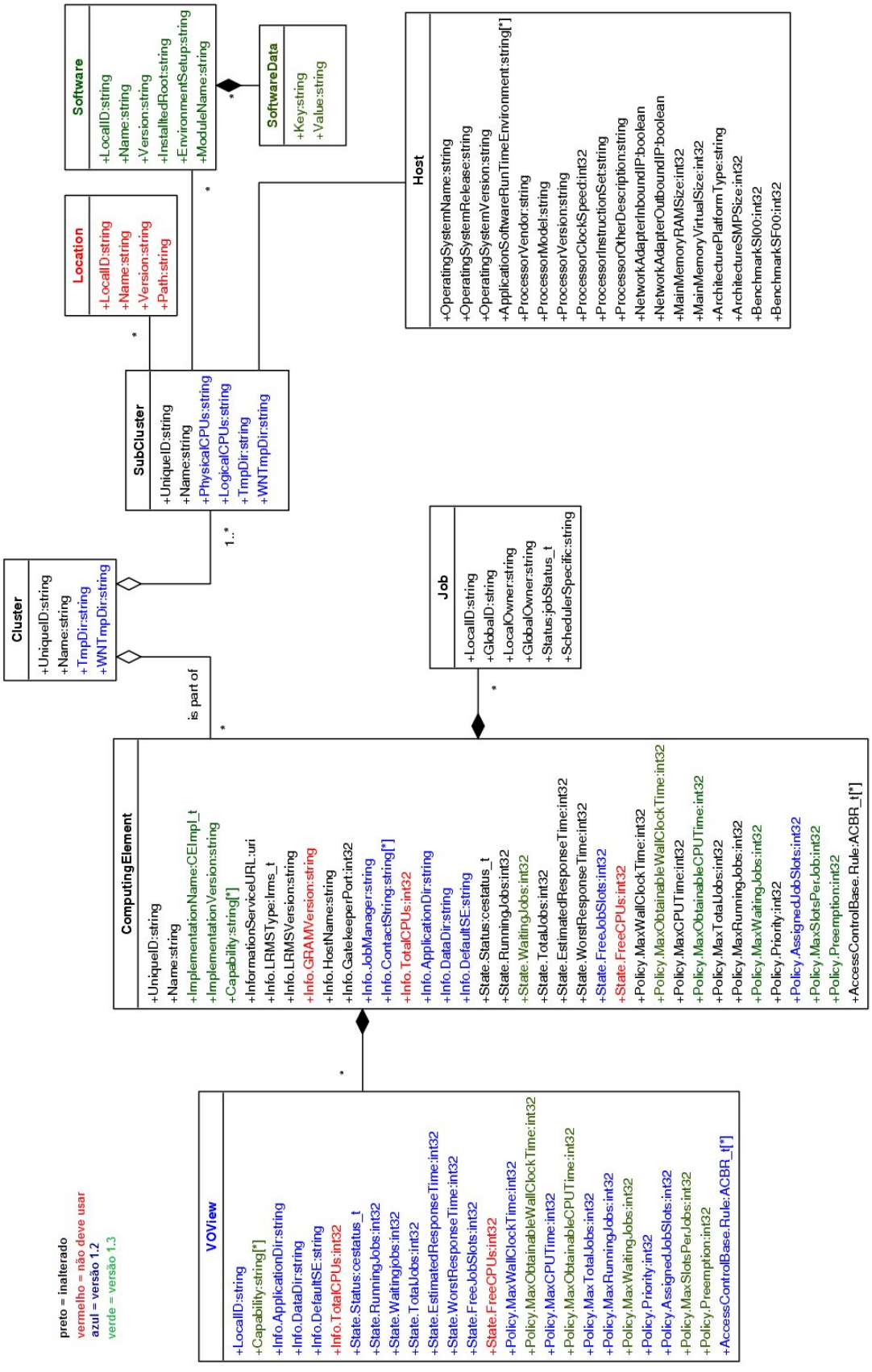


Figura 5.4: Diagrama de Classe UML do Elemento de Computação (ANDREOZZI, 2009)

foi utilizado o editor Protégé-OWL versão 4.0 (*Build* 114)[3], definindo algumas das classes do GLUE *Schema* relacionados com o Ganglia, como mostrado na Figura 5.5. Optou-se pela utilização desse editor, o qual foi descrito na Seção 2.3.6 primeiramente, por ser um *software* livre e apresentar maior quantidade de *plugins* disponíveis.

Sendo assim, caracterizadas todas as classes, subclasses, propriedades, atributos e instâncias, componentes esses da construção da ontologia, conforme a Seção 2.3.3, um conjunto preliminar de requisitos especificamente voltados à ontologia dos recursos da grade foi definido, fornecendo mais significados semânticos. As propriedades das classes capturam as diferentes variáveis, relativas ao domínio da ontologia, sendo em OWL, caracterizadas como *Object properties* e *Data properties*, as quais são necessárias à definição do *domain* (domínio) e o *range* (valor) da propriedade.

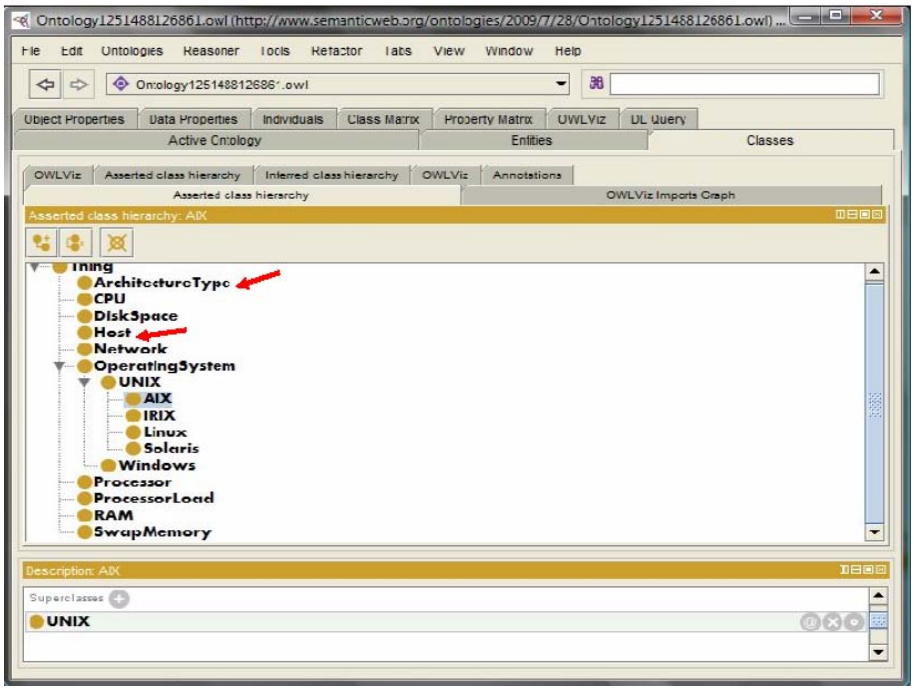


Figura 5.5: Hierarquia de classes *asserted* (assertiva) – Protégé

- *Object properties* (Propriedades de objeto) têm o papel de relacionar uma classe à outra. Por exemplo, as classes *Host* e *ArchitectureType* (Figura 5.5) foram relacionadas com a propriedade *hasArchitectureType* (Figura 5.6). O *domain* e o *range* dessa propriedade são, respectivamente, *Host* e *ArchitectureType*;

[3]http://protege.stanford.edu/download/registered.html

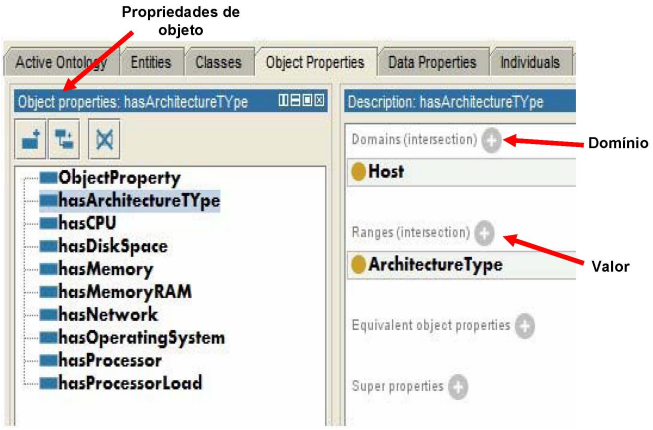


Figura 5.6: *Object properties* (Propriedades de objeto) – Protégé

- *Data properties* (Propriedades de dados) se diferenciam da *Object properties* por utilizarem uma variável para representar qualquer coisa no domínio abordado. Nesse tipo de propriedade, também é necessário definir o *domain* ao qual ela pertence, bem como seu *range*, que não mais será uma classe, mas um elemento do tipo *string*, *boolean*, *int*, dentre outros. Por exemplo, a propriedade *machine_type* (Figura 5.7) tem por *domain ArchitectureType*, sendo o *range* do tipo *string*.

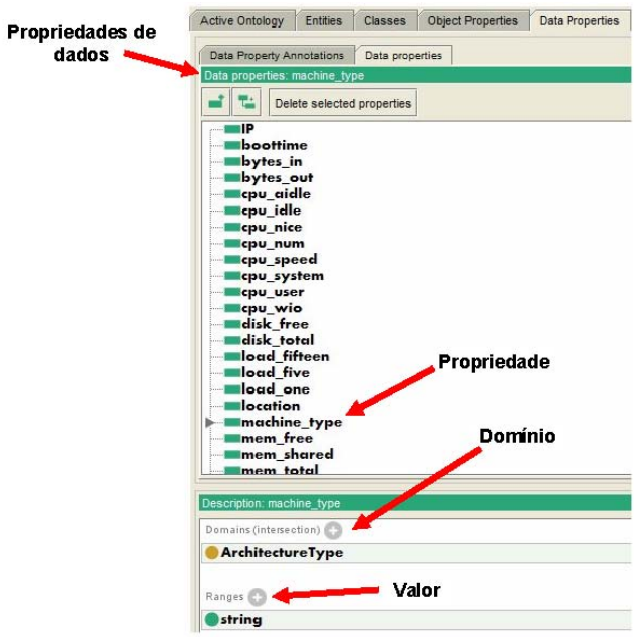


Figura 5.7: *Data properties* (Propriedades de dados) – Protégé

Como mostrado na Figura 5.8, *template*[4] foi definido como a ontologia dos recursos da grade que prevêem a hierarquia de classes. Para a sua visualização, foi utilizado o OWLViz[5], o qual permite que as hierarquias de classe numa ontologia OWL possam ser vistas e navegadas incrementalmente.

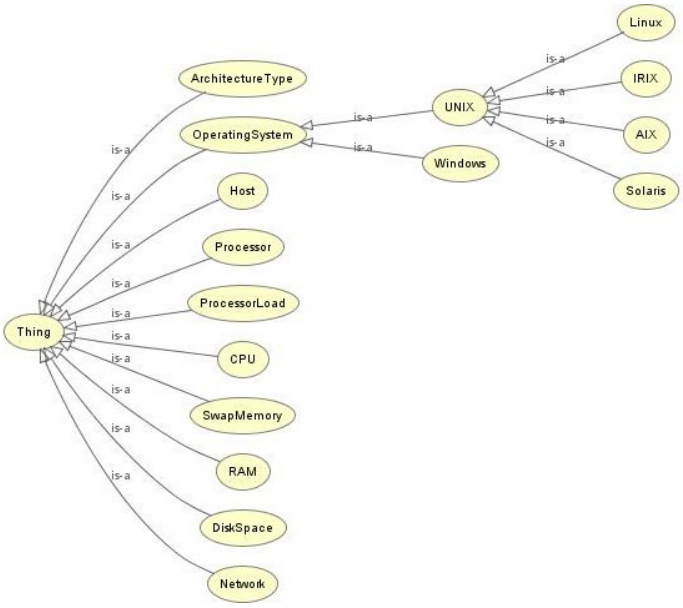


Figura 5.8: *Template* da ontologia – OWLViz

Essa ontologia é formada por classes e propriedades que correspondem aos valores das instâncias e propriedades constituintes do Repositório Semântico, não sendo ela definitiva e, por isso, podendo sofrer evoluções, definindo um mínimo de classes e propriedades providos pelo serviço de informação de recursos MDS4/Ganglia para cada um dos *hosts* da grade.

Base de Conhecimento

De modo a instanciar o *template* da ontologia, primeiramente, os recursos da grade definidos no formato em XML foram vinculados para classes Java, utilizando o *Document Object Model* (DOM). O DOM[6] é uma interface de programação, baseada no modelo de objetos, que permite a manipulação e transformação de documentos em XML. Em seguida, para criar dinamicamente uma base de conhecimento dos recursos da grade, foi utilizado o *framework* Jena (versão 2.6.2)[7], o qual disponibiliza uma API que permite uma aplicação para manipular ontologias. Através do Jena,

[4]http://www.semanticweb.org/ontologies/2009/7/28/Ontology1251488126861.owl
[5]OWLViz é um OWL *plugin* para Protégé – http://protegewiki.stanford.edu/index.php/OWLViz
[6]http://www.w3.org/DOM/
[7]http://jena.sourceforge.net/

foi desenvolvido um algoritmo para fazer a leitura dos recursos do Repositório com as Informações em XML, a fim de instanciar com as classes Java, que permitem salvar no Repositório Semântico o *template* da ontologia instanciada no formato OWL. O mecanismo de inferência Pellet[8] foi instanciado com o *framework* Jena, a fim de interagir com a base de conhecimento, para a recuperação semântica da informação dos recursos da grade.

O Jena oferece também suporte ao *Sistema de Gerenciamento de Banco de Dados* (SGBD), MySQL. O MySQL provê o armazenamento escalável e consulta de bases de dados, usando SQL para uso em aplicações *standalone*, J2EE e outras estruturas de aplicativo.

O Repositório Semântico foi instanciado por um modelo persistente em MySQL, usando Jena. Para criar esse modelo persistente, foram necessários três passos:

- Carregar o driver do BD utilizado:

  (*String DB_ URL = "jdbc:mysql://localhost/jena"*);

- Estabelecer a conexão com o BD:

  IDBConnection conn = new DBConnection();

- Instanciar a *classe ModelMaker* e invocar *ModelMaker*:

  OntModelSpec spec = new OntModelSpec(OntModelSpec.OWL_MEM); spec.setModelMaker(maker); String uri = ”http://www.semanticweb.org/ontologies/2009/7/28/Ontology1251488126861.owl#”; Model base = maker.createModel(uri); OntModel m = ModelFactory.createOntologyModel(spec,base); m.read(uri);

5.1.3 Camada de Aplicação

Finalmente, a camada de Aplicação é a que fica no topo do *framework*, provendo aos usuários a busca dos recursos da grade e possibilitando a seleção dos mesmos. A interface *Web* permite que o requisitante de recursos submeta consultas e obtenha o resultado semântico de suas solicitações através do serviço de ontologia.

Essa interface foi desenvolvida em *JavaServer Pages* (JSP). O JSP (SANTOS; JORGE, 2009) é uma tecnologia utilizada no desenvolvimento de aplicações para

[8] http://clarkparsia.com/pellet/

Web, similar às tecnologias *Active Server Pages* (ASP) da Microsoft ou PHP. Por ser baseada na linguagem de programação Java, tem a vantagem da portabilidade de plataforma, que permite a sua execução em diversos sistemas operacionais, como o Windows, Unix e Linux. Essa tecnologia permite ao desenvolvedor de páginas para *Internet* produzir aplicações que acesse banco de dados. Uma página criada com a tecnologia JSP, após instalada num servidor de aplicação compatível com a tecnologia Java EE, é transformada num *Servlet*.

Foi desenvolvido um algoritmo, fazendo uso da linguagem de busca SPARQL, a fim de enviar para o usuário o resultado de suas requisições, utilizando o banco de dados MySQL, ou seja, Repositório Semântico. Sempre que uma solicitação do usuário chega ao algoritmo, cria-se uma consulta dinâmica de acordo com as informações fornecidas pelo usuário. Realizada a correspondência de recursos, recupera-se o nome da classe de instâncias. Depois de se ter o nome da classe, cumpre recuperar classes equivalentes, se houver. Agora se obtém a lista de todas as instâncias da classe especificada, bem como de classe equivalente. Essas instâncias são recuperadas do Repositório Semântico e o resultado é devolvido ao usuário. Optou-se pelo uso da linguagem de busca SPARQL, por seguir um padrão recomendado pelo W3C e possuir também uma sintaxe semelhante à da linguagem SQL, como mostrada na Seção 2.3.8.

5.2 Estudo de Caso

Neste trabalho, foi proposta uma abordagem flexível e extensível para a realização da descoberta de recursos da grade, utilizando-se a correspondência de recursos, baseada em ontologia.

Para se avaliar a proposta do serviço de descoberta de recursos de grade baseado em semântica, foram realizadas algumas consultas, tendo como parâmetros as características dos recursos computacionais. Como estudo de caso deste trabalho, foram executadas algumas consultas para as seguintes características de recursos: arquitetura do processador, carga da CPU (usuário), quantidade de memória RAM total, quantidade de memória RAM disponível e nome do sistema operacional. No experimento, foram incluídas consulta simples (ou seja, apenas uma característica do recurso) e consultas múltiplas (ou seja, mais de uma característica do recurso).

5.2.1 Ambiente de Teste

Como ambiente de teste, foi utilizado o Laboratório de Instrumentação e Computação dos departamentos de Física e Teleinformática da UFC. Na Figura 5.9 é ilustrado o ambiente de teste, enquanto na Tabela 5.1 são apresentados arquitetura do processador, carga da CPU (usuário), quantidade de memória RAM total, quantidade de memória RAM disponível e nome do sistema operacional. Essas informações foram obtidas automaticamente pelo Ganglia, num determinado instante.

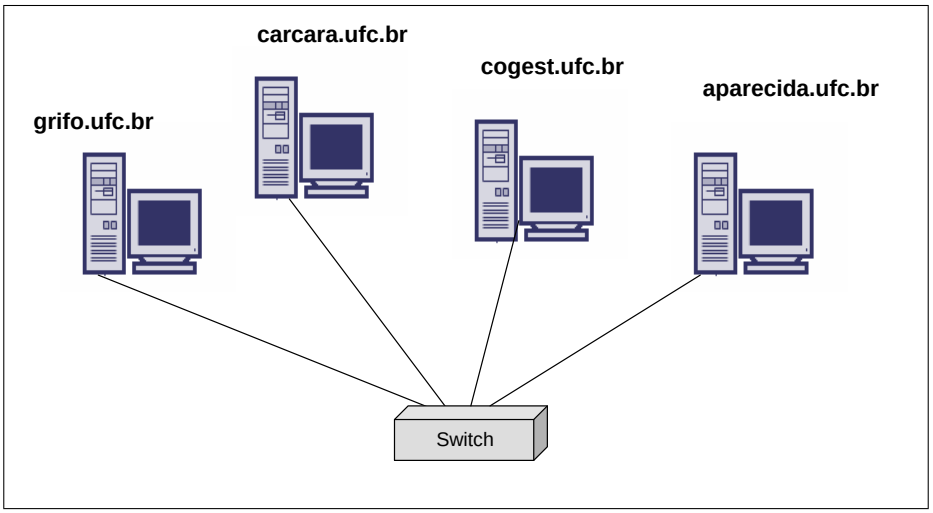


Figura 5.9: Ambiente de Teste – UFC

Tabela 5.1: *Ambiente de Grade Experimental*

Host	Arquitetura	Carga CPU	Memória Total	Memória Disponível	SO
aparecida.ufc.br	x86	1%	2052	1245	Linux
grifo.ufc.br	x86	14%	1026	587	Linux
carcara.ufc.br	x86	5%	508	12	Linux
cogest.ufc.br	x86	4%	1526	784	Solaris

Instalação do Ambiente

Para que o ambiente de grade pudesse funcionar, foram necessárias as instalações e configurações de algumas ferramentas nas máquinas do laboratório de pesquisa. Por exemplo, foi feita a instalação do Globus *Toolkit* (versão 4.2.0)[9] e foram criados certificados de segurança para cada uma das máquinas. Depois foi instalado o SGBD MySQL (versão 5.0)[10], pois é necessário que o mesmo esteja em execução para

[9]http://www.globus.org/toolkit/docs/4.2/4.2.0/admin/quickstart/
[10]http://www.mysql.com/

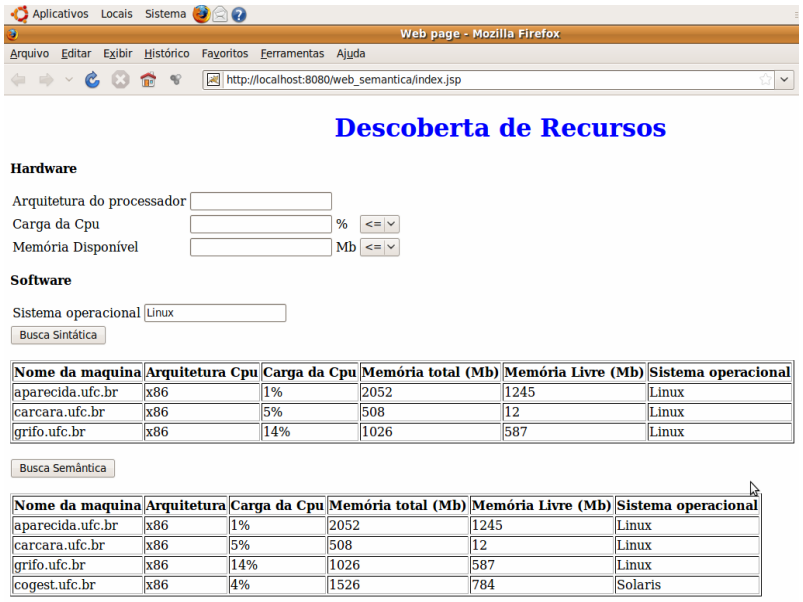
realizar algumas tarefas na grade. Posteriormente, foi instalado também o *container Web* Tomcat (versão 6.0.20)[11], responsável por fornecer, via *Web*, as informações coletadas sobre os recursos das máquinas, através da interface gráfica conhecida por *WebMDS*. Finalmente, o Ganglia (versão 3.1.1)[12] foi instalado e configurado, coletando as informações sobre os recursos das máquinas da grade.

5.2.2 Resultados Experimentais

Nesta seção, apresentam-se alguns resultados de consultas feitas pelos usuários, sendo consideradas duas abordagens de descoberta de recursos: a sintática ou simétrica; e a semântica.

- Na **Consulta 1 (simples)**, o usuário faz uma solicitação com a seguinte característica: recurso - Sistema operacional = Linux. Resultado da busca sintática:as máquinas *aparecida.ufc.br*, *carcara.ufc.br* e *grifo.ufc.br*, como mostrado na Figura 5.10, as quais possuem a característica requisitada.

  Resultado da busca semântica:as máquinas *aparecida.ufc.br*, *carcara.ufc.br*, *grifo.ufc.br* e *cogest.ufc.br*, como mostrado na Figura 5.10. Semelhante à solicitação, a máquina *cogest.ufc.br* possui a característica equivalente à solicitação.



Nome da maquina	Arquitetura Cpu	Carga da Cpu	Memória total (Mb)	Memória Livre (Mb)	Sistema operacional
aparecida.ufc.br	x86	1%	2052	1245	Linux
carcara.ufc.br	x86	5%	508	12	Linux
grifo.ufc.br	x86	14%	1026	587	Linux

Nome da maquina	Arquitetura	Carga da Cpu	Memória total (Mb)	Memória Livre (Mb)	Sistema operacional
aparecida.ufc.br	x86	1%	2052	1245	Linux
carcara.ufc.br	x86	5%	508	12	Linux
grifo.ufc.br	x86	14%	1026	587	Linux
cogest.ufc.br	x86	4%	1526	784	Solaris

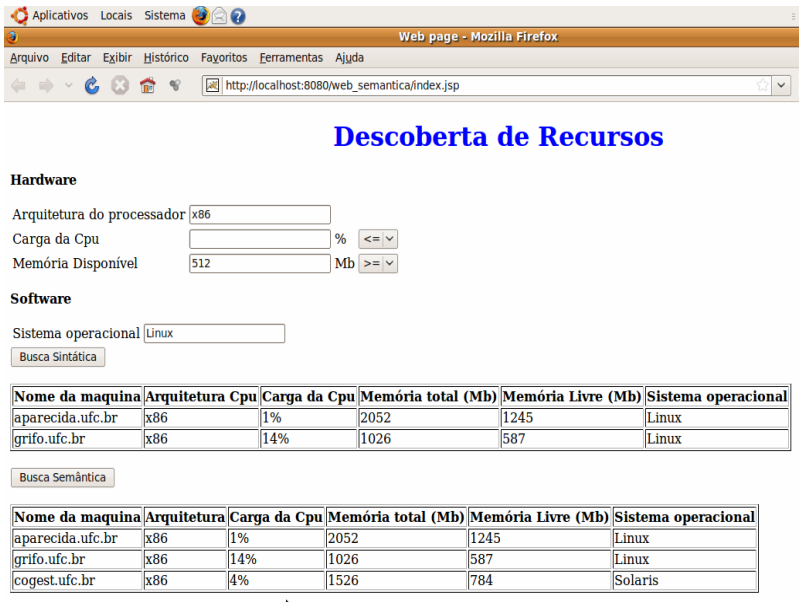
Figura 5.10: Consulta 1 – Buscas Sintática/Semântica

[11] http://tomcat.apache.org/
[12] http://ganglia.info/?tag=3-1-1

- Na **Consulta 2 (múltipla)**, o usuário faz uma solicitação com as seguintes características: recursos - Arquitetura do processador = x86, Memória disponível >= 512 e Sistema operacional = Linux.

  Resultado da busca sintática:as máquinas *aparecida.ufc.br* e *grifo.ufc.br*, como mostrado na Figura 5.11, as quais possuem as características requisitadas.

  Resultado da busca semântica:as máquinas *aparecida.ufc.br*, *grifo.ufc.br* e *cogest.ufc.br*, como mostrado na Figura 5.11. Semelhante à solicitação, a máquina *cogest.ufc.br* possui característica equivalente.



Nome da maquina	Arquitetura Cpu	Carga da Cpu	Memória total (Mb)	Memória Livre (Mb)	Sistema operacional
aparecida.ufc.br	x86	1%	2052	1245	Linux
grifo.ufc.br	x86	14%	1026	587	Linux

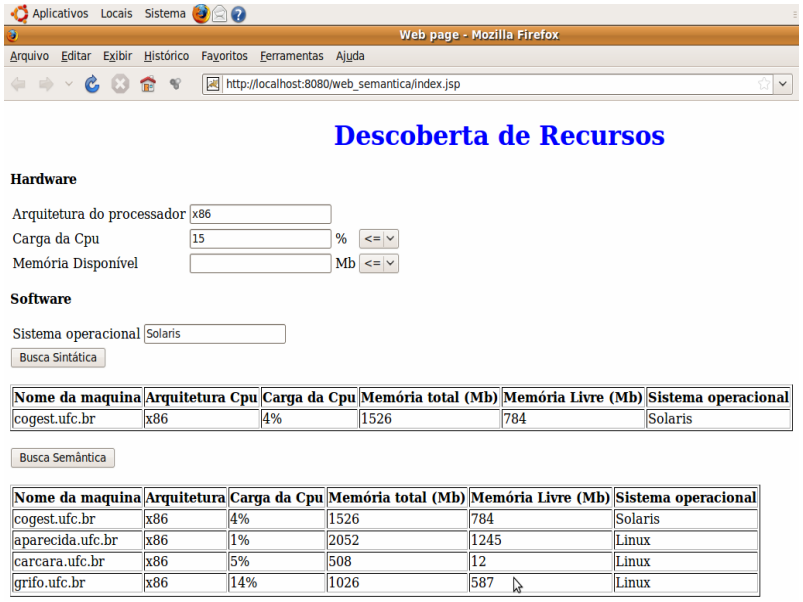
Nome da maquina	Arquitetura	Carga da Cpu	Memória total (Mb)	Memória Livre (Mb)	Sistema operacional
aparecida.ufc.br	x86	1%	2052	1245	Linux
grifo.ufc.br	x86	14%	1026	587	Linux
cogest.ufc.br	x86	4%	1526	784	Solaris

Figura 5.11: Consulta 2 – Buscas Sintática/Semântica

- Na **Consulta 3 (múltipla)**, o usuário faz uma solicitação com as seguintes características: recursos - Arquitetura do processador = x86, Carga da CPU <= 15% e Sistema operacional = Solaris.

  Resultado da busca sintática:Como mostrado na Figura 5.12, a máquina *cogest.ufc.br* possui as características requisitadas.

  Resultado da busca semântica:as máquinas *aparecida.ufc.br*, *carcara.ufc.br*, *grifo.ufc.br* e *cogest.ufc.br*, como mostrado na Figura 5.12. As máquinas *aparecida.ufc.br*, *carcara.ufc.br* e *grifo.ufc.br* são semelhantes à solicitação, possuindo a característica equivalente.



Nome da maquina	Arquitetura Cpu	Carga da Cpu	Memória total (Mb)	Memória Livre (Mb)	Sistema operacional
cogest.ufc.br	x86	4%	1526	784	Solaris

Nome da maquina	Arquitetura	Carga da Cpu	Memória total (Mb)	Memória Livre (Mb)	Sistema operacional
cogest.ufc.br	x86	4%	1526	784	Solaris
aparecida.ufc.br	x86	1%	2052	1245	Linux
carcara.ufc.br	x86	5%	508	12	Linux
grifo.ufc.br	x86	14%	1026	587	Linux

Figura 5.12: Consulta 3 – Buscas Sintática/Semântica

5.2.3 Análise dos Resultados obtidos

Pelos testes realizados, pôde-se comprovar a diferença de resultados obtidos com os dois tipos de busca. No caso da busca semântica, não só a quantidade de recursos retornada é maior, mas também a qualidade da consulta, já que as máquinas consideradas semanticamente equivalentes também podem estar aptas para utilização pelo usuário, pois possuem as mesmas características semânticas da máquina que foi requerida.

Ao se analisarem as solicitações feitas com a busca sintática, verificou-se que a taxa de acerto é bem baixa, chegando a haver um caso de falha, quando o usuário solicita, por exemplo, um sistema operacional, o qual foi escrito incorretamente. Nesse caso, a busca sintática retorna estritamente apenas aquilo que foi digitado pelo usuário. Já na busca semântica, quando o pedido foi escrito incorretamente, a taxa de acerto foi bem acentuada, sendo que para os testes realizados não houve caso de falha. Isso se deve ao fato de o protótipo implementado realizar um tratamento de não-sensibilidade de caixa, o que ignora letras maiúsculas e minúsculas.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho de dissertação forneceu uma visão da grade computacional, bem como das várias abordagens e algoritmos usados para a descoberta dos recursos num ambiente de grade e dos problemas encontrados nessas abordagens. Em seguida, descreveu-se a ontologia dos recursos da grade e implementação duma abordagem para descoberta dos recursos baseada em ontologia como uma solução para os mesmos.

A descoberta por um recurso que atenda a uma solicitação do usuário é um assunto suficientemente importante em ambiente de grade, devido à sua natureza heterogênea, dinâmica e distribuída. Contudo, não há, por enquanto, nenhuma norma comum para descrever recursos da grade. Uma vez que cada *middleware* descreve um recurso à sua própria maneira, as informações recolhidas de diversas fontes tendem a ser semanticamente heterogêneas. Com relação a esse tipo de informação, usamos uma abordagem baseada em semântica, sendo o tema desta dissertação, haja vista que a descoberta dos recursos baseada em semântica provê uma melhoria no gerenciamento dos mesmos.

Nesta pesquisa, foi atingida a interoperabilidade semântica através do desenvolvimento de uma ontologia com as informações dos recursos da grade. A ontologia desenvolvida descreveu as classes dos recursos da grade e as relações entre elas. Em particular, essa abordadem baseada em ontologia é melhor do que a abordadem da descoberta dos recursos baseada em simetria, mostrada da seguinte maneira:

- Descoberta dos recursos baseada em ontologia proporciona uma abordagem

flexível, já que não dependente da sintaxe utilizada para descrição de recursos de diferentes *middlewares*;

- A ontologia fornece um grau de liberdade para o usuário, o qual pode inserir a consulta de acordo com seus conhecimentos, sem ter qualquer informação prévia sobre o exato nome dos recursos disponíveis, nomes que podem variar, dependendo do provedor de recursos;

- A ontologia também dá melhores resultados, por encontrar recursos que têm conceitos similares, e não apenas as mesmas descrições. Assim, ela retorna todos os recursos pertinentes, e não apenas aqueles com palavras-chave de correspondência.

Como trabalho futuro, sugerimos testar a abordagem desta dissertação em Serviços de Informação de outros *middlewares* de grade, uma vez que o Repositório de Recursos foi projetado para usar o formato XML. Outra sugestão é aprimorar as regras semânticas da ontologia utilizada, permitindo a captura de outros recursos de máquina, que não são providos atualmente pelo Ganglia. Para isso, seria necessário o estudo de outros sistemas de monitoramento de recursos da grade. Uma outra forma de permitir a adição de novos conceitos no *template* da ontologia seria possibilitar que o próprio usuário do sistema mapeasse esses novos recursos de máquina. Desse modo, o Repositório Semântico poderia ser construído de modo colaborativo, tendo sua ontologia aumentada na mesma proporção em que a comunidade de usuários do sistema também fosse aumentada.

Apêndice A

Publicação Internacional

Francisca A. P. Pinto, Giovanni C. Barroso, Antônio B. Serra, Mario F. Aguilar, *A Semantic Framework for Resource Discovery Based on Ontology*, The Sixth International Conference on Networking and Services (ICNS' 2010), isbn 978-0-7695-3969-0, Cancun-México, IEEE Computer Society, Março, 2010.

Referências Bibliográficas

ANDREOZZI, S. GLUE Schema Specification version 1.3. Acesso: Dezembro 2009. Disponível em: <http://glueschema.forge.cnaf.infn.it/Spec/V13>.

BAKER, M.; BUYYA, R.; LAFORENZA, D. The Grid: Internacional Efforts in Global Computing. *In Proceedings of the Internacional on Advances in Infrastructure for Eletronic Business Science, and Education on the Internet – (SSGRR)*, Italy, 2000.

BERNERS-LEE, T. Semantic web on xml. Acesso: Novembro 2009. Disponível em: <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, May 2001.

BORST, W. N. *Construction of Engineering Ontologies*. Tese (Doutorado) — Centre for Telematica and Information Technology – University of Twenty, Enschede-HL, 1997.

BRATOSIN, C. et al. A reference model for grid architectures and its analysis. In: _____. Berlin, Heidelberg: Springer-Verlag, 2008. (LCNS, v. 5331), p. 898–913.

CORCHO, O.; FERNáNDEZ-LóPEZ, M.; GóMEZ-PéREZ, A. Methodologies, tools and languages for building ontologies. where is their meeting point? *Data e Knowledge Engineering*, v. 46, 2003.

CZAJKOWSKI, K. et al. A resource management architecture for metacomputing systems. *Workshop on Job Scheduling Strategies for Parallel Processing – (IPPS/SPDP'98)*, 1998.

FIKES, R.; HAYES, P.; HORROCKS, I. OWL-QL - A language for deductive query answering on the semantic web. *Knowledge Systems Laboratory Technical Report*, Stanford University, Stanford, CA, 2003.

FOSTER, I.; KELSSELMAN, C. *The Grid2: Blueprint for a New Computing Infrastructure*. 2th. ed. [S.l.]: Morgan Kaufmann, 2004.

FOSTER, I.; KELSSELMAN, C.; TUECKE, S. The anatomy of the grid: Enabling scalable virtual organization. *Internacional Journal of High Performance Computing Applications*, 2001.

GASSEN, J. B. et al. Uma comparação entre linguagens para consultas sobre ontologias owl. *Seminário de Informática (SEMINFO)*, Rio Grande do Sul, RS, Novembro 2008.

GLOBUS. The role of the Globus Toolkit in the grid ecosystem. Acesso: Dezembro 2009. Disponível em: <Disponível: http://www.globus.org>.

GOBLE, C. Sense and sensibility: Semantic infrastructure for bioinformatics. Harvard, January 2007.

GRUBER, T. R. A translation approach to portable ontology specifications. *Appeared in Knowledge Acquisition – Revised*, April 1993a.

GRUBER, T. R. Grande challenges for ontology design (or is it vente?). Acesso: Novembro 2009. Disponível em: <http://tomgruber.org/writing/index.htm>.

GUARINO, N. Formal ontology and information systems. *In Proceedings of the 1st International Conference on Formal Ontologies in Information Systems – (FOIS'98)*, Trento-Italy, June 1998.

JAISWAL, S. K. *Ontology based Resource Discovery Approach in Grid Environment.* Dissertação (Mestrado) — Thapar University – Patiala, May 2007.

JENSEN, K. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Berlin: Springer-Verlag, 1997.

JENSEN, K.; KRISTENSEN, L. M. *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Berlin Heidelberg: Springer-Verlag, 2009.

KAILASH, S. et al. Semantic resource description for grid. *IEEE/Proceedings of the First Asia Internacional Conference on Modelling and Simulation (AMS'07)*, 2007.

NASSIF, L. N.; NOGUEIRA, J. M. Grades computacionais: Uma tecnologia para compartilhamento de recursos em rede. *Revista Fonte*, v. 6, 2007.

NOY, N. F.; MCGUINESS, D. L. Ontology development 101: A guide to creating your first ontology. May 2009. Disponível em: <http://tw.rpi.edu/proj/portal.wiki/images/3/38/KSL-01-05.pdf>.

PEREZ, A. G.; BENJAMINS, V. R. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. *Proceedings of the (IJCAI'99) – Workshop on Ontologies and Problem-Solving Methods*, Stockholm-Sweden, August 1999.

PERNAS, A. M.; DANTAS, M. A. R. Using ontology for description of grid resources. *IEEE/Proceedings of the 19th Internacional Symposium on High Performance Computing Systems and Applications (HPCS'05)*, 2005.

ROURE, D. D. A brief history of the semantic grid. *In Semantic Grid: The Convergence of Technologies C. Goble, C. Kesselman, and Y. Sure, Eds. 05271*, Acesso: Novembro/2009, 2005. Disponível em: <http://drops.dagstuhl.de/opus/volltexte/2005/398/>.

ROURE, D. D.; JENNINGS, N. R.; SHADBOLT, N. R. The semantic grid: A future e-science infrastructure, grid computing making the global infrastructure a reality. *John Wiley & Sons*, 2003.

ROURE, D. D.; JENNINGS, N. R.; SHADBOLT, N. R. The semantic grid: Present, past, and future. *In Proceedings of IEEE*, v. 93, n. 3, p. 669–681, Acesso: Novembro/2009, 2005. Disponível em: <http://eprints.ecs.soton.ac.uk/9976/1/procieee.pdf>.

SAID, M. P.; KOJIMA, I. S-MDS: Semantic Monitoring and Discovery System for the grid. *J Grid Computing (2009)*, Springer Science, 2009.

SANTOS, R. C. dos; JORGE, E. M. F. Tutorial: *JavaServer Pages* (JSP). acesso: Dezembro 2009. Disponível em: <http://uploads.javafree.com.br/files_user/files/A/74/F8/Tutorial_JSP1.pdf'>.

SCHOPF, J. M. Grid resource management: state of the art and future trends. Kluwer Academic Publishers, 2004.

SCHOPF, J. M. et al. Monitoring the grid with the globus toolkit MDS4. *Journal of Physics*, 2006.

SKILLICORN, D. B. Motivating computational grids. *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

SMITH, G.; BAKER, M. A flexible monitoring and notification system for distributed resources. *International Symposium on Parallel and Distributed Computing*, IEEE, 2008.

SMITH, M. K.; WELTY, C.; MCGUINNESS, D. L. OWL web ontology language guide – W3C recommendation. Acesso: Outubro 2009. Disponível em: <http://www.w3.org/2004/OWL/>.

SOMASUNDARAM, T. S. et al. Semantic-based grid resource discovery and its integration with the grid service broker. *IEEE/In Proceedings of the 14th International Conference on Advanced Computing and Communications*, 2006.

STUDER, R.; BENJAMINS, R.; FENSEL, D. Knowledge engineering principles and methods. *Data and Knowledge Engineering (DKE)*, 1998.

TANGMUNARUNKIT, H.; DECKER, S.; KESSELMAN, C. Ontology-based resource matching in the grid - The grid meets the semantic web. *In The SemanticWeb - (ISWC'03) - LNCS*, Springer-Verlag, Berlin - Heidelberg, v. 2870, p. 706–721, 2003.

TRCKA, N. et al. Evaluating a data removal strategy for grid environments using colored petri nets. *Proceedings of the 12th International Conference on Principles of Distributed Systems*, Luxor, Egypt, p. 538–541, 2008.

VIDAL, A. C. et al. Applying semantics to grid middleware. John Wiley & Sons, February 2009.

ZHANG, Y.; SONG, W. Semantic description and matching of grid services capabilities. *Proceedings of the 3rd UK e-Science All Hands Meeting*, Nottingham, UK, 2004.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo