UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ana Marilza Pernas Fleischmann

ONTOLOGIAS APLICADAS À DESCRIÇÃO DE RECURSOS EM *GRIDS* COMPUTACIONAIS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Mário Antônio Ribeiro Dantas, Dr.

Florianópolis, Agosto de 2004.

ONTOLOGIAS APLICADAS À DESCRIÇÃO DE RECURSOS EM *GRIDS* COMPUTACIONAIS

Ana Marilza Pernas Fleischmann

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Raul S. Wazlawick, Dr.

Banca Examinadora

_____

Mário Antônio R. Dantas, Dr.

_____

Alba Cristina M. de Melo, Dra.

_____

Luis Fernando Friedrich, Dr.

_____

Frank Augusto Siqueira, Dr.

Dedicatória

Dedico este trabalho especialmente ao meu marido, Anderson, por toda sua atenção, paciência e por ter estado sempre do meu lado, com todo o seu amor e carinho.

Á minha mãe, Marilza, e meus irmãos, por terem sempre me incentivado a atingir meus objetivos e serem um exemplo a ser seguido.

Agradecimentos

Agradeço ao meu orientador, Prof. Mário Dantas, que ajudou muito na realização deste trabalho, indicando sempre os passos a serem realizados, e por todo seu conhecimento e experiência passados durante este período.
Ao pessoal do LABWEB (e visitantes) pela amizade, conselhos e bom humor, mesmo nos momentos difíceis.
A Vera Sodré, pela paciência e por toda ajuda dada nos momentos em que mais precisei.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API	Application Programming Interface
DAML	DARPA Markup Language
GWSDL	Grid Web Service Description Language
GB	GigaByte
HTML	Hyper Text Markup Language
MB	MegaByte
MDS	Metacomputing Directory Service
NPACI	National Partnership for Advanced Computational Infrastructure
OCML	Operational Conceptual Modelling Language
OML	Ontology Markup Language
OIL	Ontology Inference Layer
OV	Organizações Virtuais
OWL	Ontology Web Language
RDF	Resource Description Framework
RMS	Resource Management System
SDK	Software Development Kit
SO	Sistema Operacional
XML	eXtensible Markup Language
XOL	Ontology Exchange Language
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WWW	World Wide Web

Resumo

Ambientes de computação em *grid* podem realizar o compartilhamento em larga escala de seus recursos e serviços. Atualmente estes ambientes estão sendo considerados como uma solução eficiente por diversas organizações, para obtenção de alto desempenho e maior disponibilidade na execução de suas aplicações. As organizações podem se constituir de universidades, empresas, centros de pesquisa, fábricas ou, até mesmo, pessoas individuais, as quais necessitam de acesso a recursos geograficamente distribuídos para realização de tarefas. Entretanto, devido à diversidade de recursos e serviços oferecidos pelos ambientes de *grid*, em adição a uma falta de uma padronização para tarefas de busca e seleção de recursos disponíveis, sua utilização pelas organizações se torna complexa. Em outras palavras, existe a necessidade de um conhecimento prévio do ambiente e dos requisitos necessários para acesso.

Com o objetivo de facilitar a busca e seleção de recursos, esta dissertação apresenta como alternativa a utilização da abordagem de ontologias. Este paradigma pode prover a descrição dos recursos disponíveis no *grid*, auxiliando nas suas maneiras de atuação e descrevendo-os sintática e semanticamente na forma de um vocabulário comum ao domínio. Nesta alternativa, as requisições de acesso agem diretamente na ontologia, distanciando os usuários das restrições para acesso ao ambiente.

Em nossos experimentos, a abordagem proposta nesta dissertação se mostrou eficiente. A pesquisa feita com relação à forma de representação do conhecimento nestes ambientes e o desenvolvimento da ontologia nos permitiu concluir que a junção destas duas áreas traz grandes benefícios às organizações, pois permite melhor compreensão do ambiente, tanto para o fornecimento quanto para o acesso aos recursos.

Abstract

Computer grid environments can share resources and services in a large-scale. These environments actually have been considered as an effective solution for many organizations to obtain high-performance and high-availability to execute their applications. Organizations can be universities, firms, research centers, industries or, in fact, individual people, which require access to geographically distributed resources to execute specific tasks. However, the use of these grid environments is very complex, because they provide a lot of different services and resources, in addition to the lack of standardization from resource search and selection. In another words, these environments claim from the organizations a previous knowledge of the access requirements.

In order to make easier the resource search and selection, this dissertation presents as an alternative the use of ontology's approach. This paradigm can provide a description of the available resources in the grid, helping for the right operation and describing resources syntax and semantics that will form a domain-common vocabulary. In this alternative, access rules act directly in the ontology, leaving users from restrictions to environment access.

In our experiments, the approach shows to be efficient. The research about knowledge representation in these environments and the ontology development let us conclude that the union of these two domains brings a lot of benefits to organizations, since it allows better comprehension about the environment.

1. Introdução

Grids computacionais consistem de ambientes voltados ao fornecimento em larga escala de recursos, os quais são em geral providos por instituições ou indivíduos particulares para serem utilizados por outros usuários, com o objetivo de fornecer alto desempenho na execução de aplicações (FOSTER et al., 2001). O interesse pela utilização destes ambientes surge pelo atual crescimento da necessidade de infra-estrutura computacional que forneça suporte ao processamento paralelo de tarefas realmente complexas, que necessitam de grande poder computacional e relativa quantidade de recursos disponíveis para acesso.

Ambientes de *grid* são geograficamente distribuídos. Todos os seus recursos e serviços disponíveis se encontram interligados por meio de uma rede de computadores, podendo, desta forma, várias organizações formarem um *grid* a partir da interconexão de seus recursos através da rede. Nesta ótica, é fácil se comparar o *grid* à *Internet*, já que esta possui vários usuários interconectados e todos podem acessar os *sites* disponíveis dentro da rede em que estão conectados.

Entretanto, estes dois ambientes possuem muitas diferenças. Uma delas vem do fato do *grid* realizar compartilhamento de seus recursos e serviços, o que não ocorre dentro da *Internet*. Este acesso a recursos e serviços fornecido pelo *grid* é disponibilizado por meio de protocolos específicos e restrições impostas pelas organizações fornecedoras, estas restrições são chamadas de políticas de utilização, e podem variar dependendo do ambiente em questão e da organização fornecedora. Uma organização que deseja acessar os recursos fornecidos pelo *grid*, deve primeiramente satisfazer estas políticas de utilização, para posteriormente obter acesso.

Neste momento, já se pode perceber o destaque de duas entidades dentro do ambiente de *grid*, que são os *fornecedores* (donos) dos serviços/recursos e os *consumidores* destes serviços/recursos, com os fornecedores estabelecendo as regras a serem satisfeitas pelos consumidores para que estes sejam autorizados a acessar o ambiente. Também se pode perceber como ponto central do ambiente o fornecimento e a utilização de serviços e recursos, sendo este o principal objetivo do ambiente e o motivo pelo qual é tão buscado pelas organizações.

As organizações que compõem o *grid* são conhecidas como *organizações virtuais*

(OV), as quais consistem de coleções dinâmicas de indivíduos ou instituições realizando fornecimento e consumo de recursos e serviços do *grid* de acordo com suas regras de compartilhamento definidas (FOSTER et al., 2001). Exemplos de organizações virtuais podem ser grupos de universidades, de centros de pesquisas, de indústrias, de empresas, ou até mesmo grupos de indivíduos vindos de diferentes instituições.

Um problema enfrentado em ambientes de *grid* - nos quais são buscados recursos computacionais de uso geral por diferentes tipos de consumidores, onde estes não necessitam fazer parte da mesma instituição - se refere às dificuldades existentes para que as OVs descubram e acessem determinados recursos do *grid*. Pois a falta de padronização no acesso aos recursos destes ambientes obriga os consumidores a terem um conhecimento a respeito do funcionamento do ambiente, o que muitas vezes é indesejável e torna o acesso complexo. Neste aspecto é que a utilização de ontologias se apresenta como alternativa para padronização da representação dos recursos aos consumidores, facilitando a forma com que estes atuam no ambiente e realizam, de maneira eficiente, diversos tipos de tarefas que possam ser necessárias.

A utilização de ontologias vem sendo buscada por diversas áreas da Ciência da Computação, as quais o fazem em geral para dotar os sistemas de *meta-conhecimento*. A utilização de ontologias para descrição semântica de um determinado vocabulário, apesar de ser um processo relativamente complexo e exigir um estudo aprofundado a respeito do domínio a ser descrito, proporciona um entendimento amplo das características e propriedades das classes pertencentes a um domínio, assim como seus relacionamentos entre si.

A comunicação por meio de ontologias provê um *frame* comum de referência e vocabulário dentro de um domínio ou subdomínio, permitindo interoperabilidade entre sistemas, baseando-se em semânticas (POUCHARD et al., 2003). Além disso, pode-se acrescentar o fato de estas serem extensíveis, o que possibilita o uso em novos domínios, pois novas classes, regras ou vocabulários podem ser adicionados para descrição de um novo domínio de aplicação. Ontologias também podem ser distribuídas e compartilhadas para uso, em conjunto, com outras ontologias ou ferramentas, possibilitando também a interoperabilidade.

A partir de uma visão orientada a serviços e das vantagens trazidas pelo uso de ontologias, esta dissertação tem como principal objetivo apresentar como proposta a

utilização de ontologias na descrição dos recursos disponíveis em ambientes de *grids* computacionais, buscando automatizar e padronizar a tarefa de busca e seleção de recursos, distanciando os usuários dos detalhes a serem tratados para obtê-los, como políticas de uso, restrições técnicas para a utilização, entre outros.

Desta forma, a proposta para utilização de ontologias, apresentada nesta dissertação, atua na tarefa de descrição dos recursos disponíveis no *grid*, fornecendo informações a respeito dos recursos e do sistema, de uma forma geral, sempre que forem recebidas novas requisições para acesso. Nesta proposta as consultas, tanto dos usuários como dos programas de aplicação, atuam diretamente na ontologia, facilitando a busca pelos recursos realmente necessários. Além disso, a ontologia apresenta, agregados a ela, os axiomas utilizados para descrever as restrições que devem ser satisfeitas para a utilização de determinado recurso. Uma vez satisfeitos estes axiomas, pertencentes à ontologia criada, é possível a utilização do recurso por parte do consumidor.

Para realização desta dissertação, primeiramente foi feita uma investigação do conhecimento apresentado pelo domínio a que se destina a ontologia. Em um segundo momento, este conhecimento foi transformado em um vocabulário para, ao final, constituir as semânticas e os conceitos usados para definição dos recursos fornecidos pelo ambiente de *grid* adotado. No próximo passo, foram criados os axiomas da ontologia, que, no caso do trabalho, são necessários para guardar todas as regras a serem respeitadas durante a inclusão de novo vocabulário na ontologia. Ao final da dissertação, foi desenvolvida uma aplicação para possibilitar a apresentação da ontologia aos consumidores, a qual foi representada em um serviço de *grid*, visando tornar a aceitação da proposta apresentada e do novo ambiente mais simples e clara.

Nesta dissertação, os capítulos 2 e 3 apresentam uma revisão da literatura necessária para realização do trabalho. O capítulo 2 trata a respeito de computação em *grid*, fornecendo a base teórica para compreensão destes ambientes e do trabalho descrito. No capítulo 3 é realizada a apresentação de vários tópicos importantes dentro do contexto de ontologias, traçando a base teórica necessária para entendimento e desenvolvimento de aplicações baseadas em conhecimento.

O capítulo 4 mostra, inicialmente, alguns trabalhos relacionados ao tema abordado nesta dissertação. Em seguida, são descritos todos os passos que foram realizados,

apresentando como foi desenvolvida a ontologia para descrição de recursos do *grid*; a metodologia utilizada para criação; a linguagem de modelagem usada; a ferramenta para desenvolvimento; e como foi desenvolvida a aplicação e o serviço descritos.

No capítulo 5 são apresentadas as conclusões e resultados obtidos durante a realização desta dissertação, juntamente com algumas sugestões para trabalhos futuros.

2. *Grids* Computacionais

A computação em *grid*, uma infra-estrutura computacional distribuída voltada para o avanço da ciência e da engenharia, foi proposta por (FOSTER & KESSELMAN, 1999) em meados dos anos 90. Deste então, esta área vem crescendo incrivelmente, apresentando novos ambientes de *grids* computacionais sendo desenvolvidos em muitas partes do mundo, os quais, apesar de possuírem entre si interesses diferenciados, buscam alto poder computacional e alta disponibilidade de recursos e serviços, oferecidos por estes ambientes.

Um conceito claro e objetivo de *grid* é apresentado em (GRID, 2003):

"Grid é um tipo de sistema paralelo e distribuído o qual permite compartilhamento, seleção e agregação dinâmica em tempo de execução de recursos autônomos geograficamente distribuídos, dependendo de sua disponibilidade, capacidade, performance, custo e requisitos de qualidade de serviço necessários aos usuários".

Nestes ambientes geograficamente distribuídos é possível prover alto poder computacional, pois vários tipos de recursos computacionais, pertencentes a organizações distintas, são interligados por meio de uma rede de computadores. Desta forma, todos recursos existentes em cada ponto de interligação são somados e vistos como se estivessem totalmente unidos, formando um único ambiente de computação em alto desempenho, isto é, um *grid*. A alta disponibilidade é ocasionada pelo mesmo motivo, pois no caso de perda de um recurso localizado em certo ponto do *grid*, como este é visto como um único ambiente, o mesmo recurso é então buscado em outros pontos disponíveis.

Para que seja possível uma melhor compreensão a respeito dos componentes que podem ser interligados através do *grid*, é mostrado, na Fig. 2.1, um exemplo de *grid*, o qual representa o ambiente desenvolvido pelo projeto NPACI (National Partnership for Advanced Computational Infrastructure). Fundado pela NSF (National Science Foundation), o NPACI tem como objetivo a criação de uma infra-estrutura computacional em *grid* onde os pesquisadores podem obter dados vindos de diferentes bibliotecas digitais, analisá-los utilizando modelos executados sobre computação em

grid, visualizar e compartilhar estes dados através da *web* e publicar os resultados obtidos para toda a comunidade científica (NPACI, 2000).

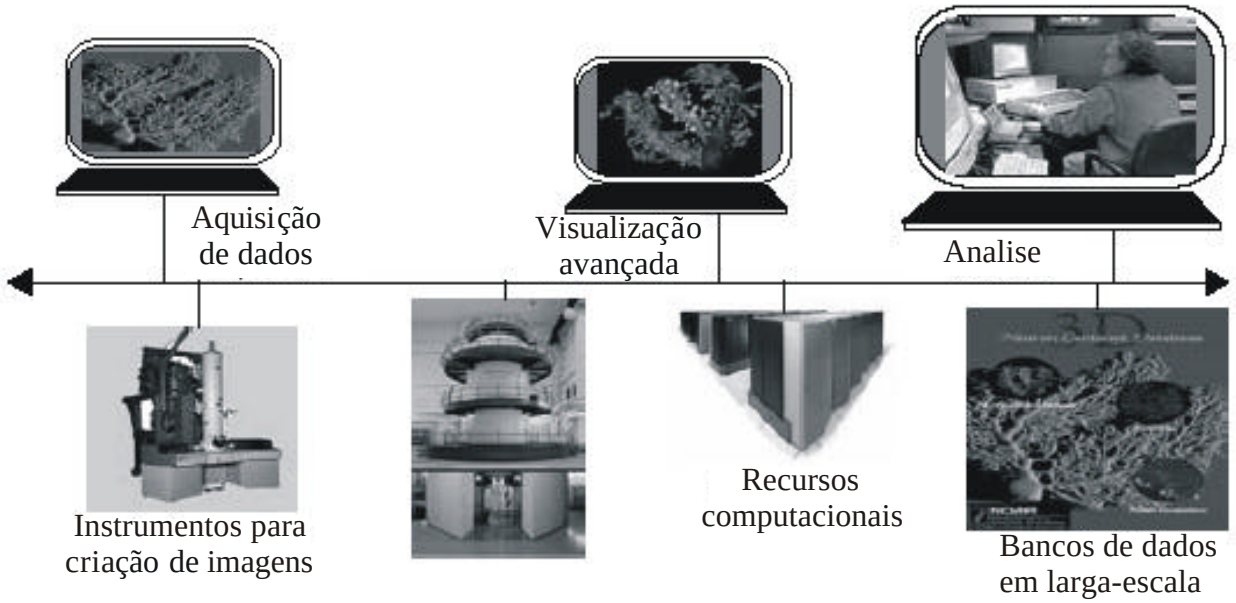

FIGURA 2.1: Ambiente *grid* desenvolvido pelo NPACI
Fonte: Adaptado de (BERMAN et al., 2003).

Com isso, já se podem perceber algumas das razões para utilização de ambientes de *grid*. Segundo FOSTER & KESSELMAN (1999), através da computação em *grid*, é fornecido aos usuários um substancial poder na execução de tarefas, possibilitando obtenção de um aumento da magnitude computacional de terceira a quinta ordem em um prazo de 5 a 10 anos. Ainda segundo o autor, este aumento será alcançado através de inovações abrangendo diversas aplicações, das quais são citadas:

- Aumento induzido da demanda de acesso a poder computacional: muitas aplicações possuem somente requisitos casuais para recursos computacionais. Se existem mecanismos situados no local necessário, permitindo acesso confiável, instantâneo e transparente aos recursos, na visão das aplicações é como se estes recursos estivessem dedicados a elas. O que possibilita a existência de sistemas *multiTeraFlops* e o aumento de terceira ordem ou mais do poder computacional aparente.

- Aumento da utilização de capacidade ociosa: computadores de poucos recursos (*personal computers* e *workstations*) podem estar freqüentemente ociosos. Estes recursos podem ser utilizados, trazendo benefícios como aumento de desempenho e disponibilidade, sem afetar os usuários destes computadores.

- Grande compartilhamento dos recursos computacionais: diariamente, a tarefa de previsão do tempo envolve aproximadamente 10^{14} operações numéricas. Se for assumido que a previsão é realizada por 10^{7} pessoas, tem-se o total de 10^{21} operações realizadas – comparado ao que é computacionalmente executado a cada dia no mundo. Alguns outros resultados computacionais ou facilidades são também compartilhados diariamente e, futuramente, podem ser compartilhados por toda comunidade científica, sendo o acesso computacional adotado pela "grande ciência", dando origem a ambientes colaborativos.

- Novas ferramentas e técnicas para solução de problemas: uma variedade de novos métodos pode aprimorar a eficiência e facilidade computacional aplicada a resolução de problemas.

Além destas razões apresentadas, *grids* computacionais são buscados por oferecerem confiabilidade e consistência no acesso aos recursos. A confiabilidade se traduz no fato dos usuários terem a certeza de que, caso seja possível a utilização do ambiente, estes irão receber freqüentemente altos níveis de desempenho e acesso prolongado aos recursos. Consistência se refere à existência de padrões nos serviços oferecidos, *interfaces* de acesso e parâmetros de operação. Essas facilidades é que fazem com que os *grids* tenham um efeito transformador na maneira com que a computação é apresentada e utilizada (FOSTER & KESSELMAN, 1999).

2.1 Áreas de Aplicação

Diversas áreas e aplicações que exigem: alto desempenho; alta disponibilidade; comunicação eficiente entre sistemas localmente distribuídos; entre outros benefícios, podem optar pela utilização de ambientes de *grid*. Entretanto, existem algumas áreas de destaque, nas quais a utilização de *grids* se torna realmente importante. Abaixo estão

descritas algumas destas áreas, as quais foram identificadas em (FOSTER & KESSELMANN, 1999), através de pesquisas experimentais, como sendo as maiores áreas de aplicação de *grids.*

- **Supercomputação Distribuída**: consiste de uma área em que os requisitos computacionais são tão altos que somente podem ser atendidos através da combinação de múltiplos recursos de alta capacidade, os quais são encontrados em um *grid*, formando um único supercomputador virtual (FOSTER & KESSELMANN, 1999).

  Um exemplo de supercomputação distribuída é o sistema *NetSolve* (CASANOVA & DONGARRA, 1995), o qual consiste de uma aplicação cliente-servidor projetada para tornar simples a resolução de problemas, relacionados a ciência da computação, através da rede. O sistema oferece vários tipos de *interfaces*, permitindo que usuários utilizando diferentes tipos de programas (C, Fortran, MATLAB ou WWW) utilizem o *NetSolve.* Devido ao seu propósito pode, dependendo da exigência, requerer muito poder computacional para possibilitar a resolução de problemas, garantindo aos usuários a existência de um ambiente de alta capacidade. Atualmente o sistema *NetSolve* faz uso efetivo da computação em *grid*, estando voltado à união de *interfaces* de programação e sistemas *desktop*, os quais são dominantes no trabalho de cientistas da computação, e o fornecimento de serviços oferecido pelos ambientes *grid* (AGRAWAL et al., 2003).

  A supercomputação é muito utilizada em áreas de pesquisa, as quais necessitam de aplicações de simulação (que, por sua vez, necessitam de grande poder computacional para ser executadas) para chegar a resultados somente obtidos a partir de teorias. Além disso, é uma área que vem crescendo em capacidade e desempenho de recursos oferecidos, sendo estes fundamentais para as aplicações atuais. Entretanto, segundo MESSINA (1999), apesar deste aumento existem aplicações em que suas requisições computacionais superam os recursos disponíveis por maior que sejam os centros supercomputacionais. Devido à existência destas aplicações, ambientes *grid* provêem acesso a uma crescente capacidade computacional, possível através do uso simultâneo e coordenado de computadores separados geograficamente, mas interligados por meio de redes de

computadores. Sendo possível, desta forma, estender-se o tamanho das aplicações para além dos recursos disponíveis em um único local.

- **Computação de Alto-*Throughput***: esta área é voltada às aplicações que requerem rápida velocidade durante a transferência de dados. Nestas aplicações, segundo FOSTER & KESSELMANN (1999), ambientes *grid* são usados para escalonar um grande número de tarefas fracamente acopladas, com o objetivo de colocar ativos computadores que não possuem muita utilização, isto é, freqüentemente ociosos.

  Existem vários ambientes que fazem uso de computação em alto-desempenho, um exemplo é o sistema para escalonamento *Condor* (CONDOR, 2004). Desenvolvido na Universidade de Wisconsin, o sistema tem como objetivo maximizar a utilização de computadores com o mínimo de interferência entre as tarefas escalonadas e as atividades de quem os utiliza, identificando computadores ociosos e escalonando tarefas sobre eles. O sistema *Condor* atua no gerenciamento das tarefas que são executadas por conjuntos de computadores, alterando o local (computador) de processamento das tarefas de acordo com a capacidade disponível no momento (LITZKOW et al., 1988).

- **Computação Sob-Demanda**: consistem de aplicações que necessitam fazer uso de recursos que não se encontram disponíveis localmente. Estes recursos podem ser classificados como repositórios de dados, pacotes de software, computacionais, e muitos outros. Pela sua descrição, parece similar à área de supercomputação distribuída, entretanto estas diferem com relação à utilização dos recursos, pois ambientes de supercomputação distribuída utilizam todos os recursos não locais visando melhoria de seu desempenho, enquanto que ambientes de computação sob-demanda apenas empregam recursos que podem solicitar de forma distribuída, uma vez que estes não se encontram localmente disponíveis (FOSTER & KESSELMAN, 1999; DANTAS, 2003).

  Como exemplo de utilização de computação sob-demanda pode-se citar o sistema desenvolvido na *Aerospace Corporation*, o qual possui como objetivo o processamento de dados para satélites meteorológicos. Este sistema realiza integração de fontes de dados, poder de processamento, alta tecnologia para

exibição de imagens e redes de computadores, formando um único recurso mais facilmente utilizável, o qual pode ser definido como sendo um ambiente de metacomputação - método uniforme para manuseio de uma coleção de recursos heterogêneos. Desta forma, o sistema permite obtenção de resposta às pesquisas processadas em quase tempo real (LEE et al., 1996).

- **Computação para Processamento Intensivo de Dados**: aplicações que utilizam processamento intensivo de dados são voltadas à união de informações obtidas a partir de dados armazenados em repositórios, bibliotecas digitais e bancos de dados, os quais estão geograficamente localizados. Esta união de informações necessita, em geral, de computação e comunicação intensiva, para poder efetivar a localização dos dados e, em seguida, seu grupamento (FOSTER & KESSELMAN, 1999).

  Uma aplicação que utiliza processamento intensivo de dados é o projeto *Digital Sky Survey*, o qual tem como objetivo integrar dados vindos de análises espaciais, realizadas através de imagens digitais de grandes áreas do espaço. Estas são criadas por meio da digitalização de fotos já existentes ou a partir da gravação de sinais digitais obtidos através de aparelhos para detecção de luz, ligados a um telescópio (MOORE et al., 1999).

- **Computação Colaborativa**: esta é voltada primariamente a possibilitar e melhorar as formas com que ocorrem interações humanas, sendo as aplicações geralmente realizadas a partir de espaços virtuais compartilhados, o que torna mais fácil o compartilhamento de arquivos e aplicações sem existência de altos custos para acesso ao ambiente (FOSTER & KESSELMAN, 1999; DANTAS, 2003).

  Um exemplo de computação colaborativa é o projeto *NICE* (*Narrative-based, Immersive, Constructionist/Collaborative Environments*), que consiste de um ambiente colaborativo para o aprendizado de crianças. Desenvolvido na Universidade de Illinois, Chicago, o projeto Nice tem como objetivo criar um ambiente virtual de aprendizado que tenha como base a colaboração entre os participantes, motivando para que utilizem o ambiente e permitindo que crianças aprendam juntas, estando ou não situadas no mesmo local (JOHNSON, 1998).

2.2 Arquitetura de Ambientes *Grid*

FOSTER et al. (2001) argumenta que o conceito de *grid* é de fato motivado por um problema real e específico. Segundo o autor, este problema consiste do compartilhamento coordenado de recursos e resolução de problemas em *organizações virtuais* (OVs) compostas por diferentes instituições. Desta forma, grande parte das ações realizadas no ambiente de *grid* estaria relacionada ao compartilhamento de recursos, os quais podem ser simples acessos a arquivos ou até mesmo acesso a computadores ou a *softwares*. E, para que este acesso a recursos seja feito da maneira correta, é necessário grande controle por parte tanto dos fornecedores (produtores) do serviço quanto dos consumidores, com ambos definindo clara e cuidadosamente quais recursos devem ser compartilhados, a quem é permitido o compartilhamento e as condições sobre as quais o compartilhamento pode ocorrer (FOSTER et al., 2001). Tendo em vista esse problema ao qual o *grid* é voltado e a natureza dos processos executados sobre ele, pode-se visualizar a arquitetura de um ambiente *grid* como sendo **orientada a serviços**.

Nesta arquitetura orientada a serviços é importante que, primeiramente, quem vai prover os serviços e recursos defina o conjunto de termos e condições que devem ser satisfeitos para possibilitar a obtenção do acesso. Em seguida, é necessária a realização de um contrato de serviço entre o fornecedor e o consumidor, onde irão constar os termos e condições sobre os quais o fornecedor concorda em prover o serviço ao consumidor (DE ROURE et al., 2003). O contrato não é necessariamente especificado somente pelo fornecedor do serviço, pois podem ocorrer casos em que é necessário que o consumidor também defina condições para sua utilização.

Para que sejam realizadas estas tarefas citadas acima, é importante que a arquitetura do *grid* esteja voltada a atender as operações realizadas pelas OVs (onde OVs são compreendidas como uma generalização tanto para fornecedores quanto consumidores de recursos e serviços), para que estas possam compartilhar recursos e serviços com qualquer outro usuário do *grid*. Portanto, identifica-se como questão central para arquitetura do ambiente o suporte à interoperabilidade nas aplicações de OVs, onde interoperabilidade significa a utilização de protocolos comuns para negociação de recursos, estabelecendo e gerenciando relações de compartilhamento.

Além disso, os protocolos possuem fundamental participação para obtenção de interoperabilidade, pois a sua definição especifica como os elementos do sistema distribuído irão interagir uns com os outros, de forma a apresentarem um comportamento esperado, e também especifica a estrutura da informação trocada durante a interação (FOSTER et al., 2001). Detalhes a respeito dos protocolos utilizados no *grid* são tratados no final desta seção.

Objetivando uma arquitetura extensível e de estrutura aberta, a partir da qual são fornecidos os requisitos para as tarefas desempenhadas pelas OVs, FOSTER et al. (2001) e DANTAS (2003) apresentam uma arquitetura para ambientes de *grid* composta de camadas, onde cada camada possui certos componentes, os quais compartilham características comuns. Esta arquitetura em forma de camadas é apresentada na Fig. 2.2.

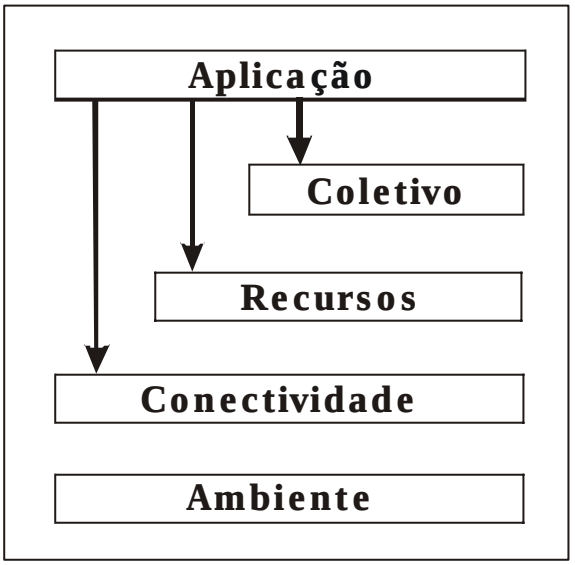

FIGURA 2.2: Arquitetura do *grid* organizada em camadas.
Fonte: (FOSTER et al., 2001).

Os autores ainda descrevem cada uma destas camadas, as quais são apresentadas de forma resumida a seguir:

- **Ambiente**: esta camada provê os recursos, os quais serão acessados de forma compartilhada por meio dos protocolos existentes no *grid*. Como exemplos de recursos, podem ser citados: recursos computacionais, recursos de rede, sistemas de armazenamento, catálogos e sensores. Seus componentes são responsáveis pelas operações locais que devem ocorrer para cada recurso, como resultado de

operações realizadas nos níveis superiores. Nesta camada é que os contratos de serviço entre o produtor e consumidor dos recursos, tratados no início desta seção, são estabelecidos, portanto deve desempenhar mecanismos que obtenham informações sobre a estrutura, o estado e as possibilidades dos recursos.

- **Conectividade**: esta camada define os protocolos centrais requeridos para comunicação e autenticação de transações específicas do *grid*. Os protocolos de comunicação permitem a troca de dados entre os recursos existentes na camada *ambiente*, enquanto que os protocolos de autenticação constroem os serviços de comunicação, provendo mecanismos criptográficos para verificação da identidade de usuários e recursos.

- **Recursos**: nesta camada são definidos os protocolos, API's (*Application Programing Interface*) e SDKs (*Software Development Kit*) que garantem segurança durante as negociações, iniciação, monitoração, controle, criação de relatórios, entre outras tarefas voltadas ao compartilhamento de recursos individuais. Implementações dos protocolos desta camada chamam funções pertencentes à camada *ambiente* para obterem acesso e controle aos recursos locais. Estes protocolos preocupam-se apenas com recursos individuais, ignorando questões a respeito de estado global e coleções distribuídas, sendo estas tratadas pela camada *coletivo.*

- **Coletivo**: nesta camada se encontram os protocolos, serviços, APIs e SDKs que não se relacionam a um recurso específico, mas que garantem a interação entre coleções de recursos. Seus componentes são construídos de acordo com as camadas *recurso* e *conectividade*, e são capazes de desenvolver vários serviços, como por exemplo:
  - Serviços de diretório – permitem que OVs saibam a respeito da existência e/ou das propriedades de recursos;
  - Serviços de monitoração e diagnóstico – oferecem suporte as OVs para monitorarem os recursos a respeito de falhas, detecção de intrusos, sobrecarga, entre outros.

- Serviços de escalonamento e co-alocação – permitem que OVs requeiram a alocação de um ou mais recursos para um propósito específico e o escalonamento de tarefas em recursos apropriados.

- **Aplicação**: esta camada compreende as aplicações dos usuários que estão ligados a uma OV, isto é, os programas de aplicação para os quais usuários necessitam dos recursos, oferecidos pelo *grid*, para que sejam corretamente executados.

Esta arquitetura em camadas representa uma visão relativamente teórica dos elementos que a compõem. De outra forma, pode-se visualizar a arquitetura do *grid* a partir dos tipos de sistemas que são empregados no seu desenvolvimento. A próxima subseção trata de forma resumida a respeito de alguns destes sistemas.

2.2.1 Sistemas que Compõem um *Grid*

A seguir serão descritos três tipos de sistemas, os quais entende-se representarem os principais em um ambiente de *grid*, tendo em vista sua complexidade e o quanto são empregados nestes ambientes.

- ***Clusters* computacionais**: conhecidos também como agregados de computadores, consistem da integração de vários computadores, geralmente tratados como nós, por meio de uma rede local de alta velocidade, sendo construídos para funcionamento como um único sistema formando, por exemplo, um recurso para processamento de dados. A configuração visa maior desempenho na execução de aplicações, através da agregação de vários processadores, o uso de algoritmos otimizados e, principalmente, a facilidade de computação paralela (FOSTER & KESSELMAN, 1999; DANTAS, 2003). Estes ambientes são altamente utilizados, pois, além de proverem as vantagens mencionadas, podem ser construídos a partir de componentes facilmente encontrados para aquisição.

  No desenvolvimento de *grids* computacionais, freqüentemente utilizam-se vários ambientes de *cluster* computacionais interligados, onde estes acabam fazendo parte da arquitetura do *grid*. Entretanto, existem diferenças nas suas configurações, de forma que não podemos tratar *grids* computacionais como sendo simplesmente a união de várias configurações de *cluster*. Na Tab. 2.1 são

apresentadas as diferenças, descritas por DANTAS (2003), existentes entre estes ambientes.

TABELA 2.1: Diferenças entre as configurações de *cluster* e *grid.*

Característica	***Cluster***	***Grid***
Domínio	Único	Múltiplos
Nós	Milhares	Milhões
Segurança do Processamento e Recurso	Desnecessária	Necessária
Custo	Alto, pertencente a Um único domínio	Alto, todavia dividido entre domínios
Granularidade do problema	Grande	Muito grande
Sistema Operacional	Homogêneo	Heterogêneo

Fonte: (DANTAS, 2003).

- ***Intranet***: consiste da união de diversos computadores a partir de uma rede local, pertencente a uma única organização. Similares aos *clusters* computacionais, podem assumir controle administrativo centralizado e, portanto, alto grau de coordenação entre os recursos. Entretanto, *intranets* possuem em geral sistemas computacionais e de rede heterogêneos, não sendo possível se obter uma única imagem de todos os sistemas compostos por ela, além disso, podem existir conflitos com relação às políticas de uso adotadas dentro deste sistema, devido a não obrigatoriedade de haver uma única administração entre sistemas individuais e também a heterogeneidade. Devido a estes problemas, os conjuntos de serviços podem não estar integrados da mesma forma que estão em configurações de *clusters* (FOSTER & KESSELMAN, 1999).

  Apesar dos problemas na integração dos sistemas, também podem fazer parte de ambientes de *grid*, pois estes também se constituem de configurações heterogêneas, possibilitando a utilização de *intranets.* Desta forma, os mesmos cuidados existentes em *grids* com relação a políticas de uso, descrição e acesso a recursos, segurança, entre outros, devem também ser aplicados a estes sistemas.

- ***Internet***: consiste de uma infra-estrutura que teve início no final da década de 60, inicialmente denominada como ARPANET, onde seu objetivo consistia, basicamente, da interconexão de computadores de centros de pesquisa em universidades, ou outras organizações, para que pudessem compartilhar seus recursos computacionais. Atualmente a *internet* forma uma infra-estrutura bem mais complexa, onde fazem parte dela os mais diversos tipos de usuários e organizações, embora ainda com o objetivo de interligar computadores geograficamente distribuídos.

  Como mostrado em configurações de *clusters*, a *internet* apresenta diferenças com relação a *grids* computacionais, uma destas se refere ao tipo de serviço fornecido por cada ambiente. No caso da *internet*, são fornecidas aplicações prontas para serem utilizadas, sem o questionamento da necessidade de interoperabilidade de acesso entre o usuário e o *site* que fornece a aplicação. No *grid*, por outro lado, a questão fundamental é o fornecimento de interoperabilidade a aplicações desenvolvidas por consumidores (DANTAS, 2003).

  Devido à grande diferença entre o propósito da *internet* e do *grid*, se torna difícil considerar ambientes *grid* fazendo uso da *internet*. Na verdade, o objetivo não é este, mas sim propor mudanças a serem realizadas na *internet* para que ambientes *grid* possam usá-la de forma adequada, essas mudanças são relacionadas a: aumento da capacidade de conexão; capacidade de gerenciamento e configuração realizada por máquina; suporte a tolerância a falhas; capacidade de processamento distribuído; localização de serviços a partir de parâmetros (POSTEL & TOUCH, 1999).

2.2.3 Protocolos de Rede

Como tratado no início desta seção, a ação dos protocolos em *grids* computacionais é que define a possibilidade de interoperabilidade entre as aplicações, pois são estes que decidem a maneira com que é realizado qualquer tipo de comunicação no ambiente. A utilização inadequada de protocolos ou o seu desenvolvimento mal projetado traz problemas a comunicação, como retardo ou perda de informações, podendo inviabilizar a sua ocorrência. Desta forma, se torna extremamente importante para a boa formação da arquitetura do *grid* definir

adequadamente os protocolos que serão utilizados, ou então como será a criação dos protocolos para cada tipo de aplicação.

Portanto, de acordo com as necessidades apresentadas pelas aplicações é que são determinados quais serviços deverão ser providos pelos protocolos. Clássicos protocolos de rede provêem serviços de transferência de dados ponto a ponto, partindo do emissor dos dados, e entrega de mensagens ordenadas de acordo com o emissor com razoável segurança e tempo de transmissão. Entretanto, cada aplicação do *grid* possui suas próprias exigências para comunicação e seus tipos de informação a serem transmitidos (FOSTER & KESSELMAN, 1999).

São definidos em (FOSTER & KESSELMAN, 1999) quais protocolos são necessários de acordo com as áreas de aplicação de *grids* computacionais, apresentadas na seção 2.1 desta dissertação. Segundo os autores:

- Ambientes de **Computação Colaborativa** são os mais exigentes com relação a utilização de protocolos, pela sua demanda e diversidade de informações tratadas. Suas aplicações necessitam de: protocolos para transferência de dados de áudio e vídeo onde esta não necessita ser finalizada para que os dados sejam reproduzidos (*data streaming*); protocolos para transferência segura de dados entre os colaboradores; protocolos de controle coordenado e protocolos associados para permitir a colaboração; protocolos para integração de módulos complexos e desenvolvidos independentemente.

- Ambientes de **Computação Sob-Demanda** também são exigentes com relação à demanda de dados transmitidos e apresentam problemas críticos devido à alta necessidade de segurança no envio dos dados. Estas aplicações necessitam de alguns tipos de protocolos, que são: protocolos de transporte de dados, os quais realizam entrega de mensagens contendo os dados mais recentes disponíveis; protocolos de controle coordenado com tolerância a falhas; protocolos para integração de módulos complexos, desenvolvidos de forma independente.

- Ambientes de **Computação para Processamento Intensivo de Dados** necessitam dos seguintes tipos de protocolos: protocolos de transporte de dados, para envio de grandes conjuntos de dados de forma eficiente, rápida e segura;

protocolos para integração de módulos complexos e desenvolvidos de forma independente; protocolos para integração e encapsulamento de módulos existentes e movimentação de códigos de programas para locais remotos, onde os dados estão armazenados.

- Ambientes de **Supercomputação Distribuída**, sendo constituídos de supercomputadores interconectados ou de *clusters* de computadores, são os que necessitam de protocolos mais refinados, que são: protocolos de transporte seguro de dados com baixo retardo entre o envio dos dados pelo emissor e entrega ao receptor (*low-latency*); protocolos síncronos com controle do tempo de retardo entre o envio dos dados pelo emissor e entrega ao receptor, os quais são escaláveis com relação ao grande número de nós existente; protocolos para integração e encapsulamento de módulos existentes e movimentação de códigos de programas para locais remotos, onde o processamento é realizado.

2.3 Ferramentas para Planejamento de *Grids*

Tarefas de planejamento e desenvolvimento de *grids* computacionais não exigem a utilização de ferramentas específicas, pois já se pode perceber que estas são possíveis através da interligação de serviços e recursos computacionais por meio de uma rede de computadores, utilizando certos protocolos para permitir a comunicação entre os serviços e recursos disponíveis. Entretanto, o planejamento sem utilização de ferramentas apropriadas e sem aplicação de uma metodologia clara pode levar a vários problemas na execução do ambiente, tornando-o não aberto à inclusão de novos serviços ou usuários, e muitos outros problemas de projeto que podem surgir.

Desta forma, esta seção apresenta algumas das ferramentas apropriadas para planejamento e programação de *grids* computacionais, as quais visão permitir a criação de ambientes mais sofisticados, de forma segura e planejada.

2.3.1 Ferramentas para Aplicações Específicas

A programação de aplicações voltadas a ambientes *grid* é uma tarefa relativamente complexa, uma vez que estas devem prever acesso a diversos recursos que podem não estar acessíveis localmente e, caso não estejam, deve ser composta por métodos para localização destes recursos no *grid* e determinação de seu estado para uso.

Esta complexidade é reduzida a partir da utilização de certas ferramentas, desenvolvidas para facilitar a criação de aplicações para *grid*. Estas ferramentas permitem que o desenvolvedor especifique seus problemas em alto nível, utilizando com freqüência abstrações adaptadas a um domínio específico de aplicação, diminuindo o tempo associado ao desenvolvimento de aplicações (CASANOVA et al., 1999).

CASANOVA et al. (1999) tratam estas ferramentas como um conjunto de ferramentas de aplicação, uma vez que nelas estão incluídas diferentes tecnologias, como: linguagens e bibliotecas específicas de aplicação ou domínio; estruturas de aplicação e tecnologias de criação; ambientes para resolução de problemas; e serviços do *grid* específicos de aplicação. Apesar de serem conjuntos de ferramentas de aplicação muitas vezes voltadas para domínios específicos, não atuam isoladamente no ambiente de *grid*, pois interagem e são construídas sob outras tecnologias existentes, entre elas: escalonadores, compiladores, tecnologias de criação, serviços básicos do *grid* e protocolos de rede. Algumas destas tecnologias estão tratadas de forma mais detalhada nas demais seções e subseções desta dissertação.

Como exemplo destas ferramentas, existem os sistemas *NetSolve*, o qual foi apresentado na seção 2.1, e *Ninf* (Network based Information Library), o qual, assim como o *NetSolve*, consiste de uma avançada infra-estrutura computacional em rede de larga distância onde se possibilita ao usuário acessar recursos computacionais, incluindo *hardware*, *software* e dados científicos, distribuídos através da rede. Além de explorar alto desempenho em um ambiente de computação paralela e distribuída, o sistema *Ninf* tem como objetivo prover serviços computacionais de alta qualidade para solução de problemas numéricos e acesso a bancos de dados científicos publicados por diversos pesquisadores. Estes serviços são fornecidos com base em sistemas cliente-servidor, de forma que os recursos computacionais podem ser acessados como bibliotecas digitais remotas, através de um computador servidor, o qual pode ser acessado por meio de um programa cliente escrito em linguagens como Fortran, C e C++ (SATO et al., 1997).

2.3.2 Linguagens e Compiladores

Da mesma forma como foi apresentado o problema a respeito do desenvolvimento de aplicações para *grid*, também são os problemas encontrados durante a programação de linguagens e compiladores para estes ambientes. Pois, para isto, o sistema deve tratar dos detalhes necessários para o mapeamento de abstrações em respectivas configurações computacionais disponíveis a cada momento. Assim, várias tecnologias têm sido desenvolvidas para exploração de alto desempenho em configurações computacionais paralelas, incluindo extensões de linguagens para tratamento paralelo de dados e técnicas de compilação para gerenciamento e tolerância a tempos de resposta variáveis.

Segundo KENNEDY (1999), linguagens paralelas de dados são fundadas, basicamente, em duas observações: 1ª - para se alcançar alto desempenho em máquinas de memória distribuída, é necessário alocar dados às diversas memórias existentes (FOX et al., 1993). Desta forma, se cada tarefa de processamento de um programa for executada no processador onde se localiza a maior parte dos dados envolvidos, então o programa pode ser executado com grande eficiência; 2ª - se a escala de paralelismo consiste de centenas ou milhares de processadores, devem-se paralelizar os dados subdividindo-os em subdomínios de dados e designando cada subdomínio resultante a diferentes processadores. Várias linguagens de programação foram desenvolvidas com base nestas observações, entre elas destacam-se: Vienna Fortran (CHAPMAN et al., 1992), Fortran D (FOX et al., 1990), C* (TICHY et al., 1991) e PC++ (BODIN et al., 1993).

Com relação à tarefa de compilação em *grids* computacionais, problemas ocorrem quando esta é realizada em tempo de execução. Pois ambientes *grid* são gerados em tempo de execução, desta forma, informações a respeito dos nós que fazem parte do *grid* não podem ser obtidas antecipadamente, mesmo quando se trata de um *grid* homogêneo (KENNEDY, 1999). Como solução para este problema, foi proposto por MIRCHANDANEY et al. (1988) o método *inspetor/executor*, onde a compilação é dividida em duas partes: na primeira, o inspetor determina um plano para execução eficiente em uma máquina paralela; e na segunda, o executor cumpre o plano anteriormente definido pelo inspetor.

Independentemente deste método em específico, o princípio consiste da divisão da complexidade existente na realização da compilação em vários passos menos

complexos, para que mesmo que ocorram problemas, estes não persistam até o final mal sucedido da compilação. Quando se trata de ambientes heterogêneos, a execução da compilação se torna ainda mais complexa.

2.3.3 Aplicação de Tecnologias para Consumo

Foi discutido no início desta seção a respeito dos problemas enfrentados no desenvolvimento de *grids* computacionais e como se faz necessária a utilização de métodos confiáveis para esta tarefa. Nesta subseção objetiva-se mostrar a proposta introduzida por FOX & FURMANSKI (1999) sobre desenvolvimento de ambientes de *grids* computacionais a partir de tecnologias criadas para serem consumidas como um produto, aplicado como uma camada intermediária em arquiteturas (um *middleware*), de forma a unir serviços sofisticados a simples aplicações de interface com o usuário. Dentre estas tecnologias classificam-se CORBA (OMG, 2004), COM (ROGERSON, 1996), *Java* (GOSLING et al., 2000), *JavaBeans* (VOSS, 1996), *JavaScript* (NETSCAPE, 2000), *ActiveX* (CHAPPEL, 1996), VRML (*Virtual Reality Markup Language*) (CAREY & BELL, 1997), HTML (RAGGETT et al., 1999), entre outras.

Segundo os autores, arquiteturas desenvolvidas para consumo em aplicações de redes de computadores são adequadas para servirem como uma arquitetura para *grids* computacionais, pois, a partir destas tecnologias, se torna mais fácil a extensão de arquiteturas distribuídas para que incorporem serviços ligados a *grids* computacionais. Desta forma, combina-se alto desempenho com as ricas funcionalidades fornecidas por estes sistemas de consumo.

Estes ainda apresentam alguns casos onde podem ser aplicadas tecnologias de consumo, como por exemplo:

- VRML ou *Java3D* podem ser usados para visualização de dados científicos;
- *Java* pode ser definida como a linguagem científica de programação comum a todas as aplicações do *grid*;
- A adoção universal do protocolo JDBC (*Java DataBase Connectivity*) para conexão de bancos de dados e o crescimento de bancos de dados conectados a

web pode resultar em um aumento na comunicação em larga escala de sistemas de bancos de dados comerciais com recursos computacionais de alta performance.

2.4 Serviços Providos

Quando foi analisada a arquitetura do *grid* (seção 2.2), constatou-se que esta é orientada a serviços, por ser o seu principal propósito o fornecimento de recursos a OV. Entretanto, para que recursos sejam fornecidos é importante que esses ambientes sejam providos de serviços para: localização, descrição e gerenciamento de seus componentes; análise do desempenho do sistema; escalonamento de recursos para que sejam utilizados; cuidados com a segurança, a partir da autenticação de usuários que entram e saem do ambiente; entre muitos outros.

Desta forma, esta seção trata a respeito de alguns destes serviços, traçando suas características e descrevendo seu funcionamento. A seção é encerrada com uma descrição do conjunto de ferramentas *Globus* (GLOBUS, 2004), o qual é utilizado para desenvolvimento de ambientes de *grid* e possui grande importância entre a comunidade ligada a estes ambientes.

2.4.1 Escalonadores

Grids computacionais permitem que recursos não existentes localmente sejam utilizados por seus usuários, devendo, portanto, prever mecanismos para lidar com o fornecimento de recursos a um grande número de usuários e, além disso, gerenciar a utilização simultânea destes recursos. A categoria de mecanismos que auxiliam na solução destes problemas é a dos escalonadores, os quais nesta dissertação podem ser denominados como *escalonadores de alto desempenho*.

Os escalonadores existentes em ambientes *grid* formam um dos serviços mais importantes e utilizados por serem essenciais para obtenção de alto desempenho. Segundo BERMAN (1999), experiências de mais de duas décadas com aplicações paralelas e distribuídas indicam o escalonamento como fundamental para o desempenho, pois emprega modelos preditos para avaliar o desempenho de uma aplicação de base do sistema, usando esta informação para determinação de tarefas,

comunicação e envio de dados a recursos, tendo como objetivo o aumento de desempenho em determinadas plataformas.

Entretanto, existem vários tipos de escalonadores em *grids*, onde cada um possui objetivo diferente com relação a performance buscada. Segundo BERMANN (1999), nestes tipos estão destacados: **escalonadores de tarefas**, os quais elevam o desempenho do sistema através da otimização do tempo de transferência de dados entre as tarefas executadas; **escalonadores de recursos**, que coordenam múltiplas requisições de acesso a um dado recurso por meio da otimização de critérios justos para permissão do acesso ao recurso; e **escalonadores de aplicação**, os quais promovem o aumento de desempenho de aplicações individuais através da otimização de medidas como tempo mínimo de execução, resolução, e outras medidas adotadas para aplicações.

Independente do tipo de escalonador tratado, sendo ele aplicado a ambientes de *grid*, devem realizar certos passos para execução de aplicações, pois estas são geralmente compostas de várias tarefas, as quais poderão interagir umas com as outras durante a execução formando ao final uma aplicação única. Segundo BERMAN (1999), escalonadores sempre devem realizar os seguintes passos:

- Selecionar um conjunto de recursos onde as tarefas da aplicação serão escalonadas, o que compreende a seleção de recursos candidatos e a determinação de quais recursos estão disponíveis para a aplicação.

- Determinar cada tarefa da aplicação para determinado recurso computacional, onde ocorre o particionamento da aplicação em tarefas.

- Distribuir os dados necessários às tarefas ou colocá-los junto ao local onde são executadas.

- Ordenar as tarefas nos recursos computacionais.

- Ordenar a comunicação entre as tarefas.

Como escalonadores sempre seguem estes cinco passos para execução das aplicações desenvolvidas em ambientes *grid*, pode-se considerar como *escalonamento* as atividades que são desempenhadas durante cada um dos passos descritos.

2.4.2 Gerenciamento de Recursos

O gerenciamento de recursos, assim como o escalonamento, é um dos serviços mais importantes do *grid*, visto que deve resolver os problemas relacionados ao motivo principal dos usuários buscarem ambientes *grid*: fornecimento/consumo de recursos computacionais.

Estes usuários, em geral cientistas, engenheiros e responsáveis por tomada de decisão, são tratados por LIVNY & RAMAN (1999) como orientados a *throughput*, pois, segundo os autores, este tipo de usuário, avalia o poder de um sistema através da quantidade de trabalho desempenhado por ele em certo tempo fixado, onde esta avaliação leva em conta tempos, que podem variar entre dias ou semanas, e a quantidade de trabalho, que pode ser limitada ou não. Devido a esta grande necessidade de poder computacional é que estes usuários buscam a utilização de *grids* computacionais, sendo estes ambientes possíveis geradores de altas quantidades de dados.

O componente principal em ambientes de computação de alto *throughput* é o *sistema de gerenciamento de recursos* (RMS – *resource management system*), o qual tem a função de administrar todos os recursos distribuídos que são disponíveis para uso. Estes recursos são utilizados pelas seguintes classes de usuários, citados de acordo com a sua importância no ambiente: fornecedores (donos) de recursos, administradores de sistema, programadores de aplicações e clientes. Esta ordem de importância existe pela prioridade com que devem ser atendidos pelo RMS, pois: 1º - o RMS deve garantir que os fornecedores terão acesso sem nenhum impedimento aos recursos que fornece, sem causar degradação a disponibilidade ou desempenho no uso do recurso; 2º - os administradores de sistema devem sentir-se confiantes com relação à funcionalidade do RMS, sabendo que este é robusto e pode executar continuamente sem necessitar de muitas intervenções; 3º - a existência de APIs inflexíveis e obscuras nos serviços fornecidos pelo RMS torna impraticável o seu total aproveitamento por parte dos programadores de aplicação, pois desta forma suas características não podem ser efetivamente garantidas (LIVNY & RAMAN, 1999).

Devido a esta hierarquia existente entre os usuários, LIVNY & RAMAN (1999) definiram o RMS como sendo composto por seis camadas:

- **RM Local**: camada de *software* que provê os serviços básicos do RM para execução de recursos em seu próprio domínio;

- **Fornecedor**: provê os serviços necessários para atender os interesses dos fornecedores dos recursos;

- **Sistema**: camada que tem como função o gerenciamento da alocação global dos recursos no RMS;

- **Consumidor**: provê os serviços necessários para atendimento aos interesses dos consumidores dos recursos;

- **RM Aplicação**: no momento em que um recurso é exigido pela camada *consumidor*, este é passado para a camada RM aplicação, a qual será responsável pela comunicação com o módulo de controle de acesso ao recurso para estabelecer um ambiente de execução para a aplicação, além de outras funções;

- **Aplicação**: representa as tarefas desempenhadas pelas aplicações dos consumidores.

2.4.4 Projeto GLOBUS

O projeto ***Globus***, desenvolvido inicialmente pelos pesquisadores Ian Foster (University of Chicago) e Carl Kesselmann (University of Southern California), consiste de um esforço multi-institucional (GLOBUS, 2004) para tornar possível a construção de *grids* computacionais. Neste esforço, o projeto desenvolveu um conjunto aberto de serviços centrais e bibliotecas de software para construção de ferramentas e aplicações voltadas a ambientes *grid*, o qual é denominado ***Globus Toolkit*** (FOSTER & KESSELMANN, 1998).

O projeto *Globus* teve seu início a partir de estudos realizados a respeito de problemas apresentados em um projeto anterior, chamado ***I-WAY*** (*Information Wide Area Year*) (FOSTER & KESSELMANN, 1998), o qual é considerado como primeiro investimento em larga escala na criação de um ambiente de *grid*, sendo conhecido também como *Internet do Futuro* (DANTAS, 2003). Este projeto, iniciado em 1995,

consistia de um ambiente experimental para construção de aplicações de realidade virtual distribuídas e exploração a respeito do gerenciamento e escalonamento de recursos geograficamente distribuídos. Para isto, foram interconectados: doze ambientes ATM (*asynchronous transfer mode*) equipados para teste; dezessete centros de supercomputadores; cinco *sites* de pesquisa em realidade virtual; e mais de sessenta grupos de aplicação, sobre os quais foram desenvolvidas aplicações em supercomputação distribuída, ambientes virtuais e colaborativos, vídeo, entre outras (DeFANTI, 1996).

O projeto I-WAY permitiu a visualização do que poderia ser desenvolvido a partir de ambientes distribuídos, dando início a uma série de pesquisas e inovações. Dentre elas surgiu o projeto *Globus*, buscando compreender os requisitos de aplicações para construção de ambientes eficientes de *grid* e as tecnologias necessárias para obtenção destes requisitos (FOSTER & KESSELMANN, 1998).

O componente central do sistema *Globus* é o *Globus Toolkit*, citado anteriormente. Este é formado por um conjunto de componentes, os quais desenvolvem serviços básicos para: segurança, alocação e gerenciamento de recursos, gerenciamento de dados, comunicação, infra-estrutura de informações, detecção de falhas e portabilidade, podendo ser utilizados de forma independente ou agrupados para desenvolvimento de aplicações (GLOBUS, 2004). Estes serviços foram criados a partir de um modelo de programação uniforme, para que usuários possam utilizá-los de forma a atender os seus próprios requisitos de projeto, independente do tipo de sistema computacional usado. Além disso, prevêem a interligação entre sistemas heterogêneos, característicos entre sistemas de *grids* computacionais. Os serviços citados estão descritos de forma resumida na Tab. 2.2.

Um exemplo da atuação do serviço de segurança é apresentado na Fig. 2.3, onde os serviços de segurança locais desenvolvem um mecanismo de segurança (*gateway*) que faz o mapeamento de credenciais autenticadas *Globus* em credenciais que podem ser reconhecidas localmente, como por exemplo: um protocolo de autenticação *Kerberos*, nomes de usuário ou senhas locais.

TABELA 2.2: Serviços centrais providos pelo *Globus Toolkit*.

Serviço	Nome	Descrição
Gerenciamento de recursos	GRAM (*Globus Resource Allocation Manager*)	Responsável pela alocação de recursos e gerenciamento de processos.
Comunicação	Nexus	Responsável pelos serviços de comunicação *unicast* (ponto a ponto) e *multicast* (um ponto a múltiplos pontos).
Informação	MDS (*Metacomputing Directory Service*)	Responsável pelo acesso distribuído a estrutura e estado de informações a respeito dos componentes de sistema.
Segurança	GSI (*Globus Security Infrastructure*)	Responsável pelos serviços de autenticação, autorização, privacidade, e outros relacionados à segurança.
Saúde e estado do sistema	HBM (*Heartbeat Monitor*)	Responsável pela monitoração da saúde e do estado de um conjunto distribuído de processos.
Acesso a dados remotos	GASS (*Global Access to Secondary Storage*)	Efetuam acesso remoto a dados por meio de *interfaces* seqüenciais e paralelas.
Gerenciamento de execução	GEM (*Globus Executable Management*)	Responsável pela identificação, alocação e criação de aplicações executáveis em ambientes heterogêneos.

Fonte: Modificado de (FOSTER & KESSELMANN, 1999).

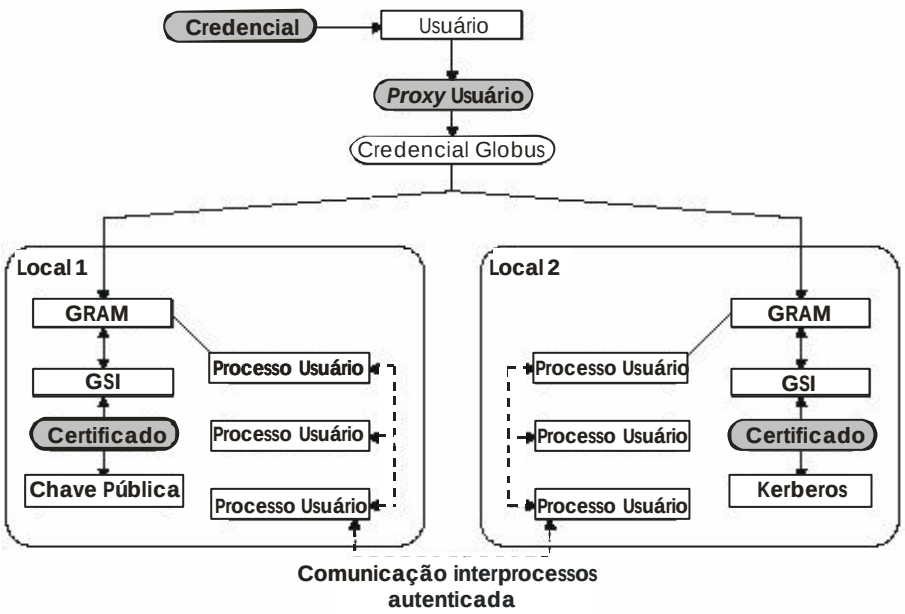

FIGURA 2.3: Infra-estrutura de segurança do sistema *Globus Toolkit*.
Fonte: Modificado de (FOSTER & KESSELMANN, 1998).

3. Ontologias

A busca pelo uso de ontologias em Ciência da Computação já vem sendo feita há vários anos, tendo sido inicialmente realizada pela Inteligência Artificial, visando criar representações que fossem além da descrição de simples instâncias do domínio considerado (GAVA & MENEZES, 2003). Atualmente, diversas áreas da ciência da computação utilizam ontologias, em geral buscando o desenvolvimento de um vocabulário contendo os conceitos relativos ao domínio de aplicação.

Na Filosofia, de onde foi originado o termo "ontologia" (GAVA & MENEZES, 2003), não são encontradas muitas controvérsias com relação ao seu significado, sendo a palavra "ontologia" geralmente definida como parte da filosofia que estuda a natureza e os seres, a metafísica. Entretanto, por ser uma palavra usada não somente pela Filosofia, mas também por diversas áreas, são encontradas várias definições para seu significado, freqüentemente dependendo do seu uso específico.

Antes de serem apresentadas as definições dadas para ontologia, percebe-se a importância de esclarecer que existe uma distinção entre o significado da palavra Ontologia (escrita com letra maiúscula) e ontologia (escrita com letra minúscula). A palavra "Ontologia" possui uma definição bem clara, se referindo a uma disciplina específica da Filosofia, enquanto que a palavra "ontologia" possui incontáveis definições, sendo primariamente dividida em dois diferentes sentidos: um assumido pela Filosofia, onde ontologia se refere ao sistema particular de categorias de acordo com certa visão do mundo, não tendo uma linguagem específica para representação; e outro assumido pela Inteligência Artificial e, em geral, toda comunidade da ciência da computação, onde ontologia se refere à artefatos de engenharia, constituídos por um vocabulário específico usado para descrever certa realidade (GUARINO, 1998). Tendo em vista a área abordada por esta dissertação, será adotado o sentido utilizado pela Inteligência Artificial para a palavra "ontologia".

Com relação aos seus diversos significados, GRUBER (1993b) define ontologia como uma especificação explícita de uma "conceituação", sendo esta uma das definições mais aceitas pela comunidade de Inteligência Artificial. Segundo o autor, uma ontologia representa a especificação de um vocabulário representativo dentro de

um domínio compartilhado, definindo classes, relações, funções e outros objetos. Tendo o desenvolvimento de uma ontologia o objetivo de *compartilhar conhecimento* (GRUBER, 1993a).

GUARINO (1998) propõe o refinamento desta definição apresentada acima, definindo ontologia ligada a outro conceito, o de *compromissos ontológicos*. Segundo o autor, ontologia consiste de uma teoria lógica representando uma significação, a qual objetiva definição de um vocabulário formal, isto é, seu compromisso ontológico para uma conceituação particular do mundo.

Com uma visão um pouco diferenciada, propondo o compartilhamento e reuso de ontologias, GRUNINGER (1996) propõe o uso de ontologias para modelagem de problemas e domínios, onde estas iriam objetivar fornecimento de uma biblioteca para fácil reutilização de classes de objetos para a modelagem. O objetivo fundamental desta proposta é o desenvolvimento de uma biblioteca de ontologias, a qual poderia ser reusada e adaptada a diferentes classes de problema e ambientes.

Em geral, estes 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 (1997) também se refere às ontologias de forma independente de domínio do conhecimento, defendendo a necessidade de se construir ontologias que possam ser reutilizáveis e compartilhadas através de múltiplos serviços e métodos.

Nesta dissertação, por ser uma definição em geral aceita pelos autores da área e não se opor a outras definições dadas, ontologia será definida como uma especificação formal e explícita de uma *conceituação* compartilhada (GRUBER, 1993, FENSEL, 2000, GAVA & MENEZES, 2003), onde, segundo STUDER et al. (1998) e FENSEL (2000), *conceituação* se refere ao modelo abstrato de algum fenômeno do mundo o qual identifica conceitos relevantes do próprio fenômeno; *explícita* significa que o tipo de conceito usado e os requisitos para o seu uso são definidos explicitamente; *formal* se refere ao fato da ontologia ser interpretável por máquina; e *compartilhada* reflete a noção de que uma ontologia captura o conhecimento apresentado não somente por um único indivíduo, mas por um grupo.

Para facilitar a compreensão dos conceitos que foram apresentados até o momento neste capítulo e apresentar outros importantes, apresenta-se abaixo um simples

glossário, sugerido por GUARINO & GIARETTA (1995), para alguns termos:

- **Conceituação:** estrutura semântica intencional, a qual codifica regras implícitas, restringindo a estrutura de parte da realidade.

- **Ontologia Formal:** desenvolvimento sistemático, formal, axiomático da lógica em todas as formas e maneiras de ser.

- **Compromisso ontológico:** uma consideração semântica parcial da conceituação intencionada de uma teoria lógica.

- **Engenharia ontológica:** ramo da engenharia do conhecimento que explora os princípios formais da *Ontologia* para o desenvolvimento de ontologias.

- **Teoria ontológica:** conjunto de fórmulas, as quais devem ser sempre verdadeiras de acordo com uma determinada conceituação.

- **Ontologia:** ramo da Filosofia que trata com a natureza e organização da realidade.

- **ontologia:** (1º sentido) teoria lógica a qual fornece uma descrição explícita e parcial de uma conceituação; (2º sentido) sinônimo de conceituação.

3.1 Motivos para Utilização

Depois de discutidos os conceitos de ontologias, torna-se importante discutir a razão pela qual são utilizadas, com qual finalidade é buscado o uso de ontologias. Refletindo-se um pouco, pode-se chegar a conclusão de que, basicamente, ontologias são aplicadas para possibilitar ou facilitar a comunicação entre diferentes pessoas, aplicações, sistemas, entre outros, os quais fazem parte do mesmo domínio do conhecimento, mas nem sempre compartilham de uma mesma conceituação a respeito dos componentes deste domínio. De acordo com GAVA & MENEZES (2003), esta falta de entendimento compartilhado leva a:

- Uma comunicação pobre entre as pessoas e as organizações;

- Dificuldades na identificação de requisitos e na definição de uma especificação do sistema;

- Problemas para permitir interoperabilidade e possibilidade de reuso e compartilhamento de conhecimento, o que é extremamente importante tendo-se em vista a grande variedade de métodos, paradigmas, linguagens e ferramentas existentes.

Um dos fatores que foram citados acima, a interoperabilidade, é muito buscada onde diferentes pessoas necessitam acessar, prover e trocar dados através do mesmo ambiente, como é o caso dos ambientes *grid* tratados no capítulo 2, ou então utilizar um mesmo *software* e programas de aplicação. Para obtenção desta interoperabilidade, ontologias são usadas para desenvolvimento de modelagens que expressem o conhecimento possuído pelo domínio, formando uma camada de comunicação única e comum a todos os usuários.

A possibilidade de reuso e compartilhamento é possível porque, no momento em que se utilizam ontologias para representação do conhecimento, este se encontra padronizado e expresso em alguma linguagem formal. Desta forma, se torna mais fácil à leitura e interpretação da ontologia por outros domínios, permitindo a sua modificação (inserindo/retirando conceitos, axiomas, relacionamentos) para se adequar a um novo domínio.

A área de Engenharia de Sistemas faz muita utilização de ontologias para obter estas facilidades citadas. Segundo USCHOLD & GRUNINGER (1996), a necessidade de confiabilidade com relação aos conceitos do vocabulário ou linguagem que se está utilizando em certo ambiente é outro forte motivo para sua utilização, pois a representação formal adquirida com a aplicação de ontologias pode tornar possível a automação da checagem de consistência, resultando em ambientes mais confiáveis. Os autores ainda mencionam a possibilidade de especificação como razão para uso de ontologias, pois o conhecimento compartilhado pode ajudar no processo de identificação de requisições e definição de especificações em sistemas de Tecnologia da Informação (TI), o que se torna especialmente verdadeiro quando as requisições

envolvem grupos diferentes usando terminologias diferentes em um mesmo domínio, ou múltiplos domínios. Na Fig. 3.1 encontram-se ilustradas estas razões para utilização de ontologias divididas em categorias.

FIGURA 3.1: Três principais categorias na utilização de ontologias.
Fonte: (USCHOLD & GRUNINGER, 1996).

3.2 Tipos de Ontologias

Ontologias são geralmente classificadas em tipos, os quais podem variar dependendo do autor em questão. Os autores VAN HEIJST (1995), VAN HEIJST et al. (1997), BORST (1997) e STUDER et al. (1998) concordam na existência de quatro tipos de ontologias:

- **Ontologias de domínio**: capturam o conhecimento válido para um tipo particular de domínio (como mecânica, medicina, biologia, entre outros). Expressam o vocabulário relativo a um domínio particular, descrevendo situações reais deste domínio.

- **Ontologias genéricas**: são similares às ontologias de domínio, entretanto os conceitos definidos por elas são considerados genéricos entre diversas áreas. Descrevem conceitos tipicamente gerais, como: estado; espaço; tempo; processo; evento; importância; ação; entre outros, os quais são independentes de um domínio ou problema particular. Conceitos em ontologias de domínio são freqüentemente definidos como especializações de conceitos de ontologias genéricas.

- **Ontologias de aplicação**: contém todos os conceitos necessários para modelagem do conhecimento requerido por uma aplicação em particular. Esses conceitos correspondem freqüentemente aos papéis desempenhados por entidades do domínio enquanto executam certa atividade (GUARINO, 1998).

- **Ontologias de representação**: não se comprometem com nenhum domínio em particular. Determinam entidades representacionais sem especificar o que deve ser representado. Ontologias de domínio e ontologias genéricas são descritas por meio das primitivas fornecidas pelas ontologias de representação.

Em (GUARINO, 1998) é apresentada uma ilustração a respeito dos diferentes tipos de ontologias, de acordo com o seu nível de generalidade, contendo flechas para representar os relacionamentos de especialização entre os tipos. Esta é mostrada na Fig. 3.2. O autor ainda descreve um outro tipo de ontologia:

- **Ontologias de tarefas**: descrevem tarefas ou atividades genéricas (como vendas ou diagnose) através da especialização dos termos introduzidos pelas ontologias genéricas.

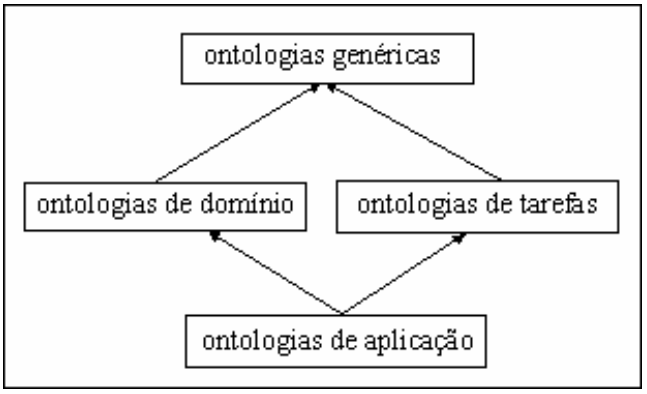

FIGURA 3.2: Tipos de ontologias.
Fonte: (GUARINO, 1998).

Apesar de serem classificadas em diferentes tipos, toda ontologia incluirá necessariamente um vocabulário de termos e especificações de seus significados ou conceitos (USCHOLD & GRUNINGER, 1996). O que varia com relação a esses vocabulários é o nível de formalismo estabelecido antes de sua criação, pois,

dependendo da aplicação para qual é voltado, o nível de formalismo requerido pode ser mais rigoroso ou não, independente do tipo da ontologia desenvolvida.

3.3 Projeto de Ontologias

Para a criação de uma ontologia, o primeiro passo a ser executado é a definição clara do propósito e escopo da sua aplicação. Para isso, é necessário realizar a captura do conhecimento, que se trata de: 1) identificação dos principais conceitos e relacionamentos do domínio de interesse; 2) definição de um texto preciso, definindo estes conceitos e relacionamentos encontrados; 3) definição dos termos usados para se referir a estes conceitos e relacionamentos (USCHOLD & GRUNINGER, 1996).

No momento em que a captura do conhecimento foi realizada é que se deve partir realmente para o projeto da ontologia. Neste ponto é muito importante se ter conhecimento dos principais critérios a serem seguidos para o desenvolvimento do projeto, assim como todos os componentes que deverão ser definidos para se tornar parte da ontologia criada. Depois de definidos os critérios de projeto e os componentes da ontologia, é também importante se definir a linguagem que será utilizada para construção e o ambiente que será usado para o desenvolvimento. Desta forma, nas próximas seções e subseções, serão descritas estas questões citadas, necessárias para o desenvolvimento de ontologias.

3.3.1 Critérios de Projeto

Para axiomatização de uma ontologia que venha a apresentar confiabilidade, seja reutilizável, possibilite compartilhamento, entre outras facilidades já apresentadas, é necessário respeitar alguns critérios durante o seu desenvolvimento (GRUBER, 1993a; USCHOLD & GRUNINGER, 1996; GAVA & MENEZES, 2003).

- **Clareza**: uma ontologia deve expressar efetivamente o significado intencionado dos termos a que define, devendo as definições ser realmente objetivas. Enquanto que a motivação para definição de um conceito pode surgir de uma situação social ou requisição computacional, sua definição deve ser independente de um contexto social ou computacional específico. Formalismo se constitui em um meio para

este fim, pois quando uma definição pode ser declarada na forma de axiomas lógicos, isso deve ser feito. Uma definição completa sempre é preferível a uma definição parcial, e todas as definições devem ser documentadas em linguagem natural e com exemplos, para reforçar a clareza.

- **Coerência**: toda ontologia deve ser coerente, isto é, deve permitir inferências que sejam consistentes com as definições. No mínimo, a definição dos axiomas deve ser consistente. Coerência deve também ser aplicada aos conceitos que são definidos informalmente, como os descritos e exemplificados em linguagem natural. Se uma sentença que pode ser deduzida a partir de axiomas contradiz uma definição ou exemplo dado informalmente, então a ontologia é incoerente.

- **Extensibilidade**: uma ontologia deve ser projetada para antecipar a utilização de um vocabulário compartilhado, devendo fornecer uma representação que possa ser estendida e especializada. Deve ser possível a definição de novos termos para outros usos, baseados no vocabulário existente de uma forma que não exija a revisão das definições existentes.

- **Influências de codificação mínimas**: a conceituação deve ser especificada em nível de conhecimento sem depender de uma codificação de nível simbólico particular. Uma influência de codificação ocorre quando a escolha de representação é feita puramente devido a conveniências de notação ou implementação. Estas influências devem ser minimizadas, pois agentes para compartilhamento de conhecimento podem ser desenvolvidos em diferentes sistemas e estilos de representação.

- **Compromissos ontológicos mínimos**: uma ontologia deve requerer compromissos ontológicos mínimos suficientes para sustentar as atividades planejadas no compartilhamento de conhecimento. Uma ontologia deve fazer um mínimo de imposições possíveis sobre o mundo que está sendo modelado, permitindo liberdade às partes comprometidas com a ontologia, para possibilitar especialização e instanciação quando necessário. Pois, como os compromissos ontológicos são baseados na utilização consistente de um vocabulário, estes

podem ser minimizados através da especificação de uma teoria mais fraca (o que permite definição de mais modelos) e da definição somente dos termos essenciais para a comunicação do conhecimento consistente com essa teoria.

3.3.2 Componentes de Projeto

Para possibilitar a representação de cada elemento presente na ontologia utiliza-se certos componentes, também conhecidos como elementos epistemológicos, os quais são definidos para identificar cada categoria de elementos da ontologia. Em (KIRYAKOV et al., 2001) são apresentados alguns dos componentes que são utilizados em geral para a identificação de elementos da ontologia, os quais se encontram descritos a seguir na forma de conjuntos, separados de acordo com a terminologia adotada pela respectiva área que os utiliza, juntamente com os seus significados dentro de cada área apresentada:

- **Conceitos**, **propriedades**, **relações**: esta terminologia é utilizada no contexto das **Redes Semânticas**, **Matemática** e **Filosofia**. *Conceitos* são usados na descrição de qualquer tipo de fenômeno semântico autônomo estática e cognitivamente, classificando as entidades do domínio considerado, sendo cada entidade pertencente ou não a certa interpretação de conceito. Entidades que pertencem a uma interpretação do conceito são chamadas instâncias do conceito. *Propriedades* representam características, aspectos ou atributos das entidades, como também as relações entre elas. Além disso, propriedades podem ser separadas em *relações* e *atributos*, como exemplos de atributos podem ser citados: idade e cor; enquanto que exemplos de relações podem ser: gostar e motivar.

- **Classes**, ***slots***, ***facets, frames***: como já indicam, estes termos são conceituados dentro de uma terminologia baseada em ***Frames***. C*lasses* correspondem a conceitos, enquanto a noção de instâncias continua sendo a de entidades que pertencem à interpretação dos conceitos. *Slots* correspondem às propriedades, sendo distintos em *template-slots* (classes ou atributos de conceitos nas redes semânticas) e *own-slot* (atributos de instâncias nas redes semânticas). *Template-*

slots são definidos em nível de classe – por exemplo, cor é um *template-slot* da classe carro. Enquanto que *own-slots* associam valores de *template-slots* a certas instâncias de classe como, por exemplo, Cores (Fusca, Verde). *Facets* consistem de propriedades de *slots* como, por exemplo, domínio, abrangência, cardinalidade mínima e documentação.

- **Conceitos, papéis**, **indivíduos**: esta é a terminologia usada na linguagem chamada ***description logic*** (DL), a qual é descendente da linguagem para representação do conhecimento KL-One. Este paradigma é muito próximo ao usado nas Redes Semânticas, entretanto, nesta terminologia *papéis* correspondem às propriedades e *indivíduos* correspondem às instâncias das redes semânticas.

- **Classes, objetos, atributos**: esta é a terminologia usada no paradigma de **orientação a objetos** (OO), o mais popular presente nas aplicações de Engenharia de Software. Neste, *classes* correspondem a conceitos, enquanto que *atributos* correspondem ás propriedades. *Objetos* são sempre instâncias de determinada classe.

- **Coleções, indivíduos, predicados, constantes**: terminologia usada pela comunidade que desenvolve a base de conhecimento Cyc, da Cyc Corp (STEPHEN & LENAT, 2002). De forma simplificada, as *coleções* correspondem aos conceitos, os *indivíduos* correspondem às instâncias e os *predicados* binários – que consistem de um tipo de coleções – correspondem às propriedades. *Constantes* são nomes dados a coleções, indivíduos ou predicados.

Em uma outra visão, independente de uma comunidade específica, GRUBER (1993b) identifica quatro componentes que devem ser definidos para especificação de uma ontologia, que são: classes, relações, funções e axiomas. Estes componentes são definidos da seguinte forma por CORCHO et al. (2000):

- **Classes**: ontologias são geralmente organizadas em taxonomias (as quais podem ser interpretadas como sendo a identificação e a classificação dos termos presentes

na ontologia). Segundo STUDER et al. (1998), muitas vezes a definição de ontologia é diluída, de forma que taxonomias são vistas como ontologias completas, entretanto, ontologias diferem de taxonomias em dois aspectos: ontologias possuem uma estrutura interna rica e refletem o consenso de um grupo. Além disso, ainda segundo os autores, ontologias incluem todas as restrições relevantes entre classes, valores de atributos, instância e relações. *Classes* ou *conceitos* são usados com ampla abrangência, de forma que conceitos podem expressar qualquer coisa sobre a qual alguma coisa é dita e, portanto, também poderia ser a descrição de uma tarefa, função, ação, estratégia, processo de raciocínio, entre outros.

- **Relações**: representam um tipo de interação entre conceitos e o domínio. São formalmente definidos como qualquer subconjunto de um produto de "n" conjuntos, isto é: R: C1 x C2 x C3 x ... x Cn. Como exemplo de relações binárias pode-se considerar: "subclasse-de" e "conectado-a".

- **Funções**: consistem de um caso especial de relações, onde o n-ésimo elemento do relacionamento é único para os n-1 elementos precedentes. Formalmente, funções são definidas como: F: C1 x C2 x C3 x ... x Cn-1 → Cn. Exemplos de funções são: Mãe-de, associando um ou vários filhos à sua mãe, e Preço-de-um-carro-usado, que calcula o preço de um carro usado dependendo do modelo do carro, data de fabricação e quilometragem.

- **Axiomas**: são usadas para modelagem de sentenças que são sempre verdadeiras. São inseridos em uma ontologia por diversos propósitos, como: definição do significado de elementos da ontologia, definição de restrições complexas para valores de atributos, argumentos de relações, entre outros. Possuem como objetivo verificar a exatidão das informações especificadas na ontologia ou chegar a conclusão de novas informações.

3.4 Linguagens

São muitas as linguagens desenvolvidas para construção de ontologias, sendo estas em geral divididas em dois grupos: as linguagens que se baseiam em uma lógica de primeira ordem e as que se baseiam em XML (*eXtensible Markup Language*) e HTML (*Hiper Text Markup Language*), desenvolvidas visando, em princípio, aplicações voltadas à *Web* Semântica.

O primeiro grupo de linguagens é usado basicamente para representação do conhecimento inserido em aplicações baseadas em conhecimento, sendo que algumas delas foram desenvolvidas a partir da adaptação de linguagens para representação do conhecimento já existentes, e outras desenvolvidas especificamente para construção de ontologias (CORCHO et al., 2001). Algumas destas linguagens, que serão descritas nesta dissertação, são: Loom (LOOM, 2003), Ontolingua (GRUBER, 1992), OCML (DOMINGUE et al., 1999) e Flogic (KIFER et al., 1995).

O segundo grupo de linguagens é grandemente usado em ambientes WWW, tendo impacto principal sobre aplicações da *Web* Semântica, a qual consiste de uma extensão da *web* já existente e conhecida, onde as informações são apresentadas a partir de significados bem definidos, possibilitando que pessoas e computadores cooperem mais facilmente entre si para o desenvolvimento de suas tarefas (BERNERS-LEE et al., 2001).

Nesta dissertação serão descritas algumas destas linguagens, das quais: RDF e RDF *Schema* (LASSILA & SWICK, 1999), SHOE (LUKE & JEFF, 2000), XOL (KARP et al., 1999), OML (KENT, 1999), OIL (FENSEL et al., 2000), DAML+OIL (HORROCKS et al., 2002) e OWL (McGUINNESS & VAN HARMELEN, 2004). Segundo CORCHO et al. (2001) estas linguagens, com exceção da linguagem SHOE, a qual tem sua sintaxe baseada em HTML, possuem sintaxe baseada em XML, e são adotadas como linguagens padrão para troca de informações na *web*. Ainda segundo os autores, RDF e RDF *Schema* não podem ser consideradas como linguagens para construção de ontologias, sendo em geral linguagens para especificação de metadados na *web*. Na Fig. 3.3 é apresentada uma pirâmide de linguagens da *Web* Semântica, mostrando os relacionamentos existentes entre elas.

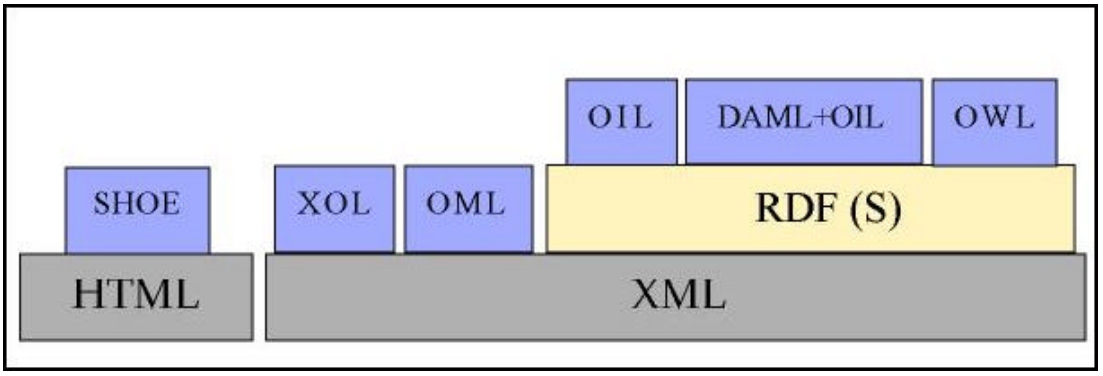

FIGURA 3.3: Pirâmide de linguagens voltadas para *web*.
Fonte: modificado de (CORCHO et al., 2001).

3.4.1 Linguagens em Lógica de 1ª Ordem

Serão apresentadas a seguir as principais linguagens para ontologias baseadas em lógica de 1ª ordem, juntamente com uma breve descrição a respeito de cada uma delas:

- **Loom** consiste de uma linguagem e um ambiente para construção de aplicações, desenvolvida em 1991 pelo ISI (Information Sciences Institute - University of Southern California). Seu ponto principal é o sistema de representação do conhecimento, o qual é usado para prover suporte conclusivo para a porção declarativa da linguagem Loom. Essa porção declarativa consiste de definições, regras definidas pela aplicação e regras definidas de forma padrão e fatos (LOOM, 2003).

  Para seu funcionamento um mecanismo dedutivo, chamado classificador, utiliza proibições, unificações semânticas e tecnologia orientada a objetos com o objetivo de compilar o conhecimento declarativo inserido em uma rede projetada para sustentar de forma eficiente a dedução *on-line* de procedimentos de consulta (LOOM, 2003).

  O Loom ainda permite a representação de conceitos, taxonomias, relações n-árias, funções, axiomas e regras de produção. Nele o raciocínio é limitado a classificações automáticas (taxonomias podem ser criadas automaticamente para a definição de conceitos), checagem de consistência e execução de regras de produção (CORCHO et al. 2001).

Esta linguagem é aceita pelas ferramentas OntoSaurus (ONTOSAURUS, 2003) e Ontolingua (FARQUHAR et al., 1996), que serão descritas na seção 3.5 desta dissertação.

- **Ontolingua** é uma linguagem para descrição de ontologias em uma forma compatível com múltiplas linguagens de representação, provendo suporte para definição de classes, relações, funções, objetos e teorias, e traduzindo definições escritas em uma linguagem declarativa em formas que são aceitas para entrada de dados em diversos sistemas de representação do conhecimento. As ontologias que são construídas a partir do Ontolingua podem ser compartilhadas por múltiplos usuários e grupos de pesquisa, cada um usando seu próprio sistema de representação (GRUBER, 1993b).

  Este sistema foi desenvolvido em 1992 pelo KSL (Knowledge Systems Laboratory - Stanford University), e sua sintaxe e semântica foram desenvolvidas com base na linguagem KIF (Knowledge Interchange Format). KIF é definida em (GENESERETH & FIKES, 1992) como uma linguagem formal para troca de conhecimento entre sistemas computacionais muito diferentes, a qual possui: semânticas declarativas, isto é, o significado das expressões existentes na representação pode ser compreendido sem recorrer a um interpretador para manipular as expressões; é compreensível logicamente, isto é, fornece suporte para expressão de sentenças arbitrárias em cálculo de predicados de primeira ordem; suporte para a representação de regras de raciocínio não-monotônicas; e também fornece suporte para a definição de objetos, funções e relações.

  A linguagem KIF contém uma base de conhecimento clara para o leitor, mas não possibilita o raciocínio automático por meio dela. Desta forma, o Ontolingua traduz definições escritas em KIF para uma forma apropriada, visando desenvolver sistemas de representação do conhecimento (GRUBER 1993b).

- **OCML** (*Operational Conceptual Modelling Language*) é uma linguagem de modelagem desenvolvida em 1993 pelo KMI (Knowledge Media Institute - Open University). Muito similar ao Ontolingua, sendo construída apresentando componentes adicionais a esta, que são: regras dedutivas e de produção e definições operacionais para funções. Permite especificação e operacionalização

de funções, relações, classes, instâncias e regras, além de apresentar uma poderosa checagem de restrições, que podem checar restrições de tipo e de cardinalidade, associadas às relações, *slots* e classes (CORCHO et al., 2001).

As ferramentas WebOnto e Apollo aceitam esta linguagem, estas serão descritas na seção 3.5.

- **FLogic** (*Frame Logic*) é um formalismo que descreve, de maneira limpa e declarativa, os mais diversos aspectos estruturais de linguagens baseadas em *frames* e orientadas a objetos, sendo, além disso, adequado para a definição, execução de consultas e manipulação de esquemas de bancos de dados (KIFER et al., 1995).

  Desenvolvido em 1995 na Universidade de Karlsruhe (Karlsruhe University - Alemanha), integra *frames* em primeira ordem e cálculo de predicados (CORCHO et al., 2001), sustentando o mesmo relacionamento para o paradigma de orientação a objetos que o cálculo de predicados mantém para a programação relacional. Em suas características apresenta: identidade de objetos, objetos complexos, herança, polimorfismo, métodos para consulta, encapsulamento, entre outros. Através dela é possível a representação de: conceitos, taxonomias, relações binárias, funções, axiomas, instancias e regras dedutivas (KIFER et al., 1995).

3.4.2 Linguagens Voltadas para *Web*

Similarmente à seção anterior, serão apresentadas a seguir algumas das linguagens para construção de ontologias voltadas a aplicações para *Web* Semântica, assim como uma descrição a respeito de cada uma delas.

- **RDF** e **RDF *Schema*** (*Resource Description Framework*) são linguagens que possuem como objetivo prover interoperabilidade entre aplicações que trocam informações interpretáveis por máquina na *web*, tendo como ênfase facilitar o processamento autônomo de recursos da *web*, tornando possível a especificação de semânticas de dados baseadas em XML, de uma forma padronizada e interoperável. Uma das áreas em que RDF é bastante aplicada é no descobrimento de recursos, onde é capaz de prover melhores capacidades aos mecanismos de

busca. O maior objetivo da linguagem RDF é definir um mecanismo para descrição de recursos que não faça suposições a respeito de um domínio de aplicação em particular, e que não defina as semânticas de nenhum domínio de aplicação, nesta a definição do domínio deve ocorrer de forma imparcial, mesmo que o mecanismo deva ser apropriado para a descrição de informações a respeito de qualquer domínio (LASSILA & SWICK, 1999).

A linguagem RDF, como muitos outros sistemas de modelagem OO, se baseia em um sistema de classes. Uma coleção de classes é chamada de *schema*, e estas classes são organizadas em uma hierarquia, provendo extensibilidade através das subclasses definidas (LASSILA & SWICK, 1999).

RDF *Schema* provê informações a respeito da interpretação dos enunciados apresentados em um modelo de dados RDF, o que difere do propósito das DTDs (*Document Type Definition*) pertencentes à linguagem XML, as quais provêm restrições específicas a estrutura de um documento XML (BRICKLEY & GUHA, 2000).

RDF e RDF *Schema* são aceitas por várias ferramentas, entre elas pode-se citar o editor Protégé-2000, que será descrito na seção 3.5 desta dissertação.

- **SHOE** (*Simple HTML Ontology Extensions*) é uma extensão da linguagem HTML, a qual fornece uma maneira de incorporar conhecimento semântico interpretável por máquina em documentos HTML, podendo também ser usado em documentos XML (LUKE & JEFF, 2000).

  A linguagem SHOE foi desenvolvida na Universidade de Maryland, em 1996, ela permite a representação de conceitos e suas taxonomias, relações n-árias, instâncias e regras de dedução, os quais são usados pelo seu mecanismo de inferência para obter novo conhecimento (CORCHO et al., 2001).

- **XOL** (*Ontology Exchange Language*) é uma linguagem para ontologias baseada em XML desenvolvida inicialmente para uso somente pela comunidade de bioinformática. Apesar de possibilitar atualmente utilização por qualquer domínio, é uma linguagem bastante restrita, a qual permite especificação somente de conceitos, taxonomias e relações binárias (CORCHO et al. 2001). Foi

desenvolvida no SRI International (Stanford Research Institute) em 1999 (SRI, 2004).

- **OML** (*Ontology Markup Language*) é uma linguagem para especificação de ontologias desenvolvida na Universidade de Washington em 1999. Esta linguagem apresenta uma estrutura ontológica e semântica, onde a estrutura ontológica é composta por: classes, relacionamentos, objetos e restrições. Possui como base a descrição lógica e conceitual da estrutura da ontologia na forma de gráficos, permitindo a representação de conceitos, definidos a partir de suas taxonomias, axiomas e relações em lógica de 1ª ordem (KENT, 1999).

- **OIL** (*Ontology Inference Layer*) consiste de uma proposta para representação e camada de inferência de ontologias voltadas a *web*, combinando as primitivas de modelagem usadas para linguagens baseadas em *frames* com as semânticas formais e os serviços de raciocínio providos pelas lógicas de descrição. OIL foi a primeira linguagem para representação de ontologias fundamentada corretamente nos padrões estabelecidos pela W3C (*World Wide Web Consortium*), como são as linguagens RDF/RDF *Schema*. Ela unifica três aspectos importantes vindos de diferentes entidades: semânticas formais e suporte eficiente ao raciocínio, providos pelas lógicas de descrição; primitivas epistemológicas de modelagem ricas providas pelos sistemas baseados em *frames*; e uma proposta padronizada para troca de notações sintáticas, como provido pela *web* (FENSEL et al., 2000).

- **DAML+OIL** (DARPA *Markup Language* + *Ontology Inference Layer*) é uma linguagem semântica voltada a aplicações ligadas a recursos da *web*. Resultado da união das linguagens DAML e OIL, ela foi desenvolvida por um grupo de pesquisadores, europeus e norte-americanos, possuindo conformidade com os padrões de linguagem estabelecidos pela W3C, assim como RDF e RDF *Schema*. Em sua linguagem para construção de ontologias, DAML+OIL tem o objetivo de descrever a estrutura de um domínio, apresentando estrutura OO com a estrutura do domínio sendo descrita em termos de classes e propriedades. Além disso, esta linguagem permite a representação de conceitos, taxonomias, relações binárias, funções e instâncias (HORROCKS et al., 2002).

- **OWL** (*Ontology Web Language*) é uma linguagem para especificação de ontologias, a qual recentemente foi recomendada como um padrão de linguagem pela W3C. Esta linguagem apresenta todos os benefícios de outras linguagens para ontologia, como por exemplo: DAML+OIL, RDF e RDF *Schema*, e outras mais, devido a esta consistir de uma revisão, incorporando já alguns melhoramentos necessários, da linguagem DAML+OIL. Um dos melhoramentos é a criação de um vocabulário mais extenso para descrição de propriedades e classes, permitindo descrição de: relacionamentos entre classes, cardinalidade, igualdade, características de propriedades, entre outras (McGUINNESS & VAN HARMELEN, 2004). A OWL é dividida em três sub-linguagens, distintas pelo nível de formalidade exigido e oferecido e liberdade dada ao usuário para a definição de ontologias: OWL *Lite*, OWL DL e OWL *Full*.

3.4.3 Comparativo entre as Linguagens

Para permitir melhor compreensão a respeito das linguagens descritas nas subseções anteriores, nesta seção serão apresentadas duas tabelas, contendo uma comparação geral entre as linguagens de acordo com certas características. A primeira tabela, Tab. 3.1, consta das linguagens especificadas nesta dissertação como linguagens em lógica de 1ª ordem, enquanto que a segunda tabela, Tab. 3.2, consta das linguagens especificadas como voltadas para *web*.

Tanto na primeira como na segunda tabela, as linguagens são comparadas de acordo com a existência ou não das seguintes características: conceitos; relações binárias; funções; instâncias; taxonomias; axiomas e restrições. As informações presentes nas tabelas significam: “S” denota que tal característica está presente na linguagem; “N” denota que não está; “S/N” denota que a característica não está presente, mas pode ser simulada através de simulações na linguagem; o caractere “?” indica que não foi encontrada nenhuma informação a respeito.

TABELA 3.1: Comparativo entre as linguagens em lógica de 1ª ordem.

Característica	LOOM	Ontolíngua	OCML	FLogic
Conceito	**S**	**S**	**S**	**S**
Relações binárias	**S**	**S**	**S**	**S**
Funções	**S**	**S**	**S**	**S**
Instâncias	**S**	**S**	**S**	**S**
Taxonomias	**S**	**S**	**N**	**S**
Axiomas	**S**	**S**	**S**	**S**
Restrições de tipo	**S**	**?**	**S**	**?**
Restrições de cardinalidade	**S**	**?**	**S**	**N**

TABELA 3.2: Comparativo entre as linguagens voltadas à *web*.

Característica	RDF-S	SHOE	XOL	OML	OIL	DAML-OIL	OWL
Conceito	**S**	**S**	**S**	**S**	**S**	**S**	**S**
Relações binárias	**S**	**S**	**S**	**S**	**S**	**S**	**S**
Funções	**?**	**N**	**S**	**?**	**S**	**S**	**S**
Instâncias	**S**	**S**	**S**	**?**	**S/N**	**S**	**S**
Taxonomias	**S**	**S/N**	**S**	**S**	**S**	**S**	**S**
Axiomas	**?**	**N**	**S**	**S**	**N**	**N**	**S**
Restrições de tipo	**S**	**S**	**S**	**S**	**S**	**S**	**S**
Restrições de integridade	**N**	**N**	**?**	**?**	**N**	**N**	**S**

Segundo as Tabs. 3.1 e 3.2 apresentadas, é possível perceber que, dentre todas as descritas, as que oferecem todas as características usadas para comparação são as linguagens LOOM e OWL.

3.5 Ambientes para Desenvolvimento

Nesta seção é apresentada uma visão geral de alguns ambientes voltados ao desenvolvimento de ontologias. Serão apresentadas as ferramentas: Apollo, OILEd, Servidor Ontolingua, OntoSaurus, Protégé-2000 e WebOnto.

- **Apollo**: é uma aplicação amigável para a modelagem de ontologias. No ambiente, a modelagem é desenvolvida através de primitivas básicas, como classes, funções, relações, instâncias, entre outras. A sua modelagem interna é construída de acordo com um sistema de *frames*, além disso, a ferramenta Apollo fornece checagem de consistência na medida em que a ontologia é desenvolvida. A ferramenta não obriga utilização de uma linguagem específica para a criação de ontologias e pode ser adaptada a diversos formatos de armazenamento, por meio da utilização de *plug-ins*. A Fig. 3.4 mostra a tela principal do Apollo, a janela superior esquerda mostra a representação hierárquica de ontologias, enquanto a janela inferior esquerda mostra a representação hierárquica de classes e instâncias da ontologia. A ferramenta é desenvolvida em linguagem *Java* e possui uma arquitetura aberta (APOLLO, 2003). Aceita as linguagens OCML, Ontolingua, entre outras.

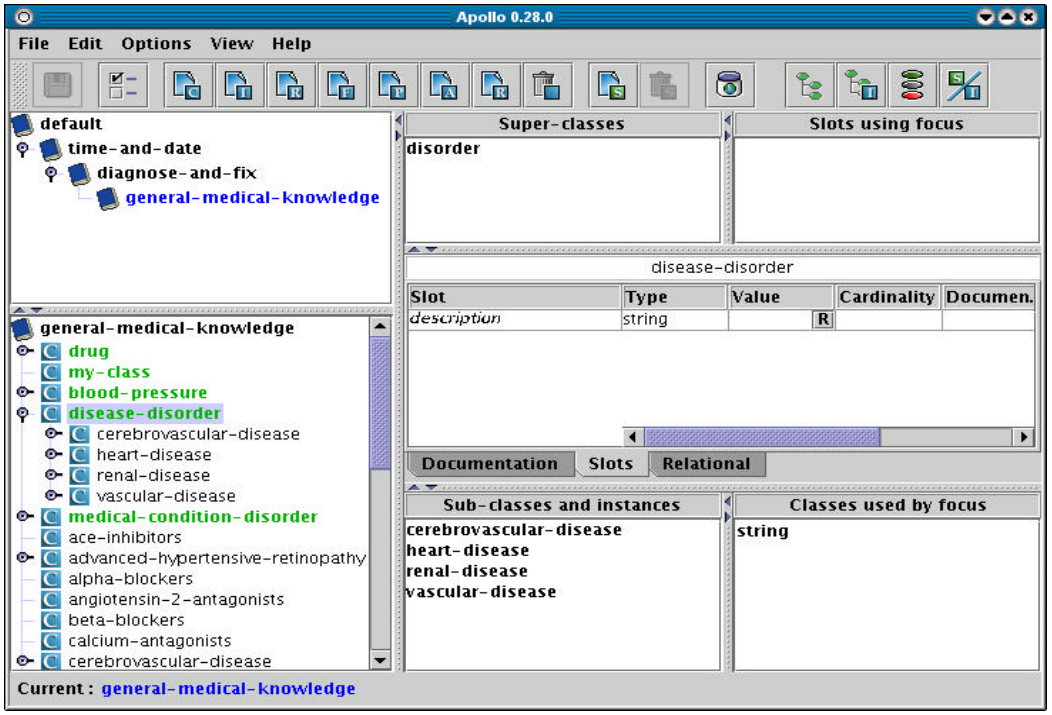

FIGURA 3.4: Tela principal da ferramenta Apollo.

- **OILEd**: é uma ferramenta simples para edição de ontologias baseadas em linguagem OIL e DAML+OIL. Em sua funcionalidade básica, a ferramenta permite a definição e descrição de classes, *slots*, entidades e axiomas, onde classes são definidas em termos de suas superclasses e restrições de propriedade. Uma inovação apresentada pelo OILEd é a utilização de raciocínio para checagem de consistência entre as classes e para inferência de classificação entre relacionamentos. Este serviço de raciocínio é provido pelo sistema FaCT, o qual consiste de um classificador de ontologias via tradução de DAML+OIL para lógica de descrição (BECHHOFER et al., 2001). A ferramenta é desenvolvida em linguagem *Java* e é disponível para uso através de licença GPL (*General Public License*). A Fig. 3.5 mostra a tela do OILEd responsável pela definição de classes. Esta ferramenta trabalha com as linguagens: RDF, RDF *Schema*, OIL e DAML+OIL.

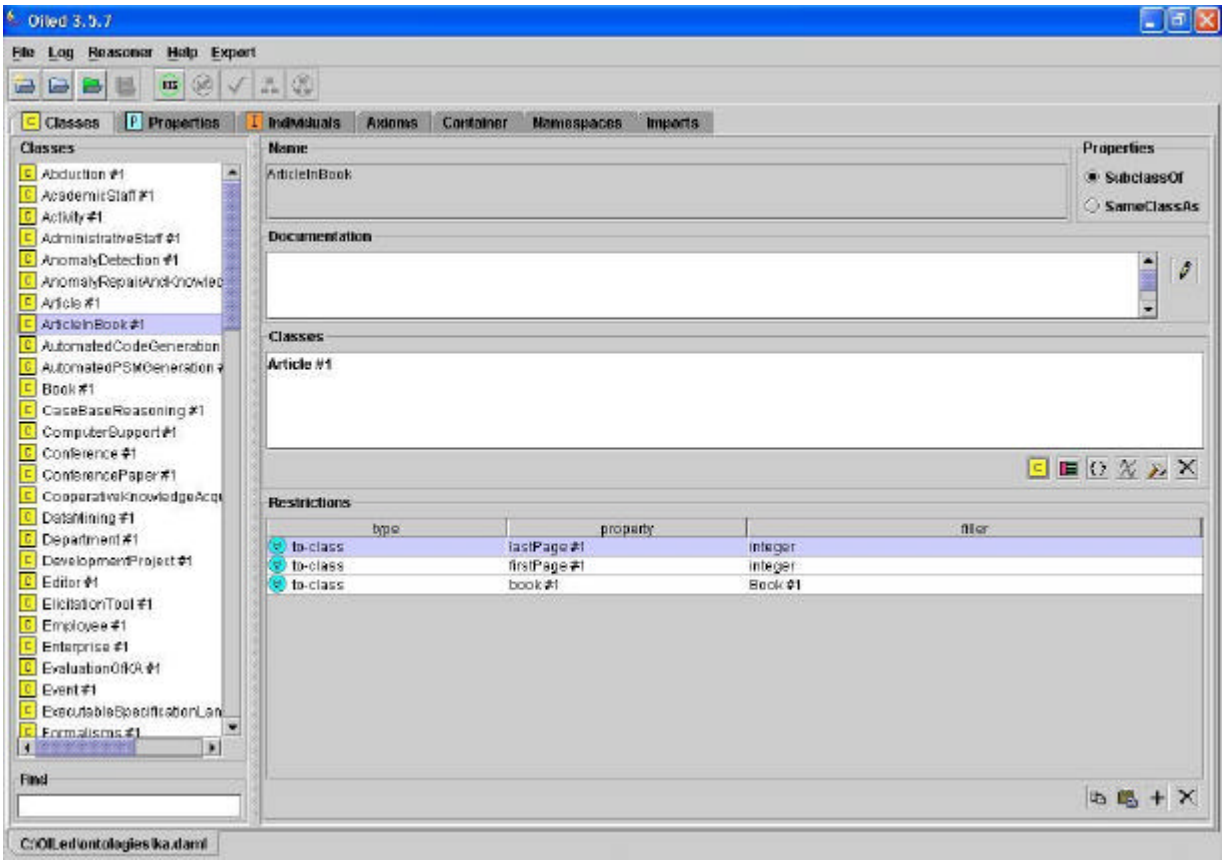

FIGURA 3.5: Painel para definição de classes do editor OILEd.

- **Servidor Ontolingua**: criada pelo KSL (Knowledge Systems Laboratory – Stanford University), consiste de um conjunto de ferramentas e serviços que suportam a construção de ontologias compartilháveis entre grupos geograficamente distribuídos (FARQUHAR et al., 1996). A arquitetura do servidor de ontologias fornece acesso a bibliotecas de ontologias, tradutores de linguagens e um editor para criação. Editores remotos podem visualizar e editar ontologias, aplicações remotas ou locais podem acessar qualquer ontologia das bibliotecas, através do protocolo OKBC (Open Knowledge Based Connectivity) (CORCHO et al., 2001). O Servidor Ontolingua aceita as linguagens: Loom, Ontolingua, entre outras.

- **OntoSaurus**: é um servidor *web* para ontologias criadas a partir da linguagem Loom, desenvolvido pelo ISI (Information Sciences Institute - University of Southern California). Este servidor consiste de um dos seus dois módulos existentes, sendo o segundo módulo um servidor de navegação de ontologias, o qual cria dinamicamente paginas em formato HTML (incluindo textos e imagens) para exibição da hierarquia de uma ontologia. Na Fig 3.6 é apresentada a tela inicial do servidor *web*, fornecido *on-line* pelo OntoSaurus, onde é mostrada a definição da linguagem LOOM, usada pelo servidor, assim como o significado de todos os ícones fornecidos e outras convenções que devem ser conhecidas para criação de ontologias.

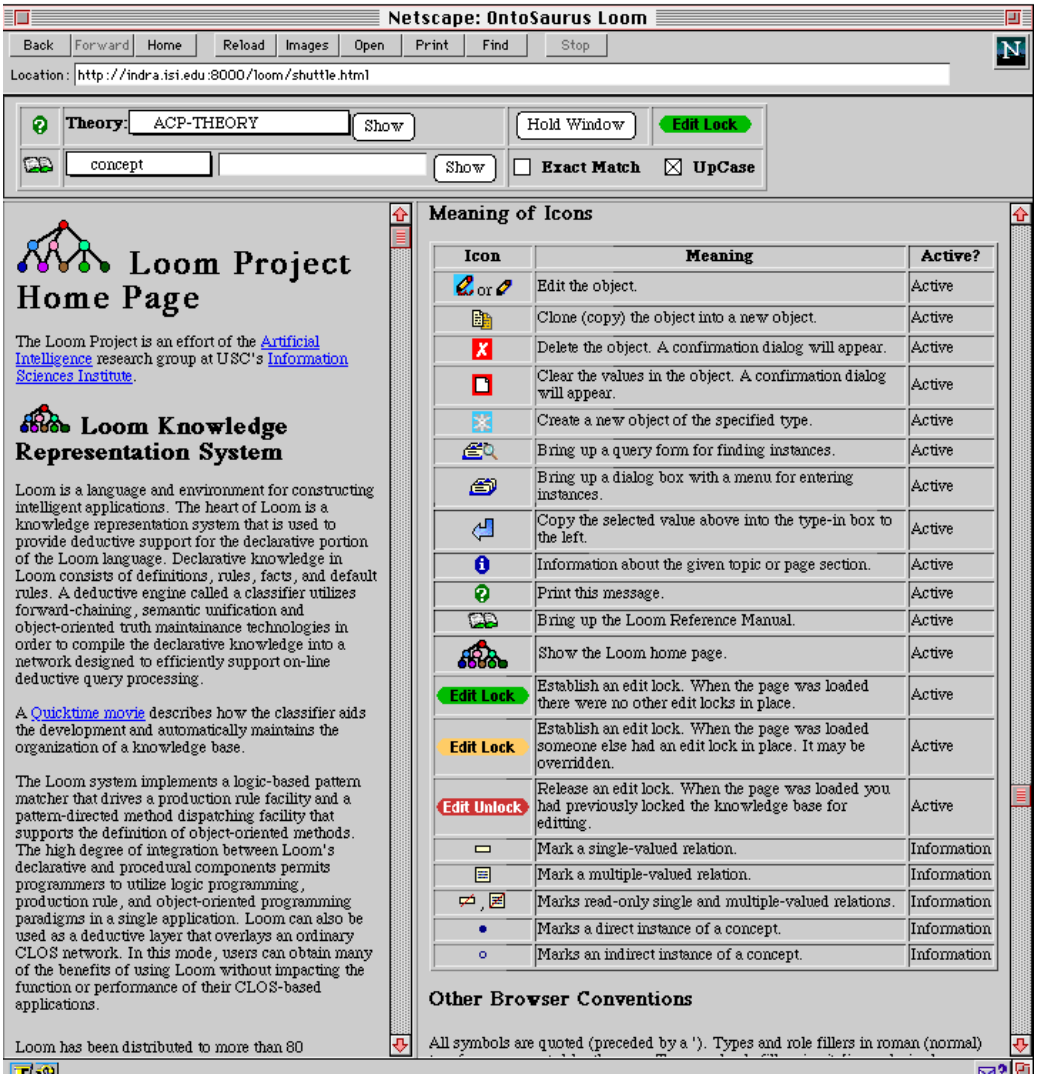

FIGURA 3.6: Servidor *web* OntoSaurus.

- **Protégé-2000**: consiste de um editor onde são descritos os conceitos pertencentes à ontologia, juntamente com seus atributos e relacionamentos. O editor Protégé-2000 possui uma arquitetura de *metaclasses*, documentos de formato padrão usados para definir novas classes em uma ontologia, que o tornam facilmente extensível e permite o seu uso juntamente com outros modelos de conhecimento. O editor Protégé-2000 foi desenvolvido tendo em vista dois objetivos: (1) apresentar interoperabilidade com outros sistemas de representação do conhecimento e (2) ser uma ferramenta para aquisição do conhecimento fácil de ser usada e configurável (NOY et al., 2000). O Protégé-2000 foi desenvolvido na Universidade de Stanford, e é disponível para utilização gratuitamente. Fornece um editor gráfico amigável para o desenvolvimento de ontologias, nele é possível

acessar informações importantes de forma rápida, podendo-se manipular diretamente a ferramenta para navegação e gerenciamento da ontologia. Na Fig. 3.7 se encontra a tela principal para descrição de classes, instâncias, propriedades de classes, entre outras tarefas do Protégé-2000.

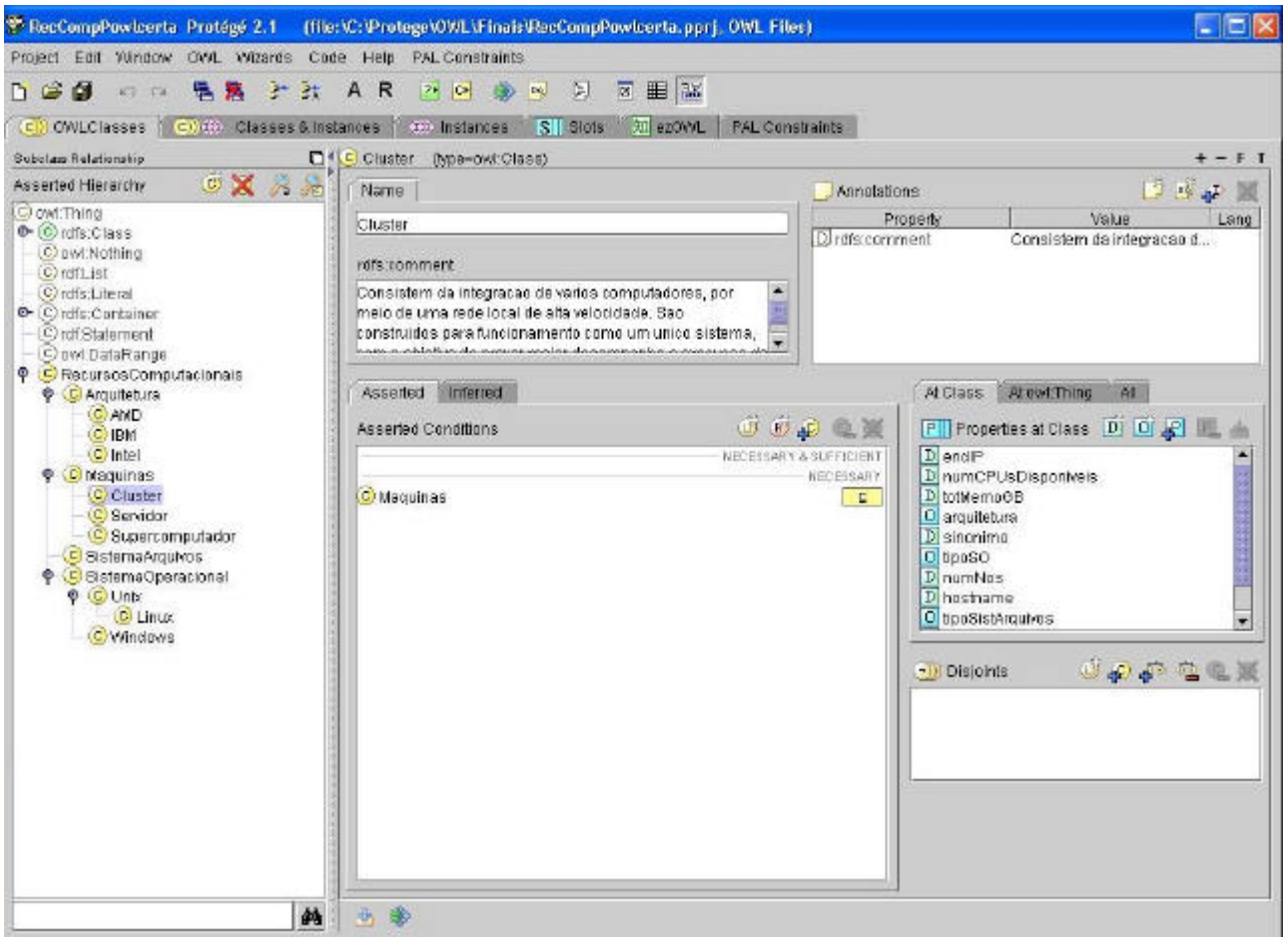

FIGURA 3.7: Tela para descrição de classes do Protégé-2000.

- **WebOnto:** ferramenta para criação de ontologias desenvolvida pelo KMI (Knowledge Media Institute – Open University). Ela suporta navegação colaborativa, criação e edição de ontologias, as quais são representadas na linguagem OCML. Suas principais características são: gerenciamento de ontologias utilizando uma interface gráfica; suporte para modelagem de tarefas; verificação de elementos, considerando herança de propriedades e checagem de consistência; suporte ao trabalho colaborativo. WebOnto é um servidor disponível gratuitamente, através dele é possível o acesso a mais de 100 ontologias, as quais podem ser visualizadas sem restrições de acesso (CORCHO et al., 2001

4. Ontologia para Descrição de Recursos do *Grid*

4.1 Trabalhos Correlatos

4.1.1 Seleção de Recursos em *Grids*

Não foram encontrados trabalhos contendo desenvolvimento completo de ontologias para descrição e seleção de recursos do *grid*. Em (TANGMUNARUNKIT et al., 2003) é apresentado o protótipo de um sistema para seleção automática de recursos (*matchmaker*) em ambiente *grid* baseada em ontologias. Neste sistema, o autor propõe a criação de ontologias para, declarativamente, descrever os recursos e as restrições de tarefas. Com o uso de ontologias, o autor objetiva facilitar e automatizar a tarefa de seleção dos recursos, pois esta seria feita utilizando termos definidos na ontologia criada. Este trabalho correlato utiliza linguagem RDF para definição da ontologia e de suas regras, sendo distinto do tema apresentado nesta dissertação pela forma com que as ontologias são usadas. Pois, na presente dissertação, o consumidor atua diretamente nas ontologias, já que o objetivo central é tornar o acesso ao *grid* transparente. Enquanto que, no trabalho correlato, as ontologias são usadas pela aplicação desenvolvida no momento de seleção dos recursos.

4.1.2 Ontologias em *Grids*

Com relação à aplicação de ontologias em *grids*, existem diversos trabalhos sendo realizados. Tendo em vista o destaque que estão apresentando entre a comunidade científica, dois deles serão resumidamente apresentados a seguir:

- *Semantic Grid*: consiste de uma nova infra-estrutura de *grid*, a *Semantic Grid*, proposta para suportar todas as atividades necessárias à comunidade pertencente a *e-Science* (DE ROURE et al., 2003). Neste contexto, a *Semantic Grid* é vista como um sistema aberto, onde cientistas e pesquisadores, das mais diversas áreas, podem acessar todos os recursos computacionais do *grid* de forma facilitada, podendo processar experimentos, verificar resultados, tanto de suas pesquisas

quanto de outros cientistas que utilizam o ambiente, e também armazenar suas pesquisas para serem acessadas por toda a comunidade. Segundo (DE ROURE et al., 2003), esta infra-estrutura seria formada por três camadas.

- *Data/computation*: trata da alocação dos dados, escalonamento e execução dos recursos, estrutura da rede computacional e interconexão entre equipamentos;
- *Information*: trata da forma com que a informação é representada, armazenada, acessada, compartilhada e mantida. Onde informação entende-se por ser os dados acompanhados de seus significados no contexto;
- *Knowledge*: trata da forma com que o conhecimento é obtido, usado, buscado, publicado e mantido. Onde conhecimento significa a informação aplicada para obtenção de um objetivo, resolução de um problema ou tomada de decisão.

Na *Semantic Grid*, ontologias são aplicadas em toda estrutura da camada *Knowledge*, onde determinam a extensão dos termos e os relacionamentos entre eles. Nesse ambiente, elas são usadas para: conceituação de objetos, com suas propriedades e relacionamentos entre si; conceituação de tarefas e processos, com suas inter-relações; conceituação dos atributos que determinada propriedade do conhecimento possui e seus inter-relacionamentos; conceituação dos atributos relevantes para estabelecimento do valor de um conteúdo; conceituação das características importantes para estabelecimento de modelos ou perspectivas (visões); definição de anotações, as quais podem relatar as razões que levaram a aquisição de certo conteúdo, porque foi modelado desta maneira e quem o sustenta ou discorda (DE ROURE et al., 2003).

- *Earth System Grid* (ESG): neste projeto, POUCHARD et al. (2003) tratam a respeito da crescente funcionalidade das ontologias no contexto de computação em *grid* voltadas a aplicações científicas, e apresenta uma ontologia desenvolvida para o domínio científico em geral, com o objetivo de prover uma base para classificação e busca de dados, economizando tempo, recursos computacionais e trazendo maior transparência aos usuários. Apresenta ainda que, para facilitar o processo de tomada de decisão por parte dos cientistas, após a busca dos arquivos de dados estes são transferidos aos pontos de acesso dos usuários (POUCHARD

et al., 2003). Para a criação da ontologia, foi utilizada a linguagem DAML-OIL, tratada no capítulo 3 desta dissertação, e para sua edição foram usados os editores OILEd e Protégé-2000, também tratados no mesmo capítulo.

4.1.3 Serviços de Informação

Com relação ao propósito da descrição de recursos em um ambiente, existe grande semelhança entre o tema desta dissertação e os serviços de informação, os quais registram as principais características e propriedades dos recursos e serviços de um ambiente para que, a partir destas, seja possível descoberta, seleção, consulta, publicação e agregação de seus recursos e serviços. Um exemplo de serviço de informação correlato com esta dissertação é o *Metacomputing Directory Service* (MDS), integrante do *Globus Toolkit*, mencionado no capítulo 2, o qual provê uma interface uniforme a diversas fontes de informação a respeito de recursos e serviços de um ambiente *grid*, possuindo toda a sua estrutura de diretórios, representação de dados e APIs definida a partir de LDAP *(Lightweight Directory Access Protocol)* (FITZGERALD et al., 1997). Entretanto, com o objetivo de suprir alguns requisitos não possíveis pelo MDS, foi proposto, em junho de 2000, o MDS-2, o qual consiste de um melhoramento do MDS, apresentando um componente provedor de informações configurável, chamado GRIS (*Grid Resource Information Service*), e um componente de diretórios agregados configurável, chamado GIIS (*Grid Index Information Service*) (CZAJKOWSKI et al., 2001).

Tanto no MDS quanto no MDS-2, a primeira interação é feita pelo fornecedor do recurso, o qual irá publicar em um serviço de informação, no caso, MDS, as informações a respeito do recurso ou serviço que está provendo. Desta forma, quando o consumidor necessitar de um serviço/recurso irá executar uma consulta ao MDS, o qual irá fornecer como resposta uma listagem dos serviços/recursos disponíveis, cabendo ao consumidor escolher qual utilizar. Neste momento, ainda não é assegurado o acesso ao consumidor, pois somente após o envio da requisição é que serão averiguadas as políticas de uso, que irão depender do fornecedor do serviço/recurso em questão.

O fato de estes sistemas suportarem somente consultas simples e não ser possível realizar descrições mais sofisticadas e expressivas a respeito dos serviços/recursos é um dos motivos pelos quais se propõe a utilização de ontologias em *grids*.

4.2 Motivação

Como já foi apresentado no capítulo de Introdução desta dissertação, ambientes de *grids* computacionais apresentam problema com relação à busca e acesso dos seus recursos fornecidos aos consumidores, por motivo da inexistência de modelos padronizados para geração de informações a respeito do estado dos recursos e suas condições para acesso. Tal fato ocasiona transtorno aos consumidores, exigindo destes um conhecimento maior a respeito das políticas de acesso ao ambiente e consumindo tempo por não serem facilmente obtidas tais informações. Neste contexto, destaca-se a inclusão de ontologias, as quais, como já detalhado no capítulo 3, constituem-se de estruturas semânticas adequadas a representação do conhecimento, fornecendo um vocabulário claro e objetivo para compreensão das características e propriedades das classes existentes em um domínio.

Desta forma, a presente dissertação propõe como alternativa para solução do problema enfrentado em *grids* computacionais a padronização das informações referentes aos recursos através da utilização de ontologias. Estas são utilizadas, primeiramente, para descrever a sintática e semântica dos recursos disponíveis no *grid* na forma de um vocabulário comum aos integrantes do domínio e interpretável por máquina. Em seguida, a ontologia é usada para apresentação dos axiomas, os quais consistem das restrições impostas pelos fornecedores, com relação a inserção de novos recursos no ambiente e também disponibilização destes para acesso pelos consumidores. A partir do uso de ontologias, não existe a necessidade de preocupação com os requisitos existentes por parte do consumidor, pois estes irão visualizar todas as informações pertinentes ao recurso a que desejam acessar, e, de acordo com estas informações, concluem se irão ou não utilizar o recurso.

A ontologia, proposta nesta dissertação, consiste da representação do conhecimento apresentado pelo domínio a que ela se destina: a comunidade que fornece/consome recursos pertencentes à *grids* computacionais. Este conhecimento é representado na forma de um vocabulário, o qual utiliza axiomas para o seu funcionamento. Além destes axiomas da ontologia, outras duas estruturas auxiliam na descrição dos recursos do *grid*, que são os **metadados** e as **visões semânticas**.

Metadados consistem de informações a respeito dos dados. No caso do trabalho, metadados armazenam informações a respeito dos recursos, essas informações são: data em que foi posto em funcionamento; por quanto tempo estima-se que esteja disponível para acesso; endereçamento do recurso, para acesso; capacidade do recurso, em espaço de armazenamento de dados e memória disponível para processamento; qual sistema operacional está em execução; a qual tipo de arquitetura pertence; entre outras informações. Por outro lado, as **visões semânticas** guardam informações sobre o estado atual do recurso. Sempre que uma visão semântica é consultada, ela devolve informações sobre o funcionamento do recurso neste momento, isto é, se está disponível ou não para uso, se está em sobrecarga de processamento, se está em manutenção, entre outros possíveis estados.

Na proposta, tanto os metadados como as visões semânticas são usados como uma referência adicional à ontologia, a qual sempre irá consultar estas duas estruturas para obter informações que possam ter sido alteradas no sistema do *grid* e ainda não colocadas como novos conceitos da ontologia. Portanto, sempre que um novo recurso for adicionado ou retirado do *grid*, essa informação deverá imediatamente ser colocada nos metadados, pois estes devem sempre refletir a real situação dos recursos. As visões semânticas, por serem geradas a cada momento pelo sistema, refletem sempre a situação instantânea do recurso computacional.

A ontologia proposta foi desenvolvida utilizando-se a linguagem OWL, a qual foi descrita na seção 3.4 do capítulo 3. Optou-se pelo seu uso devido ao fato de apresentar todas as funcionalidades de outras linguagens, como mostrado na Tab. 3.2 do capítulo 3, incluindo o fato de ser um padrão de linguagem para criação de ontologias, reconhecido pela W3C. Entre as três sub-linguagens pertencentes à OWL, optou-se pela utilização da OWL *Full* por esta oferecer um bom nível de formalidade e liberdade, uma vez que, para realização da proposta apresentada, foi necessária uma liberdade adicional para possibilitar a definição, na ontologia, das políticas de uso dos recursos, traduzidas na forma de axiomas.

Para a edição da ontologia proposta, foi usado o editor Protégé-2000, onde são descritos os conceitos pertencentes à ontologia, juntamente com seus atributos e relacionamentos. Optou-se pela utilização deste editor, primeiramente, por ser um *software* livre e disponível para as plataformas: *Windows*; *LINUX*; Mac OS (*Macintosh*

Operational System); AIX; *Solaris*; HP UX (*Hewlett-Packard UNIX*); e diversas plataformas *UNIX*, o que em geral faz com que tenha um grande número de usuários e, por conseqüência, suporte mais facilmente obtido.

Em segundo lugar, por ser um dos editores com maior quantidade de *plug-ins* disponíveis, apresentando novas funcionalidades. Estes *plug-ins* são desenvolvidos em geral por empresas ou indivíduos que utilizam o editor Protégé e sentem a necessidade de agregar novas funcionalidades a ele. Como exemplos destes *plug-ins*, podem-se citar: *UML Storage Backend* – um *plug-in* voltado ao armazenamento de bases de conhecimento, geradas no Protégé, em formato UML (*Unified Modeling Language*); *XML Tab Widget* – plug-in que possibilita a geração de ontologias a partir de arquivos em formato XML e também a geração de arquivos XML a partir de ontologias criadas no Protégé; *OWL Storage Backend* – *plug-in* utilizado neste trabalho, que permite a criação e edição de ontologias em linguagem OWL no Protégé; além de muitos outros *plug-ins* voltados aos mais diversos propósitos.

4.3 Arquitetura do *Grid* Baseada em Ontologias

A ontologia proposta nesta dissertação deve atuar diretamente no serviço de diretórios do *grid*, onde as consultas a respeito dos recursos são feitas sobre o vocabulário definido pela ontologia. Portanto, observou-se a necessidade de propor também uma mudança na arquitetura para o ambiente *grid*, adicionando-se a ela a camada aonde se localizam as ontologias, além das camadas responsáveis por alocar os metadados e as visões semânticas definidas.

Na nova arquitetura, são eliminadas possíveis ocorrências de interpretações ambíguas na busca e na leitura de informações a respeito do ambiente, pois todos os conceitos que serão usados pelas aplicações, tanto dos consumidores como dos fornecedores, como referência para apresentação e busca de informações sobre o sistema, possuem apenas um significado. Neste vocabulário criado pelas ontologias, são armazenados todos os significados dos conceitos definidos no domínio. A partir de um significado, são iniciadas buscas de subdomínio em subdomínio, através dos relacionamentos entre os conceitos, para que se venha a definir o resultado da busca.

Observando-se a Fig. 4.1, modificada de (GOBLE & DE ROURE, 2002), é possível perceber que a ontologia se localiza em uma camada quase que à parte do *grid*, pois as requisições vindas dos consumidores devem primeiramente fazer uma consulta à ontologia, que por sua vez utiliza os metadados e as visões semânticas para obter maior informação a respeito dos recursos. Os metadados e as visões semânticas, que recebem informações vindas diretamente dos recursos, arquivos de dados, entre outros, devolvem as informações consultadas pela ontologia e, então, esta devolve a resposta ao consumidor. Somente no momento em que o consumidor obtiver a resposta relativa à sua requisição é que o recurso é ou não fornecido para utilização.

Nesta nova arquitetura, observa-se o destaque ao serviço, voltado a *grids* computacionais, MDS, mencionado no capítulo 2, subseção 2.4.4, e no início deste capítulo. Pois, como a função deste serviço é a de prover informações a respeito da estrutura e estado dos componentes do sistema, percebeu-se a possibilidade das visões semânticas acessarem este serviço para obter as informações de que necessitam. Portanto, a Fig. 4.1 mostra flechas apontando para estes dois componentes, indicando sua troca de dados.

Na Fig. 4.1, as flechas nos dois sentidos, Ontologia - Metadados e Visões Semânticas – Recursos Computacionais, Catálogos e Arquivos de Dados, ilustram a troca de informações entre estes tanto para obter a informação pedida em uma consulta quanto para devolver a resposta consultada.

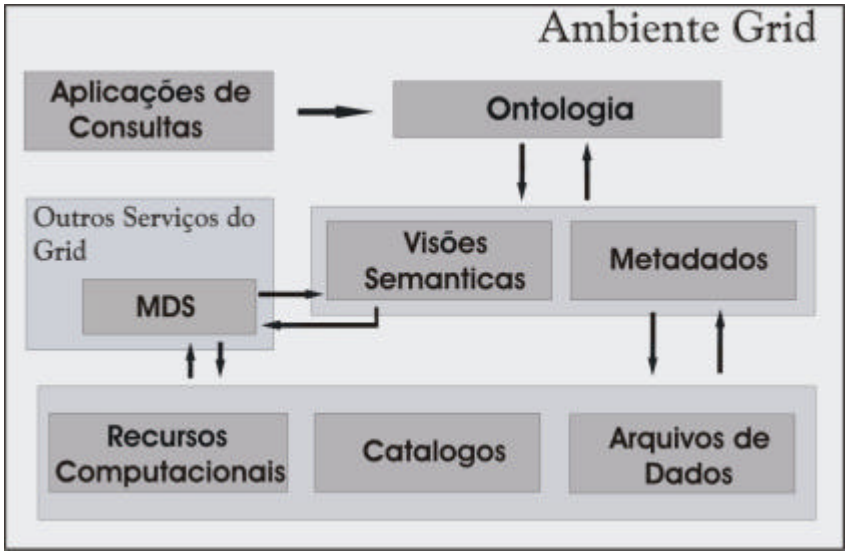

FIGURA 4.1: Arquitetura proposta para o *grid*.

4.4 Metodologia

Antes de se iniciar a construção de uma ontologia, é importante fazer uma boa investigação a respeito da metodologia a ser utilizada, pois para se garantir que a ontologia criada poderá ser reutilizada posteriormente em outros sistemas, deve-se adotar o uso de uma metodologia que garanta sua portabilidade. Desta forma, para que a ontologia criada fosse mais facilmente adaptável para descrição de novos recursos, sua definição foi feita de acordo com a metodologia indicada por GRUBER (1993b), onde é descrito um mecanismo para definição de ontologias portáteis entre diferentes sistemas de representação, sendo estas mais facilmente adaptadas aos problemas encontrados em vários outros sistemas, o que permite sua reutilização.

Por se buscar, então, desenvolver uma ontologia reutilizável, é necessário um principal cuidado com relação à forma com que irão ocorrer os processos de planejamento e de documentação da ontologia. Desta forma, a definição destas tarefas foi feita com base nos passos descritos por GOMÉZ-PÉRES et al. (1996) para o ciclo de vida de definição e padronização de uma ontologia. Estes passos estão resumidamente apresentados abaixo:

- **Primeiro Passo**: captura do conhecimento de um dado domínio e desenvolvimento de um documento de especificação dos requisitos;

- **Segundo Passo**: conceituação do conhecimento capturado em um conjunto de representações intermediárias, com o objetivo de sumarizar o conhecimento. Neste passo são realizadas diversas atividades, as quais não possuem uma ordem específica para serem efetivadas. Estas atividades são:
  - Identificar os conceitos, suas instâncias, atributos e seus valores em um *Dicionário de Dados*;
  - Classificar os grupos de conceitos em uma *Árvore de Classificação de Conceitos*;
  - Descrever as constantes da ontologia em uma *Tabela de Constantes*;
  - Descrever os atributos de instâncias e os atributos de classe em *Tabelas de Atributos de Instâncias* e *Tabelas de Atributos de Classe*;
  - Descrever as instâncias em *Tabelas de Instâncias*;

- Caso a ontologia possua valores numéricos inferidos a partir de atributos, descrever as fórmulas usadas para obtê-los em uma *Tabela de Fórmulas*;
- Reunir a seqüência inferida dos atributos em *Árvores de Classificação de Atributos*;

- **Terceiro Passo**: desenvolvimento do modelo conceitual em uma linguagem formal, como as linguagens: Ontolingua, RDF, entre outras, descritas no capítulo 3;

- **Quarto Passo**: avaliação da ontologia com relação às referências usadas durante cada fase e entre as fases do ciclo de vida.

Todos estes passos foram realizados para o desenvolvimento da ontologia, o qual é descrito na próxima seção.

4.5 Desenvolvimento da Ontologia

Primeiramente, é importante se considerar que a ontologia desenvolvida visa exclusivamente à descrição de recursos computacionais providos em ambientes de *grid*, consistindo este ambiente, portanto, de um *grid* voltado a serviços para o fornecimento de recursos computacionais. Entretanto, nada impede que esta ontologia descrita seja estendida para outros ambientes de *grids* computacionais, com outros propósitos que não o de fornecimento de recursos computacionais. Foi escolhido este escopo em específico por se acreditar ser o mais associado aos objetivos destes ambientes. Com relação aos serviços necessários ao fornecimento real de recursos, novos conceitos deveriam ser introduzidos, pois existiria a necessidade de agregar informações relativas ao envio destes recursos por meio de uma rede de computadores.

4.5.1 Obtenção e Documentação do Conhecimento

Para o desenvolvimento da ontologia, primeiramente foi feita uma pesquisa a respeito dos conceitos que deveriam ser obtidos para descrição geral de recursos computacionais, isto é, qual o vocabulário utilizado pela comunidade quando se deseja expressar do que é composto determinado recurso computacional, e quais recursos são geralmente disponibilizados em ambientes de *grid*.

Para realização deste primeiro passo, foram pesquisados diversos ambientes de *grid* computacionais existentes e em pleno funcionamento. Na pesquisa realizada, foram estudados os seguintes ambientes:

- **NPACI** – este projeto (mencionado no início do capítulo 2 desta dissertação) tem como objetivo a criação de uma infra-estrutura computacional em *grid*, voltada à pesquisa científica. Liderado pela Universidade da Califórnia e pelo Centro Supercomputacional de San Diego (SDSC), conta com recursos computacionais, como supercomputadores e *clusters* computacionais, providos pelas: Universidade do Texas; Universidade de Michigan; Universidade de Berkeley e Universidade de Santa Bárbara – Califórnia; Instituto de Ciências da Informação-Universidade do Sul da Califórnia e Universidade de Virgínia (NPACI, 2000).

- **ESG** (*Earth System Grid*) – consiste de um projeto, financiado pelo Departamento de Energia dos Estados Unidos (U.S. DOE), para desenvolvimento de um ambiente de *grid* computacional, com o objetivo de prover alto poder computacional de forma transparente, possibilitando nova geração de pesquisas climáticas. Este ambiente conta com supercomputadores, servidores e *clusters* computacionais de alto desempenho, fornecidos pelos seguintes centros norte-americanos: Laboratório Nacional de Oak Ridge (ORNL); Laboratório Nacional de Argonne (ANL); Laboratório Nacional de Lawrence Berkeley (LBNL); Centro Nacional de Pesquisas Atmosféricas (NCAR); Instituto de Ciências da Informação-Universidade do Sul da Califórnia (ISI) (FOSTER, et al., 2003).

- ***NASA's Information Power Grid*** (IPG) – consiste de um ambiente para fornecimento de computação em alto desempenho e *grids* de dados, possível pela integração de computadores, bases de dados e instrumentos geograficamente

localizados. Fundado pelo NASA Ames Research Center (ARC), possui apoio dos seguintes centros, os quais fornecem diversos recursos computacionais de alto desempenho: NASA Glenn Research Center (GRC); NASA Langley Research Center (LRC); Partnerships for Advanced Computational Infrastructure (PACI); Laboratório Nacional de Argonne (ANL); Instituto de Ciências da Informação (ISI) (IPG, 2002).

- ***The Distributed ASCI Supercomputer Project 2*** (DAS - 2) – consiste de um ambiente experimental para desenvolvimento de aplicações paralelas e distribuídas, desenvolvido pela Advanced School for Computing and Imaging (ASCI). O ambiente DAS consiste de diversos *clusters* computacionais, os quais somam um total de 200 nós. Estes *clusters* estão localizados nas seguintes Universidades: Vrije Universiteit Amsterdam; Leiden University; University of Amsterdam; Delft University of Technology; University of Utrecht (VERSTOEP, 2002).

No momento em que foram obtidos os conceitos relativos ao domínio de conhecimento, foi iniciado o segundo passo, com o objetivo de gerar toda a documentação necessária para a criação real da ontologia. Esta documentação consiste dos seguintes elementos:

- **Dicionário de Dados** - agrega todas as classes e algumas das instâncias de classe da ontologia, juntamente com suas significações. É apresentado na Tab. 4.1, onde se pode observar que a todas as classes da ontologia foi atribuído um conceito e, como algumas classes não possuem um conceito bem fundamentado e conhecido por todos os pesquisadores da área de *grid*, buscou-se descrevê-los de forma puramente teórica. Para alguns destes conceitos não foram encontradas referências formais, adotando-se, desta forma, um conceito mais coloquial.

- **Árvore de Classificação de Conceitos** - agrupa todas as classes e subclasses da ontologia, apresentada na Fig. 4.2;

- **Tabelas de Atributos de Instâncias** e de **Atributos de Classe** - apresentam, respectivamente, para cada instância e para cada classe da ontologia, todos os seus

atributos. Exemplos destas são mostrados nas Tabs. 4.2 - referente aos atributos da instância IBM Power4 - e 4.3 – referente aos atributos da classe *Cluster*;

- **Tabelas de Instâncias** - apresentam a descrição, atributos e valores de cada instância da ontologia. A tabela referente à instância **IBM** é mostrada na Tab. 4.4.

- **Árvores de Classificação de Atributos –** mostram, na forma de um gráfico, todos os atributos que são inferidos através da existência de outros atributos hierarquicamente superiores. Na Fig. 4.3 é apresentada uma destas árvores criadas, referente aos atributos *tipoSO + arquitetura -> IBM-SP24*, pertencentes às classes **SistemaOperacional** – **Arquitetura** e **Supercomputador**

Não foram geradas as **Tabelas de Constantes** e as **Tabelas de Fórmulas**, que constam na metodologia usada, pois não foi identificada nenhuma constante e também nenhuma fórmula na ontologia desenvolvida.

TABELA 4.1: Dicionário de Dados da ontologia proposta.

Conceito	Descrição	Instancias	Atributos de Classes	Atributos de Instâncias
Recursos Computa-cionais	Classe principal, que contém especificações relacionadas à: Sistemas Operacionais, Sistemas de Arquivos e Arquitetura dos recursos computacionais fornecidos pelo *grid*.	----	----	----
Arquite-tura	Conceito relativo ao padrão aplicado à estrutura interna dos recursos computacionais e demais componentes do *grid*.	----	nome fabricante	----
Sistema Operacio-nal	Camada de software localizada entre o hardware e os programas que executam tarefas para os usuários. Responsável por realizar o acesso aos periféricos e tornar a utilização do computador mais eficiente e conveniente (OLIVEIRA et al., 2000).	----	nome	----

Conceito	Descrição	Instancias	Atributos de Classes	Atributos de Instâncias
Sistema Arquivos	Parte do Sistema Operacional (SO) responsável pelo gerenciamento geral dos discos rígidos e pelo armazenamento lógico dos dados no disco. Dentre outras tarefas, determina como arquivos são armazenados, alterados, copiados ou removidos do disco (OLIVEIRA et al., 2000).	EXT2 EXT3 FAT16 FAT32 NTFS NFS ReiserFS	nome espacoDisco	----
Tipode-Maquina	Elementos físicos usados para execução de processamento em ambientes de *grids* computacionais.	----	nome dataFunc tempoEstim	----
Unix	SO comercial e corporativo, muito usado em organizações. Um dos primeiros SOs criados e um dos mais antigos ainda em uso. Serviu como base para criação de muitos outros SOs, entre eles o Linux.	AIX Solaris SunOS	nome tipo	versao maxCPUProcess maxMemProcess sinonimo
Windows	SO proprietário. Desenvolvido e fornecido pela empresa americana Microsoft.	WindowsNT	nome	versao maxCPUProcess maxMemProcess sinonimo
Linux	SO baseado no SO Unix. Foi desenvolvido por meio da Licença Geral e Pública GNU, sendo fornecido de forma livre e de código aberto. (LINUX, 2004).	Conectiva RedHat Fedora Debian Mandrake Slackware Suse	nome tipo	versao maxCPUProcess maxMemProcess sinonimo
AMD	Classe referente a um tipo específico de processadores computacionais, desenvolvidos e fornecidos pela empresa AMD – *Advanced Micro Devices*.	AthlonCluster OpteronCluster	nome fabricante	tipoProcessador modelo sinonimo
IBM	Classe que agrega os processadores computacionais que possuem seu tipo de arquitetura desenvolvido pela empresa IBM - *International Business Machines*	Power4 SP	nome fabricante	tipoProcessador modelo sinonimo

Conceito	Descrição	Instancias	Atributos de Classes	Atributos de Instâncias
Intel	Classe referente aos processadores computacionais pertencentes à arquitetura Intel.	BladeSBX44 Itanium2	nome fabricante	tipoProcessador modelo sinonimo
Cluster	Consiste da integração de vários computadores, geralmente tratados como *nós*, apresentando funcionamento como um único sistema. A configuração visa um maior desempenho com a agregação de vários processadores, o uso de algoritmos mais otimizados e a facilidade da computação paralela (DANTAS, 2002).	ClusterIBM ClusterIBMSP ClusterLinux CrayDellCluster ClusterWindows	nome dataFunc tempoEstim	hostname endIP arquitetura tipoSO tipoSistArquivos numNos numTotProcess numProcesPorNo numCPUsDispon totMemoGB totEspDiscoGB sinonimo
Servidor	Estrutura computacional mais robusta e segura do que os computadores pessoais, utilizada para fornecer serviços computacionais a outros computadores interligados a ela por meio de uma rede de computadores.	IBMPower4	nome dataFunc tempoEstim	hostname endIP arquitetura tipoSO tipoSistArquivos numTotProcess numCPUsDispon totMemoGB totEspDiscoGB sinonimo
Supercomputador	Consistem de sistemas computacionais compostos por diversos processadores, os quais executam suas funções de forma intensivamente paralela, buscando alto desempenho na realização de tarefas. Possuindo arquitetura multiprocessada, realiza acesso de maneira uniforme a memória, estando ela localizada de forma central ou distribuída.	CrayT3E	nome dataFunc tempoEstim	hostname endIP arquitetura tipoSO tipoSistArquivos numTotProcess numCPUsDispon totMemoGB totEspDiscoGB sinonimo

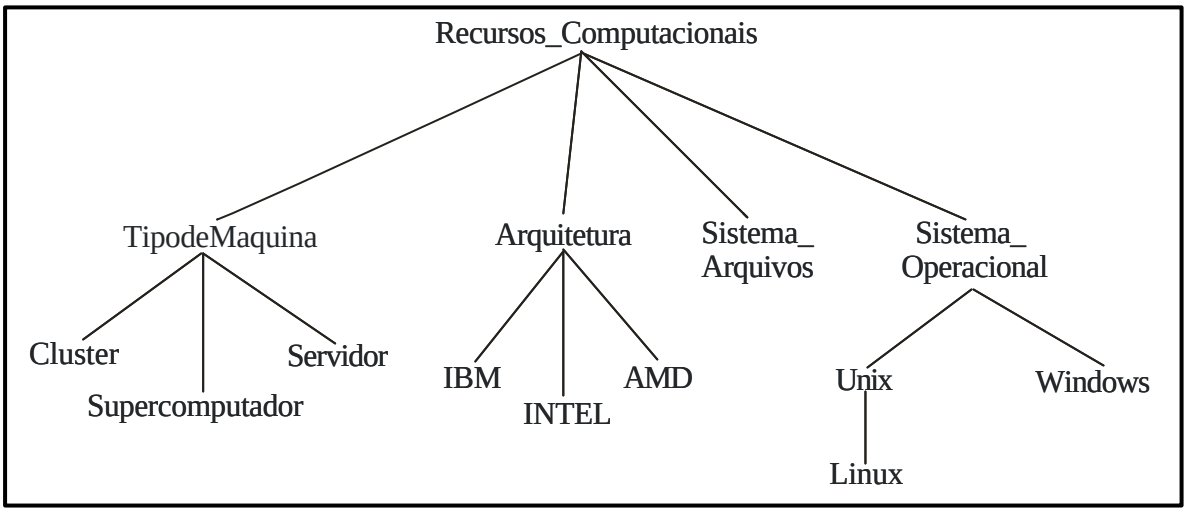

FIGURA 4.2: Árvore de Classificação de Conceitos

TABELA 4.2: Tabela de Atributos de Instâncias: instância *IBM Power4*.

nome	IBM Power4
dataFunc	10/03/2004
tempoEstim	12 meses
hostname	lab.teste.grad
endIP	129.116.97.200
arquitetura	IBMPower4
tipoSO	AIX
tipoSistArquivos	EXT3
numTotProcess	224
numCPUsDispon	36
totMemoGB	512
totEspDiscoGB	7134
sinonimo	Máquina servidora IBM Power 4

TABELA 4.3: Tabela de Atributos de Classe: conceito *Cluster*.

Nome Atributo	Relacionamento Lógico	Valor Mínimo	Tipo	Unidade de Medida
hostname	=	1	Caracter	-----
endIP	=	1	Caracter	-----
arquitetura	=	1	Instância	-----
tipoSO	=	1	Instância	-----
tipoSistArquivos	=	1	Instância	-----
numNos	>=	1	Inteiro	-----
numTotProcess	>=	2	Inteiro	-----
numProcesPorNo	>=	1	Inteiro	-----
numCPUsDispon	>=	1	Inteiro	-----
totEspDiscoGB	-----	-----	Inteiro	GigaBytes (GB)
totMemoGB	-----	-----	Inteiro	GigaBytes (GB)
sinonimo	-----	-----	Caracter	-----

TABELA 4.4: Tabela de Instâncias: instância *IBM*.

Nome Instância	Descrição	Atributos	Valores
IBM	Classe que agrega os processadores computacionais que possuem seu tipo de arquitetura desenvolvido pela empresa IBM - *International Business Machines*	nome	-----
		fabricante	IBM
		modelo	[“eSeries pServer”, “Scalable Power Systems – SP”]
		tipoProcessador	[“Power3”,“Power4”]
		sinonimo	-----

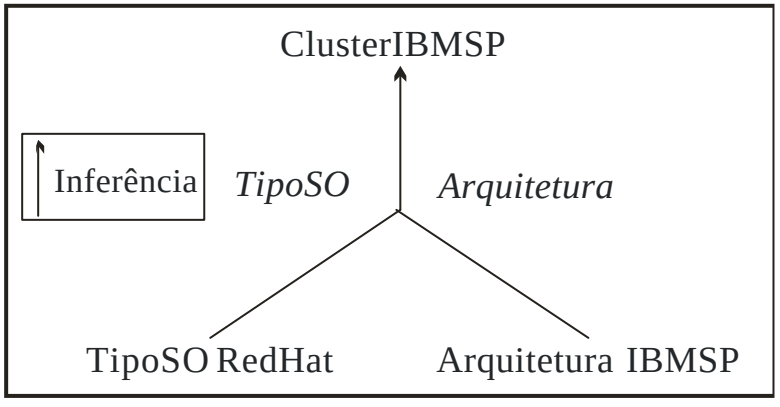

FIGURA 4.3: Árvore de Classificação de Atributos: domínio *Cluster*.

4.5.2 Representação do Conhecimento

No terceiro passo, foi feita a reprodução do conhecimento, obtido nos passos anteriores, em linguagem OWL, onde foram representadas todas as classes e instâncias nesta linguagem formal. Esta reprodução foi feita por meio do editor Protégé-2000, combinado ao *plug-in* responsável pelo reconhecimento da linguagem OWL, neste foi possível se descrever todas as classes da ontologia, suas instâncias, atributos e seus relacionamentos. Um pequeno trecho do código OWL desenvolvido é mostrado na Fig. 4.4, a qual descreve a criação da classe *Maquinas* e de suas subclasses *Cluster* e *Supercomputador*. Algumas telas do editor Protégé, mostrando a ontologia desenvoldida, são mostradas no Anexo II desta dissertação.

```
<owl:Class rdf:ID="TipodeMaquinas">
 <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      > Elementos fisicos usados para execucao de processamento em ambientes de
      grids computacionais.</rdfs:comment>
  <rdfs:subClassOf>
     <owl:Class rdf:about="#RecursosComputacionais"/>
 </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Cluster">
 <rdfs:subClassOf>
      <owl:Class rdf:about="#Maquinas"/>
 </rdfs:subClassOf>
 <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Consiste da integracao de varios computadores, geralmente tratados como nos,
      apresentando funcionamento como um unico sistema. A configuracao visa um
      maior desempenho com a agregacao de varios processadores, o uso de algoritmos
      mais otimizados e a facilidade da computacao paralela. </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="Supercomputador">
  <rdfs:subClassOf>
     <owl:Class rdf:about="#Maquinas"/>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype=http://www.w3.org/2001/XMLSchema#string
      > Consistem de sistemas computacionais compostos por diversos processadores,
      os quais executam suas funcoes de forma intensivamente paralela, buscando alto
      desempenho na realizacao de tarefas. Possuindo arquitetura multiprocessada,
      realiza acesso de maneira uniforme a memoria, estando ela localizada de forma
      central ou distribuida.</rdfs:comment>
</owl:Class>
```

FIGURA 4.4: Trecho de código da ontologia criada.

No editor Protégé-2000 também foram criados os axiomas da ontologia, utilizando-se a linguagem PAL (*Protégé Axiom-Language*), a qual é facilmente incorporada ao Protégé por meio de um *plug-in*, chamado *Pal Tab Widget* (CRUBÉZY, 2002). Os axiomas, no princípio, foram criados para garantir a consistência das instâncias a serem inseridas à ontologia. Na Fig. 4.5 encontra-se o axioma desenvolvido para garantir que o atributo "*tipoSO*", pertencente a qualquer instância da classe *Supercomputador*, não tenha, ao mesmo tempo, valores referentes à classe *Windows* e à classe *Linux*.

```
(defrange ?Supercomputador :FRAME Supercomputador)
(defrange ?tipoSO :FRAME Windows tipoSO)
(defrange ?tipoSO :FRAME Unix tipoSO)

    (forall ?Supercomputador
        (forall ?Windows
            (forall ?Unix
                (not (and ('tipoSO' ?Supercomputador ?Windows)
                          ('tipoSO' ?Supercomputador ?Unix))))))
```

FIGURA 4.5: Axioma criado para consistência da classe *Supercomputador*.

Foi também criado um axioma, mostrado na Fig. 4.6, para garantir os valores mínimos dos atributos "*numTotProcess*" e "*numProcesPorNo*", apresentados pelas instâncias da classe *Cluster*, uma vez que o número total de processadores existentes em um *Cluster* não pode ser maior do que o número de processadores localizados em um único nó do *cluster*.

```
(defrange ?Cluster :FRAME Cluster)

    (forall ?Cluster
        (> ( 'numTotProcess' ?Cluster)
            ('numProcesPorNo' ?Cluster)))
```

FIGURA 4.6: Axioma criado para consistência da classe *Cluster*.

Não houve a necessidade de serem criados outros axiomas relevantes, pois o editor Protégé-2000 controla a entrada de dados nas propriedades das instâncias através das regras definidas no momento da definição da ontologia, como: cardinalidade; intervalo de valores aceitos; quais classes devem ser aceitas durante a inserção de valores de propriedades, onde estas são instâncias de classe, por exemplo, a propriedade "*arquitetura*", existente em instâncias das classes *Cluster*, *Servidor* e *Supercomputador*, é definida como uma **instância de classe**, portanto, somente são permitidos os valores vindos das classes *AMD, IBM* e *INTEL* para inserção nesta propriedade; entre outros.

Na seqüência, foi criado um axioma relativo às restrições, determinadas pelos consumidores dos recursos do *grid*, que devem ser atendidas pelo ambiente para que o recurso seja acessado. Tendo em vista que a aplicação desenvolvida nesta dissertação, para apresentar o ambiente ao consumidor, mostra toda a configuração do recurso computacional pesquisado, não seria necessário desenvolver este axioma. Entretanto, este foi criado porque a *interface* existente em outros ambientes pode ser apresentada de outra forma, na qual seja necessário o uso de axiomas. O axioma, apresentado na Fig. 4.7, foi criado de acordo com os conceitos definidos na ontologia proposta, nele é definido que o consumidor deseja acessar qualquer recurso computacional, desde que o recurso apresente *Sistema Operacional UNIX AIX*, tenha, no mínimo, 40 GBs de *espaço em disco* e 64 GBs de *memória*.

```
(defrange ?tipodemaquina :FRAME TipodeMaquina tipoSO)
(defrange ?unix :FRAME Unix)

   (findall ?tipodemaquina
      (exists ?unix
         (and ('tipoSO' ?unix ?tipodemaquina)
              (= ('nome' ?AIX) ('tipoSO' ?tipodemaquina))
              (exists ?tipodemaquina
                 (and
                    (> ('totEspDiscoGB' ?tipodemaquina) 40)
                    (> ('totMemoGB' ? tipodemaquina) 64))))))
```

FIGURA 4.7: Axioma responsável pela restrição de acesso do consumidor.

Os metadados da ontologia também foram definidos utilizando-se o editor Protégé-2000 e a linguagem OWL, os quais foram definidos conceituando cada uma das estruturas da ontologia, sejam elas classes, instâncias ou atributos. Desta forma, se torna mais simples ao consumidor, ao acessar o ambiente, saber como interagir com ele e qual o significado de cada conceito apresentado durante a interação. Além disso, como comentado no início deste capítulo, os metadados auxiliam a própria ontologia a obter todas as informações referentes ao recurso requisitado pelo consumidor. É importante salientar que as informações contidas nos metadados devem ser alteradas de forma manual pelo gerenciador do ambiente a cada alteração que possa ocorrer, o que não se aplica às visões semânticas. A forma com que os metadados são apresentados aos consumidores pode ser vista na próxima subseção, quando é descrita a aplicação desenvolvida para esta dissertação.

4.6 Apresentação da Ontologia aos Consumidores

A ontologia foi proposta com o objetivo de tornar o acesso ao ambiente de *grid* mais fácil e transparente aos consumidores, entretanto, existe ainda a necessidade de se determinar como e onde, exatamente, esta ontologia será apresentada aos consumidores. Neste ponto, assumiu-se como forma mais eficiente para representação da ontologia a criação de um serviço, pois, como já foi visto no capítulo 2, todo tipo de interação realizada sobre o ambiente *grid* somente é possível por meio de serviços, os quais são desenvolvidos de acordo com o escopo do ambiente. Tendo em vista este objetivo, primeiramente foi desenvolvida uma aplicação, utilizando-se a linguagem *Java*, para servir como *interface* no acesso a ontologia por meio do serviço. Em seguida, foi criado o serviço propriamente dito, que será acessado pelos consumidores.

4.6.1 Aplicação Desenvolvida

A aplicação, descrita nesta subsecção, foi criada visando possibilitar a interação do consumidor com o ambiente e com a ontologia, por meio do serviço, de forma prática e simples. Dessa forma, foi desenvolvida uma *interface*, a qual acessa diretamente a ontologia e permite que o consumidor realize consulta aos seus

componentes, tendo, assim, uma visão mais abrangente de todos os recursos computacionais existentes no ambiente e de toda a sua configuração.

Ainda com o objetivo de tornar o entendimento claro, por parte do consumidor, é possível a visualização dos metadados criados para a ontologia, os quais são apresentados através do rótulo "Documentação", listado na *interface*. Este rótulo está presente em toda a estrutura da ontologia, descrevendo: classes, instâncias de classes e propriedades.

Foi escolhida a linguagem de programação *Java* para desenvolvimento da aplicação, principalmente porque a partir desta linguagem é possível a interação com as APIs do Protégé-2000, uma vez que este editor é totalmente desenvolvido em linguagem *Java.* Utilizando-se as APIs, foi possível desempenhar toda a programação de forma tradicional, sem haver a necessidade de se criar rotinas novas para interpretarem os dados existentes no editor.

No desenvolvimento do programa foi utilizado o editor para criação de programas em linguagem *Java*, Eclipse (ECLIPSE, 2004). Este foi escolhido após pesquisa realizada a respeito da interação de ontologias, desenvolvidas no Protégé-2000, com outros *softwares*, na qual foram encontradas bibliotecas, desenvolvidas pelo centro americano Mayo Clinic - College of Medicine, que possibilitam executar, dentro do editor Eclipse, o editor Protégé-2000. Desta forma, não é necessário trabalhar com os dois editores ao mesmo tempo já que, através do Eclipse, torna-se possível iniciar a utilização do Protégé-2000 e realizar alterações em qualquer ontologia desenvolvida neste editor. Em MAYO (2004), são descritos os benefícios do *software* Eclipse e como ele pode ser adaptado para permitir a inclusão de novas APIs, diferentes das nativas à linguagem *Java.*

Utilizando-se as bibliotecas citadas acima, foi possível a interpretação das APIs do editor Protégé-2000 pela máquina virtual *Java.* Além destas, foi criada uma nova classe, para possibilitar a utilização das APIs normalmente no programa e para que fossem criados os métodos responsáveis pela apresentação na *interface*: dos dados da ontologia, das consultas de classes e propriedades, e das buscas de recursos requeridas pelo consumidor. Para desenvolvimento destes métodos são necessários comandos específicos da API do Protégé, os quais definem como deve ser feita a manipulação dos dados contidos em ontologias, da mesma forma que as APIs nativas da linguagem *Java.*

Na Fig. 4.8 são mostrados alguns dos comandos para manipulação de estruturas da ontologia, no caso, responsáveis pela listagem dos *slots* e instâncias de uma classe.

```
...
Collection cls3 = cls2.getDocumentation();
listagem.addElement(" Classe: " + cls2.getName());
listagem.addElement("  Documentação: "+cls2.getDocumentation());
      Iterator pesqI = cls2.getDirectInstances().iterator();
Iterator ind3 = cls2.getTemplateSlots().iterator();
while (ind3.hasNext()) {
      Slot slot = (Slot) ind3.next();
      Collection values = slot.getDocumentation();
      listagem.addElement("   " + slot + " :  " + values);
}
listagem.addElement(" Instancias da classe " + cls2.getName()+":");
while (pesqI.hasNext()) {
      Instance instance = (Instance) pesqI.next();
      listagem.addElement("   Instancia: " + instance.getName());
}
```

FIGURA 4.8: Alguns comandos para manipulação de dados da ontologia.

A aplicação está dividida em quatro módulos, os quais são descritos a seguir:

1º) **Apresentação**: mostrada na Fig. 4.9, consiste da tela inicial da *interface* desenvolvida, onde é feita a apresentação do ambiente, explicando quais tipos de recursos são fornecidos e informando a respeito da existência de uma ontologia específica, voltada ao domínio de *grid*, usada para apresentação dos conceitos. Na apresentação também são descritas as funcionalidades das demais telas existentes.

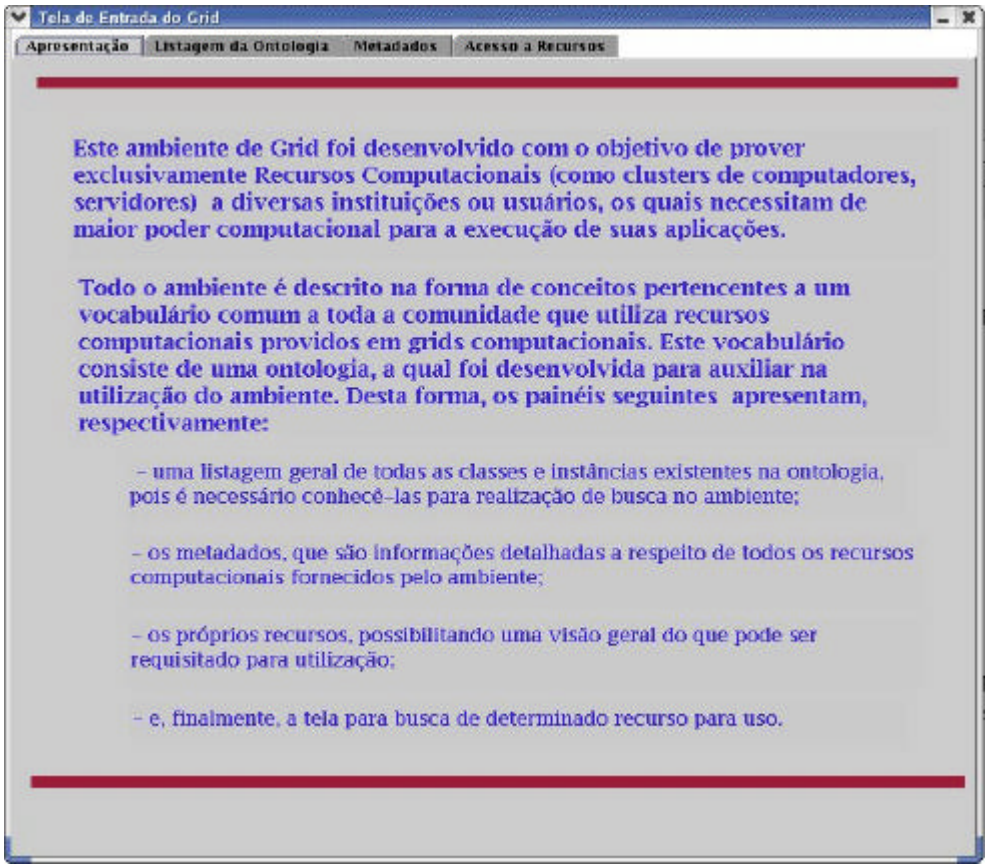

FIGURA 4.9: Tela para apresentação do ambiente ao consumidor.

2º) **Listagem da Ontologia**: realiza a listagem geral de todas as classes e instâncias definidas na ontologia, sendo a partir destes nomes de classes e instâncias apresentados que o consumidor deve realizar consultas a metadados e recursos nos módulos seguintes. Este módulo é apresentado na Fig. 4.10.

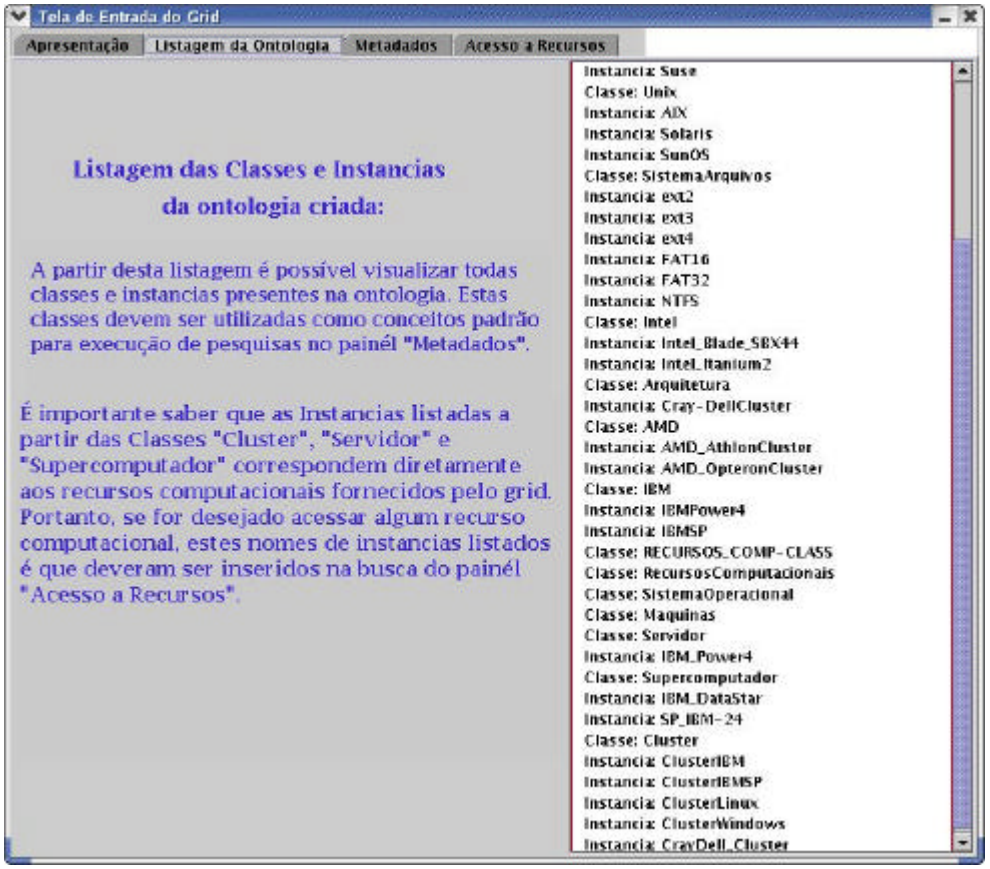

FIGURA 4.10: Tela de listagem das classes e instâncias da ontologia.

3º) **Metadados**: neste é possível que o consumidor pesquise os metadados a respeito de qualquer uma das classes listadas na ontologia, apenas digitando o nome da classe para ser pesquisada. Na Fig. 4.11 é apresentado um exemplo de busca a classe *Servidor*, exibindo os dados resultantes.

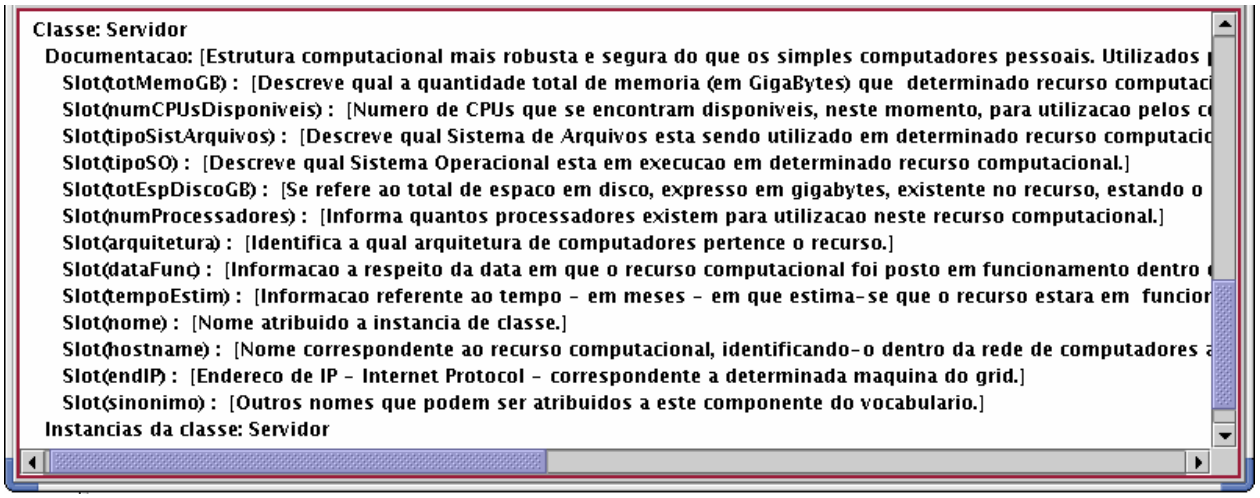

FIGURA 4.11: Resultado obtido após busca de metadados: classe *Servidor*.

4º) **Acesso a Recursos**: aqui é permitido ao consumidor entrar com o nome de qualquer um dos recursos existentes no ambiente, os quais são listados no 2º módulo como instâncias das classes: Cluster, Servidor ou Supercomputador, para pesquisa com relação aos seus dados de configuração. Caso a configuração apresentada pelo recurso seja satisfatória e este esteja disponível para uso, o consumidor pode requisitar o acesso. Um exemplo de pesquisa à recursos é mostrado na Fig. 4.12, onde pode-se visualizar o resultado obtido após pesquisa a respeito do recurso computacional *"ClusterLinux"*. Nesta pesquisa de recursos, todas as informações listadas são trazidas diretamente da ontologia, criada no editor Protégé-2000. Desta forma, na listagem constam também alguns dados próprios do editor, os quais são usados internamente, para controle dos conceitos inseridos na ontologia. Exemplo disso são os identificadores, representados por *FrameID*, presentes na listagem.

Como o ambiente não consiste de um *grid* computacional real, a tarefa efetiva de acesso ao recurso não é oferecida pela aplicação, uma vez que seria necessário dispor de recursos computacionais em funcionamento regular e de toda infra-estrutura real de *grid* para conexão.

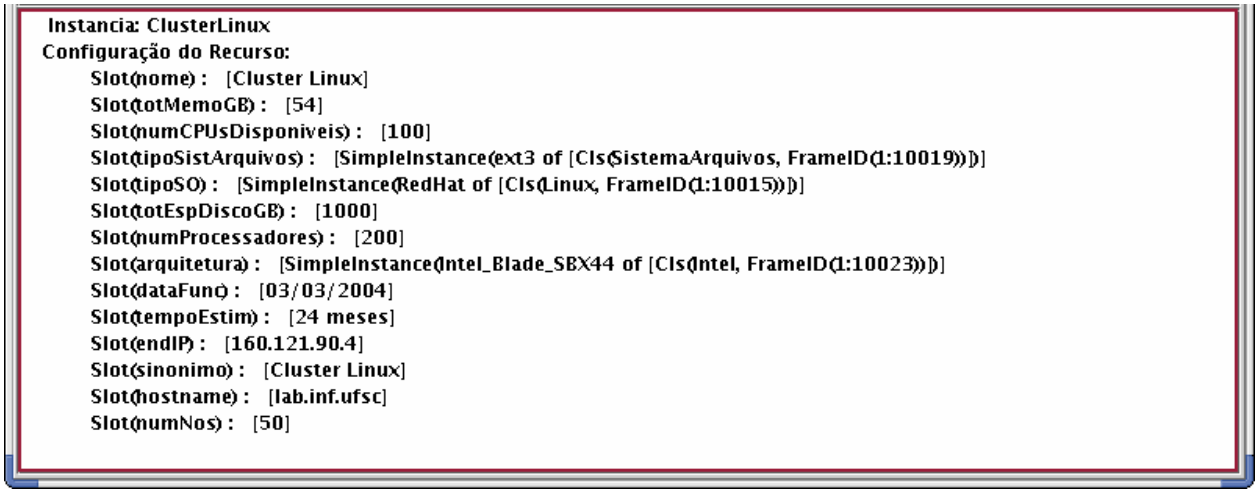

FIGURA 4.12: Resultado obtido após pesquisa de recursos: recurso *ClusterLinux*.

4.6.2 Criação do Serviço

Para desenvolvimento de um serviço a ser fornecido dentro de um ambiente *grid* é primeiramente necessário, sem dúvida, dispor do ambiente. Como já apresentado no capítulo 2, existem muitos cuidados que devem ser levados em consideração para criação de um *grid* computacional, sendo uma tarefa realmente complexa de ser realizada de maneira eficiente. Desta forma, o ambiente foi definido a partir da ferramenta *Globus*, também tratada do capítulo 2, a qual objetiva tornar mais simples e segura a criação de um *grid*, além de ser facilmente obtida, de forma gratuita, contando com vasta documentação e manuais sobre seu funcionamento.

O primeiro passo realizado foi, portanto, instalar a ferramenta *Globus*. Optou-se por instalar a versão 3.2, pois já estava sendo disponibilizada como uma versão estável pela *Globus Alliance* (GLOBUS, 2004) e também por possuir boa documentação. Como o objetivo constava apenas da criação do serviço para viabilização da ontologia, a instalação da ferramenta foi realizada em um único computador, possuindo a configuração básica descrita na Tab. 4.5, sendo configurado para apresentar ambos os comportamentos de fornecedor e de consumidor de recursos.

Apesar de a ferramenta *Globus* ter sido instalada em uma máquina, ao final foram gerados dois certificados de autoridade distintos para acesso ao ambiente, pois é necessário garantir a existência segura tanto do fornecedor de recursos (servidor) quanto do consumidor (cliente), uma vez que o certificado gerado para o servidor se diferencia do certificado necessário ao consumidor para acesso ao ambiente de *grid Globus*, e cada cliente necessita do seu próprio certificado para obter acesso.

TABELA 4.5: Configuração básica do computador usado para o ambiente.

Processador	AMD Athlon XP 2000 – 1.7 Ghz
Capacidade de Memória	512 MB
Capacidade de Disco	30 GB
Sistema Operacional	Linux Red Hat 9.0 – Kernel versão 2.4.20-8

No momento em que a ferramenta *Globus* encontrava-se corretamente instalada e em funcionamento, iniciou-se a tarefa de geração do serviço. Segundo SOTOMAYOR (2004) a criação de um serviço de *grid* consiste da realização de cinco passos básicos,

os quais foram seguidos para o desenvolvimento do serviço e serão brevemente descritos a seguir.

Para criação do serviço, o primeiro passo realizado foi definir quais métodos, desenvolvidos pela aplicação do fornecedor (servidor), seriam visíveis pela aplicação dos consumidores (clientes) para permitir o acesso e manipulação dos conceitos presentes na ontologia. Estes métodos correspondem aos três últimos módulos descritos na subseção anterior: Listagem da Ontologia, Metadados e Acesso a Recursos.

No ambiente *Globus* o padrão XML é usado em diversos documentos, por ser este adotado de forma geral em ambientes *web*, onde também são fornecidos os servidos de *grids* computacionais. Desta forma, para a criação do serviço, os métodos já existentes foram reproduzidos para o formato usado em uma linguagem especial, derivada da linguagem WSDL (*Web Service Description Language* – a qual, por sua vez, é derivada da linguagem XML), denominada GWSDL (*Grid Web Service Description Language*). A linguagem GWSDL, assim como a WSDL, é usada para especificação de quais operações são oferecidas por um serviço disponibilizado via *web* (SOTOMAYOR, 2004). Porém, a linguagem GWSDL difere da WSDL por ser voltada especialmente à descrição de serviços de *grids* computacionais, sendo adotada pela especificação OGSI (*Open Grid Services Infrastructure*), usada pela ferramenta *Globus* (GLOBUS, 2004).

O próximo passo na criação do serviço consistiu da definição da classe responsável pelo desenvolvimento de todos os métodos descritos em linguagem GWSDL, especificada pelo fornecedor (servidor) do serviço e que será acessada por todos os consumidores que buscam recursos presentes no ambiente. Como a aplicação, descrita na seção anterior, já havia sido finalizada, somente foram feitas adaptações necessárias para o *Globus*.

O passo de finalização do serviço criado, por ser puramente operacional e poder variar de acordo com a versão da ferramenta *Globus* utilizada, está descrito no Anexo III desta dissertação.

Após execução destes passos, o serviço encontra-se pronto para ser acessado por qualquer consumidor. Entretanto, a apresentação da ontologia e a visualização da aplicação em funcionamento são importantes para ter-se segurança da não ocorrência de falhas durante a criação do serviço, por isso, a aplicação descrita na subseção anterior foi novamente adaptada para funcionamento no ambiente *Globus*, na forma da aplicação

utilizada para acesso do consumidor (cliente). O resultado obtido após as etapas descritas é apresentado na Fig. 4.13, onde é possível observar, por meio da listagem dos serviços existentes no ambiente *Globus*, a qual é apresentada por ele no momento de sua inicialização, a presença do serviço criado.

FIGURA 4.13: Listagem destacando o serviço criado: *GgridService*.

5. Conclusões e Trabalhos Futuros

Nesta dissertação abordamos um problema específico dentro da área de *grids* computacionais: a dificuldade na busca e acesso de recursos computacionais por parte de consumidores não familiarizados com as configurações dos *grid* computacionais. Nesta abordagem, adotamos como possível causa do problema a falta de padronização na forma com que as informações, relacionadas aos recursos computacionais do *grid*, são apresentadas aos consumidores. Desta forma, nossa proposta de solução para o problema foi a utilização do paradigma de ontologias.

Com este objetivo, efetuamos a construção de um vocabulário contendo os conceitos relativos a um determinado domínio de um *grid*, para que fosse possível fornecer recursos computacionais de uma forma amigável para um grupo de usuários.

Embora a princípio pareça ser disjunta a pesquisa da aplicação de ontologias em ambientes de *grid*, sua utilização foi verificada como um diferencial para o uso de recursos e serviços distribuídos. Em nossa proposta, o vocabulário é usado para facilitar a interação com o ambiente, já que através do mesmo ocorre uma demonstração na forma descritiva. Com nossos experimentos, verificamos que para um consumidor que utiliza o ambiente fica mais fácil compreender o significado dos diversos recursos e serviços disponíveis, ao invés de não possuir nenhum tipo de simbolismo para sua utilização.

Como alternativa para apresentação deste vocabulário desenvolvido aos consumidores, foi construída uma aplicação, onde os conceitos pudessem ser visualizados e consultados através de uma *interface*. Posteriormente, esta aplicação foi transformada em um serviço para *grids* computacionais, demonstrando a possibilidade de inserção da ontologia criada dentro de um ambiente real de *grid*.

Através da pesquisa realizada com relação à aplicação de ontologias em ambientes de *grid* podemos constatar que, apesar de serem áreas distintas, sua utilização em conjunto é realmente viável. Pois a área de *grid* computacional utiliza os mesmos conceitos de outras áreas, como por exemplo, arquitetura de computadores. Entretanto, estes conceitos apresentam certas diferenças, por serem voltados à computação em alto desempenho. Um exemplo disso é a unidade de medida aplicada a certos recursos que tende a ser sempre em grau maior.

Podemos também constatar que a ontologia auxilia ao tornar transparente para os consumidores a utilização e o acesso aos recursos, e reduz a possibilidade de problemas devido a ambigüidades na busca de dados, facilitando, desta forma, que diferentes organizações acessem os recursos fornecidos em *grids* computacionais.

Para que fosse possível ter um sentimento a respeito da clareza e facilidade de uso da *interface*, desenvolvida na aplicação, esta foi disponibilizada para total utilização pelos alunos do 1º semestre do curso de Bacharelado em Ciência da Computação da Universidade Federal de Pelotas (UFPel). Como a *interface* foi desenvolvida visando o domínio de consumidores de recursos de ambientes *grid*, foi necessário apenas explicar aos alunos do que consiste um ambiente de *grid* e para quê é usado, não havendo a necessidade de explicar o conceito de ontologia. Os alunos utilizaram a aplicação de forma livre, realizando consulta de metadados e busca de recursos computacionais. Acreditamos que o resultado obtido após esta utilização foi o esperado, pois algumas modificações foram feitas, tornando a compreensão mais ampla, e as telas da aplicação foram dispostas na forma de um “fichário”, permitindo navegação a qualquer momento por todas elas. De maneira geral, a *interface* foi bem aceita pelos alunos, os quais conseguiram realizar consultas e explorar o vocabulário sem encontrar dificuldades.

A criação da ontologia, seguido do desenvolvimento da aplicação e do serviço, permitiram obtenção das seguintes conclusões:

- A linguagem OWL se mostrou realmente adequada à definição da ontologia, provando que possui todos os pontos necessários para criação de ontologias completas. O fato de existirem três tipos diferentes desta mesma linguagem também ajudou na realização do trabalho, pois no momento em que constatamos os problemas a serem enfrentados pela utilização da linguagem OWL DL, esta foi abandonada para utilização da linguagem OWL *Full*, pois existe a possibilidade de, durante o projeto, passar a utilizar outra subdivisão da linguagem OWL. Entretanto, caso exista no projeto alguma regra não suportada pela nova sub-linguagem adotada, esta deve ser anteriormente modificada.

- O editor Protégé-2000 apresenta diversas funcionalidades e *plug-ins*, permitindo o desenvolvimento de qualquer tipo de ontologia, juntamente com definição de restrições de integridade e axiomas, geração de gráficos, importação e exportação para outros tipos de linguagens para ontologias, como OWL, RDF e DAML-OIL, entre outras funcionalidades. Além disso, possui uma lista de discussão realmente grande e efetiva, tornando mais rápida a resolução de possíveis problemas.

- Apesar destas vantagens, o editor Protégé-2000 ainda não possui bons *plug-ins* para reprodução de uma ontologia, seja em linguagem OWL ou não, para o padrão XML. Existem *plug-ins* para esta tarefa, mas apresentam problemas na reprodução da ontologia, não reconhecendo totalmente certas estruturas, como: cardinalidade; múltiplas ocorrências de mesma propriedade; herança múltipla e documentação.

- A linguagem fornecida pelas APIs do Protégé-2000 para manipulação da ontologia se mostrou bastante intuitiva, pois segue o mesmo padrão das APIs da linguagem *Java* e a mesma nomenclatura usada diretamente pelo editor, onde a palavra "*class*" faz relação à classes da ontologia, "*instance*" faz relação à instâncias, entre outros exemplos.

- A linguagem para definição de serviços GWSDL não reconhece facilmente estruturas complexas, como por exemplo, estruturas do tipo *Lista.* Na definição do serviço, o mais usual é a definição de estruturas simples, como *String* e *Int*, nos parâmetros de entrada e saída dos métodos *Java* criados. Não foi encontrado nenhum meio para o tratamento de estruturas complexas em linguagem GWSDL.

Podemos concluir que os objetivos de pesquisa, propostos inicialmente nesta dissertação, foram alcançados. Com a criação e o funcionamento do serviço pode-se visualizar a aplicação em funcionamento em um ambiente de *grid*. Não foi possível o desenvolvimento das visões semânticas, as quais necessitariam da simulação de diversos recursos computacionais em funcionamento no ambiente, para que o serviço MDS fornecesse as informações necessárias para geração e atualização das visões semânticas.

Como trabalhos futuros, são sugeridos: a geração das visões semânticas, através da simulação ou não (no caso de recursos reais) de vários recursos computacionais em funcionamento no ambiente *grid*; ampliação da ontologia desenvolvida, inserindo conceitos usados em outros tipos de *grids* computacionais, como *grids* voltados ao fornecimento de recursos de rede, recursos de dados, entre outros; ampliação da ontologia, agregando novos métodos necessários para o efetivo acesso ao recurso pelo consumidor; recriação do serviço em outros Sistemas Operacionais, para constatar a possibilidade de ser interoperável.

Referências Bibliográficas

AGRAWAL, Sudesh, DONGARRA, Jack, SEYMOUR, Keith, VADHIYAR, Sathish. NetSolve: Past, Present, and Future - A Look at a Grid Enabled Server. In: Making the Global Infrastructure a Reality. Berman, Fran; Fox, Geoffrey; Hey, Tony. (eds). Wiley Publishing, April, 2003. p. 613-622. Disponível por www em: http://icl.cs.utk.edu/news_pub/submissions/netsolve-ppf.pdf

APACHE Ant Project, The. 2004. Disponível por www em: http://ant.apache.org/

APOLLO Project. 2003. Disponível por www em: http://apollo.open.ac.uk

BECHHOFER, Sean; HORROCKS, Ian; GOBLE, Carole; STEVENS, Robert. OilEd: a Reason-able Ontology Editor for the Semantic Web. Proceedings of KI2001, Joint German/Austrian Conference on Artificial Intelligence, Vienna, September 19-21, 2001. p. 396-408. Disponível por www em: http://oiled.man.ac.uk/publications.shtml

BERMAN, Fran; FOX, Geoffrey; HEY, Tony. The Grid: past, present, future. In. Grid Computing – Making a Global Infrastructure a Reality. Berman, Fran; Fox, Geoffrey; Hey, Tony (eds). John Wiley & Sons, 2003, ISBN: 0470853190. Disponível por www em: http://www.grid2002.org/

BERMAN, Francine. High-Performance Schedulers. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.279-309

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The Semantic Web. Scientific American, May, 2001. Disponível por www em: http://www.w3c.org/2001/sw/

BODIN, François; BECKMAN, Peter H.; GANNON, Dennis; YANG, Shelby; KESAVAN, S.; MALONY, Allen D.; MOHR, Bernd. Implementing a Parallel C++ Runtime System for Scalable Parallel System. In. Proceedings of the 1993 Supercomputing Conference, Portland, Oregon, November, 1993, p. 588-597. Disponível por www em: http://citeseer.ist.psu.edu/bodin93implementing.html

BORST, Willem N. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. PhD Thesis, University of Twente, Enschede, 1997. Disponível por www em: http://www.ub.utwente.nl/webdocs/inf/1/t0000004.pdf

BRICKLEY, Dan; GUHA, Ramanathan. Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation. March, 2000. Disponível por www em: http://www. w3.org/TR/2000/CR-rdf-schema-20000327/

CAREY, Rikk; BELL, Gavin. The Annotated VRML97 Reference Manual. April, 1997. Disponível por www em http://www.web3d.org/x3d/vrml/tutorials/

CASANOVA, Henri; DONGARRA, Jack. NetSolve: A Network Server for Solving Computational Science Problems. Technical Report CS-95-313, University of Tennesee, November, 1995. Disponível por www em: http://citeseer.ist.psu.edu/casanova95netsolve.html

CASANOVA, Henri; DONGARRA, Jack; JOHNSON, Chris; MILLER, Michelle. Application-Specific Tools. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.159-180.

CHAPMAN, Barbara; MEHROTRA, Piyush; ZIMA, Hans. Programming in Vienna Fortran. Scientific Programming, 1992. p.31-50. Disponível por www em: http://citeseer.ist.psu.edu/chapman92programming.html

CHAPPEL, David. Understanding ActiveX and OLE. Redmond, WA: Microsoft Press, 1996. ISBN 1-572-31216-5. Disponível por www em: http://www.microsoft.com/com/resources/books.asp

CONDOR High Throughput Computing. The Condor Project Homepage. 2004. Disponível por www em: http://www.cs.wisc.edu/condor/

CORCHO, Oscar; FERNÁNDEZ-LÓPES, Mariano; GÓMEZ-PÉRES, Asunción. OntoWeb – Tecnical Roadmap v 1.0. 2001. Disponível por www em: http://www.ontoweb.org/download/deliverables/D11_v1_0.pdf

CRUBÉZY, Monica. The Protégé Axiom Language and Toolset ("PAL"). April, 2002. Disponível por www em: http://protege.stanford.edu/plugins.html

CZAJKOWSKI, Karl; FITZGERALD, Steven; FOSTER, Ian; KESSELMAN, Carl. Grid Information Services for Distributed Resource Sharing. In Proceedures of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August, 2001.

DANTAS, Mario R. Grids Computacionais - Fundamentos, Ambientes e Experiências. In. Computação em Cluster. Pitanga, Marcos. Rio de Janeiro: Editora Brasport. 2003.

DANTAS, Mario R. Tecnologias de Redes de Comunicação e Computadores. Rio de Janeiro: Axcel Books, 2002. 328p.

DeFANTI, Tom; FOSTER, Ian; PAPKA, M; STEVENS, R; KUHFUSS, T. Overview of the I-WAY: Wide Area Visual Supercomputing. International Journal of Supercomputer Applications, 1996. p. 123-130. Disponível por www em: http://www.globus.org/research/papers.html

DE ROURE, David; JENNINGS, Nicholas; SHADBOLT, Nigel. The Semantic Grid: A Future e-Science Infrastructure, 2003. Disponível por www em: http://www.semanticgrid.org/documents/semgrid-journal/semgrid-journal.pdf

DOMINGUE, John; MOTTA, Enrico; GARCIA, Oscar C. Knowledge Modelling in WebOnto and OCML: A User Guide. v. 2.4. 1999. Disponível por www em: http://kmi.open.ac.uk/projects/ocml/

ECLIPSE. Página oficial do editor Eclipse. 2004. Disponível por www em: http://www.eclipse.org.

FARQUHAR, Adam; FIKES, Richard; RICE, James. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Knowledge Systems Laboratory, September, 1996. Disponível por www em: http://www-ksl-svc.stanford.edu:5915/doc/project-papers.html

FENSEL, Dieter. Ontologies: Silver Bullet for Knowledge Management and Eletronic Commerce. Springer – Verlag, Berlin, 2000.

FENSEL, Dieter; HORROCKS, Ian; VAN HARMELEN, Frank; et al. OIL in a Nutshell. Lecture Notes In Computer Science, Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, 2000. Disponível por www em: http://www.cs.vu.nl/~ontoknow/oil/downl/oilnutshell.pdf

FITZGERALD, Steven; FOSTER, Ian; KESSELMAN, Carl; VON LASZEWSKI, Gregor; SMITH, Warren; TUECKE, Steven. A Directory Service for Configuring High-Performance Distributed Computations. In. Proceedures of 6th IEEE Symposium on High Performance Distributed Computing (HPDC'97), Portland, OR, August, 1997, pp. 365-375.

FOSTER, Ian; MIDDLETON, Don; WILLIAMS, Dean. The Earth System Grid II:Turning Climate Model Datasets into Community Resources. January, 2003. Disponível por www em: https://www.earthsystemgrid.org/about/documentsPage.do

FOSTER, Ian; KESSELMAN, Carl; NICK, Jeffrey; TUECKE, Steven. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Presented at GGF4, 2002. Disponível por www em: http://www.globus/research/papers/ogsa.pdf

FOSTER, Ian; KESSELMAN, Carl; TUECKE, Steven. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 2001.

FOSTER, Ian; KESSELMAN, Carl (eds). The Grid: Blueprint for a New Computing Infrastructure. São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. 677p. ISBN 1-55860-475-8.

FOSTER, Ian; KESSELMAN, Carl; The Globus Project: a Status Report. In Proceedings of Seventh Heterogeneous Computing Workshop (HCW 98), IEEE Computer Society Press, 30 March, 1998. p. 4-18. Disponível por www em: http://citeseer.ist.psu.edu/foster98globus.html

FOX, Geoffrey C.; HIRANANDANI, Seema; KENNEDY, Ken; KOELBEL, Charles; KREMER, Ulrich; TSENG, Chau-Wen; WU, Min-You. Fortran D Language Specification. Technical Report TR90-141, Dept. of Computer Science, Rice University, Houston, TX, December, 1990, 31p. Disponível por www em: http://citeseer.ist.psu.edu/fox91fortran.htm

FOX, Geoffrey C.; WILLIAMS, Roy D.; MESSINA, Paul C. Parallel Computing Works. San Francisco: Morgan Kauffmann Publishers, 1994. ISBN 1-55860-253-4. Disponível por www em: http://www.npac.syr.edu/copywrite/pcw/

GAVA, Tânia; MENEZES Crediné. Especificação de Software Baseada em Ontologias. In. III Escola de Informática, 2003. p. 167-205.

GENESERETH, Michael; FIKES, Richard. Knowledge Interchange Format, Version 3.0 Reference Manual, Knowledge Systems Laboratory-Stanford University, June, 1992. Disponível por www em: http://www.ksl.stanford.edu/KSL_Abstracts/KSL-92-86.html

GLOBUS Alliance. 2004. Disponível por www em: http://www.globus.org

GOBLE, Carole; DE ROURE, David. Semantic Web and Grid Computing, September, 2002. Disponível por www em: http://www.semanticgrid.org/documents/swgc/swgc-final.pdf

GOMÉZ-PÉRES, Asunción; FERNÁDEZ, Mariano; VICENTE, Antonio de. Towards a Method to Conceptualize Domain Ontologies, Workshop on Ontological Engineering/ECAI96, Budapest, Hungary, 1996. Disponível por www em: http://citeseer.nj.nec.com/483876.html

GOSLING, James; JOY, Bill; STEELE, Guy L.; BRACHA, Gilad. The Java Language Specification, Second Edition. Sun Microsystems, 2000. Disponível por www em: http://java.sun.com/docs/books/jls/index.html

GRID Infoware. The Grid Computing Information Centre. 2003. Disponível por www em: http://www.gridcomputing.com

GRUBER, Thomas R.. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Presented at the Padua Workshop on Formal Ontology, March, 1993a. Disponível por www em: http://ksl-web.stanford.edu/KSL_Abstracts/KSL-93-04.html

GRUBER, Thomas R.. A Translation Approach to Portable Ontology Specifications. In Knowledge Acquisition, 1993b. p. 199-220. Disponível por www em: http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html

GRUBER, Thomas R. Ontolingua: A Mechanism to Support Portable Ontologies. Technical Report, Knowledge Systems Laboratory, Stanford University, Stanford, June, 1992. http://citeseer.ist.psu.edu/gruber92ontolingua.html

GRUNINGER, Michael. Designing and Evaluating Generic Ontologies. In Proceedings of the 12th European Conference of Artificial Intelligence, August, 1996. Disponível por www em: http://citesser.ist.psu.edu/gruninger96designing.html

GUARINO, Nicola. Formal Ontology and Information Systems. In Proceedings of FOIS'98, Trento, Italy, 6-8 June, 1998. p. 3-15.

GUARINO, Nicola. Understanding, Building and Using Ontologies. In Proceedings of Tenth Knowledge Acquisition for Knowledge Based Systems Workshop, Catelonia, Spain, October, 1997.

GUARINO, Nicola; GIARETTA, Pierdaniele. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Towards Very Large Knowledge Bases – Knowledge Building and Knowledge Sharing (KBKS). Amsterdam: IOS Press.1995. p. 25-32.

HORROCKS, Ian; PATEL-SCHNEIDER, Peter; VAN HARMELEN, Frank. Reviewing the Design of the DAML+OIL: An Ontology Language for the Semantic Web. Presented at 18th National Conference on Artificial Intelligence (AAAI), 2002. Disponível por www em: http://www.cs.man.ac.uk/~horrocks/Publications/download/2002/AAAI02IHorrocks.pdf

IPG, Information Power Grid – Nasa's Computing and Data Grid. What is the IPG?. October, 2002. Disponível por www: http://www.ipg.nasa.gov/aboutipg/what.html

JOHNSON, Andrew; ROUSSOS, Maria; LEIGH, Jason; et al. The NICE Project: Learning Together in a Virtual Word. In Proceedings of VRAIS '98, Atlanta, Georgia, 14-18 March, 1998. p.176-183. Disponível por www em: http://www.evl.uic.edu/tile/NICE/NICE/PAPERS/VRAIS/vrais98.2.html

KARP, Peter D.; CHAUDHRI, Vinay K.; THOMERE, Jerome. XOL: An XML-Based Ontology Exchange Language. August, 1999. Disponível por www em: http://www.ai.sri.com/~pkarp/xol/

KENNEDY, Ken. Compilers, Languages, and Libraries. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.181-204.

KENT, Robert. Conceptual Knowledge Markup Language: The Central Core. In. Twelfth Workshop on Knowledge Acquisition, Modeling and Management. 1999. Disponível por www em: http://sern.ucalgary.ca/ksi/kaw/kaw99/papers/Kent1/CKML.pdf

KIFER, Michael; LAUSEN, Georg; WU, James. Logical Foudations of Object-Oriented and Frame-Based Languages. Journal of the Association for Computing Machinery, May, 1995. Disponível por www em: http://citeseer.ist.psu.edu/kifer90logical.html

KIRYAKOV, Atanas K.; DIMITROV, Martin; SIMOV, Kiril Iv. OntoMap – The Guide to the Upper-Level. In. Proceedings of the International Semantic Web Working Symposium (SWWS), Stanford University, California, USA, 2001. Disponível por www em: http://www.ontotext.com/publications/swws01.pdf

LASSILA, Ora; SWICK, Ralph. Resource Description Framework (RDF) Schema Specification. W3C Recommendation. February, 1999. Disponível por www em: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

LEE, Craig A.; KESSELMAN, Carl; SCHWAB, Stephen. Near-real-time Satellite Image Processing: Metacomputing in C++. IEEE Computer Graphics and Applications, 16(4). 1996. p. 79-84.

LINUX Online. What is Linux. 2004. Disponível por www em: http://www.linux.org

LITZKOW, Michael J.; LIVNY, Miron; MUTKA, Matt W. Condor - A Hunter of Idle Workstations. In Proceedings of 8th International Conference of Distributed Computing Systems, San Jose, CA, USA, June, 1988. p. 104-111. Disponível por www em: http://citeseer.ist.psu.edu/context/10482/0

LIVNY, Miron; RAMAN, Rajesh. High-Throughput Resource Management. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.311-337.

LOOM Project Home Page. Overview: Loom Knowledge Representation System. University of Southern California, 2003. Disponível por www em: http://www.isi.edu/isd/LOOM/LOOM-HOME.html

LUKE, Sean; JEFF, Heflin. SHOE 1.01 - Proposed Specification. SHOE Project. April, 2000. Disponível por www em: http://www.cs.umd.edu/projects/plus/SHOE/spec.html

MAYO Clinic. Protégé within Eclipse. Mayo Foundation for Medical Education and Research. 2004. Disponível por www em: http://informatics.mayo.edu/

McGUINNESS, Deborah; VAN HARMELEN, Frank. OWL Web Ontology Language Overview. W3C Recommendation, February, 2004. Disponível por www em: http://www.w3.org/TR/owl-features/

MESSINA, Paul. Distributed Supercomputing Applications. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.55-73.

MIRCHANDANEY, R,; SALTZ, J. H.; SMITH, R. M. Principles of runtime support for parallel processors. In Proceedings of the 2nd international conference on Supercomputing, France, 1988. p. 140-152. Disponível por www em: http://portal.acm.org/citation.cfm?id=55378&jmp=references&dl=GUIDE&dl=ACM

MOORE, Reagan W.; BARU, Chaitanya; MARCIANO, Richard; et al. Data-Intensive Computing. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.105-129.

NETSCAPE Communications Corporation. Core JavaScript Guide. v. 1.5, September, 2000. Disponível por www em: http://devedge.netscape.com/central/javascript/

NOY, Natalya; SINTEK, Michael; DECKER, Stefan; et al. Creating Semantic Web Contents with Protégé-2000. IEEE Intelligent Systems, 2001. p. 60-71.

NOY, Natalya; FERGERSON, Ray; MUSEN, Marl. The knowledge model of Protégé-2000: combining interoperability and flexibility. 12th International Conference on Knowledge Engineering and Knowledge Management-Europe Knowledge Aquisition Workshop (EKAW), French Riviera, 2-6 October, 2000. Disponível por www em: http://citeseer.nj.nec.com/noy01knowledge.html

NPACI - National Partnership for Advanced Computational Infrastructure. Partnership Report. 2000. Disponível por www em: http://www.npaci.edu/About_NPACI/index.html

OLIVEIRA, Rômulo S.; CARISSIMI, Alexandre S.; TOSCANI, Simão S. Sistemas Operacionais. Porto Alegre: Editora Sagra Luzatto, 2000. 233p.ISBN 85-241-0643-3.

OMG – Object Management Group. Common Object Request Broker Architecture: Core Specification. Version 3.0.3. Needham, MA: 02494, USA, March, 2004. Disponível por www em: ftp://ftp.omg.org/pub/docs/formal/04-03-12.pdf

ONTOSAURUS – Loom OntoSaurus Browser. 2003. Disponível por www em: http://www.isi.edu/isd/ontosaurus.html

POSTEL, Jon; TOUCH, Joe. Network Infrastructure. In. The Grid: Blueprint for a New Computing Infrastructure. Foster, Ian; Kesselman, Carl (eds). São Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p.534-566.

POUCHARD, Line; CINQUINI, Luca; DRACH, Bob; et al. An Ontology for Scientific Information in a Grid Environment: the Earth System Grid. In Proceedings of the 3th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03), Japan, Tokyo, 10-15 May, 2003.

POUCHARD, Line. Ontologies and The Earth System Grid. NIEeS Metadata Workshop, 10-11 September, 2002. Disponível por www em: https://www.earthsystemgrid.org/about/documentsPage.do

RAGGETT, Dave; LE HORS, Arnaud; JACOBS, Ian. HTML 4.01 Specification. W3C Recommendation, December, 1999. Disponível por www em: http://www.w3.org/TR/html401/

ROGERSON, Dale. Inside COM. Redmond, WA: Microsoft Press, 1996. ISBN 1-572-31349-8. Disponível por www em: http://www.microsoft.com/com/resources/books.asp

SATO, Mitsuhisa; NAKADA, Hidemoto; SEKIGUSHI, Satoshi; et al. Ninf: a Network based Information Library for Global World-Wide Computing Infrastructure. In Proceedings of High-Performance Computing Network (HPCN'97), Europe, 1997. p. 491-502. Disponível por www em: http://citeseer.ist.psu.edu/sato97ninf.html

SOTOMAYOR, Borja. The Globus Toolkit 3 Programmer's Tutorial. August, 2004. Disponível por www em: http://www.casa-sotomayor.net/gt3-tutorial/

SRI International - an independent, nonprofit R&D organization dedicated to client sucess. 2004. Disponível por www em: http://www.sri.com/

STEPHEN, Reed; LENAT, Douglas. Mapping Ontologies into Cyc. Conference Workshop on the Ontologies For The Semantic Web, Edmonton, Canada, July, 2002. Disponível por www em: http://www.cyc.com/cyc/technology/whitepapers

STUDER, Rudi; BENJAMINS, Richard; FENSEL, Dieter. Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering, 1998. Disponível por www em: http://citeseer.ist.psu.edu/225099.html

TANGMUNARUNKIT, H.; DECKER, S.; KESSELMAN, Carl. Ontology-based Resource Matching - The Grid meets the Semantic Web. 1th Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid) at the Twelfth International World Wide Web Conference, Budapest, May, 2003. Disponível por www em: http://www.isi.edu/~stefan/SemPGRID

TICHY, Walter F.; PHILIPPSEN, Michael; HATCHER, Phil. A Critique of the Programming language C*. Communications of the ACM. June, 1992, p. 21-24. Disponível por www em: http://citeseer.ist.psu.edu/tichy92critique.html

USCHOLD, Mike; GRUNINGER, Michael. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, June, 1996. p. 93-155. Disponível por www em: http://citeseer.ist.psu.edu/uschold96ontologie.html

VAN HEIJST, Gertjan; SCHREIBER, Guss A.; WIELINGA, Bob, Using Explicit Ontologies in KBS Development. International Journal of Human - Computer Studies, 1997. p. 183-292. Disponível por www em: http://ksi.cpsc.ucalgary.ca/IJHCS/VH/VH1.html

VAN HEIJST, Gertjan. The Role of Ontologies in Knowledge Engineering. PhD Thesis, University of Amsterdam, May, 1995. Disponível por www em: http://www.swi.psy.uva.nl/usr/gertjan/thesis.html

VERSTOEP, Kees. The Distributed ASCI Supercomputer 2 (DAS-2). May, 2002. Disponível por www em: http://www.cs.vu.nl/das2/

VOSS, Greg. Introducing Java Beans. November, 1996. Disponível por www em: http://java.sun.com/developer/onlineTraining/Beans/Beans1/index.html

Anexo I

Publicações

Título: **Aplicação de Ontologias na Descrição de Recursos em Grids Computacionais.**

Evento: III Simpósio de Informática da Região Centro do RS, Santa Maria – RS.

Data: de 18 a 20 de Agosto de 2004.

Autores: Ana Marilza Pernas Fleischmann e Mario Antonio Ribeiro Dantas

Anexo II

Telas Protégé

Neste anexo são apresentadas algumas das telas geradas pelo editor Protégé-2000. A Fig. A1 ilustra a tela principal para criação de ontologias do Protégé-2000, onde podemos observar, no canto direito, a estrutura hierárquica da ontologia, no centro, as instâncias geradas em cada uma das classes existentes, e no canto esquerdo, o valor dado a cada um dos atributos das instâncias criadas. Nesta figura é ilustrada a instância Intel_Itanium2, juntamente com seus atributos.

A Fig. A.2 mostra a árvore, gerada pelo editor, que representa, de forma hierárquica (de cima para baixo): a classe principal, todas as subclasses geradas a partir da classe principal; as demais subclasses da ontologia; as instâncias geradas a partir de cada subclasse. Na figura, está destacado o caminho percorrido na árvore para se chegar à instância *ext3*.

A Fig. A.3 mostra todas as classes geradas pelo editor, onde as setas indicam os relacionamentos existentes entre as classes. São também mostrados os atributos referentes a cada uma das classes.

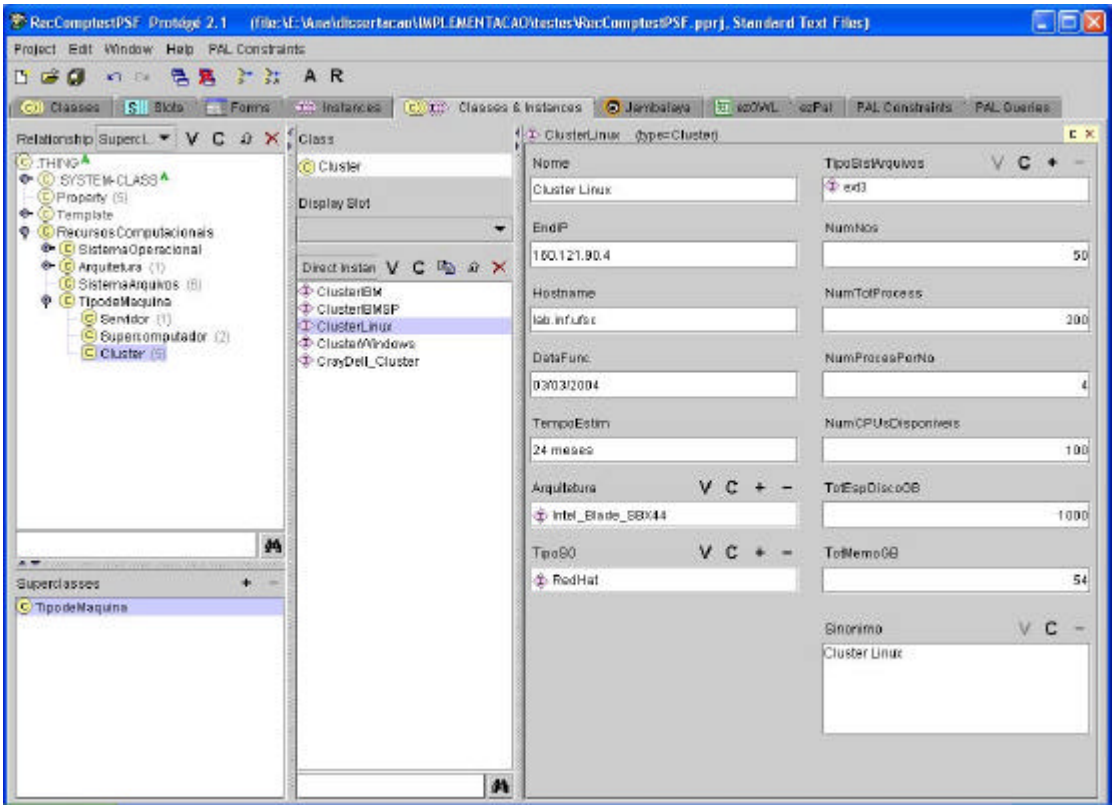

FIGURA A.1: Tela principal com a listagem da ontologia criada no Protégé-2000.

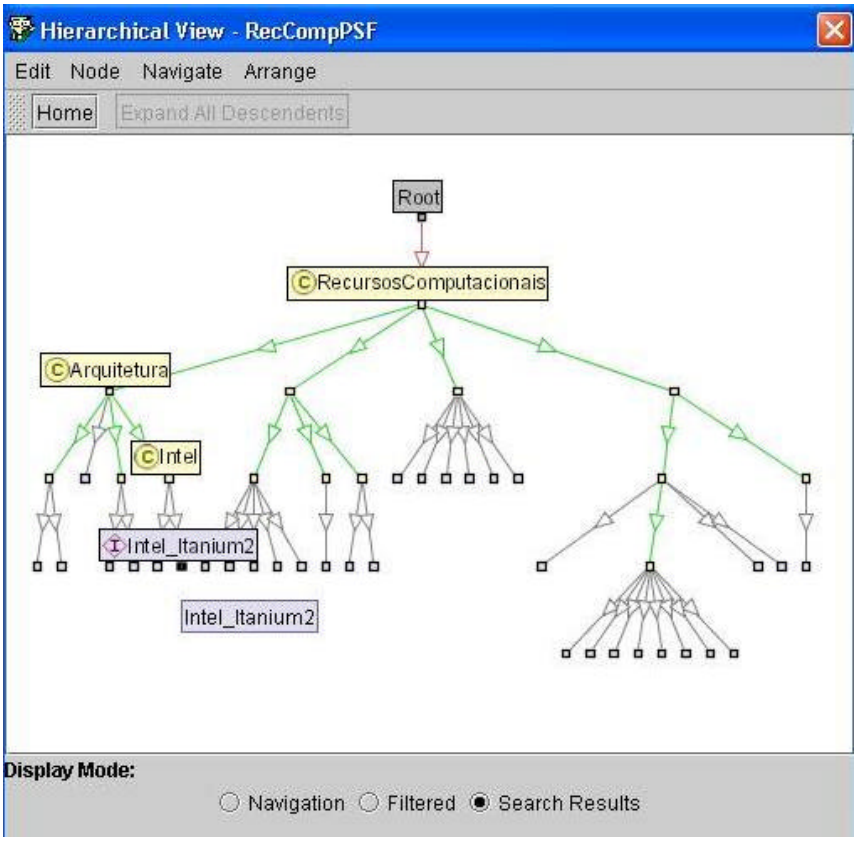

FIGURA A.2: Estrutura em árvore das classes e instâncias da ontologia.

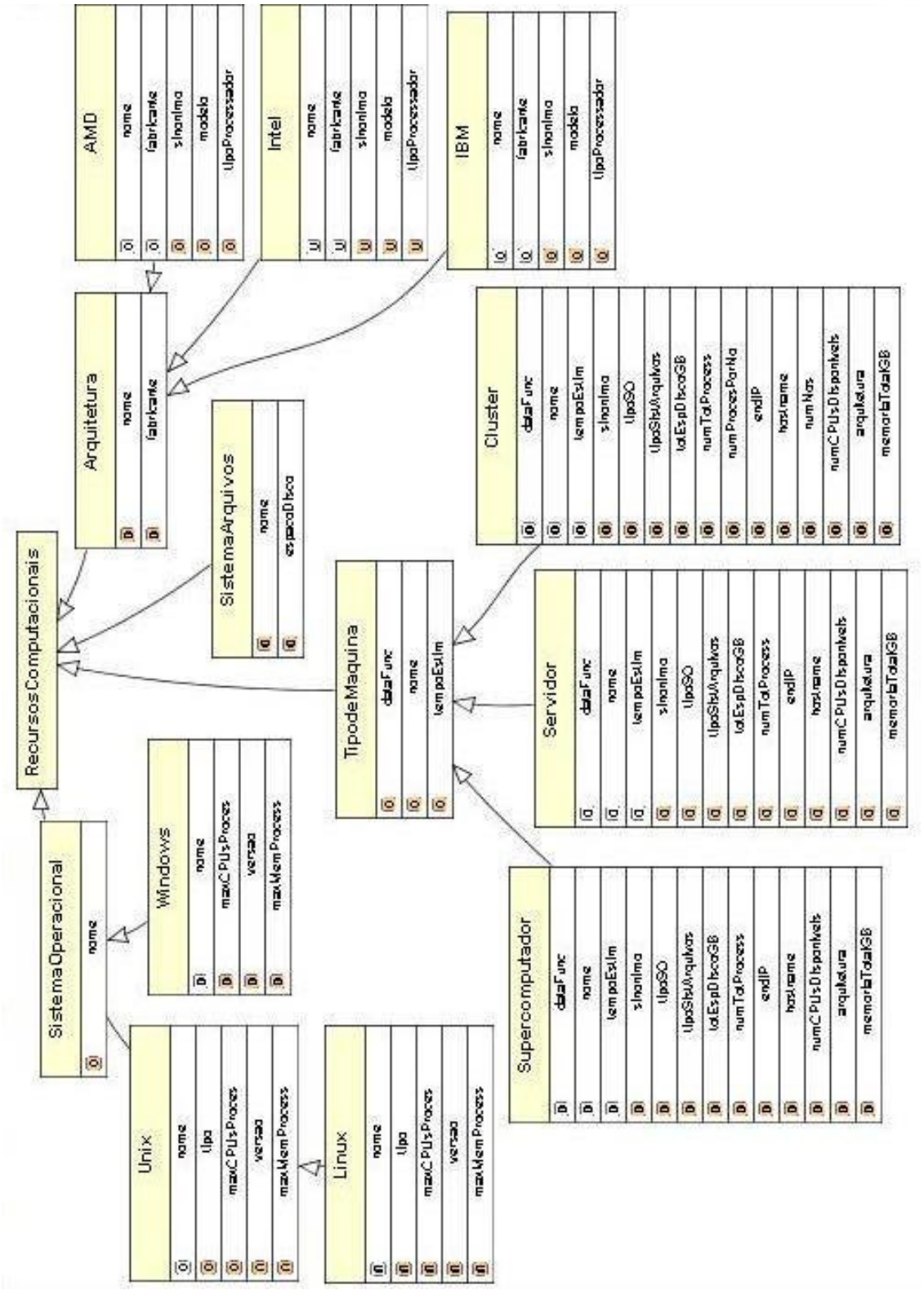

FIGURA A.3: Gráfico com todas as classes e seus relacionamentos.

Anexo III

Passo Final para Criação do Serviço

A etapa mais complexa na criação do serviço de *grid*, descrito do capítulo 4, foi desempenhada pela ferramenta ***Apache Ant*** (APACHE, 2004), a qual é executada por meio de linhas de comando e realiza, entre outras tarefas, a compilação de todas as classes e a geração de todos os demais arquivos necessários para a execução do serviço. Na Fig. A4 são mostrados os principais arquivos, referentes ao serviço criado nesta dissertação, gerados e compilados durante a execução da ferramenta *Ant*.

A Fig. A4 apresenta dois tipos de arquivos: Arquivos de criação GT3 (*Globus Toolkit 3*) e o arquivo *build.xml*, os quais são responsáveis por guiar as operações realizadas pela ferramenta *Ant*. Os Arquivos de criação GT3 indicam as localizações dos arquivos necessários que pertencem a ferramenta *Globus*, enquanto que o arquivo *build.xml* informa à ferramenta *Ant* quais arquivos do serviço devem ser compilados, como e em qual ordem (SOTOMAYOR, 2004).

Ainda na Fig. A.4, o arquivo *server-deploy.wsdd* é responsável por definir como o serviço deve ser apresentado, por exemplo, quais métodos definidos no arquivo GWSDL podem ser visualizados pela aplicação do consumidor. Todos os arquivos que são gerados pela ferramenta *Ant* são armazenados na pasta "*GAR*", presente na parte inferior da Fig. A.4.

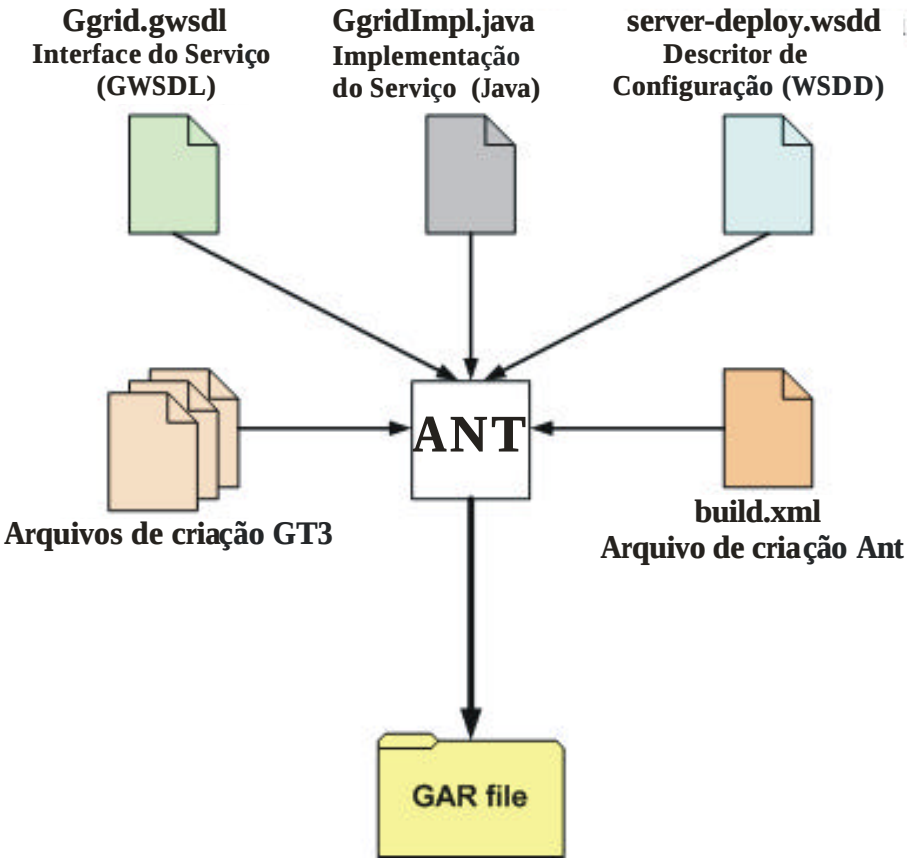

FIGURA A.4: Interação entre os arquivos do serviço criado e a ferramenta *Ant*.
Fonte: modificado de (SOTOMAYOR, 2004).