

FÁBIO JOSÉ RODRIGUES PINHEIRO

**UMA PROPOSTA DE UM SISTEMA DE
IMAGEM ÚNICA PARA USO DE
COMPUTAÇÃO EM GRADE EM
ORGANIZAÇÕES VIRTUAIS**

**FLORIANÓPOLIS
2005**

**UNIVERSIDADE FEDERAL DE SANTA
CATARINA
CURSO DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA**

**UMA PROPOSTA DE UM SISTEMA DE
IMAGEM ÚNICA PARA USO DE
COMPUTAÇÃO EM GRADE EM
ORGANIZAÇÕES VIRTUAIS**

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

FÁBIO JOSÉ RODRIGUES PINHEIRO

Florianópolis, Setembro de 2005.

UMA PROPOSTA DE UM SISTEMA DE IMAGEM ÚNICA PARA USO DE COMPUTAÇÃO EM GRADE EM ORGANIZAÇÕES VIRTUAIS

FÁBIO JOSÉ RODRIGUES PINHEIRO

‘Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Informática Industrial*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.’

Prof. Ricardo José Rabelo. Dr.
Orientador

Prof. Alexandre Trofino Neto, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Ricardo José Rabelo. Dr.
Presidente

Prof. Mário Antônio R. Dantas, Dr.

Prof. Joni da Silva Fraga, Dr.

Prof. Rômulo Silva de Oliveira, Dr.

Aos meus pais...

AGRADECIMENTOS

Num primeiro momento, como não poderia haver de deixar de ser, sinto-me como que obrigado, e da mesma forma um prazer imenso, a agradecer àqueles que foram e ainda são responsáveis pela minha existência aqui na Terra. Iniciando pelo Ser superior que a todos governa e a quem todos chamamos de Pai; e em seguidas aos meus amados pais. Sim, paiinho e mainha. Estes que desde o dia do meu nascimento estão ao meu lado me protegendo e me defendendo de todo mal possa vir a me acontecer. Nada mais do que MUITO OBRIGADO!! Amo vocês!

Em segundo, mas não menos merecedoras, a minhas irmãs lindonas: Biça e Nanda. Senti muitas saudades de vocês, viu!?! Também amo vocês!!

Wanessa... mulher encantadora que eu conheci aqui e que agora ficará para sempre ao meu lado. Muito obrigado por me suportar numa fase tão complicada. Te amo demais, minha linda!

Agradeço à PGEEL por me dar a possibilidade de realizar este mestrado. Como também à FAPEL (Fundação de Amparo à Pesquisa do Estado de Alagoas) pelo apoio financeiro, sem o qual isto não seria possível.

Se não fosse o apoio do professor (e por fim orientador) Ricardo Rabelo, talvez eu não tivesse dado continuidade e, conseqüentemente, finalizado o meu mestrado. Obrigado, professor, pelos seus ensinamentos e pela confiança depositada em mim.

Citemos os "broders": Cristiano, Roberto Demente, Carlinhos (Zaca), Heitor, Rosfran, Pastor, André Aranha e tantos outros. Só quem os conhece, e também conhece a nossa relação de amizade, sabe o quantos somos importantes uns para os outros. Queria agradecer em especial ao Cristiano e ao Roberto, que além serem os maiores, deram uma boa "esticada" e vieram me visitar, me fazendo sentir mais perto de casa.

Tércio, Augusto e Marcelo... meus companheiros de morada... verdadeiros irmãos em Floripa. Primeiro o Tércio: esse me deu todo apoio na minha chegada à cidade. Ele mesmo sabe que esse apoio foi indispensável. Eu não saberia (e não tinha) a quem mais recorrer se não fosse ele. Valeu mesmo, meu chapa! Segundo, o Augusto: o companheiro de mais longa duração. Dois anos e meio, não foi? Foi um bom tempo... às vezes passado tão lentamente (quando a saudade de casa batia forte), e às vezes tão rápido (quando viamos todos os prazos se estourando). 'Brigadão pela sua companhia!! E por fim, Marcelo... vindo do "velho oeste", se tornou de fato (e em pouco tempo) um grande irmão. Da mesma forma em que

discutíamos com certa frequência, também nos divertíamos muito. Sem dúvida aprendemos bastante juntos. Acredito que isso só demonstra a admiração que sentimos um pelo outro. Irmão, espero reencontrá-lo, certo?!

Graças a Deus eu tive a oportunidade de fazer grandes amigos em Floripa. Amigos esses que espero, sinceramente, sejam eternos. Uma grande abraço Neves, Emerson, Ricardo, Fernando, Cassia, Felipe, Rodrigo, Pinga, XT, Karine, Andreza, Baldo, Rui, Gesser, Leandro, Thiago, Favarim, Patrícia, Michele, Zezão, e a todos aqueles que por ventura acabei não citando aqui. Acreditem... vocês foram fundamentais. Muito obrigado!!

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

UMA PROPOSTA DE UM SISTEMA DE IMAGEM ÚNICA PARA USO DE COMPUTAÇÃO EM GRADE EM ORGANIZAÇÕES VIRTUAIS

Fábio José Rodrigues Pinheiro

Setembro/2005

Orientador: Ricardo José Rabelo

Área de Concentração: Automação e Sistemas

Palavras-chave: Computação em Grade, Organizações Virtuais, Colaboração, Compartilhamento de Recursos, Confiança

Número de Páginas: 1 + 131

A crescente necessidade de grandes e constantes investimentos para se atualizar os recursos computacionais tem se tornado um fator complicador dentro das organizações. Os atuais sistemas empresariais estão cada vez mais complexos e demandado por mais poder de processamento. Isso torna-se mais crítico se considerarmos que o mercado atual é composto, quase que na sua totalidade, de micro, pequenas e médias empresas, que cada vez mais dependem de tecnologia de informação para se manterem competitivas. No entanto, dentro de uma empresa, um recurso computacional é ocupado apenas por uma pequena parte do seu tempo, levando a possibilidade de serem utilizados em outras situações durante o tempo em que estiver ocioso. O paradigma empresarial de Organizações Virtuais (OV) permite que organizações/empresas colaborem em busca de um objetivo em comum, compartilhando temporária e dinamicamente habilidades, informações e recursos, através de redes de computadores. Neste trabalho buscou-se melhor explorar o compartilhamento de recursos entre os parceiros de OVs, permitindo que esses façam uso dos recursos de forma racional e transparente, evitando um investimento desnecessário em tecnologia. Isto significa se beneficiar da colaboração intrínseca em uma OV. A Computação em Grade (*Grid Computing*) foi pesquisada para ser a base desta visão, representando a tecnologia central utilizada neste trabalho. Uma arquitetura-cliente enxuta e baseada em padrões foi proposta como meio de suportar a transparência desejada no uso de recursos de uma OV como também numa utilização mais racional. Também é oferecida flexibilidade na seleção dos recursos via a indicação de três critérios: confiança, desempenho e custo. Resultados experimentais são apresentados a partir do protótipo desenvolvido como também uma avaliação geral do trabalho.

Abstract of Dissertation (Thesis) presented to UFSC as a partial fulfillment of the requirements for the degree of Master (Doctor) in Electrical Engineering.

A PROPOSAL OF A CLIENT SIDE ARCHITECTURE FOR USING GRID COMPUTING IN VIRTUAL ORGANIZATIONS

Fábio José Rodrigues Pinheiro

Setembro/2005

Advisor: Ricardo José Rabelo

Area of Concentration: Automation and Systems Control

Key words: Grid Computing, Virtual Organization, Collaboration, Resource Sharing, Trust

Number of Pages: 1 + 131

Organizations have been facing tremendous difficulties to keep their computing resources updated due to their constant and increasing costs. Typical enterprise systems are getting more and more complex and computing bound. These difficulties are more relevant when considering the reality where almost all companies in the world are composed of micro, small and medium size organizations, more and more dependent on the information technology to keep being competitive. However, within a company, computing resources are, in practice, used only during some time, meaning that they could be used in other situations in the time they are idle. The Virtual Organizations (VO) enterprising paradigm aims at allowing different organizations to collaborate towards a common objective, sharing skills, information and resources temporarily and dynamically, by means of computing networks. This work aimed to better exploit the resources sharing among VO partners, providing the possibility to make use of their idle computing resources in a more rational and transparent way, meaning that unnecessary investments on information technology can be avoided. In other words, it means to benefit of the intrinsic collaboration existent within a given VO more properly. Grid Computing technology was investigated to be the basis for supporting this vision, and it represents the core information technology dealt with this work. A lean and standard-compliant client-architecture has been proposed as a way to support the desired transparency in the use of resources of a VO as well as their more rational utilization. Some flexibility in the selection of resources is also offered via the indication of three criteria: trust, performance and costs. Experimental results are presented upon the developed prototype as well as a global assessment of the proposed work is provided in the end.

Sumário

1	Introdução	1
1.1	Motivação	1
1.2	Objetivo Geral	3
1.3	Objetivos Específicos	3
1.4	Organização do Trabalho	4
2	Organizações Virtuais	5
2.1	Conceituação	5
2.2	Colaboração em Organizações Virtuais	10
2.2.1	Compartilhamento	13
2.2.2	Confiança	14
2.2.3	Ontologia	16
3	Computação em Grade	19
3.1	Conceituação	19
3.2	Arquitetura	22
3.2.1	Ambiente (<i>Fabric</i>)	23
3.2.2	Conectividade (<i>Connectivity</i>)	24
3.2.3	Recursos (<i>Resource</i>)	24
3.2.4	Coletivo (<i>Collective</i>)	25
3.2.5	Aplicação (<i>Application</i>)	26

3.3	Áreas de Aplicação	26
3.3.1	Supercomputação distribuída	26
3.3.2	Computação de alta vazão	27
3.3.3	Computação sob demanda	27
3.3.4	Computação de processamento intensivo de dados	28
3.3.5	Computação colaborativa	28
3.4	Comparação com outras tecnologias	28
3.4.1	P2P	29
3.4.2	Cluster	32
3.5	Padronização	34
3.5.1	<i>Open Grid Service Architecture</i>	35
3.5.2	<i>Open Grid Service Infrastructure</i>	36
3.5.3	Serviços de Grade (<i>Grid Services</i>)	36
3.5.4	<i>Web Service Resource Framework</i>	39
3.6	Ferramentas	41
3.6.1	<i>Legion</i>	41
3.6.2	<i>MyGrid</i> e <i>OurGrid</i>	43
3.6.3	<i>Condor</i>	44
3.7	<i>Globus Toolkit</i>	47
3.7.1	Núcleo	48
3.7.2	Segurança	49
3.7.3	Gerenciamento de Dados	49
3.7.4	Gerenciamento de Recursos	50
3.7.5	Serviço de Informação	52
3.8	Outros Serviços	53
3.8.1	Escalonadores	53

3.8.2	Mediador de Recurso	55
3.9	Trabalhos Relacionados	56
3.9.1	Grades no lado cliente	57
3.9.2	Semântica em Grades	60
4	Modelo Conceitual	64
4.1	Introdução	64
4.2	Proposta	67
4.3	Arquitetura	68
4.3.1	Federação de Recursos	71
4.3.2	Interface Cliente de Acesso à Grade	71
4.3.3	Requisitos das Aplicações	72
4.3.4	Serviço de Informação	73
4.3.5	Provedor de Informações de Confiança	74
4.3.6	Publicador de Informações	75
4.3.7	Serviço Buscador de Recursos	76
4.3.8	Cliente Buscador de Recursos	78
4.3.9	Avaliador de Uso da Grade	79
4.3.10	Segurança	80
4.3.11	Submissão de Tarefas	80
4.3.12	Transferência de Dados	81
4.4	Conclusão	82
5	Protótipo	83
5.1	Implementação	83
5.1.1	Ferramentas	83
5.1.2	Cenário de Implementação	84

5.1.3	Modelo de Implementação	86
5.1.4	Serviços	96
5.2	Execução	102
5.2.1	Contexto	102
5.2.2	Instalação do Protótipo	102
5.2.3	Configuração do Protótipo	104
5.2.4	Fluxo de Execução	107
5.3	Resultados, Dificuldades e Problemas Encontrados	110
6	Conclusões	112
6.0.1	Sugestões para Trabalhos Futuros	113
A		115
A.1	GWSDL do Serviço <i>SystemInformation</i>	115
A.2	GWSDL do Serviço <i>TrustInformationProvide</i>	117
A.3	GWSDL do Serviço <i>ResourceFinder</i>	118
A.4	Diagrama de Classes	119

Lista de Figuras

2.1	Ciclo de vida de uma Organização Virtual	9
2.2	Organização Virtual	11
3.1	Exemplo de uma Grade	22
3.2	Arquitetura de uma Grade (adaptado de (Foster et al., 2001))	23
3.3	Exemplo de uma rede P2P	30
3.4	Arquitetura de um <i>Cluster</i> de computadores (adaptada de (Buyya, 1999))	32
3.5	Evolução para a especificação WSRF (adaptada de (Sotomayor, 2004))	40
3.6	Cenário da integração do <i>MyGrid</i> e o <i>OurGrid</i> (adaptada de (Andrade et al., 2003))	45
3.7	Submissão de aplicação utilizando <i>Condor-G</i> , com gerenciamento de recursos do <i>Globus</i>	46
3.8	Serviços de Grade (adaptada de (Sotomayor, 2004))	48
3.9	Exemplo de um documento RSL	51
3.10	Submissão de uma tarefa para o gerenciador de recursos (adaptada de (Czajkowski et al.,))	52
3.11	Estrutura do Serviço de Informação (adaptada de (Joseph e Fellenstein, 2003))	53
3.12	Hierarquia de escalonadores e meta-escalonadores (adaptada de (Joseph e Fellenstein, 2003))	54
3.13	Mediador de Recursos	56
4.1	Uso de recursos computacionais de um Ambiente Virtual de Incubação (AVI)	66
4.2	Arquitetura proposta	69

4.3	Fluxo de execução	70
4.4	Árvore de classificação de recursos (adaptada de (Fleischman, 2004))	74
4.5	Publicação das informações do usuário, no Sistema de Informação	76
4.6	Processo de busca e seleção de recursos	78
4.7	Avaliador de recurso	79
4.8	Ciclo da submissão	81
5.1	Cenário de execução	85
5.2	Diagrama de Caso de Uso	87
5.3	Diagrama de Classes	89
5.4	Diagrama de Classes	90
5.5	Exemplo de um documento RSL Cliente	91
5.6	Diagrama de Classes	92
5.7	Exemplo de um arquivo grid-mapfile	93
5.8	Diagrama de Classes	93
5.9	Diagrama de Classes	94
5.10	Diagrama de Classes	95
5.11	Trecho do código da classe <i>GridNecessityTester</i>	96
5.12	Diagrama de seqüência da submissão de aplicações	97
5.13	Trecho do XML <i>Schema</i> que define o <i>SystemInfoData</i>	99
5.14	Diagrama de seqüência da publicação de informações	100
5.15	Retorno de uma consulta que busca um recurso de bom desempenho	102
5.16	Tela principal	103
5.17	Tela de Configuração	104
5.18	Requisição do certificado de segurança	105
5.19	Definição do arquivo <i>grid-mapfile</i>	106
5.20	Publicação das informações	106

5.21	Requisitos da aplicação	108
5.22	Criação do <i>proxy</i>	108
5.23	Submissão da aplicação	109

Lista de Tabelas

3.1	Principais classes de aplicações em Grade (adaptada de (Foster e Kesselman, 1999))	29
3.2	Diferenças entre <i>Clusters</i> e <i>Grades</i> ((Dantas, 2003))	34
4.1	Parâmetros de um RSL Cliente	73
5.1	Dicionário de termos comuns, no domínio de recursos computacionais	92
5.2	Computadores utilizados nos testes	104
5.3	Informações publicadas	106
5.4	Informações publicadas	109

Glossário

API *Application Programming Interface*

ASP *Application service providers*

AVI *Ambiente Virtual de Incubação*

B2B *Business to Business*

CA *Certificate Authority*

CoG *Commodity Grid Kit*

CORBA *Common Object Request Broker Architecture*

CSF *Community Scheduler Framework*

CVP *Comunidade Virtual de Profissionais*

EV *Empresa Virtual*

EVs *Empresas Virtuais*

FTP *File Transfer Protocol*

GGF *Global Grid Forum*

GRAM *Grid Resource Allocation and Management*

GRIP *Grid Interoperability Project*

GSH *Grid Service Handle*

GSI *Grid Security Infrastructure*

GSR *Grid Service Reference*

GT *Globus Toolkit*

GWSDL *Grid Web Services Description Language*

IDL *Interface Definition Language*

MPMEs Micro, Pequenas e Médias Empresas

MPI *Message Passing Interface*

OGSA *Open Grid Service Architecture*

OIL *Ontology Inference Layer*

OGSI *Open Grid Service Infrastructure*

OML *Ontology Markup Language*

OV Organização Virtual

OVs Organizações Virtuais

OWL *Ontology Web Language*

OpenPBS *Open Portable Batch System*

P2P *Peer-to-peer*

PBSPro *Portable Batch System Pro*

PVC *Professional Virtual Community*

QoS *Quality of Service*

RDF *Resource Description Framework*

RFT *Reliable File Transfer*

RLS *Replica Location Service*

ROC Redes de Organizações Colaborativas

RSL *Resource Specification Language*

SDE *Service Data Element*

SDK *Software Development Kit*

SMP Multiprocessadores simétricos

SOA *Service Oriented Architecture*

SSL *Secure Socket Layer*

SSP *Storage service providers*

SGE *Sun Grid Engine*

TIC *Tecnologias de Informação e Comunicação*

TV *Time Virtual*

URI *Uniform Resource Identifier*

VBE *Virtual Breeding Environment*

VPN *Virtual Private Network*

WSDL *Web Services Description Language*

WSRF *Web Service Resource Framework*

Capítulo 1

Introdução

1.1 Motivação

Uma grande quantidade de investimentos em recursos tecnológicos e computacionais vem sendo exigidos de empresas para que estas possam lidar com o problema de crescimento de complexidade de sistemas empresariais, como por exemplo sistemas de ERP e sistemas de gerenciamento de Organizações Virtuais (OVs). Isto se torna mais problemático quando se leva em conta que cerca de 98% das companhias são Micro, Pequenas e Médias Empresas (MPMEs)¹, e que o investimento em tecnologia de informação tem sido visto como primordial para que elas se mantenham competitivas (Camarinha-Matos e Afsarmanesh, 1999).

No entanto, os recursos computacionais dentro de uma organização/empresa são utilizados, de fato, durante uma pequena parte do seu tempo. Segundo a IBM (Berstis e Ferreira, 2002), em torno de 5% do tempo. Isso significa dizer que estes recursos são subutilizados e que durante o tempo de ociosidade eles poderiam ser melhor aproveitados para ajudar na execução de programas mais complexos, ou ainda ser compartilhados com outros usuários que no momento não dispusessem de recursos computacionais suficientes para solucionar os seus problemas. Ou seja, se houvesse uma colaboração “digital”, muitos ganhos poderiam ser obtidos. Contudo, esta questão da colaboração ainda não se encontra presente nas atuais organizações.

Porém, se nas formas tradicionais de organizações empresariais a questão da colaboração não foi sentida como questão essencial, e muito menos para as próprias sobrevivências, em algumas formas mais avançadas, como o de Organizações Virtuais, ela já o é.

O paradigma empresarial de Organizações Virtuais tem na colaboração o seu ponto-chave para que a aliança estratégica formada entre organizações/empresas obtenha sucesso

¹Fonte: IBGE 2005

na execução do seu processo de negócio. Uma OV é definida como uma agregação lógica e temporária de organizações autônomas, cooperantes e heterogêneas, que é estratégica e dinamicamente formada para atender a uma certa oportunidade de negócios, e que cujo atendimento é operacionalizado através de um compartilhamento coordenado de habilidades, recursos e informações, integralmente via rede de computadores (Rabelo et al., 2004). O paradigma de OV tem sido visto como um posicionamento estratégico de MPMEs para competirem no mercado global com alianças mais fortes (Camarinha-Matos e Afsarmanesh, 1999).

Devido a esta forte visão da colaboração presentes em OVs, tais ambientes são altamente propícios ao compartilhamento de recursos dado o natural interesse de seus membros que os demais executem sua tarefas com a maior qualidade e tempo possíveis, de forma a que, no todo, a OV atinja seus objetivos da forma mais eficiente possível. Porém, para que isso ocorra, se faz necessário algum mecanismo/tecnologia que forneça o suporte necessário para que este compartilhamento se dê de forma coordenada e segura.

No final dos anos noventa, um grupo de cientistas da comunidade de Processamento de Alto Desempenho criou o conceito de **Computação em Grades** (*Grid Computing*). A proposta era prover uma infraestrutura de *hardware* e *software* que permitisse o acesso a grandes capacidades computacionais geograficamente distribuídas, de forma confiável, consistente, econômica e persistente. Apesar de se parecer com algumas práticas mais antigas (como é o caso dos *Clusters* de computadores), as Grades Computacionais apresentam um diferencial no tocante à sua grande distribuição geográfica, e também à grande variedade de recursos envolvidos.

Contudo, a simples aplicação da Computação em Grades em uma OV não representa uma solução completa para o compartilhamento de recursos. Em primeiro lugar, apesar dos contínuos investimentos que a área vem recebendo nos últimos tempos, o acesso aos recursos através de ambientes de Grades ainda é bastante complexo, principalmente se considerarmos a necessidade de grande especialização dos usuários para utilizar esta tecnologia. Portanto, existe uma poderosa tecnologia, mas ela é, sob ótica do nível de complexidade que ela se coloca para usuários normais de MPMEs - caso típico de membros de OVs -, praticamente inacessível. Em segundo lugar, existe a relação de confiança existente entre os parceiros de uma OV. Mesmo havendo algum tipo de contrato/acordo entre as organizações quando de um negócio que estarão envolvidas, a disponibilização dos seus recursos para usuários/empresas até então desconhecidas deve ser baseada em algum critério. No caso de uma OV, dado que o aspecto *confiança* é o mais crítico para o seu sucesso, é importante que este seja fortemente considerado como critério na disponibilização e seleção de recursos disponíveis na Grade.

Com isso, é de considerável utilidade prover meios para que esses usuários/empresas possam não apenas compartilhar e fazer uso de recursos compartilhados, mas também que o seja da forma mais simples e transparente possível.

1.2 Objetivo Geral

Este trabalho tem como objetivo geral prover uma arquitetura para o lado cliente que possibilite um usuário fazer uso de recursos ociosos existentes numa OV, de forma simples e transparente, utilizando serviços de Grades Computacionais. A idéia é permitir que usuários comuns, integrantes de OVs, não tenham que dispor de grande quantidade de recursos computacionais localmente para executar suas aplicações. Para isso, o usuário terá a flexibilidade na definição dos critérios para a busca e seleção dos recursos. Além do critério de desempenho (relativos ao *hardware* do computador), também serão considerados outros dois, relevantes em OVs, que são: o nível de confiança de cada parceiro, e o custo de acesso aos recursos.

1.3 Objetivos Específicos

Derivadas do objetivo geral pretendido, os seguintes objetivos específicos são propostos:

- Avaliação da viabilidade de utilização da tecnologia de Grades computacionais em Organizações Virtuais;
- Desenvolvimento de uma ferramenta de *software* que implemente a arquitetura proposta e que sirva de base para a mencionada avaliação;
- Prover um nível de flexibilidade à ferramenta que, não apenas seja simples e permita um acesso e uso transparente a Grades computacionais, mas que seja esperta no sentido de reconhecer automaticamente quando da necessidade do uso de Grades.

Com base no levantamento bibliográfico efetuado (relatado no capítulo 3), acredita-se que este trabalho possui algumas facetas de inovação, principalmente quanto as questões de transparência e a introdução de métricas mais qualitativas na decisão acerca dos recursos a usar da Grade, como a confiança.

A arquitetura será avaliada através da implementação de um protótipo, demonstrando um estudo de caso que envolve um passo importante dentro da formação de uma Organização Virtual (OV), que é a seleção da mais adequada combinação de organizações para compor o grupo que formará uma Organização Virtual (OV). O desenvolvimento do protótipo será feito utilizando o *middleware* que já se tornou padrão *de facto* no desenvolvimento de Grades, o *Globus Toolkit*.

Este trabalho foi desenvolvido no âmbito dos projetos *ECOLEAD* e IFM (Instituto Fábrica do Milênio), dentro do Grupo de Sistemas Inteligentes de Manufatura (GSIGMA), do Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina.

1.4 Organização do Trabalho

O capítulo 2 apresenta os principais conceitos de Organizações Virtuais, que é o domínio de aplicação no qual a proposta desta dissertação é voltado.

O capítulo 3 aborda a parte conceitual de Computação em Grades que é a tecnologia de base utilizada para o desenvolvimento desta dissertação. Este capítulo inclui também uma descrição detalhada do *Globus Toolkit*, assim como são apresentados vários trabalhos correlatos à dissertação.

O capítulo 4 apresenta o modelo conceitual proposto, basicamente representado por uma arquitetura para atuar no lado cliente de um plataforma de Grades Computacionais.

O capítulo 5 discute as questões de implementação, e apresenta de forma detalhada o protótipo desenvolvido.

Por fim, o capítulo 6 apresenta as conclusões do trabalho, os resultados obtidos, e sugestões de trabalhos futuros.

Capítulo 2

Organizações Virtuais

2.1 Conceituação

A evolução das redes de computadores e tecnologias associadas trouxe a possibilidade de inovar e flexibilizar a forma como processos de negócios podem ser realizados. Através do uso dessa infraestrutura tecnológica, organizações e empresas (principalmente as pequenas e médias) ganham uma importante ferramenta para cooperarem entre si, fortalecendo-se e melhorando suas competitividades no mercado global.

Apesar da cooperação entre organizações não se tratar necessariamente de uma nova abordagem de negócio, o uso de tecnologias de informação e comunicação (TIC) se tornou uma característica-chave para que organizações e empresas possam atender a essas oportunidades da forma mais ágil possível. Essa forma de cooperação resume-se em um conceito já concretizado, que é o de **Organizações Virtuais (OVs)**.

Esse novo paradigma de cooperação trouxe desafios na forma como os sistemas empresariais são gerenciados e planejados. Para se manterem mais competitivas frente às grandes corporações, empresas e organizações têm tido que unir suas habilidades e recursos. A concretização deste paradigma, embora possível pelo desenvolvimento recente de tecnologias comunicação, redes de computadores, e logística, primeiro requer uma definição de uma arquitetura de referência adequada para cooperação e o desenvolvimento de uma plataforma de suporte flexível, e em segundo lugar o desenvolvimento de protocolos e mecanismos apropriados (Afsarmanesh e Camarinha-Matos, 1997), (Rabelo e Camarinha-Matos, 1996).

Mesmo sendo uma área de pesquisa que se encontra cada vez mais em evidência, ainda não existe uma definição padrão de Organização Virtual. Em (Bultje e Wilk, 1998), (Filos e Devine, 2000), uma OV é vista com uma coleção geograficamente distribuída de funcionalidades e entidades culturalmente diversas, conectadas através de uma infraestrutura de

telecomunicações, e que estão comprometidas a alcançar um objetivo coletivo através do compartilhamento de suas competências e recursos.

Nesta dissertação tomar-se-á a definição dada em (Camarinha-Matos e Afsarmanesh, 2004a), onde uma OV é definida como uma composição de organizações independentes, que compartilham recursos e habilidades para realizar uma determinada missão ou objetivo, mas que não estão limitadas a alianças de empresas com fins lucrativos. Apesar de OVs sempre serem compostas por uma certa quantidade de parceiros, externamente essas são vistas com uma **única organização**, que provêm um conjunto de serviços e funcionalidades.

Organizações Virtuais são, na verdade, um tipo de aliança estratégica de colaboração entre organizações. Acredita-se que nos próximos dez anos a grande maioria das organizações deverá estar envolvida em alguma aliança como forma de se manter competitiva, fazendo parte de alguma Redes de Organizações Colaborativas (ROC) (ECOLEAD, 2004).

Outros tipos de alianças são as cadeias de fornecimento (*supply chain*), empresa estendida (*extended enterprise*), agrupamento de empresas (*cluster of enterprises*), Empresas Virtuais (EVs) e Comunidade Virtual de Profissionais (CVP) (*Professional Virtual Community* (PVC)).

O paradigma emergente de Redes de Organizações Colaborativas muda fundamentalmente o modo como atividades comerciais, industriais, culturais e sociais são organizadas. A rápida evolução das tradicionais cadeias de fornecimento e práticas de terceirização, tem levado a uma crescente tendência onde tarefas são realizadas por grupos autônomos de um pequeno número de pessoas ou de pequenas e médias empresas, montados como contratantes independentes ou pequenas firmas, conectados por uma rede de computadores (Camarinha-Matos e Afsarmanesh, 2004a). Da forma como estas abordagens de colaboração vêm crescendo, chegou-se a conclusão que o desenvolvimento de ROCs devem ser baseados em contribuições de natureza multidisciplinar. Ou seja, além das tecnologias de informação e comunicação, essas contribuições devem vir dos mais diversas ramos, como: sócio-econômico, ciência cognitiva, gerenciamento organizacional e de negócio, segurança social e áreas éticas.

Uma OV é bastante abrangente quanto aos seus integrantes, podendo englobar, além de empresas, órgãos governamentais, ONGs e profissionais liberais, podendo até mesmo englobar os demais tipos de aliança.

Um ponto importante a ser ressaltado é diferença da idéia que se tem de uma OV no contexto de Redes de Organizações Colaborativas, e no contexto de Grades Computacionais. A comunidade de Computação em Grade tem uma visão bastante específica sobre este conceito, já que apenas é considerado o compartilhamento de recursos, não levando em conta as habilidades dos parceiros. Portanto, fora o capítulo 3 sobre Grades Computacionais, sempre que se mencionar OVs, será no contexto de ROCs.

Uma Comunidade Virtual de Profissionais possui uma característica singular, que é a participação de profissionais liberais ao invés de empresas e organizações. Em (Camarinha-Matos e Afsarmanesh, 2004a) uma CVP é definida como um termo que representa a combinação de conceitos de comunidade virtual e comunidade profissional. Comunidades virtuais são definidas com sistemas sociais de redes de pessoas que usam tecnologias de informação para mediar seus relacionamentos. Comunidades de profissionais fornecem ambientes para que profissionais compartilhem conhecimentos de suas profissões, como culturas de trabalho similares, percepções de problemas, técnicas para solução de problemas, valores profissionais e comportamento. Quando profissionais adotam redes de computadores e a maioria das práticas e ferramentas de comunidades virtuais, eles se tornam uma Comunidade Virtual de Profissionais.

Já um Ambiente Virtual de Incubação (*Virtual Breeding Environment* – VBE) representa uma associação ou agrupamento de organizações e suas instituições de suporte (e.g. provedores de serviços, ferramentas e ontologias), que possuem o potencial e o propósito de cooperar umas com as outras, através do estabelecimento de um acordo de cooperação de longa duração (Camarinha-Matos e Afsarmanesh, 2004a). Um Ambiente Virtual de Incubação (AVI) é uma associação aberta, mas de limites controlados de seus membros. O seu objetivo é de melhor preparar os seus membros (organizações) para se unirem a potenciais e futuras OVs, portanto provendo um “berço” para um estabelecimento ágil e dinâmico de redes colaborativas dirigidas a oportunidades (Afsarmanesh, 2004). Portanto, um AVI facilita a formação de novas OVs ou de qualquer outra forma de redes colaborativas.

Tradicionalmente, essas associações são estabelecidas em uma região geográfica, com a vantagem de possuírem uma cultura de negócios comum, e tipicamente focarem em um dos setores específicos da região. Atualmente, os desafios estão principalmente direcionados a diminuir esta restrição, ou seja, permitir que empresas geograficamente dispersas e que atuam em vários setores possam buscar uma colaboração de formas mais facilitada. Assim, AVIs trazem em si uma noção de uma aliança, um agrupamento lógico formal de empresas bastante heterogêneas, quer tecnológica quer organizacionalmente.

A fim de operacionalizar mais eficazmente uma colaboração, faz-se necessário que seus membros tenham acesso a uma adequada infraestrutura de comunicações e de serviços de suporte, ou seja, estejam preparadas para colaborar na forma de uma Organização Virtual.

Um conjunto de características relacionadas às organizações podem ser utilizadas como base para distinguir diferentes classes de ambientes dessas organizações. Essa classificação facilita na escolha das melhores características necessárias em uma OV em certos tipos de negócios. Em (Camarinha-Matos e Afsarmanesh, 1999), as seguintes características são utilizadas na classificação de organizações:

- Duração: algumas alianças entre organizações são estabelecidas para uma única opor-

tunidade de negócio e são dissolvidas no final do processo. Esta situação corresponde ao caso típico encontrado em organizações virtuais. No entanto, há também alianças de longa duração que são estabelecidas para executar um número indeterminado de processos de negócios, ou para executar um processo de negócio de longo tempo.

- **Topologia:** De acordo com a topologia de rede, existem situações onde a natureza se apresenta **variável** ou **dinâmica**, em que algumas organizações podem, dinamicamente, se unir ou deixar as alianças de acordo com as fases do processo de negócios, ou outros fatores de mercado. Neste caso faz-se necessário o auxílio de funcionalidades específicas para a busca e seleção de parceiros (fornecedores e provedores de serviços) e de procedimentos para a junção/saída de parceiros. Por outro lado, organizações virtuais podem possuir uma topologia com estrutura fixa (com pequenas variações em termos de fornecedores e clientes).

Existem trabalhos em que esta característica é abordada de maneira um pouco diferente. No caso de (Ouzounis, 2001), existem dois tipos de organizações virtuais, considerando-se a sua topologia. Quando um parceiro pode dinamicamente entrar ou sair da aliança enquanto o processo de negócio está sendo executado, sem afetar o seu funcionamento, esta aliança é **dinâmica** ou **variável**. Contudo, quando essa entrada e saída de parceiros ocorre com pouca frequência, ou simplesmente não ocorre, a aliança é caracterizada como **fixa** ou **estática**.

- **Participação:** Um outro lado a ser considerado é a possibilidade de uma organização estar participando simultaneamente de **múltiplas alianças**, ou então, dedicando-se apenas a uma **única aliança** (exclusividade de associação). No caso de OV's múltiplas, a infraestrutura de suporte deve ser capaz de lidar com vários espaços virtuais de participação, e de controlar a visibilidade das informações de acordo com os requisitos individuais de cada organização.
- **Coordenação:** Quanto à coordenação, várias abordagens podem ser encontradas. Quando uma organização dominante centraliza a coordenação, impondo seus próprios padrões, como os modelos de processos de negócio, os mecanismos de troca de informação e direitos de acesso, tem-se a abordagem conhecida como **estrutura de coordenação em estrela**. Em outros casos, as companhias participam de diferentes OV's, sem existir uma organização dominando, formam uma **aliança democrática**. Neste tipo de rede, todos os nós cooperam em igualdade, prevalecendo sua autonomia e agregando suas competências. Por fim, tem-se uma **federação**, que consiste de uma OV onde as organizações fazem uso de benefícios mútuos no gerenciamento comum dos recursos e habilidades, criando um tipo de estrutura de coordenação comum (federação).
- **Visibilidade:** As características de topologia e coordenação estão diretamente relacionadas com o aspecto de **visibilidade de escopo**, que significa o quanto da configuração

de uma OV um nó pode enxergar. Na maioria dos casos, um nó vê apenas os seus vizinhos diretos (fornecedores, clientes), tendo-se a **visibilidade de único nível**. Em situações de coordenação mais avançada, um nó pode ter alguns direitos de visibilidade maiores sobre outras organizações, que é o caso da **visibilidade de múltiplos níveis**. Isto é importante em tarefas como planejamento, escalonamento, previsão de demanda, distribuição de carga de trabalho e gerenciamento otimizado de recursos a nível de OV (e não apenas a nível interno a uma organização).

Uma OV ao longo da sua vida passa por um ciclo (Camarinha-Matos e Afsarmanesh, 1999), cuja etapas são ilustradas na figura 2.1. A seguir é feita uma descrição de cada uma destas etapas:

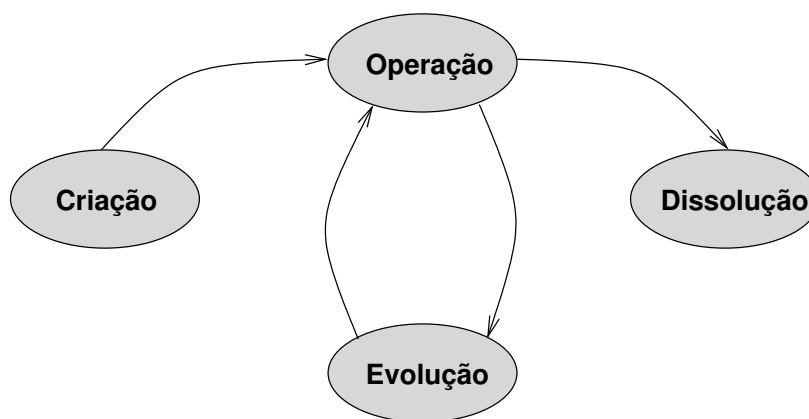


Figura 2.1: Ciclo de vida de uma Organização Virtual

- **Criação:** nesta fase são identificados os parceiros que satisfaçam os requisitos que foram especificados pelo processo de negócio. A procura por parceiros em potencial pode ser facilitada pelo uso de ferramentas como bases de dados *on-line* (diretório externo de fornecedores), diretórios de cluster e serviços de procura na Internet. Também são especificadas as regras e ajustados os parâmetros para o processo ao qual a OV se destina.
- **Operação:** esta é a fase em que a OV executa seu processo de negócio visando alcançar seus objetivos comuns. É nesta fase que as ordens de produção são executadas. Todo este processo precisa ser constantemente verificado para garantir que a OV possa alcançar seu objetivo, podendo ocorrer de os processos de negócio terem as suas especificações alteradas durante a própria execução, caso seja detectada alguma falha na condução do processo.
- **Evolução:** refinamentos podem ser necessários durante a operação da empresa virtual. Esta situação acontece quando é necessária a inclusão, exclusão ou substituição de um parceiro ou ainda uma alteração nos direitos de acesso / visibilidade de informação.

Isto pode ocorrer devido a algum evento excepcional, tal como a incapacidade de um parceiro na execução de sua tarefa ou da necessidade de incrementar a carga de trabalho.

- Dissolução: esta fase ocorre quando a OV termina seus processos de negócio e se dissolve (“separação dos parceiros”). A dissolução pode ocorrer devido a duas situações: quando os objetivos são alcançados com sucesso, ou por decisão dos parceiros para interromper a operação da OV (cancelamento da OV).

Assim como as CVPs, os AVIs têm também códigos de conduta para seus membros, regras de colaboração e, muito importante, indicadores de desempenho dos parceiros. Estes indicadores são tanto do ponto de vista histórico de cada parceiro, indicando sua “qualidade” sob vários pontos de vista, como também indicam metas de “qualidade” a serem perseguidas. Ainda, estes indicadores, além dos tradicionais quantitativos, passam a incorporar aspectos também qualitativos. Tendo em vista os requisitos existentes para se colaborar num CVP ou AVI, um deles é de especial relevância para este trabalho: a **confiança**.

Nas próximas seções o aspecto confiança é detalhado, e é enquadrado primeiramente como um dos aspectos dentro de uma filosofia de colaboração entre empresas; no caso sendo operacionalizada na forma de **partilha de informações e de recursos computacionais**. Dado que as necessidades e as disponibilidades devem poder ser expressas de alguma forma, faz-se necessário abordar o aspecto **ontologias**.

2.2 Colaboração em Organizações Virtuais

Cada vez mais empresas, organizações e profissionais têm procurado atividades complementares que os permitam participar de oportunidades de negócio em novos mercados ou em pesquisas científicas. As redes colaborativas têm sido uma resposta a essas necessidades através do desenvolvimento de arcabouços conceituais e tecnológicos para dar suporte aos seus vários tipos de alianças/arranjos estratégicos, tais como cadeias de fornecimento, empresas virtuais/organizações virtuais, comunidades virtuais de profissionais e laboratórios virtuais colaborativos. Estas formas de colaboração representam apenas um início nesta tendência, já que o número de projetos de pesquisa tem crescido largamente e vários casos práticos de diferentes formas de colaboração têm sido relatados (Camarinha-Matos e Afsarmanesh, 2004d).

Muito do que se foi feito nesta área foi conduzido por uma extensivo conhecimento empírico. Para isso, tornou-se um fator primordial consolidar e sintetizar os conhecimentos existentes, definindo uma base consistente para futuras pesquisas nesta área. O estabelecimento de uma nova disciplina científica para redes colaborativas é um forte instrumento para este propósito (Camarinha-Matos e Afsarmanesh, 2004d). A organização de tal disciplina

científica serviria como um efeito impulsionador no desenvolvimento e consolidação da área, tanto em termos de pesquisa, quanto em implantações práticas.

O projeto europeu ECOLEAD (European Collaborative Networked Organizations Leadership Initiative) (ECOLEAD, 2004), dentro do qual este trabalho de dissertação se insere (no subprojeto WP6 - Desenvolvimento de uma infraestrutura de comunicações e de serviços para Redes de Organizações Colaborativas), corresponde a uma dessas iniciativas que visam contribuir na consolidação da área.

Além da colaboração entre empresas e organizações, as pesquisas mais recentes nessa área buscam cada vez mais o entendimento das características e necessidades de redes colaborativas também baseadas em pessoas. Com isso, tornou-se um desafio definir a interação entre os conceitos de OV, AVI e CVP. De forma geral, a colaboração entre parceiros pode ocorrer nos seguintes níveis:

- Entre empresas e/ou organizações, membros de um ou vários AVIs, formando uma Empresa Virtual (EV);
- Entre profissionais liberais, membros de um ou vários CVPs, formando um Time Virtual (TV);
- Ou entre empresas, organizações e profissionais liberais, formando uma Organização Virtual (OV).

Por atingir uma maior abrangência de integrantes, a proposta desse trabalho é enquadrada em um ambiente de OVs, como é apresentado na figura 2.2.

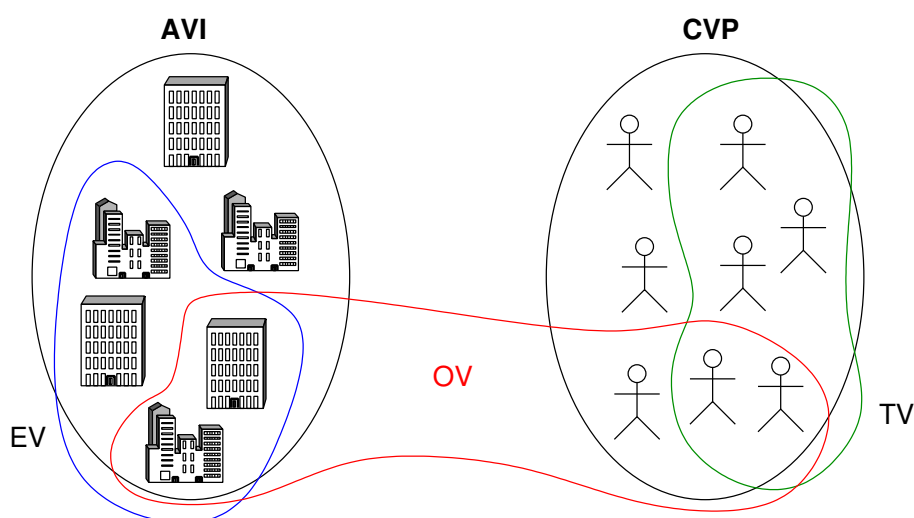


Figura 2.2: Organização Virtual

É visto que, no caso dos CVPs, é mais provável que esses façam uso dos recursos providos por empresas e organizações, e não que forneçam recursos para os demais membros da

OV. Isso se deve essencialmente ao fato destes profissionais possuírem uma quantidade de recursos computacionais bastante inferior ao de empresas e organizações, tipicamente cada um possuindo apenas um computador pessoal e freqüentemente acessível na rede via linha discada e de forma não contínua (nas 24 horas). Em outras palavras, apesar de os membros dos AVIs e CVPs terem recursos computacionais teoricamente disponíveis, na prática, em função da referida dificuldade nos membros dos CVPs, apenas os recursos de membros de AVIs - empresas - parecem fazer sentido serem compartilhados em função das requisições de aplicações-cliente externas, sejam elas oriundas de usuários de AVIs ou de CVPs.

O projeto e desenvolvimento de uma infraestrutura de tecnologia de informação e comunicação que seja transparente, de fácil uso e financeiramente viável, é um pré-requisito para a implantação efetiva e em larga escala de redes colaborativas, tais como OVs e EVs (Camarinha-Matos e Afsarmanesh, 2004c). Dentro de uma infraestrutura de redes colaborativas, a computação em Grades pode oferecer algumas funcionalidades de suporte durante algumas etapas (criação e operação) do ciclo de vida de OV (Camarinha-Matos e Afsarmanesh, 2004b):

- Busca e seleção de parceiros: feita com base em serviços de diretório, que possuem todas as informações dos recursos disponíveis na Grade, e o seu estado atual. Neste trabalho, o serviço diretório é exclusivamente utilizado na busca por recursos ociosos na rede;
- Negociação de acordo/contrato: a cada novo nó na Grade é possível definir quais usuários podem usar o recurso;
- Escalonamento e planejamento distribuído de processos de negócio: é feita uma especificação de requisitos para execução de tarefas. Através dos requisitos é feito o escalonamento para os recursos mais adequados. E através de sistemas de *workflow* é feito um plano de coordenação para execução destas tarefas;
- Avaliação dinâmica de desempenho: a cada aplicação submetida à Grade, a execução das tarefas e os recursos podem ser monitorados, o que é bastante importante para questões de gerenciamento dos recursos.

Como comentado anteriormente, a concretização da efetiva disponibilização dessas funcionalidades depende de vários fatores, de diferentes naturezas. Para o trabalho nesta dissertação, observa-se que os principais são o compartilhamento de recursos; a confiança entre os integrantes da OV, e a definição de vocabulário comum de termos.

2.2.1 Compartilhamento

Os princípios de compartilhamento refletem as regras internas seguidas durante o ciclo de vida de AVIs. Estes princípios englobam diversas categorias, como: organizacional, processo, recursos, perspectivas de negócio e sistemas de valores, e interação (Afsarmanesh, 2004). Devido ao contexto do trabalho, esta seção apenas abordará os princípios relacionados a gerenciamento de recursos, onde estes recursos englobam: tecnologias de informação e comunicação, conhecimento, direitos de propriedade intelectual e bens físicos.

As propriedades de um AVI estão diretamente ligadas aos direitos de propriedade. Um gerenciamento adequado destes direitos de propriedade, incluindo bens compartilhados, é uma questão chave para uma operação efetiva do AVI. Estes bens podem ser (Afsarmanesh, 2004):

- Base de competências: o conjunto de competências (habilidades, aptidões, e capacidade de aplicá-las) são os bens mais importantes de um AVI;
- Tecnologias de Informação e Comunicação (TIC): são ferramenta utilizadas durante o ciclo de vida do AVI. A qualidade de integração entre a infraestrutura e os serviços/aplicações de alto nível representa um tipo de bem que pode ser gerenciado por um membro central;
- Informação e Conhecimento: produzidos do uso e do gerenciamento de conhecimento, os direitos de propriedade intelectual requerem regras para uso, compartilhamento e exploração, a serem propriamente definida pelo AVI;
- Propriedades físicas: o administrador/gerenciador do AVI deve gerenciar os bens físicos compartilhados, ligados às competências dos membros do AVI. Este administrador deve também focar no crescimento e integração da capacidade de bens físicos de membros.

Um aspecto fundamental para AVIs é a colaboração baseada em conhecimento. Os membros do AVI geram informações e conhecimentos ao longo de suas operações, que são coletadas e armazenadas pela administração do AVI. Através de regras apropriadas, o acesso e uso de tais conhecimentos/ informações pode ser definido, motivando o compartilhamento entre os membros.

Já os bens físicos podem ser classificados em dois grupos (Afsarmanesh, 2004):

- O primeiro, é de bens que são parte do gerenciamento do AVI. Este tipo de bem geralmente é para uso gerencial, e não para atividades comuns de OVs. Portanto, o seu compartilhamento não é usual;

- O segundo grupo está relacionado aos bens pertencentes aos membros do AVI. Neste caso, o compartilhamento dos recursos entre os parceiros é uma prática mais comum.

Regras e políticas devem controlar o uso destes bens. Ferramentas de TIC também podem ser utilizadas para gerenciar estes bens.

Tendo-se definidas estas regras/políticas de compartilhamento de recursos, torna-se possível a aplicação de tecnologias como Grades computacionais, que oferecem um suporte robusto para melhor coordenar o compartilhamento. Por exemplo, no caso da ferramenta *Globus*, através de módulos como o *Grid Resource Allocation and Management* (GRAM) (seção 3.7.4) e o *GridFTP* (seção 3.7.3), é possível ter um maior controle dos recursos e dados compartilhados, respectivamente.

2.2.2 Confiança

Confiança é um fator crítico para uma colaboração mais eficiente e efetiva entre membros do AVI. Ela tem um papel essencial na criação de vantagens competitivas, reduzindo custos de aquisição, custos de transação entre organizações, e impacta positivamente a criação do conhecimento. Confiança permite uma comunicação aberta, compartilhamento de informações e gerenciamento de conflitos de um modo mais claro, ajudando a acelerar o processo de contrato (Afsarmanesh, 2004). Também em (Afsarmanesh, 2004) é dada uma definição para confiança, que é a adotada neste trabalho: “confiança é a expectativa positiva que uma pessoa tem em outra ou em uma organização, baseada nos seus desempenhos passados e nas suas garantias verdadeiras. Confiança é sobre expectativa do futuro. Ela acumula-se na medida que pessoas e organizações realizam bons trabalhos.”

A confiança exerce uma importante função na criação de OVs e na consequente escolha dos seus parceiros. No entanto, a construção desta confiança é uma tarefa complexa, que leva tempo, e que requer novos mecanismos a serem suportados por AVIs. Grande parte desta confiança é gerada a partir dos resultados de informações detalhadas das organizações, de acordo com desempenhos passados e presentes, que são armazenados no AVI. Dentro de um AVI, membros precisam interagir e colaborar com outros membros, que até então podem lhes ser completamente desconhecidos. Considerando a dinamicidade com que surgem as oportunidades de negócio, é muito importante um suporte para rápida construção de confiança entre estes membros.

Um conjunto bem definido de políticas que acompanhe num processo colaborativo pode ajudar na criação e crescimento de confiança entre parceiros em um AVI. Para se definir essas políticas, três importantes pontos devem ser considerados (Afsarmanesh, 2004):

- Políticas internas: estão relacionadas aos processos de definição e operação no AVI;

- Políticas externas: referem-se ao relacionamento entre cliente e os parceiros da OV;
- Políticas sobre colaboração: referem-se a relação entre os parceiros da OV, onde estão descritos as informações de confiança, políticas no processo de trabalho e para oferta de produtos e serviços.

Assim como em AVIs, o conceito de confiança se expande para CVPs. Desta forma, a confiança também se torna uma base fundamental para o controle de CVPs. O modo como configurar e garantir a confiança dentro de um CVP, da mesma forma que em um AVI, é um fator crucial, particularmente quando se está lidando com a virtualidade do relacionamento entre profissionais (Bifulco, 2005).

Neste trabalho, a questão da confiança é explorada na seleção de recursos. Considerando que os membros de um AVI e CVP se desconhecem à priori, o posicionamento natural é de que, apesar da necessidade e dos benefícios de uma colaboração, essas organizações sentem-se naturalmente receosas em abrir, partilhar, seus recursos. Por isso, a confiança acaba sendo um bom instrumento de ponderação sobre “quanto” de partilha cada parceiro está disposto a oferecer, consoante a quanto confia nos outros. Portanto, será de acordo com o nível de confiabilidade de um parceiro que será tomada a decisão final de selecionar ou disponibilizar um determinado recurso. A procura por métricas adequadas para se definir esse **nível de confiabilidade** entre parceiros é uma tarefa extramente importante e é essencial para o propósito do modelo proposto neste trabalho. Contudo, muito poucas empresas tem construído alguma ferramenta de indicadores, relevante e aplicável (Afsarmanesh, 2004). O que existe são propostas de se desenvolver um método simples e coerente para medição de confiança em um processo colaborativo, possibilitando a disponibilização dessa informação ao nível do AVI e ao do seus membros. Para fins gerenciais, a apresentação de indicadores de desempenho deve ser simples e de uso amigável.

O processo de construção de confiança depende do tamanho do AVI e da estrutura organizacional (Afsarmanesh, 2004):

- AVIs de pequena proporção: não há a necessidade de se construir um sistema computacional sofisticado para AVIs que possuem uma pequena quantidade de organizações. AVIs deste porte podem fazer um uso mais intenso de suportes mais simples de comunicação ao invés do uso fortemente dependente de tecnologias de informação;
- AVIs de grande proporção: para este tipo de AVI uma ferramenta de suporte deve ser desenvolvida voltada para facilitar o armazenamento de informação, validação e busca de informações sobre a confiança entre os parceiros. O gerenciamento da confiança é auxiliado por um uso intensivo de tecnologias de informação.

2.2.3 Ontologia

A partir do momento em que há a decisão de compartilhamento, seja de recursos ou de informações, torna-se essencial a definição de um vocabulário comum que amenize problemas de semântica entre as partes. Uma tendência atual para descrever modelos de informação/conhecimento para redes colaborativas é utilizar sistemas de ontologia. Vários projetos (e.g., *Ontobroker* (Decker et al., 1998), *OntoSeek* (Guarino et al., 1999), *MASTER-Web* (Freitas, 2002)) confirmam o crescente interesse em desenvolver abordagens baseadas em ontologias como tecnologia emergente, que permitam uma independência semântica de informação, fornecendo um conjunto de regras bem definidas com os quais ambigüidades na comunicação entre aplicações distintas são evitadas.

Na filosofia, ontologia é parte da ciência que estuda o ser e seus relacionamentos. Esta definição é bastante ampla, permitindo diversas interpretações mais específicas de acordo com a área de aplicação, seja ela sistemas de informação, lingüística ou ciência da informação (Sampaio, 2003). Contudo, de acordo com o contexto do trabalho, a definição que melhor se aproxima com o que vai ser aplicado, é dada em (Gruber, 1993), onde ontologia é definida como uma especificação formal e explícita de uma conceituação compartilhada.

Uma ontologia é um tipo de uma representação de conhecimento que descreve a conceitualização de algum domínio. Ela especifica um vocabulário incluindo termos chaves, interconexões semânticas e algumas regras de inferência (Singh e Huhns, 2005). Uma representação compartilhada é essencial para o mútuo entendimento de comunicações. De forma geral, pode-se dizer que 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 (Fleischman, 2004).

Segundo (Gava e Menezes, 2003), esta falta de entendimento compartilhado acarreta em problemas com o da interoperabilidade e da possibilidade de reuso e compartilhamento do conhecimento. A interoperabilidade é altamente desejável quando, por exemplo, diferentes sistemas necessitam acessar, prover e trocar dados através do mesmo ambiente. Em Grades Computacionais isso não é diferente. Devido a larga abrangência alcançada por ambientes de Grades, onde há uma grande quantidade de participante das mais diversas organizações, há a presente necessidade por uma forma de se modelar o conhecimento. Um claro exemplo disso encontra-se na descrição de recursos computacionais onde cada organização pode descrevê-los de modo diferenciado. Esta falta de “padronização” na descrição dificulta na localização de recursos, comprometendo a interoperabilidade entre as diferentes organizações. Para obtenção desta interoperabilidade, ontologias são usadas no desenvolvimento de modelagens que expressem o conhecimento possuído pelo domínio, formando uma camada de comunicação única e comum a todos os usuários (Fleischman, 2004).

Ontologias tem sido aplicadas em várias áreas onde o conhecimento explícito é desejável. Desta forma, foram identificados três grupos principais onde ontologias podem ser empregadas (Sampaio, 2003):

- **Troca de Informação:** é bastante utilizado em Sistemas Multiagente em suas interações. Isso é necessário porque os agentes de uma sociedade devem compartilhar do mesmo conhecimento, o que pode ser obtido através de ontologias. Além disso, a aplicação de ontologias, neste caso, permite a especialização de um conhecimento utilizado pelos agentes;
- **Estruturação da Informação:** surgiu a partir da necessidade de se adicionar semânticas às informações - *Web Semântica* - provendo um contexto compreensível para agentes inteligentes, facilitando a troca de informações entre eles;
- **Recuperação de Informação:** atuais máquinas de busca não utilizam, ou pouco utilizam, ontologias. Um exemplo é o *Yahoo*, que faz um uso parcial de ontologias, com sua estrutura montada em diretórios e sub-diretórios. Porém, não segue, necessariamente, uma hierarquia de classes.

Atualmente, existem diversas linguagens/ferramentas que auxiliam na definição de ontologias. As linguagens são necessárias para que se possa expressar as especificações de conceituações de forma legível em sistemas computacionais. Segundo (Corcho e Gomez-Perez, 2000), as linguagens existentes podem ser classificadas em dois tipos: as linguagens de representação de ontologias tradicionais; e linguagens de representação de ontologias baseadas na *Web*.

O primeiro tipo de linguagem, conhecida como **ontologia tradicional**, é baseada em lógica de primeira ordem. É utilizada basicamente para representar o 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 e Gomez-Perez, 2000). Entre as mais conhecidas, estão: Ontolingua (Farquhar et al., 1997), LOOM (MacGregor, 1991) e OCML (Motta, 1998).

Já o segundo tipo de linguagem, possui o foco voltado para aplicações de *Web Semântica*. Com isso, padrões desenvolvidos pelo W3C (W3C, 1994), como XML e *Resource Description Framework* (RDF), são utilizados no desenvolvimento de linguagens ontológicas e construção de ontologias baseadas na *Web*. Baseadas nestas duas linguagens, foram definidas diversas outras linguagens específicas para ontologia, como: *Ontology Markup Language* (OML) (Kent, 1999), *Ontology Inference Layer* (OIL) (Fensel et al., 2001), DAML+OIL (Horrocks et al., 2002) e *Ontology Web Language* (OWL) (McGuinness e van Harmelen, 2004).

Além das linguagens citadas acima, existem ferramentas para construção de ontologias que minimizam o esforço durante a sua criação. Para prover essa facilidade, essas ferramentas incluem documentação, importação e exportação de diferentes formatos, visualização gráfica, bibliotecas e mecanismos de inferência, permitindo a criação de ontologias desde o seu início, ou a partir de ontologias reutilizáveis. Atualmente, existem algumas ferramentas disponíveis, como: Apollo, LinkFactory, Ontolingua e Protégé-2000, sendo essa última, a mais utilizada.

De acordo com o apresentado na seção 3.9 sobre os trabalhos relacionados, o uso de ontologias vem ganhando bastante força dentro da computação em Grades. Por se tratar de um ambiente de grande abrangência, e altamente dinâmico, é importante que sejam definidos vocabulários comuns, para que se garanta interoperabilidade entre os sistemas. Apesar da existência de outras alternativas para a representação do conhecimento, como palavras-chave, dicionários e taxonomias (Singh e Huhns, 2005), para este propósito ontologias têm se apresentado como a opção mais utilizada pelos pesquisadores da área.

Capítulo 3

Computação em Grade

3.1 Conceituação

A tecnologia de Computação em Grade oferece uma nova abordagem para os tradicionais sistemas distribuídos, mas com a diferença de focar em compartilhamento de recursos em larga escala. Com a constante melhoria do desempenho que redes de computadores vêm experimentando, naturalmente surgiu a idéia de se utilizar computadores independentes conectados em rede como plataforma para execução de aplicações paralelas (Foster e Kesselman, 1999). Com isso se trouxe a possibilidade de executar aplicações complexas, que requerem um grande poder computacional, alocando recursos disponíveis via Internet, a um custo muito menor do que as alternativas tradicionais (e.g. supercomputadores paralelos).

Uma comparação (de forma metafórica) entre Grades Computacionais e a Rede Elétrica (*Electric Grid*) é comumente feita (Chetty e Buyya, 2002). Do mesmo modo que um usuário tem acesso a energia elétrica, de forma transparente e sob demanda, com Grades pretende-se prover poder computacional, seja ele processamento, espaço em disco, memória ou até mesmo aplicações. Um usuário que tenha tal necessidade não deve se preocupar como, ou onde, conseguir tais requisitos, da mesma forma que ao se conectar um aparelho eletrônico numa tomada, não tem-se a noção de onde (e.g. sub-estação elétrica) está vindo, ou o quanto está vindo, de energia elétrica. Apenas tem-se a garantia de que seus requisitos serão atendidos. É claro que por ser uma tecnologia recente, vários obstáculos ainda devem ser vencidos para se atingir esse nível de transparência. De forma geral, uma Grade deve ser vista com uma plataforma de execução de aplicações paralelas.

O termo "Grade" (*Grid*) foi criado em meados dos anos 90 para denotar a então proposta de infraestrutura de computação distribuída para ciência e engenharia avançada (Foster e Kesselman, 1999). As Grades Computacionais nasceram da comunidade de Processamento de Alto Desempenho (PAD). Apesar de só há pouco tempo essa tecnologia ter entrado em

evidência, alguns projetos mais antigos já fizeram uso de sua filosofia, como é o caso do pioneiro *SETI@home* (SETI@home, 2005), que surgiu em 1997, e atualmente conta milhões de participantes, chegando a uma média 14 (quatorze) TeraFLOPS de poder processamento, ou seja, 14 (quatorze) trilhões de operações por segundo. Esse projeto tem o intuito de utilizar recursos ociosos de usuários comuns que estejam conectados na Internet para a busca de sinais de vida extraterrestre a partir de dados obtidos por um radiotelescópio. Para contribuir com o projeto, basta apenas que o usuário instale um programa fornecido no *site* do projeto. Da mesma forma que o SETI@home, vários outros projetos invocam a participação de voluntários que queiram fornecer seus recursos enquanto ociosos, como por exemplo: *Compute Against Cancer* (Cancer@Home, 2005), *Fight AIDS @ Home* (FightAIDS@Home, 2005) e *Genome @ Home* (Genome@Home, 2005).

Segundo (Foster et al., 2001), o problema real e específico que fundamenta o conceito de Grades é como coordenar o compartilhamento de recursos e a solução de problemas em Organizações Virtuais dinâmicas e multi-institucionais. Isto não está primariamente relacionado a troca de arquivos, mas sim ao acesso direto a computadores, aplicações, dados, e outros recursos. Este compartilhamento é, necessariamente, altamente controlado, com consumidores e provedores de recursos definidos clara e cuidadosamente, o que é compartilhado, quem é permitido compartilhar, e as condições sobre as quais ocorrem o compartilhamento. Um conjunto de indivíduos e/ou instituições definidos por tais regras de compartilhamento forma o que é chamado de Organização Virtual (OV). É importante ressaltar as diferenças existentes na conceituação de **Organizações Virtuais** para a comunidade de Grades Computacionais e para a comunidade de Redes de Organizações Colaborativas. Estas diferenças são abordadas no capítulo 2 sobre Organizações Virtuais. Contudo, neste capítulo, a utilização do termo "Organizações Virtuais" será uma referência ao conceito dado pela comunidade de Computação em Grade.

Em (Buyya, 2002) são apresentadas as principais características de uma Grade, que a classifica como uma estrutura distribuída, heterogênea e complexa, diferenciando-se das tradicionais distribuídas. Quatro principais aspectos caracterizam uma Grade:

- **Múltiplos Domínios Administrativos e Autonomia:** os recursos de uma Grade estão geograficamente distribuídos em múltiplos domínios administrativos e são pertencentes a diferentes organizações. A autonomia dos proprietários dos recursos precisa ser respeitada juntamente com as suas políticas de gerenciamento e uso.
- **Heterogeneidade:** uma Grade envolve uma grande quantidade de recursos que são heterogêneos e compostos por diferentes tecnologias.
- **Escalabilidade:** uma Grade pode crescer de uma pequena quantidade de recursos até a faixa de milhares. Com isso surge o problema de degradação do potencial de

desempenho. Conseqüentemente, aplicações que requerem uma grande quantidade de recursos que encontram-se dispersos geograficamente, devem ser projetadas para serem tolerantes a variação de largura de banda e de latência.

- **Dinamicidade e Adaptabilidade:** em uma Grade, a falha de um recurso é uma regra e não uma exceção. De fato, com inúmeros recursos em uma Grade a probabilidade de algum recurso falhar é alta. Os gerenciadores de recursos ou das aplicações devem saber como adaptar o seu comportamento dinamicamente e fazer uso dos serviços e recursos de maneira eficiente e efetiva.

Como descrito acima, as características apresentadas são bastante particulares às Grades, portanto não presentes nas atuais tecnologias de computação distribuída (Foster et al., 2001). A Internet, por exemplo, é uma tecnologia voltada para comunicação e troca de informações entre computadores, mas que não provê uma forma integrada de coordenar o uso dos recursos. As tecnologias de empresas virtuais e de *Business to Business* (B2B) estão focados principalmente no compartilhamento de informações e, em alguns casos, de aplicações e dispositivos físicos. Tecnologias como *Common Object Request Broker Architecture* (CORBA) e J2EE, para sistemas distribuídos em empresas, permitem o compartilhamento de recursos dentro de uma única organização. Em *Storage service providers* (SSP) e *Application service providers* (ASP), permite-se que organizações terceirizem requisitos de armazenamento e computação, mas com algumas restrições (e.g. criação de uma *Virtual Private Network* (VPN)). Em resumo, pode-se observar que as atuais tecnologias, ou não suportam uma certa variedade de tipos de recursos, ou não fornecem flexibilidade e controle dos compartilhamento, características necessárias para se criar uma OV de grande escala. A partir dessas necessidades, é que se fez surgir a tecnologia de Grades.

Um conceito que consegue resumir a proposta de Grades Computacionais é dada por (Infoware, 2005):

”Computação em Grade é um tipo de sistema paralelo e distribuído que permite compartilhamento, seleção e agregação dinâmica, em tempo de execução, de recursos autônomos geograficamente distribuídos, dependendo da disponibilidade, capacidade, desempenho e custos desses recursos, assim como dos requisitos de qualidade de serviço especificados pelos usuários.”

A figura 3.1 ilustra a dimensão que pode ser atingida por uma Grade, onde um simples usuário pode ter acesso a recursos geograficamente dispersos. É bastante comum dizer que, assim como a Internet está para os dados, a Computação em Grade está para os recursos.

Como um exemplo básico de utilização de Grades, podemos citar a submissão de uma aplicação para uma (ou mais) máquina(s) remota(s). O computador no qual a aplicação é normalmente executada pode estar ocupada no momento em que usuário necessitar efetuar sua execução. Então, a aplicação em questão pode ser enviada para uma máquina disponível

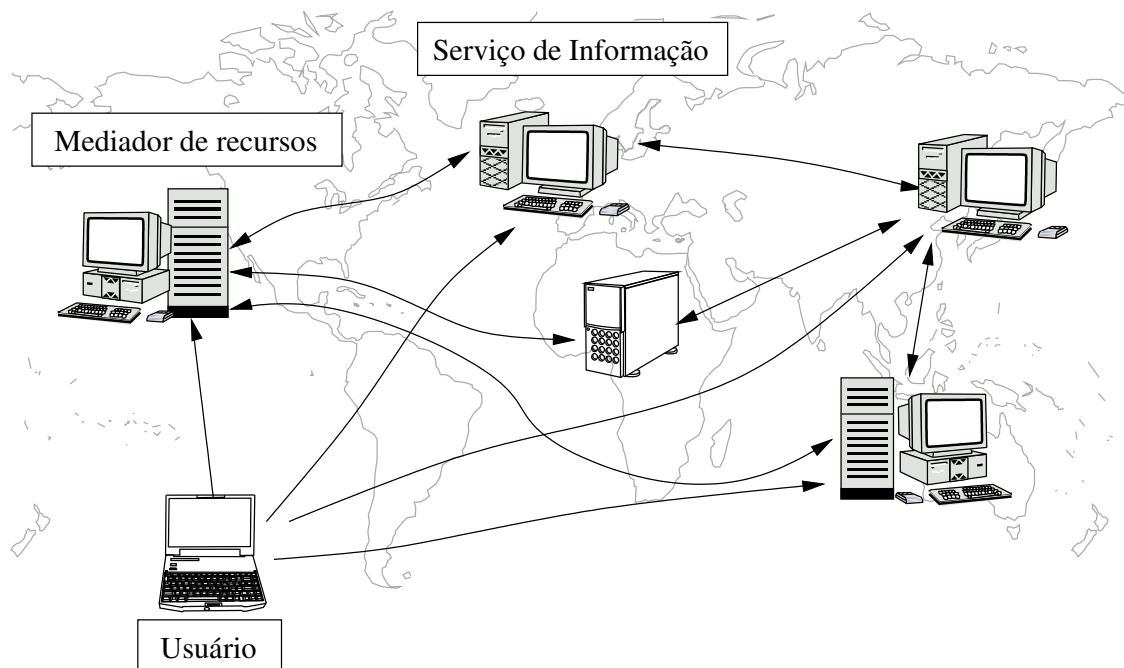


Figura 3.1: Exemplo de uma Grade

em qualquer lugar da Grade. Na maioria das organizações, há uma grande quantidade de recursos computacionais subutilizados. Como dito em (Berstis e Ferreira, 2002), a maioria dos computadores pessoais são ocupados menos de 5% do seu tempo. Até mesmo os computadores servidores podem passar uma parte do seu tempo disponível. A computação em Grade fornece um arcabouço (*framework*) para exploração de recursos ociosos, dando a possibilidade de aumentar a eficiência de uso destes.

Uma outra considerável contribuição das Grades é de tornar possível e simples a colaboração entre um grande número de parceiros, permitindo que esses compartilhem os seus recursos entre si. São fornecidos padrões que permitem sistemas heterogêneos trabalharem juntos para formar um grande sistema computacional virtual oferecendo uma variedade de recursos, portanto, virtuais. A computação em Grade também pode servir como um grande acréscimo a organizações que desejam compartilhar recursos, mas não necessariamente voltado para computação de alto desempenho. Por exemplo, um sistema composto de diversos módulos, que podem ser processados de forma independente, pode fazer uso dessa infraestrutura em prol de sua flexibilização, distribuindo esses módulos para execução nos diversos recursos da Grade.

3.2 Arquitetura

Considerando-se o que foi dito anteriormente, as mais diversas comunidades (e.g. cientistas, empresários, engenheiros) podem fazer uso dos benefícios da Computação em Grade,

e sendo assim cada uma fará uso de serviços que atendam as suas peculiaridades, de acordo com suas próprias necessidades e características.

Por se tratar de uma estrutura de grande abrangência, que envolve uma grande quantidade e variedade de parceiros e recursos, é essencial que a arquitetura de uma Grade seja projetada com base em protocolos padrões, definindo-se, assim, mecanismos básicos com os quais as OV vão estabelecer, gerenciar e explorar suas relações de compartilhamento.

Uma arquitetura aberta baseada em padrões facilita a escalabilidade, interoperabilidade, portabilidade e compartilhamento de código (Foster e Kesselman, 2004). Para uma arquitetura com tais propósitos, a interoperabilidade é fator fundamental pois irá permitir que os compartilhamentos possam se iniciados entre as partes, acomodando dinamicamente novos participantes, através de diferentes plataformas, linguagens, e ambientes de programação.

Em (Foster et al., 2001) é proposta uma arquitetura que identifica componentes fundamentais, especifica o propósito e funções desses componentes, e como os componentes interagem uns com os outros. Estes componentes são organizados na forma de camadas, sendo classificados pelas características que apresentam em comum (figura 3.2). A seguir, as camadas da arquitetura são descritas de forma resumida.

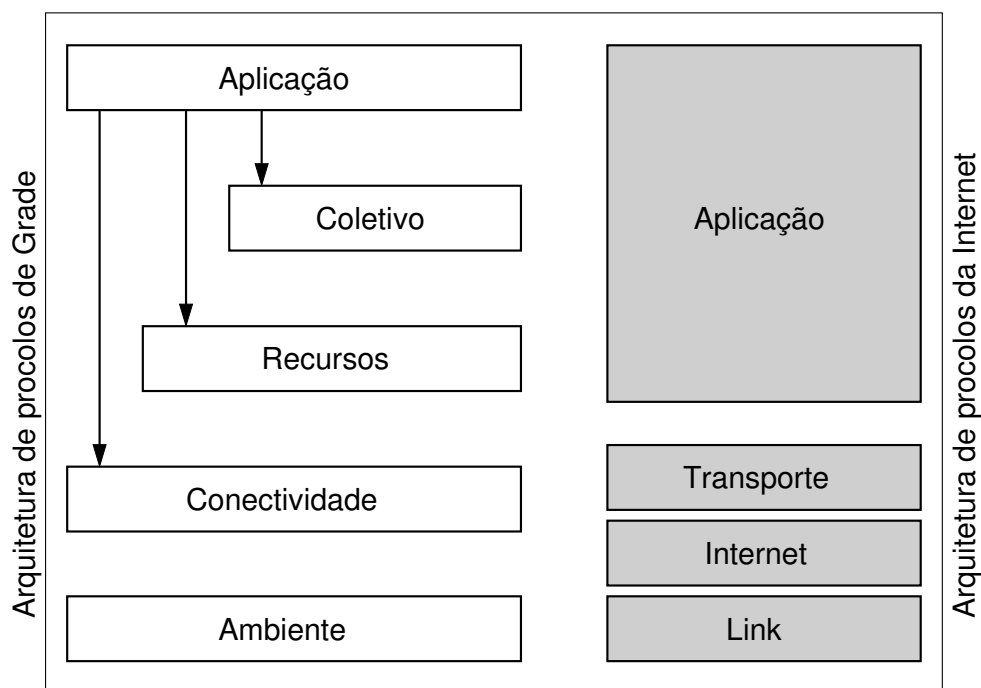


Figura 3.2: Arquitetura de uma Grade (adaptado de (Foster et al., 2001))

3.2.1 Ambiente (*Fabric*)

Essa camada fornece os recursos para os quais o acesso compartilhado é mediado por protocolos existentes na Grade, como por exemplo: recursos computacionais ou de rede, sistemas

de armazenamentos, catálogos e sensores. Os componentes dessa camada implementam as operações locais, específicas de cada recurso, que ocorrem como resultado de operações de compartilhamento em níveis superiores. Os recursos compartilhados podem ser físicos ou lógicos, como um sistema de arquivos distribuído, um *cluster* de computadores, ou um grupo de computadores distribuídos.

Alguns mecanismos são considerados básicos para o funcionamento da camada Ambiente. Por exemplo, um mecanismo de pesquisa que permita descobrir informações sobre a estrutura, estado e capacidade do recurso. E um segundo, para o gerenciamento de recursos a fim de fornecer algum controle de garantia de qualidade de serviço (*Quality of Service* – QoS).

3.2.2 Conectividade (*Connectivity*)

Essa camada define o protocolos centrais de comunicação e autenticação requeridos por transações específicas de rede em uma Grade. Os protocolos de comunicação permitem a troca de dados entre os recursos da camada **Ambiente**. Os protocolos de autenticação são construídos sobre serviços de rede, fornecendo mecanismos de criptografia para verificação da identidade de usuários e recursos.

Os protocolos utilizados na comunicação são baseados na pilha de protocolos TCP/IP, englobando os serviços de transporte, roteamento e nome. Assim como na comunicação, os aspectos de segurança da camada de **Conectividade** também são baseadas em protocolos padrões.

Algumas características são essenciais na escolha de soluções para ambientes de OV:

- **Assinatura única:** permite que um usuário se autentique uma única vez para ter acesso aos diversos recursos;
- **Delegação:** dá possibilidade de um usuário delegar suas permissões de acesso a outro usuário, permitindo que utilize os recursos em seu benefício;
- **Integração com várias soluções locais de segurança:** os provedores de recursos podem utilizar uma variedade de soluções de segurança (e.g. *Kerberos* e *Unix*);
- **Relação de confiança baseada no usuário:** permite que um usuário utilize recursos de diferentes provedores, sem a necessidade que esses provedores interajam entre si.

3.2.3 Recursos (*Resource*)

Nessa camada são definidos os protocolos, *Application Programming Interfaces* (APIs) e *Software Development Kits* (SDKs) para negociação segura, iniciação, monitoração, controle,

geração de relatórios, dentre outras tarefas para o compartilhamento de recursos individuais. As implementações desses protocolos invocam funções da camada **Ambiente** para acessar e controlar os recursos locais. Esses protocolos estão inteiramente relacionados com recursos individuais e portanto ignoram questões globais, que são tratadas na camada **Coletivo**.

Dois protocolos primordiais nessa camada, são descritos a seguir:

- **Protocolos de Informação:** são usados para obter informações sobre a estrutura e o estado do recurso (e.g. configuração, carga do processador, memória disponível, custo de acesso);
- **Protocolos de Gerenciamento:** são usados na negociação de acesso a um recurso compartilhado. Além de garantir às operações requisições aos recursos, esse protocolo deve prover mecanismos de monitoramento do estado e controle das operações.

As camadas **Recursos** e **Conectividade** formam o gargalo da arquitetura, portanto os protocolos a serem utilizados devem ser limitados a um conjunto pequeno e bem focado.

3.2.4 Coletivo (*Collective*)

Ao contrário da camada de **Recursos**, que foca em interações com um único recurso, essa camada provê protocolos, APIs e SDKs que efetiva as interações entre coleções de recursos. Seus componentes, construídos sob as duas camadas anteriores, fornecem uma grande variedade de serviços, sem exigir novos requisitos dos recursos que estão sendo compartilhados. Os serviços são:

- **Serviços de diretório:** permitem participantes de OVs descobrirem a existência e/ou propriedades dos recursos;
- **Serviços de co-alocação, escalonamento e mediação (*brokering*):** permitem participantes da OVs requisitarem a alocação de um ou mais recursos para um propósito específico, e o escalonamento de tarefas em recursos apropriados;
- **Serviços de monitoramento e diagnóstico:** dão suporte a monitoração de falhas, ataques, sobrecarga, em recursos da OV;
- **Serviços de replicação de dados:** dão suporte ao gerenciamento de recursos de armazenamento, aumentando o desempenho de acesso a dados;
- **Serviços de programação para a Grade:** fornecem modelos de programação para serem usados em ambientes de Grades, englobando diversos serviços, como: descoberta de recursos, segurança, alocação, etc.;

- **Serviço de descoberta de software:** permitem a descoberta e seleção da aplicação e plataforma de execução que sejam mais adequadas, baseada nos parâmetros do problema a ser resolvido;
- **Servidores de autorização:** forçam a definição de política de acesso a recursos;
- **Serviços de contabilização e pagamento:** coletam informações de uso dos recursos, para propósitos de cobrança pelo uso;
- **Serviços de colaboração:** dão suporte a troca coordenada de informações, podendo ser síncronas ou assíncronas.

3.2.5 Aplicação (*Application*)

A última camada da arquitetura está relacionada às aplicações dos usuários que operam dentro de um ambiente de OVs. O desenvolvimento dessas aplicações é feito a partir de chamadas a serviços fornecidos pelas demais camadas.

Apesar de a arquitetura apresentada representar as necessidades de um ambiente de Grades computacionais, e de ser a principal referenciada em trabalhos da área, ainda não existe uma arquitetura padrão. Tal padronização, faz parte dos esforços do *Global Grid Forum* (GGF) (GGF, 2005), que busca trazer benefícios à comunidade de Grades, principalmente, no tocante à interoperabilidade entre diferentes sistemas de Grades.

3.3 Áreas de Aplicação

Dentre as diversas aplicações que podem fazer uso da infraestrutura proporcionada pela Computação em Grade, as que obtêm uma maior relevância, são agrupadas em 5 principais classes, definidas em (Foster e Kesselman, 1999).

3.3.1 Supercomputação distribuída

Este tipo de aplicação faz uso de Grades para agregar uma quantidade considerável de poder computacional que possa resolver problemas que sistemas simples não conseguem. Esses recursos agregados podem variar de uma grande quantidade de supercomputadores de diversas organizações, ou do conjunto de todas as estações de trabalho de uma empresa.

A supercomputação distribuída tem uma grande usabilidade nas áreas que necessitam fazer simulações de problemas complexos, tanto na pesquisa científica, como física (e.g. o

maior acelerador de partículas do mundo que fica no CERN¹); biologia (e.g. projeto do genoma humano); meteorologia; como no campo militar, economia, finanças e computação gráfica.

Relativo à arquitetura, alguns aspectos devem ser destacados para essa classe de aplicações. Nessa perspectiva, os principais desafios estão relacionados à escalabilidade de protocolos e algoritmos para dezenas ou até centenas de milhares de nós, algoritmos tolerantes a latências, execução e suporte a altos níveis de desempenho em sistemas heterogêneos.

3.3.2 Computação de alta vazão

Nessa classe, a Grade é utilizada para escalonar uma grande quantidade de tarefas independentes ou fracamente acopladas², com o objetivo de utilizar recursos ociosos (geralmente estações de trabalho).

Apesar de, assim como a supercomputação distribuída, a computação de alta vazão buscar reunir uma certa quantidade de recursos disponíveis para solução de um único problema, há uma diferença sutil entre essas duas classes de aplicação. No caso desta, o fato de as tarefas serem independentes permite a exploração de diferentes tipos de problemas e métodos de solução de problemas, como por exemplo, em algoritmos de criptografia.

Um projeto bastante conhecido (e citado na introdução) é o SETI@Home, para busca vida extra-terrestre, que distribui dados para serem analisados em máquinas disponíveis na Internet.

3.3.3 Computação sob demanda

Aplicações sob demanda utilizam a Grade para fazer uso, por um curto período, de recursos que não podem ser mantidos localmente por questões financeiras. Esses recursos são um pouco mais diversificados nessa classe, tais como: computadores, aplicações, repositório de dados, sensores, etc.

Ao contrário das anteriores, as decisões de uso de recursos nas aplicações desta classe costumam ser orientadas por questões financeiras (e.g. o custo de se utilizar um determinado recurso) ao invés de questões de desempenho.

¹Organização Européia de Pesquisa Nuclear

²Aplicações fracamente acopladas são aplicações cujas tarefas são independentes, ou seja, não trocam, ou trocam poucas informações entre si.

3.3.4 Computação de processamento intensivo de dados

Nas aplicações que fazem uso intensivo de dados, o foco está em sintetizar novas informações a partir de dados que são mantidos em repositórios, bibliotecas digitais e bancos de dados geograficamente distribuídos. Esse processo de síntese geralmente faz um uso computacional e de comunicação intensivos.

Esse tipo de computação tem aplicações em experimentos de física, na astronomia e em sistemas de previsão meteorológica, que chegam a gerar *terabytes* de dados por dia.

Os maiores desafios nessa área estão relacionados com o escalonamento e configuração de complexos fluxos, com alta quantidade de dados, através de múltiplos níveis hierárquicos.

Por focar essencialmente em dados, essa área de aplicação de Grades ganhou um termo específico: Grades de Dados (*Data Grids*).

3.3.5 Computação colaborativa

As aplicações colaborativas têm a preocupação de melhorar a interação homem-a-homem, onde, geralmente, são definidas em termos de um espaço virtual estruturado. Muitas dessas aplicações buscam melhorar o uso compartilhado de recursos computacionais, como arquivos de dados e simulações, neste caso, se assemelhando com outros tipos de aplicações.

Exemplos práticos dessa classe podem ser: ambientes de realidade virtual, de entretenimento e de educação.

Os principais desafios de aplicações colaborativas, de uma perspectiva da arquitetura de Grades computacionais, são os requisitos de tempo real impostos e a grande variedade de interações que podem ocorrer.

Mesmo tendo-se essa classificação, ocorrem casos em que uma aplicação se enquadra em mais de um tipo, como é o caso de aplicações meteorológicas, por exemplo. Na tabela 3.1 tem-se um resumo das áreas de aplicação e suas características.

3.4 Comparação com outras tecnologias

É certo de que Grade não é a primeira e nem a única plataforma para execução de aplicações paralelas. Nesta seção será feito um comparativo de Grades computacionais com outras duas tecnologias: Par-a-par (*Peer-to-peer* (P2P)) e *Clusters*.

Categoria	Características
Supercomputação distribuída	Problemas complexos que necessitam de grande poder de processamento, memória, etc.
Alta Vazão	Reunir a maior quantidade possível de recursos (e.g. estações de trabalho) disponíveis, para aumentar a vazão
Sob demanda	Recursos remotos integrados com computação local, freqüentemente por um período de tempo limitado
Uso intensivo de dados	Extração informação a partir de grandes fontes de dados
Colaborativa	Trabalho colaborativo entre diversos participantes

Tabela 3.1: Principais classes de aplicações em Grade (adaptada de (Foster e Kesselman, 1999))

3.4.1 P2P

Com o surgimento da tecnologia P2P, uma mudança considerável ocorreu em relação aos paradigmas existentes, de forma que nessa abordagem não há uma dependência de uma organização central ou hierárquica, além de dispor aos seus integrantes as mesmas capacidades e responsabilidades (Parameswaran et al., 2001). Através dessa tecnologia qualquer dispositivo pode acessar diretamente os recursos de outro, sem nenhum controle centralizado.

O termo **par-a-par** (de *peer-to-peer*) refere-se a uma classe de sistemas e aplicações que utilizam recursos distribuídos para executar tarefas de forma descentralizada (Milojicic et al., 2002). A inexistência de um servidor central significa que é possível cooperar para a formação de uma rede P2P sem qualquer investimento adicional em *hardware* de alto desempenho para coordená-la (Rocha et al., 2004). Outra vantagem é a possibilidade de agregar e utilizar a capacidade de processamento e armazenamento que fica subutilizada em máquinas ociosas. Além disso, a natureza descentralizada e distribuída dos sistemas P2P torna-os inerentemente robustos a certos tipos de falhas muito comuns em sistemas centralizados. Finalmente, o modelo P2P apresenta o benefício da escalabilidade para tratar de crescimentos no número de usuários e equipamentos conectados, capacidade de rede, aplicações e capacidade de processamento.

Dessa forma, estabelece-se que redes P2P são redes virtuais com o objetivo de compartilhar recursos entre os seus participantes (Rocha et al., 2004). Em geral, é aceito pela comunidade científica que sistemas P2P devam suportar os seguintes requisitos (Rocha et al., 2004):

- Os nós podem estar localizados nas bordas da rede;
- Nós com conectividade variável, ou mesmo temporária, com endereços também variáveis;

- Capacidade de lidar com diferentes taxas de transmissão entre os participantes;
- Autonomia parcial ou total dos nós em relação a um servidor central;
- Assegurar que os nós possuam capacidades iguais de fornecer e consumir recursos de seus pares;
- A capacidade dos nós se comunicarem diretamente uns com os outros.

As características acima denotam uma rede P2P, mesmo que determinadas funções de controle da rede estejam localizadas em um servidor central. A figura 3.3 ilustra a rede P2P, onde cada recurso tem acesso direto a qualquer um dos participantes.

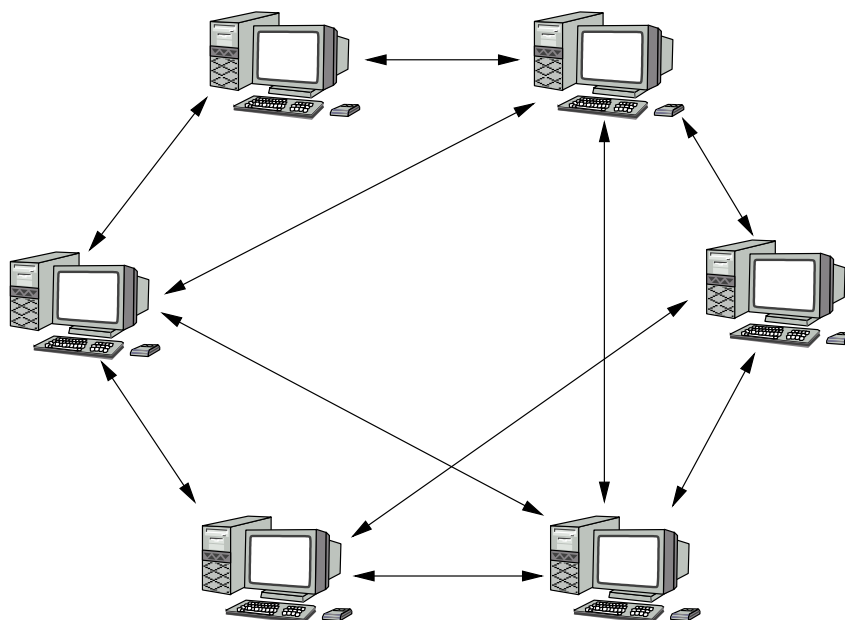


Figura 3.3: Exemplo de uma rede P2P

A partir dos conceitos e características descritas acima, pode-se notar que P2P e Grades computacionais possuem vários pontos em comum, principalmente pelo fato de agregar recursos ociosos distribuídos pela rede, e utilizá-los de forma coordenada. Apesar disso, diferentes comunidades fazem parte dessas tecnologias, da mesma forma que os focos de pesquisa também são distintos. A comunidade de Grades está mais focada em agregar recursos distribuídos de alta capacidade (e.g. *clusters*), enquanto que as comunidades de P2P buscam o compartilhamento de recursos mais simples (e.g. PC) (Buyya, 2002).

Em (Foster e Iamnitchi, 2003) são destacadas as principais características comuns às duas tecnologias:

- Ambas tecnologias estão preocupadas com o mesmo problema geral, ou seja, a organização do compartilhamento de recursos dentro de comunidades virtuais;

- As duas tecnologias buscam resolver este problema através da criação de estruturas adicionais que façam uso das estruturas organizacionais existentes;
- Cada uma das tecnologias teve várias avanços significativos, como também ainda possui várias limitações. Por exemplo, a computação em Grade tem se preocupado mais com a questão de infra-estrutura e não com as falhas, enquanto que P2P está mais preocupada com as falhas do que com a estrutura;
- A natureza complementar das forças e limitações das duas abordagens, sugere que os interesses das duas comunidades estão cada vez mais se aproximando.

Também em (Foster e Iamnitchi, 2003), são apresentados pontos que diferenciam as duas tecnologias:

- Em Grades, há uma maior ênfase na integração de recursos de alto desempenho, fornecendo melhor qualidade de serviço. Embora a computação P2P apresente um número bem maior de participantes, conseqüentemente um alto poder computacional, não há a preocupação em fornecer qualidade de serviço;
- Os serviços providos nas Grades geralmente são mais complexos;
- A computação em Grade surgiu da necessidade por processamento de alto desempenho, estando suas comunidades concentradas na pesquisa, engenharia e indústria, enquanto que P2P tornou-se popular com aplicações que permitem o compartilhamento de arquivos (e.g. *Napster* e *Kazaa*), sendo sua comunidade formada por anônimos com pouco incentivo a agir cooperativamente;
- A questão de segurança em P2P, em alguns pontos, e em algumas aplicações, não é abordada. Um exemplo é o de aplicações para compartilhamento de arquivos, onde o usuário não precisa se identificar;
- A computação em Grade se preocupa com a criação de uma infra-estrutura de múltiplo propósito, enquanto que P2P tende a focar na integração de recursos mais simples (computadores individuais) via um conjunto de protocolos desenvolvidos para prover funcionalidades esta integração.

Algumas aplicações para Grades Computacionais utilizam uma abordagem P2P para suas operações, como por exemplo, SETI@home (SETI@home, 2005). Essas aplicações podem ser caracterizadas como "Grades par-a-par", ou seja, aplicações de Grade que são suportadas por serviços de redes P2P.

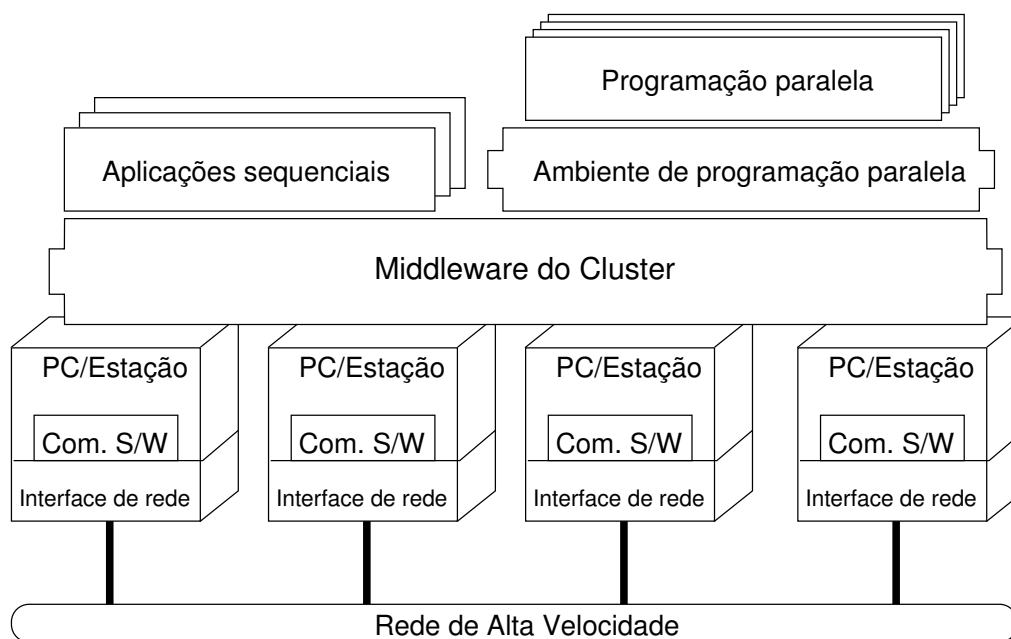


Figura 3.4: Arquitetura de um *Cluster* de computadores (adaptada de (Buyya, 1999))

3.4.2 Cluster

A computação em *Cluster* surgiu no início da década de 90, quando observaram que com a paralelização de processamento, ou seja, interligando-se dois ou mais computadores, poderia se obter uma grande quantidade de poder computacional, ao invés de se preocupar em construir processadores cada vez mais potentes (Baker e Buyya, 1988).

Um *cluster* é um tipo de sistema de processamento paralelo, que consiste de uma coleção de computadores interconectados trabalhando juntos como se fossem um único recurso integrado. O que tem contribuído para que os *clusters* sejam difundidos como uma plataforma prática para processamento paralelo é a padronização de ferramentas e utilitários usados por aplicações paralelas, como por exemplo a biblioteca *Message Passing Interface* (MPI) (Bruck et al., 1995).

Cada nó (recurso) de um *cluster* pode ser um sistema simples ou multiprocessado, com memória, interfaces de I/O e um sistema operacional. Os recursos podem ficar localizados num único lugar, ou ficarem fisicamente separados e conectados por uma LAN. A arquitetura típica de um *cluster* é apresentada na figura 3.4 (Buyya, 1999).

Dentro do que foi apresentado, os componentes abaixo são os que se destacam na arquitetura (Buyya, 1999):

- Múltiplos computadores de alto desempenho (PCs, estações de trabalho (*workstation*) ou Multiprocessadores simétricos (SMP));
- Sistemas operacionais;

- Redes e *switches* de alto desempenho;
- Placas de interface de rede;
- Protocolos e serviços para comunicação rápida;
- *Middleware* do *Cluster* (Imagem de sistema único e Infraestrutura de disponibilidade do sistema);
- Ambientes de ferramentas para programação paralela (e.g. compiladores, máquinas virtuais paralelas, e MPI);
- Aplicações: seqüenciais e paralelas.

Em (Baker e Buyya, 1988), são citadas as principais motivações na utilização de *clusters*:

- Estações de trabalho estão se tornando cada vez mais poderosas;
- A largura de banda para comunicação entre estações estão cada vez maiores;
- *Clusters* são mais facilmente integrados a redes de computadores existentes do que computadores paralelos específicos;
- As ferramentas para estações de trabalho estão mais maduras do que as soluções proprietárias para computadores paralelos, principalmente devido a falta de padronização de muitos sistemas paralelos;
- Estações de trabalho apresentam um menor custo e maior disponibilidade do que plataformas de computação de alto desempenho;
- A capacidade de estações de trabalho podem ser facilmente aumentadas, por exemplo, atualizando o processador ou aumentando a quantidade de memória.

A distinção entre *clusters* e Grades Computacionais está, principalmente, no modo como os recursos são gerenciados. No caso dos *clusters*, a alocação de um recurso é feita por um gerenciador de recursos central, e todos os nós trabalham cooperativamente para a solução de um problema. Já nas Grades, cada nó tem o seu próprio gerenciador de recurso, e não objetiva fornecer uma visão única do sistema.

É comum que *clusters* façam parte de estruturas de Grades, já que agregam um grande poder computacional a estas. Contudo, além das já citadas acima, outras diferenças, destacadas em (Dantas, 2003), podem ser observadas na tabela 3.2 abaixo:

Características	Clusters	Grades
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

Tabela 3.2: Diferenças entre *Clusters* e *Grades* ((Dantas, 2003))

3.5 Padronização

No final dos anos 90, pesquisadores da área de Grades computacionais criaram o *Grid Forum*, que posteriormente tornou-se o GGF (GGF, 2005). Desde 1999 essa organização internacional vem focando no desenvolvimento de práticas comuns, acordos, e especificações que irão promover a interoperabilidade e reuso de tecnologias de Grades.

O maior esforço está no desenvolvimento do *Open Grid Service Architecture* (OGSA) (Foster et al., 2002), um arcabouço, que foi desenvolvido juntamente com a especificação *Open Grid Service Infrastructure* (OGSI) (Tuecke et al., 2003). O arcabouço OGSA ajuda na identificação de peças que estão faltando no sistema como um todo, enquanto que a especificação OGSI leva em conta o desenvolvimento independente e paralelo de funções que irão interoperar usando a OGSI (Catlett, 2003).

Outros projetos têm trabalho em conjunto o GGF para o desenvolvimento de padrões em Grades, como é o caso do *Grid Interoperability Project* (GRIP) (GRIP, 2005), que também trabalha para interoperabilidade entre os *middlewares Globus* e *Unicore* (Erwin e Snelling, 2001).

Segundo (Berman et al., 2003), padrões como esses devem ser desenvolvidos para que se possa definir serviços centrais para uma grande variedade de áreas, como:

- Gerenciamento e automação de sistemas;
- Gerenciamento de carga de trabalho e desempenho de sistemas;
- Segurança;
- Gerenciamento de serviços;
- Gerenciamento de recursos lógicos;

- Serviços de *clustering*;
- Gerenciamento de conectividade;
- Gerenciamento de recursos físicos.

Nas seções seguintes, será feito um detalhamento das especificações OGSA e OGSI, que até o momento apresentam-se como as propostas mais promissoras de padronização.

3.5.1 *Open Grid Service Architecture*

O modelo de composição, proporcionado pela abordagem orientada a serviços (*Service Oriented Architecture* (SOA)) apresenta vantagens sobre as abordagens tradicionais. Essa característica é de grande valia, no sentido que cada vez mais as aplicações de Grades estão migrando de um forte acoplamento, para um fraco acoplamento. Uma visão que se tem de uma Grade deste tipo é a idéia de que tais Grades poderiam proporcionar serviços com fins científicos e comerciais para qualquer usuário, a qualquer momento, em qualquer lugar, e usando qualquer dispositivo. A estruturação dessas Grades neste cenário de orientação por serviços fornece funcionalidades importantes para que seja possível a formação das organizações virtuais dinâmicas (Foster et al., 2001).

O arcabouço *Open Grid Service Architecture* (OGSA) (Foster et al., 2002) define uma arquitetura simples, aberta, baseada em padrões, voltada para aplicações para Computação em Grade. Com o intuito de realizar a visão da orientação a serviços, houve uma convergência de tecnologias da área de computação de alto desempenho com a de padrões bem consolidados pela indústria. Isso ocorreu através da união de tecnologias e conceitos de Grades com Serviços *Web* (*Web Services*) (Cirne e Santos-Neto, 2005). O objetivo da OGSA é de padronizar praticamente todos os serviços encontrados nas aplicações de Grades (e.g. gerenciamento de recursos, gerenciamento de tarefas, segurança), especificando um conjunto de interfaces-padrão para estes.

O que são ditos como **serviços** podem ser: recursos computacionais ou de armazenamento, redes de computadores, programas e bancos de dados. No OGSA é definido o conceito de **Serviço de Grade** (*Grid Service*) (Tuecke et al., 2002; Foster et al., 2002; Tuecke et al., 2003). Os Serviços de Grade foram criados a partir da definição de várias extensões sobre Serviços *Web*. Na seção 3.5.3 serão discutidos as características e aprimoramentos presentes nos Serviços de Grade. Na mesma seção, também será abordada a questão dos **Dados de Serviço** (*Service Data*), que foi a forma definida para se trabalhar com estado persistente (*statefulness*).

3.5.2 *Open Grid Service Infrastructure*

A especificação *Open Grid Service Infrastructure* (OGSI) (Tuecke et al., 2003), definida pelo GGF, define um conjunto de convenções e extensões no uso de *Web Services Description Language* (WSDL) e *Schema XML* (W3C, 2001b) para permitir a criação de Serviços *Web* que possuam estados (i.e. que os seus dados sejam persistentes). São definidas abordagens para: criação, nomeação e gerenciamento do tempo de vida das instâncias dos serviços; declaração e análise de dados dos estados dos serviços; notificação assíncrona de mudança do estado dos serviços; e representação e manipulação de coleções de instâncias de serviços.

A especificação feita na OGSA não entra em detalhes sobre o que realmente é um Serviço de Grade, como também não no funcionamento destes serviços. Neste documento, apenas são identificados os serviços básicos e é apresentado o modelo da arquitetura. Desta forma, a OGSI veio sanar esta falta de detalhes, definindo uma especificação formal do que vem a ser um *Grid Service*, e descrevendo suas interfaces e comportamentos.

Dentre as definições presentes da especificação OGSI, as principais são (Cirne e Santos-Neto, 2005):

- Um conjunto de extensões para a linguagem WSDL, definido como *Grid Web Services Description Language* (GWSDL);
- Padrões de estrutura e operação, em WSDL, para representação pesquisa e atualização de dados sobre os serviços;
- As estruturas *Grid Service Handle* (GSH) e *Grid Service Reference* (GSR) usados para referenciar um serviços;
- Formato para mensagens que indicam falhas, sem modificar o modelo de mensagens de falha da linguagem WSDL;
- Conjunto de operações que permitem a criação e destruição de *Grid Services*;
- Conjunto de operações para criação e uso de coleções de Serviços *Web* por referência;
- Mecanismos que permitam notificação assíncrona, caso haja mudança nos valores dos dados dos serviço.

3.5.3 Serviços de Grade (*Grid Services*)

A atual tecnologia de Serviços *Web* oferece um poderoso arcabouço para integração de aplicações. Padrões bem definidos, como a linguagem *Web Services Description Language* (WSDL), permitem descrições concisas das interfaces dos serviços, como também a utilização

de *XML Schema* na definição de modelos de tipos. Contudo, alguns fatores impedem que essa tecnologia seja adotada por completo na infraestrutura de Grades, sendo necessárias algumas adaptações (Sandholm e Gawor, 2003).

Duas “deficiências” principais foram identificadas em relação à tecnologia de Serviços *Web*:

- A primeira diz respeito aos Serviços *Web* não possuírem estado, característica conhecida como *stateless*. Ou seja, o serviço não tem a capacidade de armazenar o seu estado entre duas requisições seguidas. Por exemplo, no caso de se executar uma cadeia de operações, seria sempre necessário enviar os resultados anteriores. Essa deficiência é sanada através da criação de uma estrutura de dados persistentes, os Dados de Serviço (*Service Data*), detalhada nas seções a seguir;
- A segunda está relacionado com a fato dos Serviços *Web* não serem transientes. Isso significa que ele permanece ativo por todas as requisições clientes. O problema com essa característica, é que a requisição de um cliente pode afetar no resultado de outro. Para isso, a especificação OGSi provê um mecanismo para criação de instância, que será melhor detalhado na seção 3.5.3.

Os aprimoramentos feitos para cobrir estas deficiências são melhor explicados a seguir.

Dados de Serviços (*Service Data*)

Os **Dados de Serviços** são considerados uma das maiores contribuições da especificação OGSi. Funciona como uma coleção estruturada de informações que é associada a uma instância de um serviço de Grade. Este foi o modo encontrado para manter a persistência dos dados de um serviço, possibilitando fazer requisição para consulta, atualização ou mudanças desses dados, conhecidos como *Service Data Element* (SDE).

Um SDE pode ser visto como uma extensão ao WSDL, onde, ao invés de se ter apenas operações, também é possível incluir atributos. Toda instância de um serviço já possui alguns SDEs padrões e cada um pode possuir tipos diferentes. Por exemplo, o serviço *Resource Information Provider Service* possui o SDE *Host* que armazena informações de *hardware* do recurso. Os valores dos dados são mantidos nos serviço, e estes podem ser consultados a qualquer momento, ou ser associados ao retorno das notificações (seção 3.5.3) quando algum desses valores for alterado. A especificação OGSi define interfaces para consultas e atualização de dados em SDEs.

De modo geral, os dados de serviços podem ser classificados em duas categorias (Tuecke et al., 2003):

- Informações do estado: são informações sobre o estado corrente do serviço, como resultado de operações, resultados intermediários e informações de tempo de execução;
- Metadados do serviço: são informações sobre o serviço em si, como dados do sistema, interfaces suportadas, custo de se usar o serviço.

Notificações

O serviço de **Notificações** cria um mecanismo que envia mensagens de uma fonte (*source*) para um receptor (*sink*). O ciclo de notificações é controlado por um **Serviço Gerenciador de Subscrição**. Essa funcionalidade permite que, a qualquer mudança (definida pelo programador do serviço) que ocorra em um serviço, todos os receptores recebam uma notificação de alteração. Para isso o serviço receptor deve discriminar os serviços os quais ele quer receber notificações de alteração.

Nome

Assim como os Serviços *Web*, um Serviço de Grade é endereçado por uma *Uniform Resource Identifier* (URI). Contudo, na especificação OGSF é definido um esquema de endereçamento mais sofisticado, o **Grid Service Handle (GSH)** (Tuecke et al., 2002). Para atender os requisitos de comunicação com o serviço, o GSH deve ser resolvido em um **Grid Service Reference (GSR)**. Dessa forma:

- GSH: aponta para um Serviço de Grade;
- GSR: especifica como se comunicar com um Serviço de Grade.

Cada GSH deve ser único e deve apontar para uma única instância de um Serviço de Grade. Ou seja, não pode haver duas instâncias com o mesmo GSH. Por outro lado, o GSR não é um apontador permanente para um serviço. Um GSR pode se tornar inválido por várias razões, como por exemplo, um serviço pode ser movido para um servidor diferente.

Gerenciamento do Ciclo de vida

O ciclo de vida de um Serviço de Grade é definido por limites que vão da criação à destruição da instância desse serviço (Tuecke et al., 2003). Os mecanismos para o gerenciamento do ciclo são fornecidos nos próprios serviços.

A criação de uma instância de um serviço é feita através de uma requisição ao mecanismo de **Fábrica** (*Factory*). Já a destruição da instância é feita através da invocação de um

método na própria instância do serviço. Esse mecanismo é definido na especificação OGSII e é implementado via uma interface Fábrica, através de uma classe do Serviço de Grade. Na requisição para a criação de uma instância, o cliente recebe como retorno o GSH do serviço, que então é resolvido para um GSR.

Grupos de Serviços

A criação de Grupos de Serviços (*Service Groups*) permite que vários serviços possam ser agregados em um só. Qualquer serviço pode ser configurado para agir como um agregador de outros serviços, disponibilizando funções como: adição de um novo serviço ao grupo, remoção de um serviços, e busca por um determinado serviço dentro grupo.

Esta funcionalidade é essencial dentro da estrutura de um serviços de diretórios para permitir o registro e busca de serviços de Grades.

3.5.4 *Web Service Resource Framework*

O WSRF foi definido pela *Globus Alliance* (Globus, 2005) e a IBM em um trabalho conjunto com a HP, com a proposta de definir convenções para manipular estados (i.e. valores de dados persistentes) para que aplicações possam buscar, analisar e interagir com recursos de um modo padrão e que seja interoperável (WSRF, 2005).

O desenvolvimento do WSRF foi baseado na especificação OGSII. Essa nova especificação é, na verdade, um melhoramento da OGSII, onde os conceitos e interfaces foram refeitos de modo a explorar os recentes desenvolvimentos na arquitetura de Serviços *Web* e os padrões existentes de XML. Além disso, a comunidade de Serviços *Web* apontou quatro críticas à especificação OGSII (Czajkowski et al., 2004):

- Apresenta uma documentação muito complexa, com grande quantidade de material, sem fazer uma separação clara das funções para que suportem uma adoção incremental;
- Não faz uso correto (i.e. formas mais comumente utilizadas) das tecnologias de Serviços *Web* e XML;
- Os conceitos de *estado persistente* e de *instâncias*, acrescentados a Serviços *Web*, acarretaram problemas em algumas implementações existentes, que passaram a não suportar a criação e destruição dinâmica de serviços;
- O fato da especificação ter acrescentado algumas extensões à versão 1.1 do WSDL, pertencentes à versão 2.0 (ainda não publicada), causaram incompatibilidades com serviços implementados com a versão atual do WSDL.

A figura 3.5 abaixo representa essa evolução, onde as especificações mais atuais de Serviços *Web* são melhor aproveitadas, e a OGSi deixa de existir como uma parte neste suporte:

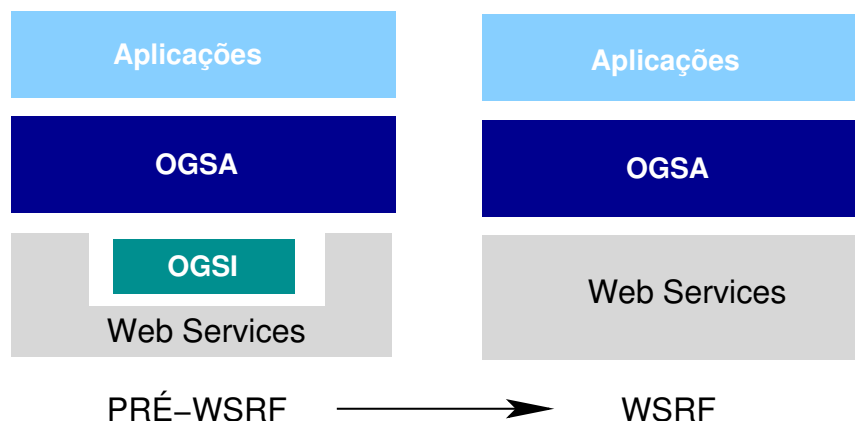


Figura 3.5: Evolução para a especificação WSRF (adaptada de (Sotomayor, 2004))

O WSRF é uma coleção de especificações para suportar serviços de grades ou outros recursos com estados persistentes. A principal contribuição dessa especificação é a intersecção entre os padrões de Grades Computacionais e Serviços *Web*, e o seu alinhamento com os princípios da abordagem orientada a serviços (SOA).

No WSRF, o conceito dos Serviços de Grade é redefinido como um *WS-Resource* (Graham et al., 2005). Um *WS-Resource* é dito como uma composição de recursos de estados persistentes com Serviços *Web*. Apesar da mudança de termo, um *WS-Resource* continua com o mesmo propósito, características e mecanismos, que foram definidos na seção 3.5.3, sobre os Serviços de Grades.

A especificação do arcabouço WSRF foi dividida em quatro grupos principais:

- *WS-Resource Properties* (WSRF-RP): esta especificação padroniza os modos pelos quais as definições das propriedades de uma *WS-Resource* devem ser declaradas como parte de uma interface de um Serviço *Web*. A declaração das propriedades do *WS-Resource* representam uma projeção, ou uma visão sobre o estado do *WS-Resource*. Esta especificação também define um conjunto padrão de troca de mensagens para permitir a consulta e atualização de informações em um *WS-Resource* (Graham e Treadwell, 2004). As propriedades de um recurso são o que anteriormente - na especificação OGSi - eram conhecidas como Dados de Serviço (*Service Data*).
- *WS-Resource Lifetime* (WSRF-RL): esta especificação define as trocas de mensagens, para padronizar os modos pelos quais *WS-Resource* deve ser destruído, e as propriedades dos recursos (*WS-Resource Properties*) que devem ser usadas para inspecionar e monitorar o tempo de vida de um *WS-Resource*. São definidos dois modos para destruir um *WS-Resource*: a destruição imediata, baseada num tempo pré-definido, e a destruição programada (Srinivasan e Banks, 2005).

- *WS-Service Group* (WSRF-SG): um Grupo de Serviços é uma coleção heterogênea de Serviços *Web* (Maguire e Snelling, 2005). Estes grupos podem ser utilizados para formar uma grande variedade de coleções de serviços, ou de *WS-Resource*. Membros de um Grupo de Serviços são representados usando componentes chamados de **entradas**, onde cada entrada é um *WS-Resource*.
- *WS-Base Faults* (WSRF-BF): esta especificação define um XML *Schema* como um conjunto básico de informações que devem aparecer em mensagens de falha (Liu e Meder, 2005).

Essa especificação foi submetida a OASIS (OASIS, 2005), em mais uma tentativa de padronização.

3.6 Ferramentas

Vários esforços vêm sendo feitos no desenvolvimento de sistemas que dêem suporte à computação em Grade, estando na área acadêmica os resultados mais significativos. Também, algumas empresas têm despendido esforços na criação de sistemas comerciais, como por exemplo o *Digipede Network* (Digipede, 2005) e o *GridServer* (DataSynapse, 2005). Contudo, devido principalmente à estas características comerciais, poucos detalhes técnicos estão disponíveis sobre o funcionamento destes. Além do mais, segundo (Cirne, 2002), questões como interoperabilidade com outros sistemas de grade ainda não são bem abordados por tais empresas. Atualmente, há um grande número de ferramentas disponíveis, e em desenvolvimento. Dentre as mais conhecidas, estão: Globus, Legion, Condor, Mygrid/OurGrid, Unicore, GridBus, Alchemi, Grip e Grace. Nas seções seguintes será feita uma explanação sobre as principais ferramentas dentre as citadas anteriormente, sendo elas: Legion, Condor e Mygrid/OurGrid. A seção 3.7 é dedicada a um maior detalhamento sobre a o Globus, ferramenta escolhida para o desenvolvimento do protótipo do modelo conceitual proposto nesta dissertação.

3.6.1 Legion

Legion (Legion, 2004) é um projeto de *software* de um meta-sistema, orientado a objetos, surgido na Universidade da Virgínia, no ano de 1993. Inicialmente, o grupo de pesquisa tinha o seu trabalho direcionado ao processamento paralelo orientado a objetos, computação distribuída e segurança, sendo que sempre focado em questões como escalabilidade, fácil programação, tolerância a falha e segurança. O *Legion* foi projetado para suportar um alto grau de paralelismo no código da aplicação e controlar a complexidade de sistemas físicos para os usuários.

Como descrito em (Grimshaw e Wulf, 1996) e em (Grimshaw et al., 1997), *Legion* é um *middleware* que combina um grande número de computadores heterogêneos, sistemas de armazenamento, bancos de dados legados, todos fisicamente distribuídos, em um único e poderoso sistema computacional. Não há uma central que controla e fiscaliza cada recurso disponível; em vez disso, cada recurso é, por si só, um elemento independente na rede *Legion*. Ele fornece meios de agrupar estes componentes dispersados, acomodando alto grau de flexibilidade e a autonomia. Isso possibilita combinar, de diferentes maneiras, os recursos disponíveis de modo a paralelizar a execução de problemas complexos e executar mais eficientemente os programas, sem ter que se preocupar com diferentes linguagens, conflitos de plataformas ou até mesmo falhas de *hardware*. Também é de responsabilidade do *Legion* suportar a abstração apresentada ao usuário final, escalonar os componentes das aplicações de forma transparente, gerenciar a migração, armazenamento e transferência dos dados, detectar e gerenciar falhas e garantir que os dados do usuário e recursos sejam protegidos adequadamente. Contudo, o *software Legion* não chegou a se apresentar como uma ferramenta completa, mas somente como um arcabouço (Dantas et al., 2002).

No *Legion*, segundo (Grimshaw e Wulf, 1996), cada ítem é representado na forma de objeto, que é um processo ativo que responde a invocações feitas por outros objetos. Estes ítems são os recursos integrantes, de *software* e de *hardware*. Nenhuma linguagem de programação ou um protocolo de comunicação é definido. Apenas define-se um formato de mensagens e um protocolo de alto nível. A definição e o gerenciamento de objetos *Legion* é feito por seu objeto de classe, que por si só, é um objeto *Legion* ativo. Esse tipo de objeto atua em nível de sistema, sobre suas instâncias, realizando a criação de novas instâncias, escalonamento de execuções, ativação e desativação das instâncias, e fornece informações sobre sua localização atual para que clientes possam comunicar-se com esta. Classes cujas instâncias são classes dela mesma são denominadas metaclasses.

Devido ao avanço nos últimos anos, nas área de redes, banco de dados e computação paralela, os projetistas do *Legion* o construíram para ser um sistema flexível e altamente suscetível a modificações. Acima de tudo, o *Legion* é um sistema aberto, feito para que terceiros desenvolvam novas e atualizadas aplicações, implementações de bibliotecas e componentes de infraestrutura básica do sistema.

Alguns objetos que fazem parte do núcleo do *Legion* fornecem serviços básicos para as classes (objetos de classe) que implementam suas próprias políticas de gerenciamento de seus objetos. Alguns serviços oferecidos são: *naming* (nomeação de recursos do sistema) e *binding*, criação de objetos, ativação/desativação e exclusão de uma instância. Esses objetos comunicam-se com outros por meio das interfaces oferecidas por estes. Essas interfaces podem ser descritas em uma linguagem de descrição de interfaces (*Interface Definition Language – IDL*).

A última versão lançada desde *middleware* foi a 1.8, em 2001. Em 1999, a empresa *Applied*

MetaComputing foi fundada para dar continuidade ao projeto *Legion*. Em 2001, adquiriu os direitos legais do *Legion*, e o nome foi mudado para *Avaki*. O *Avaki* foi lançado como produto comercial em Setembro do mesmo ano. Segundo (Grimshaw et al., 2004), na versão comercial, o núcleo da arquitetura e os princípios sobre os quais ela operava se mantiveram o mesmo.

Além da comercialização do *Legion*, a nova equipe procurou focar os seus trabalhos em tolerância a falhas (computação autônoma) e negociação de políticas de segurança entre organizações. Ainda, todo trabalho está sendo feito no contexto de padrões abertos definidos pelo GGF e a *Globus Alliance*: OGSA e OGSF.

3.6.2 *MyGrid* e *OurGrid*

O projeto *MyGrid* (Costa et al., 2004), criado na Universidade Federal de Campina Grande (UFCG), é um dos pioneiros na área de Grades Computacionais no Brasil.

A proposta do *MyGrid*, conforme (Cirne, 2002), é construir um sistema **simples, completo e seguro**. Por **simples** entende-se que a complexidade para utilização do *MyGrid* deve ser mínimo. Por **completo**, denota-se a necessidade de cobrir todo o ciclo de uso de um sistema computacional, do desenvolvimento à execução, passando por instalação e atualização e incluindo também a manipulação de arquivos. E por **seguro**, a necessidade de não introduzir vulnerabilidades ao ambiente computacional do usuário, ou seja, evitar que falhas de segurança em qualquer uma das máquinas que o usuário possa utilizar sejam propagadas para sua máquina base (computador usado pelo usuário).

Para atingir tais objetivos, *MyGrid* focou no suporte de aplicações do tipo *Bag of Tasks* (BoT), não aceitando qualquer tipo de aplicação. Aplicações *Bag of Tasks*, ou de fraco acoplamento, são aquelas cujas tarefas são independentes, isto é, não se comunicam e podem ser executadas em qualquer ordem. Segundo (Cirne, 2002), esses tipos de aplicações são importantes porque são usadas por várias áreas, tais como mineração de dados, pesquisas massivas (como quebra de chave de segurança), varredura de parâmetros, processamento genômico, fractais e manipulação de imagens (e.g. tomografia). Além disso, são bastante apropriadas para execução em Grades devido exatamente ao seu fraco acoplamento (pois o usuário pode, em princípio, se beneficiar de quaisquer processadores que ele tenha acesso), e por se adequar mais facilmente a ampla distribuição e heterogeneidade das Grades.

Além da restrição a aplicações *Bag of Tasks*, o *MyGrid* não aborda a questão de como formar a Grade. Para o *MyGrid*, segundo (Cirne, 2002), a Grade de um dado usuário é composta por todas as máquinas que o usuário pode acessar, onde esta pode conter as máquinas de seu laboratório, de outros laboratórios com que o usuário desenvolve atividades conjuntas, de algum provedor contratado para fornecer ciclos, ou até de algum amigo que forneceu um login para sua máquina.

Contudo, a solução para acesso a outros domínios administrativos não é escalável, pois uma conta para o usuário deverá ser criada em cada domínio que ele necessitar acessar. Para contornar essa limitação, foi desenvolvido o projeto *OurGrid* (OurGrid, 2004), que visa a desenvolver tecnologias para a utilização em larga escala da computação em Grade. O *OurGrid* objetiva prover mecanismos para o usuário obter acesso aos recursos que necessita, livrando o usuário de ter que negociar pessoalmente com o proprietário deste recurso.

O *OurGrid* é uma Grade par-a-par, também especializada em aplicações *Bag of Tasks* e baseada em “rede de favores” (Andrade et al., 2003). Na arquitetura dessa ferramenta cada instituição possuirá um ou mais nós *OurGrid*. Cada nó tem duas responsabilidades principais:

- gerenciar os recursos disponibilizados pelo sítio à comunidade;
- gerenciar as necessidades de recurso das Grades locais.

Quando a demanda por processamento exceder as capacidades locais, os escalonadores poderão acessar um nó *OurGrid* para que este contacte nós das outras instituições associadas para alocar mais processadores.

O *MyGrid* tem a capacidade de utilizar provedores de recursos para atender às suas demandas locais de processamento. O *OurGrid*, por sua vez, poderá desempenhar esta função em uma Grade, fornecendo, assim, os recursos necessários às aplicações submetidas no *MyGrid*. Ou seja, o *MyGrid* desempenha a função de escalonador de aplicação, ao passo que o *OurGrid* é o escalonador de recursos.

A integração entre o *MyGrid* e o *OurGrid* é representado na figura 3.6. O cenário apresenta um ambiente de Grade composto por várias instituições, onde cada instituição contém o seu nó *OurGrid* e um ou mais escalonadores de aplicação (*MyGrid*). As aplicações seriam submetidas pelo *MyGrid*, ao gerenciador de recursos *OurGrid*, que alocaria os recursos locais e de outras instituições que estivessem disponíveis para a requisição.

Atualmente o *MyGrid* e o *OurGrid* encontram-se na versão 3.0.

3.6.3 *Condor*

O projeto *Condor* (Condor, 2005) surgiu em 1988 a partir dos resultados do projeto *Remote Unix* (RU), e como continuação de um trabalho de gerenciamento de recursos distribuídos, na Universidade de *Wisconsin*, nos Estados Unidos. Seguindo seus predecessores, o projeto continuou focando na necessidade de alto poder de processamentos e em ambientes com recursos heterogêneos distribuídos.

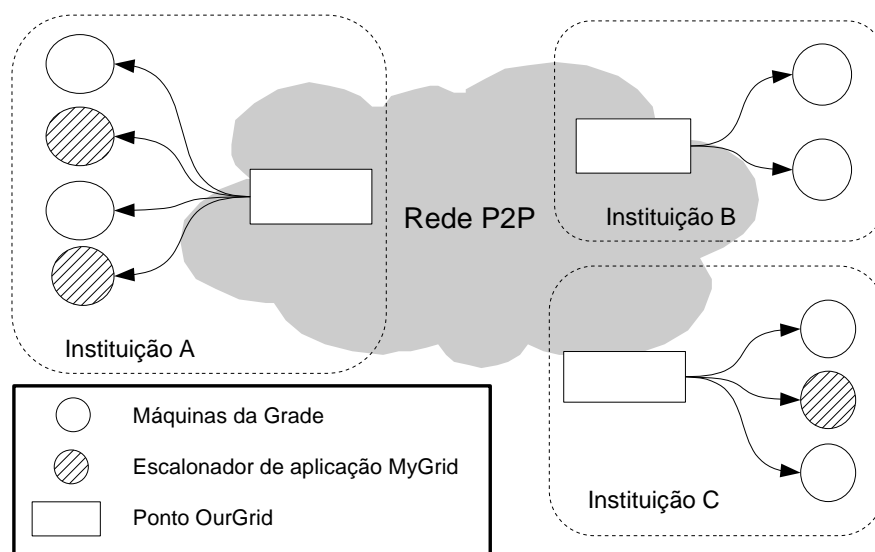


Figura 3.6: Cenário da integração do *MyGrid* e o *OurGrid* (adaptada de (Andrade et al., 2003))

O objetivo do sistema *Condor* é de fornecer grande quantidade de poder computacional a médio e longo prazos (dias a semanas) utilizando recursos ociosos na rede (Litzkow et al., 1988), tendo como foco alta vazão (*high throughput*) e não alto desempenho (*high performance*) (Litzkow et al., 1988) e (Frey et al., 2001). Segundo (Cirne, 2002), isto pode ser entendido que *Condor* visa fornecer desempenho sustentável em tais prazos, mesmo que o desempenho instantâneo do sistema possa variar consideravelmente. O *Condor* pode ser utilizado tanto no gerenciamento de nós de *clusters* dedicados, como na exploração de poder computacional de computadores pessoais ociosos.

Conforme (Thain et al., 2002), o *Condor* é um sistema de gerenciamento de recursos (*Resource Management System*, RMS) para tarefas de computação intensiva, que provê um mecanismo de gerenciamento de tarefas, políticas de escalonamento, esquemas de prioridade e monitoramento de recursos. No *Condor*, tarefas independentes são submetidas para execução. Após a submissão da(s) tarefa(s), o próprio *Condor*, baseado nas políticas, escolhe quando e onde executar a(s) tarefa(s).

O sistema *Condor* é dividido em duas partes. A primeira é responsável pelo gerenciamento das tarefas. Esse gerenciador oferece funções essenciais, tais como: mostrar fila de tarefas, fazer submissão de novas tarefas, colocar tarefas em espera, requisitar informações sobre tarefas que tenham sido completadas. A outra parte do sistema é o gerenciamento de recursos. Este é responsável por: controlar as máquinas que estão disponíveis para executar aplicações, informar como nós disponíveis devem ser utilizados dado todos os usuários que querem executar aplicações nelas, e informar, também, quando uma máquina não está mais disponível.

O gerenciador de tarefas do *Condor* é chamado de *Condor-G*. Como dito anteriormente,

este gerenciador permite submeter tarefas em uma fila, ter acesso a detalhes do registro das atividades (*log*) do ciclo de vida das tarefas gerenciar os arquivos de entrada e saída de dados. Ao invés de utilizar os protocolos próprios do *Condor* para fazer a comunicação com sistemas remotos e então submeter tarefas, o *Condor-G*³ utiliza o *Globus Toolkit*. Do *Globus*, são usados os protocolos para comunicação segura interdomínio e para acesso padronizado para uma variedade de sistemas. A figura 3.7 (adaptada de (Frey et al., 2001)) representa a integração do *Condor* com o *Globus*, onde o gerenciador de tarefas *Condor-G* utiliza o gerenciador de recursos do *Globus* para fazer submissão de aplicações para diversos nós da Grade.

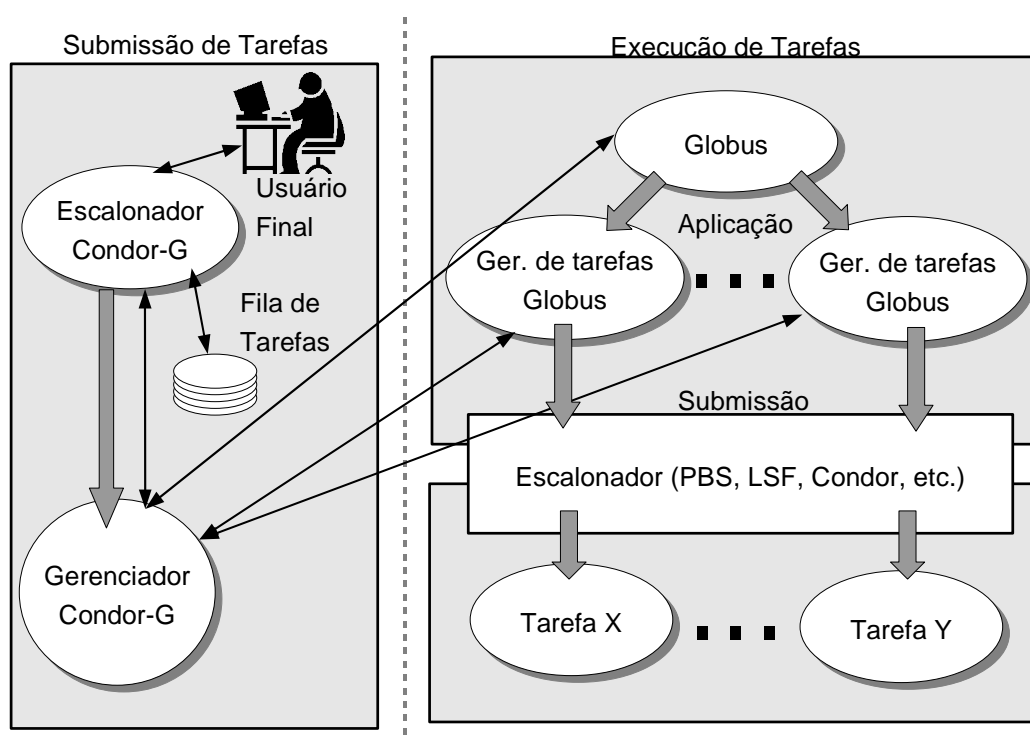


Figura 3.7: Submissão de aplicação utilizando *Condor-G*, com gerenciamento de recursos do *Globus*

A arquitetura do *Condor* se sobressai em áreas onde os tradicionais sistemas de gerenciamento de recursos têm um fraco desempenho - áreas como computação de alta vazão e computação oportunista. Segundo (Litzkow et al., 1988), o objetivo de um ambiente de computação de alta vazão é fornecer grandes quantidades de poder de computação tolerante a faltas durante um longo período de tempo, utilizando, efetivamente, todos os recursos disponíveis na rede. E o objetivo da computação oportunista é de ter a habilidade de utilizar os recursos sempre que eles estiverem livres, não requerendo 100% de disponibilidade. Sendo, assim, esses dois conceitos encontram-se naturalmente ligados; a computação de alta vazão é mais facilmente alcançada através de meios oportunistas.

³a letra 'G' indica justamente a utilização de protocolos do *Globus Toolkit*

Dentre os mecanismos fornecidos pelo *Condor*, os principais são (Thain et al., 2002):

- *ClassAds*: mecanismo que fornece extrema flexibilidade na combinação de recursos requisitados com os recursos oferecidos;
- Ponto de checagem (*checkpoint*) e migração de tarefas: com alguns tipos de tarefas o *Condor* pode transparentemente gravar um **ponto de checagem** e posteriormente reiniciar a execução da tarefa do mesmo ponto. A gravação periódica do ponto de checagem provê uma forma de tolerância a falhas.
- Chamadas a sistemas remotos: durante a execução de tarefas em máquinas remotas o *Condor* pode preservar o ambiente de execução local via chamadas a sistemas remotos. Desta forma, os usuários não precisam disponibilizar os arquivos de dados nas estações remotas antes do *Condor* executar os seus programas mesmo na ausência de um sistema de arquivos compartilhado.

Esses mecanismos tornam possíveis o “escalonamento de resumo preemptivo” (*preemptive resume scheduling*) em *clusters* dedicados. Isso permite ao *Condor* fazer escalonamento em *clusters*, baseado em prioridades. Desta forma, o *Condor* pode ser perfeitamente utilizado para combinar todo o poder computacional de uma organização em um único recurso.

Atualmente, o projeto *Condor* envolve cerca de 30 faculdades, com equipes de trabalho em tempo integral, compostas de graduandos e pós-graduandos, trabalhando na Universidade de *Wisconsin-Madison*. Juntos, os grupos possuem mais de uma centena de experiências de conceitos e práticas em computação distribuída, projetos e desenvolvimento de sistema, e em engenharia de *software* (Thain et al., 2002).

3.7 *Globus Toolkit*

A descrição a ser feita sobre o *Globus Toolkit* será sobre a versão 3.2 (GT3), baseada nas especificações OGSA/OGSI, na qual foi desenvolvido todo este trabalho. A versão 1.0 do GT foi lançada em 1998 pela *Globus Alliance*, e atualmente encontra-se na versão 4.0 (baseada nas especificações OGSA/WSRF), lançado no mês de Maio de 2005.

O *Globus Toolkit* (Foster e Kesselman, 1997) é um conjunto de serviços, de código e arquiteturas abertos, e uma biblioteca de funções que dão suporte a Grades Computacionais, e a suas aplicações (Foster et al., 2002). O GT3 é uma implementação completa da OGSI, além de fornecer uma variedade de outros serviços, programas e utilitários. A figura 3.8 enquadra os papéis da OGSA, OGSI e Globus, ilustrando com estas entidades se relacionam, formando um cenário de padrões de implementações de tecnologias para Serviços de Grade. Desta forma, tem-se representado que o OGSA define os Serviços de Grade, que por sua vez

são uma extensão de Serviços *Web*. A especificação do comportamento dos Serviços de Grade é feita pela OGSI, que é implementada pelo GT3.

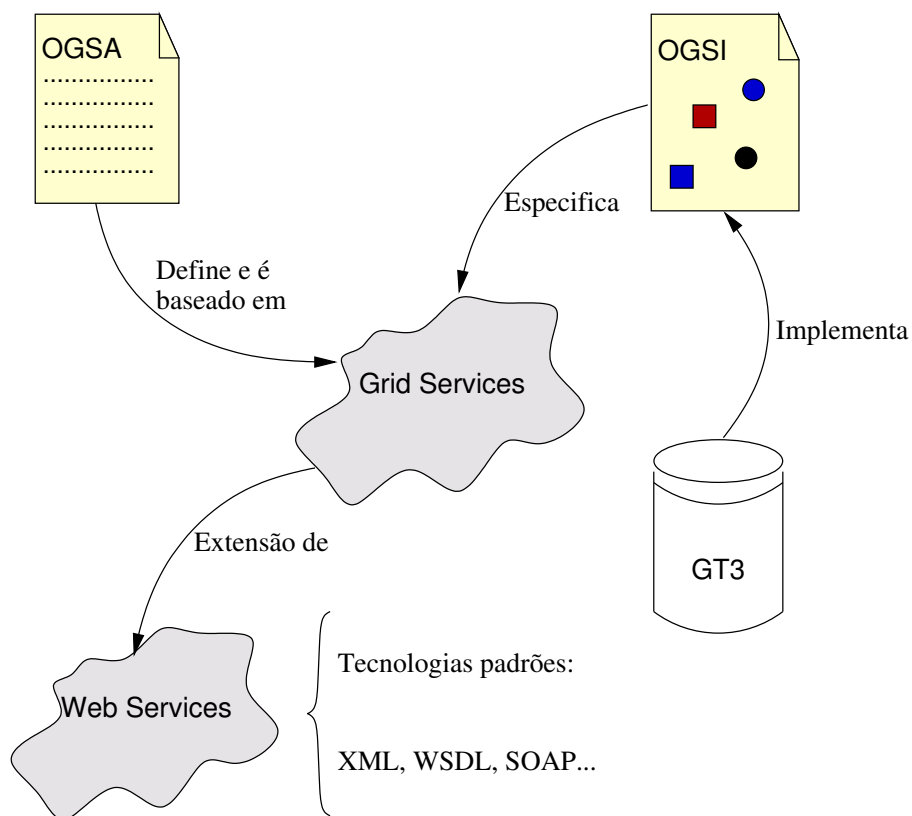


Figura 3.8: Serviços de Grade (adaptada de (Sotomayor, 2004))

O GT3 é dividido em cinco principais componentes que tratam as questões de segurança, serviços de informação, gerenciamento de recursos, gerenciamentos de dados, comunicação, detecção de falhas e portabilidade (Globus, 2005). Estes módulos são descritos nas seções a seguir.

Os componentes do *Globus* são organizados de tal forma que três deles - o **gerenciamento de recursos**, **gerenciamento de dados** e o **serviço de informação** - são construídos sobre um quarto componente, o de segurança, sendo que todos esses são suportados pelas funções básicas providas pelo **núcleo**.

3.7.1 Núcleo

O *núcleo* do GT3 contém a infraestrutura básica necessária para a criação de Serviços de Grade, oferecendo um ambiente de tempo de execução capaz de hospedá-los. Este ambiente faz a mediação entre a aplicação e o suporte de rede.

3.7.2 Segurança

Através do componente *Grid Security Infrastructure* (GSI), o GT provê serviços de autenticação de usuário para a utilização de Grades. O GSI usa criptografia de chave pública como base para suas funcionalidades. A infraestrutura é baseada no protocolo *Secure Socket Layer* (SSL), e certificados x.509.

Esse modelo contempla propriedades como autenticação, confidencialidade, integridade, controle de acesso e auditoria. O GSI tem como premissa que tais mecanismos não sejam instanciados nas aplicações, mas sim fornecidos independentemente pela infraestrutura. Com isso, cada aplicação tem a independência de instanciar um conjunto diferente de serviços de segurança.

As funcionalidades oferecidas por esse módulo são:

- Comunicação segura entre elementos de uma Grade;
- Suporte a segurança entre organizações, evitando um sistema de segurança centralizadamente gerenciado;
- Suporte a *single sign-on* para usuários, incluindo delegação de certificados para computações envolvendo múltiplo recursos.

Uma das principais características presentes nesse módulo é a autenticação *single sign-on*. Isto elimina a necessidade de o usuário se autenticar a cada submissão de uma tarefa. Isto é feito a partir da criação de um *proxy* (Welch, 2004). Um certificado proxy é um certificado de chave pública X.509, semelhante aos tradicionais, com a diferença de que ele não é assinado por uma autoridade certificadora (CA), mas sim pela entidade que está delegando poderes a outra.

3.7.3 Gerenciamento de Dados

O módulo gerenciador de dados do GT dá suporte ao acesso e a manipulação de dados distribuídos, seja em banco de dados ou arquivos.

GridFTP

Este componente provê um suporte para transferência de arquivos entre máquinas de uma Grade, e para a gerência destas transferências. O GridFTP é um procolo confiável para transferência de dados, seguro, de alto desempenho, otimizado para redes de computadores

de banda larga, e de grande abrangência. Este protocolo é baseado no protocolo *File Transfer Protocol* (FTP), que é o padrão na Internet para transferência de arquivos.

Algumas características foram adicionadas ao protocolo original (FTP) para atender os requisitos necessários em aplicações em Grades. A seguir serão citadas as principais funções do GridFTP:

- Provê níveis de integridade de dados e/ou confiabilidade para o usuário final através do suporte ao GSI;
- Permite que um usuário ou aplicação realize a transferência de dados entre dois diferentes nós. Isto é o que é conhecido como transferência *third-party*, ou seja, uma terceira “parte” faz transferência de dados entre duas outras “partes”;
- Suporta transferência paralela de dados através de extensões de comandos do FTP e de canais de dados;
- Possui extensões que permitem o particionamento de arquivos, que são enviados para diferentes servidores;
- Permite a transferência de apenas parte de arquivos, ao invés deles completos. Este tipo de transferência parcial é bastante útil em aplicações que necessitam apenas de uma parte dos dados. O FTP padrão apenas dá suporte à transferência completa do arquivo, ou do restante de uma transferência já iniciada. O GridFTP introduz novos comandos que permitem que seja selecionado qualquer parte de um arquivo;
- Dá suporte a transferência de dados, confiável e reiniciável.

Além do GridFTP, o módulo de gerenciamento de dados do *Globus* oferece dois outros serviços:

- *Replica Location Service* (RLS): é projetado para prover uma maior confiança, evitando pontos de falhas, como também provendo melhor balanceamento de carga, desempenho e escalabilidade no acesso a dados;
- *Reliable File Transfer* (RFT): tem como função fazer a transferência confiável de arquivos entre dois servidores GridFTP.

3.7.4 Gerenciamento de Recursos

Este componente é o núcleo da infraestrutura para execução remota de programas do *Globus*. O GRAM permite que aplicações sejam executadas remotamente, utilizando um

conjunto de interfaces WSDL cliente, para submissão, monitoração e finalização de uma tarefa *job*.

Dada uma especificação de uma tarefa, o serviço GRAM fornece as seguintes propriedades:

- Criação de um ambiente para uma tarefa;
- Transferir arquivos de/para o ambiente;
- Submissão da tarefa para um escalonador local;
- Monitoração da tarefa;
- Envio de notificações de mudança de estados da tarefa.

Uma das grandes contribuições do componente GRAM do Globus foi a criação da *Resource Specification Language* (RSL) (Globus, 2000), que é uma linguagem para especificação de recursos, definida em XML (W3C, 2001a), e interpretada pelo serviço gerenciador de tarefas. Em um documento RSL são definidos valores como: nome do executável, argumentos, variáveis de ambiente, quantidade de execução, arquivos de entrada e saída de dados, quantidade mínima e máxima de memória, tempo máximo de utilização da CPU.

Um trecho de um documento RSL é exemplificado na figura 3.9, onde estão descritos o executável “CalculaNumPrimo”, o seu argumento “1234567”, e a quantidade mínima de memória “512MB” necessária que a aplicação seja executada.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <rsl:rsl xmlns:rsl="http://www.globus.org/namespaces/2004/02/rsl">
3   <gram:job>
4     <gram:executable>
5       <rsl:path>
6         <rsl:stringElement value="CalculaNumPrimo" />
7       </rsl:path>
8     </gram:executable>
9     <gram:arguments>
10      <rsl:stringArray>
11        <rsl:string>
12          <rsl:stringElement value="1234567" />
13        </rsl:string>
14      </rsl:stringArray>
15    </gram:arguments>
16    <gram:minMemory>
17      <rsl:integer value="512" />
18    </gram:minMemory>
19    ...
20  </gram:job>
21 </rsl:rsl>
```

Figura 3.9: Exemplo de um documento RSL

Na figura 3.10 encontra-se representada uma submissão de uma tarefa, onde o cliente a especifica através de um documento RSL, sendo em seguida a tarefa enviada para ser executada em um recurso computacional remoto.

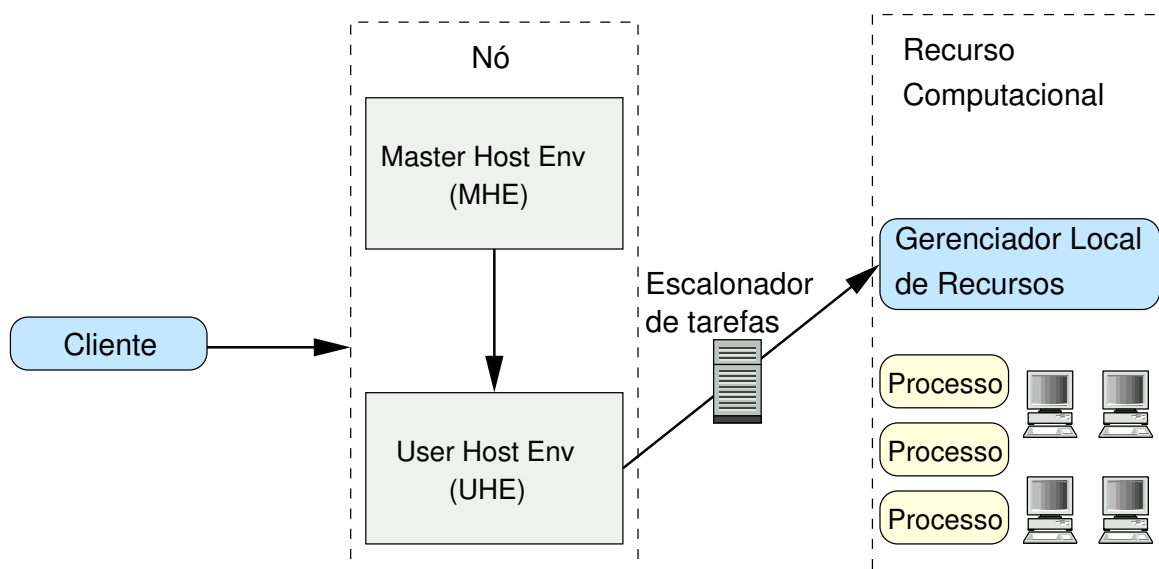


Figura 3.10: Submissão de uma tarefa para o gerenciador de recursos (adaptada de (Czajkowski et al.,))

3.7.5 Serviço de Informação

Através deste componente é possível a busca e seleção de recursos existentes em uma Grade. Ele fornece informações essenciais sobre os recursos, necessárias para a operação de Grades e para a construção de aplicações.

O serviço de informação é implementado na forma de um Serviço de Indexação (*Index Service*), que utiliza um arcabouço extensível para manipular dados estáticos e dinâmicos, em Grades. Este arcabouço fornece as seguintes funcionalidades:

- Cria e gerencia Dados de Serviços (*Service Data*) dinâmicos via programas provedores de Dados de Serviços;
- Agrega Dados de Serviços de múltiplas instâncias de Serviços de Grade;
- Registra instâncias de Serviços de Grade.

O serviço de informação tem um papel muito importante dentro da estrutura de uma Grade computacional, pois dá suporte à descoberta inicial, e posterior monitoração da existência e das características de recursos, serviços e outras entidades presentes numa Grade (Fitzgerald, 2001).

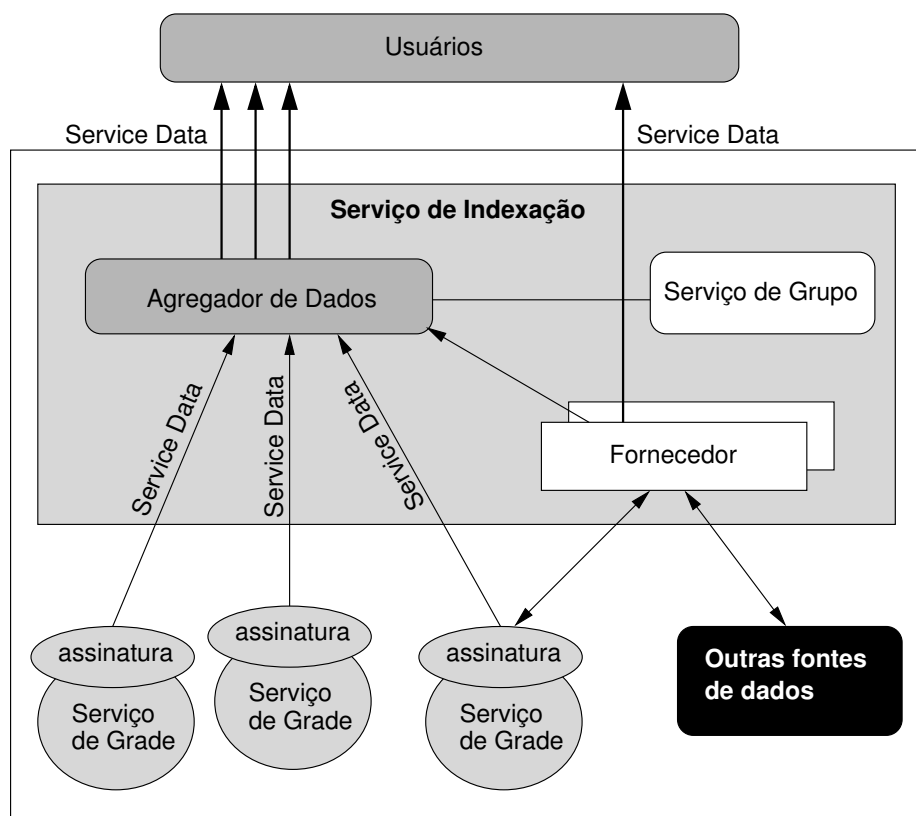


Figura 3.11: Estrutura do Serviço de Informação (adaptada de (Joseph e Fellenstein, 2003))

Empresas como a Oracle, IBM e HP estão trabalhando em conjunto com a *Globus Alliance*, produzindo ferramentas, APIs, extensões, etc. Como será observado na capítulo 5, a plataforma Globus foi a escolhida para implementar a proposta conceitual desta dissertação.

3.8 Outros Serviços

Além dos citados acima, dois outros serviços essenciais da estrutura de uma Grade não são providos pela maioria das plataformas de Grade - como é o caso do Globus -, mas sim desenvolvidos como *softwares* a parte: **Escalonador** e o **Mediador de Recurso** (*Resource Broker*).

3.8.1 Escalonadores

A partir do momento em que se permite o compartilhamento distribuído de recursos, torna-se necessário dispor de um mecanismo que faça o controle do uso simultâneo destes. Esse serviço essencial dentro da estrutura de uma Grade é fornecido pelos **escalonadores de recurso**, ou simplesmente, escalonadores.

Escalonadores são tipos de aplicações responsáveis pelo gerenciamento de tarefas, tais

como alocação de recursos necessários por uma tarefa específica; particionamento de tarefas para execução paralela; gerenciamento de dados; e capacidade de gerenciamento a nível e serviços (Joseph e Fellenstein, 2003).

Dessa forma, um escalonador faz controle de acesso aos recursos. Ou seja, não há como fazer uso de um recurso sem se ter permissão do escalonador. Uma de suas principais características é que ele deve arbitrar os vários usuários que fazem requisições de acesso e alocação.

Segundo (Cirne e Santos-Neto, 2005), devido à grande escala, ampla distribuição e existência de múltiplos domínios administrativos, não é possível construir um escalonador de recursos global para Grades. Uma razão para isto é que sistemas distribuídos que dependem de uma visão global coerente (necessária ao controle dos recursos) apresentam problemas de escalabilidade. Além disso, é muito difícil convencer os administradores dos recursos que compõem a Grade a abrirem mão do controle de seus recursos.

Portanto, os escalonadores são organizados de forma hierárquica. Na raiz dessa estrutura ficam os **meta-escalonadores**, e num nível mais baixo da estrutura, encontram-se os escalonadores, conforme é representado na figura 3.12. Os meta-escalonadores coordena a comunicação entre os múltiplos escalonadores pertencentes à estrutura. Esses escalonadores podem ser definidos como escalonadores locais, voltados para a execução de uma tarefa específica; como escalonadores de *clusters* para execução de aplicações paralelas; ou até mesmo como um outro meta-escalonador, responsável por uma outra estrutura de escalonadores.

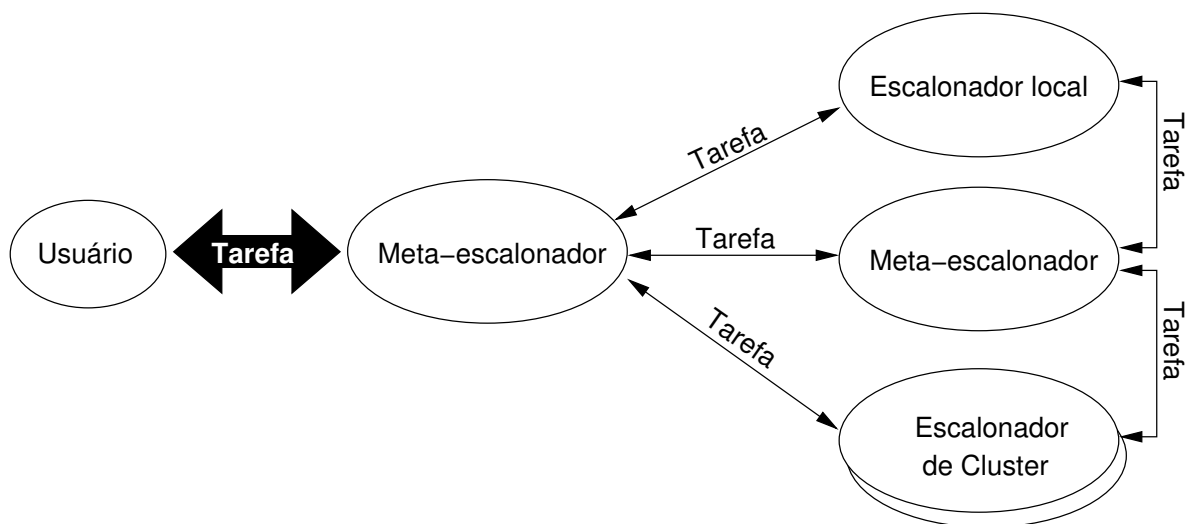


Figura 3.12: Hierarquia de escalonadores e meta-escalonadores (adaptada de (Joseph e Fellenstein, 2003))

As tarefas submetidas para os escalonadores em Grades, são avaliadas tomando como base os seus requisitos a nível de serviço e posteriormente os respectivos recursos são alocados para execução. Todo esse processo envolve um complexo gerenciamento na transferência de aplicações e de dados.

Dentre os vários escalonadores e meta-escalonadores disponíveis, os mais utilizados pela comunidade de Grades computacionais são: *Condor*, *Portable Batch System Pro* (PBSPro), *Open Portable Batch System* (OpenPBS), *Sun Grid Engine* (SGE), LSF, *Community Scheduler Framework* (CSF) e Josh. Segundo (Joseph e Fellenstein, 2003), um escalonador deve prover as seguintes funcionalidades:

- Reserva antecipada de recursos;
- Garantia e validação de acordo a nível de serviço;
- Gerenciamento de tarefas e recursos para melhor tempo de execução dentro das restrições orçamentárias permitidas;
- Monitoração da execução e do estado das tarefas;
- Re-escalonamento e ações corretivas no caso de situações de falha.

3.8.2 Mediador de Recurso

O **Mediador de Recursos** também desempenha um papel importante dentro de uma estrutura de Computação em Grades, provendo uma camada intermediária entre o usuário e os recursos disponíveis na Grade. Através dessa camada, ele fornece o serviço de casamento (*matching*) entre o recurso requisitado e os recursos disponíveis (Czajkowski et al.,). O processo de casamento envolve funções de alocação e suporte, tais como (Joseph e Fellenstein, 2003):

- Alocação do recurso apropriado, ou a combinação de recursos para a execução de tarefas;
- Suporte a restrições de orçamento e prazo de execução do usuário, para otimização de escalonamento.

A figura 3.13 mostra como o mediador se encaixa dentro da estrutura de uma Grade.

As informações a respeito dos recursos, utilizadas pelo mediador, geralmente são obtidas de um serviço de informação, como por exemplo, o MDS do *Globus*. As informações coletadas estão relacionadas a disponibilidade do recurso, modelos de uso, capacidades de *hardware* e custo de uso. Através deste serviço provido pelo mediador, o usuário livra-se da tarefa de ser ele a procurar pelos recursos que supram as suas necessidades. E se considerarmos o fato da dinamicidade presente em um ambiente de Grades, essa tarefa torna-se ainda mais complexa.

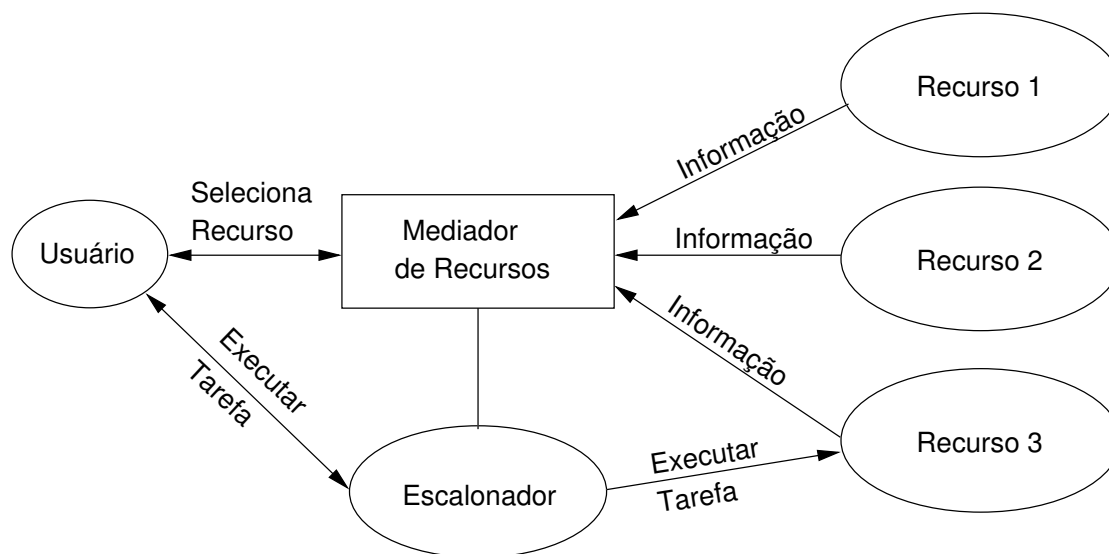


Figura 3.13: Mediator de Recursos

3.9 Trabalhos Relacionados

Até então, forneceu-se um panorama geral sobre os aspectos conceituais da Computação em Grades, bem como das plataformas de suporte mais relevantes. Tem-se observado claramente que os desenvolvimentos de ferramentas na área têm se focado em melhorias que, em suma, se concentram no lado do servidor, onde há que se gerenciar a execução das tarefas.

De acordo com o comentado no capítulo de Introdução, em virtude desse foco no servidor a tecnologia de Grades computacionais encontra grandes dificuldades de ser utilizada por um usuário normal, tipicamente encontrado em micros, pequenas e médias empresas (MPME), que compõem a quase totalidade do tecido empresarial brasileiro e internacional. Portanto, restringe-se fortemente a sua utilização ao se observar o nível que ela é apresentada a um usuário, exigindo deste grandes conhecimentos computacionais e a trabalhar com interfaces de baixo nível. Por conseguinte, seu uso fica majoritariamente reduzido a grandes organizações e instituições de pesquisa.

O objetivo maior da proposta a ser apresentada neste trabalho é a de potencializar a sua utilização através do desenvolvimento de um ambiente, uma “camada amigável” e mais transparente possível, tornando-as mais acessível a usuários de MPMEs.

Neste sentido, o centro de gravidade de desenvolvimento deve ser complementarmente redirecionado para o lado cliente. Assim sendo, trabalhos em termos de arquiteturas e de suporte a interação com usuários são necessários.

Nas duas seções que se seguem, são apresentados alguns trabalhos nessas duas perspectivas, e que por igualmente trabalharem no lado cliente, são vistos como trabalhos correlatos a o desta dissertação.

3.9.1 Grades no lado cliente

Por ser uma área de pesquisa relativamente nova, a grande maioria dos trabalhos em Grades estão focados no aprimoramento de sua infraestrutura, na criação de protocolos mais eficientes, na definição de padrões de interoperação, e no desenvolvimento de *middlewares*, escalonadores e ferramentas para criação de aplicações que façam uso dos benefícios da Grades. Portanto, o “lado servidor” tem sido o mais explorado pelos pesquisadores da área, em busca de uma infraestrutura mais robusta, segura e eficiente.

Por outro lado, apesar de todos esses investimentos, a utilização de toda essa infraestrutura de um ambiente de Grades ainda é bastante complexa. Esta complexidade, vai desde a configuração da plataforma e de quesitos de segurança, especificação dos requisitos da aplicação, até a localização dos serviços e recursos adequados a execução e o monitoramento da aplicação em ambiente remoto.

Um projeto que vem ganhando bastante espaço dentro da comunidade de Grades é o *Commodity Grid Kit* (CoG) (CoG, 2005). O CoG define e implementa um conjunto de componentes universais que mapeiam as funcionalidades de Grades em um ambiente/arcahouço para o desenvolvimento de aplicações (von Laszewski et al., 2001). O *kit* fornece uma série de APIs, nas linguagens de programação Java e Python, que facilitam a criação de aplicações que façam uso de serviços de Grades, como o de gerenciamento e busca de recursos, segurança e monitoramento de execução das aplicações (von Laszewski et al., 2000). Além disso, a proposta de se definir e implementar os componentes universais visa o mapeamento das funcionalidades entre as mais diversas plataformas de Grades. Essa integração não é feita somente através da definição de interfaces entre as plataformas e sim está mais relacionada a como os conceitos de Grades e o seus serviços são melhor expressados em termos dos conceitos e serviços de uma plataforma em particular. Atualmente, as implementações existentes estão sendo testadas na plataforma *Globus* e no *Access Grid* (Grid, 2005).

A utilização de um sistema de *workflow* dentro de uma estrutura de Grades pode prover facilidades e abstrações quanto ao controle e gerenciamento de execução das aplicações das aplicações em recursos remotos. Em (von Laszewski e Hategan, 2004), um *Workflow* de Grade (*Grid Workflow*) é referido como um conceito que assiste na instanciação de um modelo de *workflow* em uma infraestrutura de Grade existente. Dentro do projeto CoG vem sendo desenvolvido um sub-projeto direcionado nesse sentido. Karajan (von Laszewski e Hategan, 2005) é composto de uma linguagem de gerenciamento de tarefas paralelas em Grades e uma máquina de execução. A sua proposta é de prover para a comunidade científica uma ferramenta de fácil utilização para definir e gerenciar tarefas complexas em uma Grade Computacional, mantendo escalabilidade e oferecendo algumas características avançadas, como: tratamento de falhas, pontos de checagem, execução dinâmica e execução distribuída. A especificações no Karajan são feitas em uma linguagem baseada em XML. Nelas são definidas

todos os detalhes relativos a aplicação e o seu processo de execução. O Karajan é um projeto recente, lançado na última versão do CoG (4.0).

Também no sentido de prover facilidades ao usuário de Grades, foi desenvolvido o projeto *Grid Application Framework for Java* (GAF4J) (Venkatakrisnan et al.,), que abstrai os detalhes de interfaceamento com a infraestrutura de Grades, através de um modelo de programação que serve de apoio para um desenvolvimento mais rápido de aplicações clientes. Criou-se um arcabouço para que aplicações Java *multi-thread* possam explorar recursos compartilhados em uma Grade. Como plataforma de infraestrutura o projeto GAF4J utiliza o *Globus Toolkit* na sua versão mais antiga, a 2.4, e ainda não foi atualizada para versões baseadas nos padrões orientados a serviços.

Alguns trabalhos buscam prover arcabouços que facilitem o gerenciamento da execução de aplicações em Grades. O projeto *GridWay* (Huedo et al., 2004) visa prover essas facilidades, através da automatização de todos os passos no escalonamento de tarefas, fornecendo mecanismos de recuperação de falhas. Em busca de melhor desempenho na submissão de aplicações, a execução das tarefas é adaptada às condições dinâmicas dos recursos e à demanda das aplicações. Este arcabouço está disponível somente para aplicações baseadas em versões antigas (2.4) *Globus Toolkit*, que não suportam a abordagem orientada a serviços.

Uma das áreas que está começando a ganhar espaço no desenvolvimento de aplicações em Grades é a construção de portais que permitem que usuários de aplicações e computadores de alto desempenho façam acesso a recursos via uma interface de página *web*. Alguns projetos como o *Open Grid Computing Environment* (OGCE) (OGCE, 2004), *GridSphere Portal* (GridSphere, 2004) e o *GridPort* (Dahan e Boisseau, 2001) tem buscado criar componentes simples que possam ser utilizados por desenvolvedores de portais na construção de páginas na Internet, que ofereçam serviços de autenticação de segurança em recursos remotos, como também informações sobre esses recursos, ajudando nas decisões de escalonamento. Além disso, são criados perfis dos usuários, permitindo que esses monitorem a execução das suas aplicações e visualizem os resultados.

Uma avaliação de como as tecnologias de Grades Computacionais e de Agentes podem colaborar entre si é feita de maneira abrangente em (Argonne et al., 2004). Nele, são traçadas algumas semelhanças entre as duas abordagens, como por exemplo na criação de comunidades para alcançar um determinado objetivo. Dentro das várias contribuições que podem ser adicionadas às Grades, como composição de sistemas e desenvolvimento de aplicações, os agentes podem desempenhar um papel eficiente na busca de serviços e recursos em uma Grade. O fato da arquitetura abstrata da *Foundation of Intelligent Physical Agents* (FIPA) (O'Brien e Nicol, 1998) reforçar uma arquitetura baseada em serviços, permite que os agentes expressem as suas habilidades e capacidades na forma de serviços. Realizar uma busca por um serviço, ou um recurso computacional em um Serviço de Informação, composto por um serviço de diretório, por exemplo, como acontece na arquitetura da FIPA, é mais fácil porque

os agentes podem procurar por características específicas do serviço, ontologias utilizadas, dentre outras informações (Assunção, 2004).

Os mediadores tem assumido cada vez mais o papel de facilitadores na busca de recursos na Grade (Afgan, 2004). Apesar de haver alguns mediadores disponíveis já consolidados, como *Nimrod/G* (Nimrod, 2005), *GridBus* (GridBus, 2004) e *GridWay* (Huedo et al., 2004), nenhum desses ainda apresenta a utilização de mecanismos de semântica (baseada em ontologias) para, por exemplo, fazer o casamento dos recursos. Na seção 3.9.2, o tópico sobre a utilização de semântica em Grades, será discutido com mais detalhes. Além disso, ainda não houve uma preocupação maior de convergi-los para a abordagem orientada a serviços. Contudo que nenhum desses citados acima dão suporte às versões mais novas (3.2 e 4.0) do *Globus Toolkit*, e somente com a sua versão mais antiga (2.4).

O trabalho descrito em (Buyya et al., 2000), (Abramson et al., 2002) apresenta um diferencial, por considerar em sua avaliação na busca por recursos, questões relacionadas ao custo de uso dos recursos, além dos fatores de *hardware* e *software*. Essa abordagem vem sendo conhecida como “Economia em Grades” (*Grid Economy*). A seleção de recursos, considerando esse critério, pode ser de duas formas. Na primeira, o mediador tenta completar a tarefa dentro do prazo e custo estipulados. No segundo caso, o usuário entra em acordo, através de uma negociação, em busca de qual recurso aceita tal quantia e prazo para executar a sua tarefa. Nesta dissertação, esta questão será abordada de forma bastante simples.

Em (Mulder e Meijer, 2004), é introduzido o conceito de *squads* - abreviação de *squadrons* - que é um grupo que age como um time para realizar uma missão específica. A idéia deste trabalho é utilizar o paradigma empresarial de Organizações Virtuais (OVs) para uma melhor coordenação, e um desenvolvimento e manutenção de *softwares* de modo mais eficiente. Isso é feito através do uso destes *squads*, que são “equipados” com dispositivos móveis para sua comunicação e acesso a um ambiente de rede central. Portanto, este trabalho propõe um modelo de um sistema de gerenciamento de informação, baseado em um Serviço de Grade, que fornece informações sobre a Grade e as aplicações que estão executando sobre este sistema.

A partir do momento em que se disponibiliza recursos em uma Grade, esses recursos estão sujeitos a aceitar todo tipo de aplicação. O trabalho feito em (Chien, 2002) e (Calder et al., 2005) descreve o projeto da Máquina Virtual Entropia (*Entropy Virtual Machine - EVM*). Este máquina virtual pretende garantir a proteção de computadores de mesa, de aplicações instáveis e/ou maliciosas, em ambientes de Grades. O sistema deve garantir que as aplicações não irão interferir o desempenho e nem danificar a máquina ou os dados do usuário. A EVM suporta a execução controlada e segura de aplicações expressas em código nativo x86, para sistemas operacionais Windows NT, 2000 e XP.

3.9.2 Semântica em Grades

A utilização de semântica vem tendo um crescimento substancial nas várias áreas de computação em Grades. Alguns trabalhos foram encontrados, demonstrando em que pontos a aplicação de semântica pode trazer maiores benefícios.

Em (De Roure et al., 2003) é traçado um relacionamento entre Semântica *Web* e as Grade computacionais, onde, da mesma forma que a computação em Grade evoluiu da *Web* (permitindo o compartilhamento de recursos, além de informações), a Semântica *Web* está evoluindo para o que está sendo chamada de **Grades Semânticas** (*Semantic Grid*).

Enquanto a visão orientada a serviços veio dar força ao problema da coordenação de recursos compartilhados, atividades nos campos da pesquisa, engenharia e indústria, surgem com outras necessidades, focando-se na interoperação de informações heterogêneas. Nestes campos tem-se a necessidade por metadados compartilháveis e acessíveis computacionalmente, para suportar descoberta, integração e agregações de informação de forma automatizada, já que operam em um ambiente global, distribuído e dinâmico (Goble e De Roure, 2002b; De Roure e Hendler, 2004).

Em (Goble e De Roure, 2002a) é feita uma classificação em que áreas de Grades a *web* semântica pode atuar:

- Na **infraestrutura** são citados dois pontos em que a semântica pode ser útil. O primeiro trata da forma como os serviços (recursos) são descritos, sendo que esse é um fator essencial na automatização de várias operações sobre serviços como: descoberta, busca, seleção, interoperação, composição, execução e monitoração. Essas operações dependem dos metadados dos serviços, ou seja, de acordo com a forma em que esses são descritos, tais operações podem ter um melhor desempenho. A classificação dos serviços baseada em suas funcionalidades tem sido adotada por diversas comunidades como uma forma eficiente de se identificar um serviço adequado. O segundo ponto aborda a integração de informação. Um questão complexa, enfrentada por pesquisadores dos mais diversos campos de pesquisa, é quando se deseja unir várias fontes heterogêneas de informações, as quais variam em formato, interfaces e estruturas. As Grades de Dados (*Data Grids*) garantem um certo nível de interoperabilidade na recuperação e acesso aos dados. Neste caso, o uso semântica acrescenta um nível a mais de interoperabilidade, possibilitando entender o significado dos dados e assim esses podem ser ligados de forma apropriada, fornecendo um suporte automatizado para esse processo de integração (Goble, 2000).
- Nas **aplicações de Grades** que trabalham com grandes quantidades de dados que podem fazer uso de mecanismos de *Web Semântica* na descoberta de informações. Este tem sido o maior investimento do uso de semântica em aplicações para Grades computacionais. Este tipo de aplicações podem usufruir dos benefícios da semântica em:

workflows, como em (Gil et al., 2004); análise de parâmetros; anotação de resultados; descrição de pessoas, laboratórios, literatura e ferramentas. Alguns trabalhos já foram desenvolvidos nesse sentido, como em (Cannataro, 2003), onde é implementado o *Knowledge Grid*, ou Grade de Conhecimento, um ambiente para projeção e execução de aplicações de descoberta de informações distribuídas geograficamente; em (Tao et al., 2003), onde é definida uma arquitetura de semântica em Grades para busca e otimização de projetos de engenharia; ou em (Ruscic et al., 2003), no qual é definido um arcabouço para construção de aplicações científicas em Grades voltado para uma área específica da química de Tabelas Termoquímicas Ativas.

Foram alguns encontrados alguns trabalhos que aplicam mecanismos de semântica no casamento de valores para seleção de recursos disponíveis na Grade. Em (Fleischman, 2004) é definida uma ontologia para descrição de recursos em ambientes de Grades. Esta ontologia deve atuar diretamente com o serviço de diretório da Grade, onde as consultas a respeito dos recursos são feitas sobre o vocabulário definido pela ontologia. Desta forma, são eliminadas possíveis interpretações ambíguas na busca e na leitura de informações a respeito do ambiente, pois todos os conceitos usados pelas aplicações, tanto dos clientes como dos provedores, possuem apenas um significado. Neste trabalho foi desenvolvido um serviço de Grade que permite a visualização dos conceitos, representados através de ontologias, como também possibilita a busca por recursos existentes.

Em (Tangmunarunkit et al., 2003) é proposta uma abordagem flexível e extensível para a seleção de recursos, fazendo casamento baseado em ontologia. Nele são definidas ontologias distintas que descrevem recursos e requisições, levando a um fraco acoplamento entre a descrição do recurso e da requisição, livrando da necessidade de coordenação entre os provedores e consumidores de recursos. Isso permite que novos vocabulários e regras de inferências sejam adicionados. Em (Harth et al., 2004) esse trabalho foi estendido, provendo acesso dinâmico a um serviço de casamento dinâmico.

Em (Brooke e Garwood, 2003), a questão da semântica é utilizada em um teste de interoperabilidade no escopo do projeto *GRIP (Grid Interoperability Project)*. No trabalho, é investigada a possibilidade de casamento entre dois modelos diferentes de descrição de recursos: o *GLUE schema*, implementado pela *Globus Alliance*; e o arcabouço *Abstract Job Object*, utilizado no projeto *UNICORE*. A partir dessa análise, são propostos métodos que trabalham voltados a um arcabouço unificado para a descrição de recursos entre diferentes *middlewares* para Grades.

No trabalho desenvolvido em (Heine et al., 2004) é proposta uma abordagem de descoberta semântica de recursos utilizando tecnologia P2P. A rede P2P é utilizada para distribuir e consultar catálogos de recursos. Cada par pode prover descrições dos recursos, e cada par pode consultar a rede por recursos existentes. Não é definida uma ontologia central para

a descrição e casamento de recursos. Cada par tem a sua própria ontologia, possivelmente incompleta, as quais serão completadas pelo conhecimento distribuído pela rede. Isso permite efetuar o casamento dos recursos, mesmo se os conceitos usados para descrever os recursos forem desconhecidos pelo provedor, já que a rede fornece a parte que falta da ontologia.

Em (Jiang e Cybenko, 2004) é definido o conceito de validação funcional em Grades. Esse conceito oferece um nível de verificação além de descrição semântica para o casamento de recursos. De acordo com (Jiang e Cybenko, 2004), uma descrição de recursos com ontologias não pode ser definida e interpretada de maneira precisa para que possa ser utilizada por *brokers* em ambientes de computação completamente distribuído e heterogêneo, como é caso de Grades computacionais. Para isso, a validação funcional oferece uma nova abordagem para comparar os requisitos dos serviços, provendo um outro nível de verificação além das descrições semânticas para o casamento de serviços.

Após o trabalho de pesquisa efetuado, ficou claro o envolvimento da comunidade de Grades pela busca de uma forma de padronizar a descrição de recursos. Além dos já citados, outros trabalhos também abordaram a questão da utilização de semântica nessa tentativa de padronização, como em (Brooke et al., 2004), (Ludwig e van Santen, 2002), (Aktas et al., 2004) e (Moreau et al., 2003).

Observou-se que ambas as sub-áreas, nas quais foram classificados os trabalhos relacionados, encontram-se em um estágio inicial de desenvolvimento. Alguns trabalhos de Grade provendo facilidades ao usuário cliente já encontram-se concretizados, como o *GridWay* e o *CoG Kit*. Contudo, no caso do *GridWay* que se propõe a trabalhar tendo o Globus como plataforma de suporte, possui a sua versão atual ainda não adaptada as especificações OGSA e OGSI, logo não funciona com as versões mais novas e mais robustas do Globus. O *CoG Kit*, apesar de não voltada diretamente para o usuário final, possui um conjunto de APIs bastante completo e que tem se mantido atualizado com as especificações mais atuais. Até por esses motivos, essas APIs foram peças fundamentais no desenvolvimento do protótipo. Já os trabalhos voltados para solucionar problemas de semântica em Grades encontram-se mais em fase de amadurecimento e pesquisa, não existindo atualmente nenhuma ferramenta de Grade que disponha de tais mecanismos semânticos. No entanto, trabalhos como (Fleischman, 2004) e (Tangmunarunkit et al., 2003) serviram de base no desenvolvimento do modelo conceitual desta dissertação, buscando uma forma de se padronizar a descrição de recursos computacionais.

Como pode ser observado, não foram encontrado trabalhos que englobassem as duas áreas citadas acima. Neste sentido, como será visto no capítulo a seguir, este trabalho traz a proposta de prover ao usuário facilidades e abstrações no uso de Grades Computacionais, sempre com a preocupação de minimizar possíveis problemas de semântica na busca/seleção de recursos. Devido a atual linha de desenvolvimento, procurou-se definir as funcionalidades na forma de serviços, seguindo a abordagem SOA. Como citado anteriormente, a grande

maioria dos trabalhos em Grades ainda não aderiu, ou está em processo de migração para essa abordagem. Além disso, apesar dos trabalhos que vêm surgindo sobre a utilização de semântica na descrição e seleção de recursos, isto ainda não foi posto em prática, não estando disponíveis nas principais ferramentas, *middlewares* ou plataformas.

Capítulo 4

Modelo Conceitual

4.1 Introdução

Como foi dito na seção 3.9.1 sobre os trabalhos relacionados, grande parte do investimento na área de Computação em Grades está voltada para pesquisa e desenvolvimento de plataformas, *middlewares*, escalonadores e todo um conjunto de ferramentas e bibliotecas para a construção de uma infraestrutura robusta e eficiente, e para criação de aplicações que possam tirar o máximo proveito do poder computacional provido pelas Grades. Pôde-se observar que pouco trabalho vem sendo feito no intuito de permitir que um simples usuário tenha acesso a computação em Grade.

Na introdução deste documento foi explicitada a necessidade de se ter uma maior flexibilidade de obter recursos computacionais dentro de uma OV, visando dar melhores condições e alguma flexibilidade dos seus membros executarem as aplicações envolvidas num dado processo de negócio. Com isso, viu-se que a idéia de compartilhar o poder computacional entre os componentes de tais organizações pode se transformar em grandes ganhos para a organização como um todo, já que cada integrante tem interesse que todos os seus parceiros tenham sucesso a fim de que o processo de negócios seja executado no menor tempo e maior qualidade possível. Em (Camarinha-Matos e Afsarmanesh, 2004b), foram identificados como diferentes requisitos de OVs podem ser suportados por tecnologias de Grades:

- Compartilhamento de informações e conhecimentos: gerenciamento de dados em Grades, geralmente envolvendo uma grande quantidade de dados armazenados em arquivos. Para lidar com o gerenciamento de arquivos, estão sendo desenvolvidos armazenadores de arquivos de alto-desempenho, sistemas gerenciadores de réplicas e diretórios de meta-dados;
- Segurança: o *Grid Security Infrastructure* (GSI) permite que usuários utilizem recursos

compartilhados de forma segura baseado na criação de certificados de segurança e numa infraestrutura de chaves públicas e privadas.

- Interoperabilidade e integração de sistemas legados: com a definição de novos padrões, como o *Open Grid Service Architecture* (OGSA), que segue a abordagem orientada a serviço, será possível a integração e invocação de aplicações através de uma estrutura de Grades;
- Ambientes Colaborativos: suportado apenas a nível de contratos entre vários nós com a in-tenção de compartilhar recursos.

O desenvolvimento deste trabalho está focado no requisito compartilhamento de recursos.

Apesar de Grades Computacionais terem surgidos para solucionar problemas que exigem uma computação de grande desempenho, e até certo ponto dar possibilidade a criação de aplicações antes não pensadas - exatamente pela inexistência desse potencial -, este trabalho foge um pouco deste escopo, focando-se essencialmente no compartilhamento de recursos, e não na execução de aplicações que exigem um alto poder computacional. Isso se deve principalmente ao fato de aplicações em ambientes de OVs, apesar de se tornarem cada vez mais complexas, não chegam a requerer tanto poder de processamento, quanto aplicações científicas (e.g., campos da biologia ou física). Em tais ambientes (OVs), é muito mais interessante que seus participantes tenham a flexibilidade de utilizar qualquer recurso disponível, no momento em que precisar.

Portanto, é de grande valia para uma OV que os seus integrantes tenham como, de maneira dinâmica e flexível, utilizar recursos disponíveis em sua rede. De acordo com o que está representado na figura 4.1, pode se verificar que um usuário, independentemente da quantidade de OVs (pertencentes a um mesmo AVI) que ele participe, a partir do momento em que ele encontra-se configurado e autenticado nesta rede (e.g. OV1 ou OV2), ele já pode fazer uso dos seus recursos.

Todavia, apesar de todo o potencial e recursos que as plataformas e tecnologia de Grades oferecem (detalhadas no capítulo 3), existe uma série de limitações ao analisar Grades sob a ótica de que em que medida ela pode ser usada facilmente por um usuário final e suas aplicações, incluindo as características de colaboração desejáveis em Organizações Virtuais.

São elas:

1. Plataformas de Grade não são projetadas para serem utilizadas por usuários comuns:
 - (a) São muito complexas de serem instaladas e configuradas;

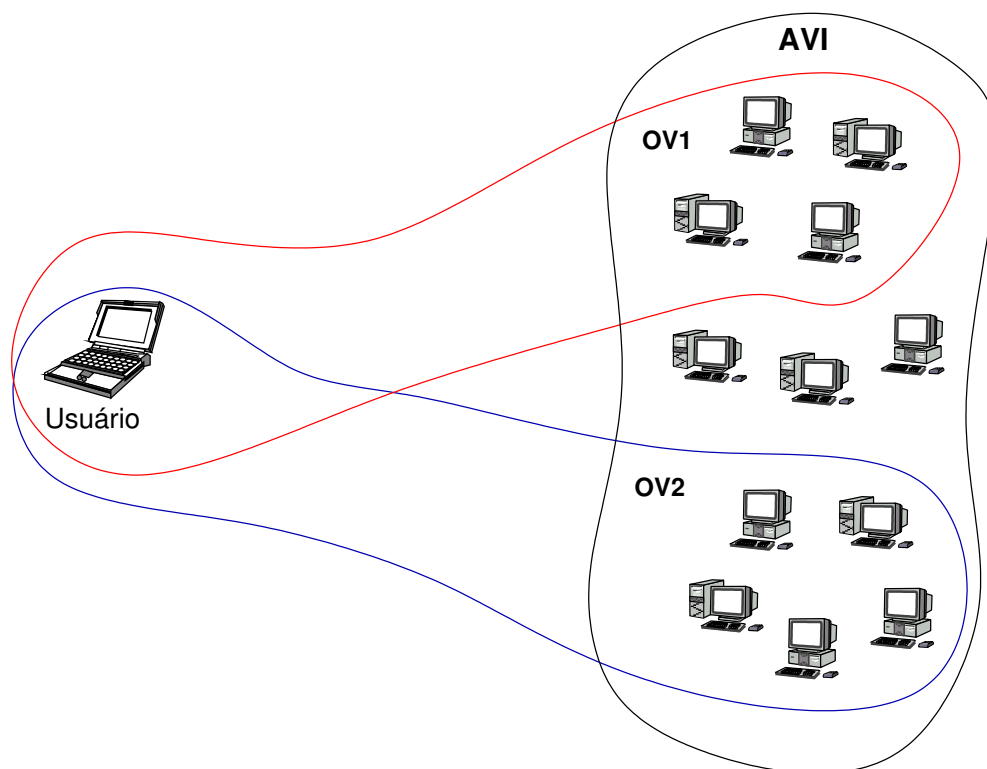


Figura 4.1: Uso de recursos computacionais de um Ambiente Virtual de Incubação (AVI)

- (b) Requerem especialistas em tecnologia de informação para usá-las;
 - (c) Não são totalmente transparentes ao usuário.
2. Atuais sistemas de *software* não são projetados para usar Grades computacionais:
- (a) Tradicionais sistemas monolíticos e até os modernos sistemas baseados em componentes, em sua maioria, não são previstos para serem executados concorrentemente;
3. Falta de flexibilidade e precisão para definir quais recursos computacionais são os mais adequados para efetuarem um dado processamento;
- (a) Atuais selecionadores de recursos na Grade não dão a flexibilidade de se priorizar outros critérios de avaliação para seleção de recursos, que não requisitos de *hardware* e *software*;
 - (b) A falta de padronização na descrição dos recursos/requisitos da aplicação acarretam em problemas de semântica, dificultando o casamento (*matching*) de recursos;

A presente proposta visa oferecer contribuições para a solução dos aspectos 1b, 1c e 3.

Mesmo com toda evolução que esta área vem sofrendo nos últimos anos - principalmente agora, que há a proposta de padronização para que a computação em Grade tenha como base uma arquitetura baseada em serviço - a sua usabilidade por usuários comuns ainda é

demasiada complexa. Esta complexidade vai desde a instalação, configuração e manutenção da plataforma/*middleware*, até a sua utilização de fato, que inclui diversos passos, como: emissão e autenticação de certificados de segurança, busca e seleção de recursos, definição de requisitos de aplicações e submissão de tarefas.

Mais concretamente, o presente trabalho propõe uma arquitetura de Cliente de Grades enxuta e transparente para permitir a um usuário final fazer uso dos mais adequados recursos computacionais ociosos existentes dentro de uma OV. Sua especificação segue os padrões de *Open Grid Service Architecture* (OGSA) e *Open Grid Service Infrastructure* (OGSI), de forma a poder ser utilizada em plataformas de Grades que também sigam esses padrões. Em termos gerais, a sua utilização não se restringe aos recursos localizados/pertencentes aos membros de uma OV, podendo encontrar e fazer uso de outros recursos quaisquer que estejam devida e previamente mapeados na Grade. No entanto, deseja-se tomar partido de algumas particularidades relevantes em OVs, nomeadamente o fato das empresas-membro estarem, à partida, interessadas em colaborar entre si e o fator confiança existente.

4.2 Proposta

Como dito anteriormente, esta dissertação tem a proposta de prover ao usuário final - mais precisamente a participantes de uma Organização Virtual (OV) - uma arquitetura e ferramenta de *software*-cliente que possibilite que este tenha acesso aos mais diversos recursos ociosos e disponíveis em uma rede (local ou até mesmo a Internet), de forma simples e transparente. A idéia da arquitetura/ferramenta é detectar quando uma aplicação precisa de recurso computacional que não dispõe localmente, e automaticamente localizar os mais adequados recurso(s) que supra(m) as suas necessidades, levando-se em conta os níveis de confiança existentes entre os membros de uma dada OV. A avaliação e a localização do recurso são feitas a partir dos requisitos da aplicação.

Para prover tal arquitetura, buscou-se identificar quais funcionalidades são essenciais para que um usuário possa fazer acesso a Grades de forma simples e transparente. Baseado nos estudos feitos sobre a estrutura de ambientes de Grades Computacionais, concluiu-se que uma arquitetura cliente com tal propósito, tem como requisitos básicos os seguintes módulos:

- Um módulo para submissão de tarefas;
- Um módulo para transferência de dados;
- Um módulo para autenticação de segurança;
- Um módulo para publicação do recurso na Grade; e
- Um módulo para busca/seleção de recursos na Grade.

Seguindo a identificação desses módulos, partiu-se para definição de outros módulos que venham agregar funcionalidades e facilidades à arquitetura. Na seção seguinte, a arquitetura será apresentada com mais detalhes, especificando as peculiaridades e responsabilidades de cada módulo.

4.3 Arquitetura

Conforme descrito anteriormente, a arquitetura é definida por um conjunto de módulos, onde cada módulo possui sua função bem especificada. Na figura 4.2 estão representados todos os módulos e a forma como são organizados. Além da divisão em módulos, a arquitetura apresenta uma divisão de maior granularidade, que a separa em duas partes: uma que engloba os módulos que não tem acesso direto aos serviços da plataforma Grade e até certo ponto são *independentes* da plataforma utilizada; e outra onde se encontram os módulos que fazem interface com os serviços providos pela plataforma e com outros serviços desenvolvidos, vistos como *dependentes* da plataforma. A camada dependente de plataforma - que engloba os módulos Cliente Buscador de Recursos, Transferência de Dados, Publicador de Informações, Autenticador da Grade e Submissão de Tarefas -, no caso deste trabalho, será representada pela plataforma *Globus*, apresentada na seção 3.7.

De forma superficial (nas seções seguintes, serão dados todos os detalhes sobre os módulos da arquitetura) , o fluxo de controle da arquitetura se dá da seguinte, como representado na figura 4.3:

1. Como primeiro passo o usuário deve configurar o seu ambiente, definindo os serviços a serem utilizados e requisitos de segurança, funcionalidades estas fornecidas pelo módulo *Interface Cliente de Acesso à Grade*;
2. Toda aplicação do usuário que é candidata a fazer uso Grade deve ser previamente registrada, informando-se os seus seguintes requisitos de *hardware* e *software*: sistema operacional, arquitetura do sistema, processador, memória principal, disco rígido disponível, o nome do executável (aplicação) e seus possíveis parâmetros de execução. A manipulação destas informações fica sob responsabilidade do módulo *Requisitos das Aplicações*, o qual tem acesso via o módulo *Interface Cliente de Acesso à Grade*. Em teoria, essa etapa é realizada apenas uma vez, num momento inicial, quando a figura de um administrador (ou de alguém que possua conhecimentos sobre os requisitos da aplicação) efetua esse cadastro;
3. Quando o usuário executa uma das aplicações cadastradas, automaticamente o módulo *Avaliador de Uso da Grade* inicia uma análise que verifica a necessidade de se fazer

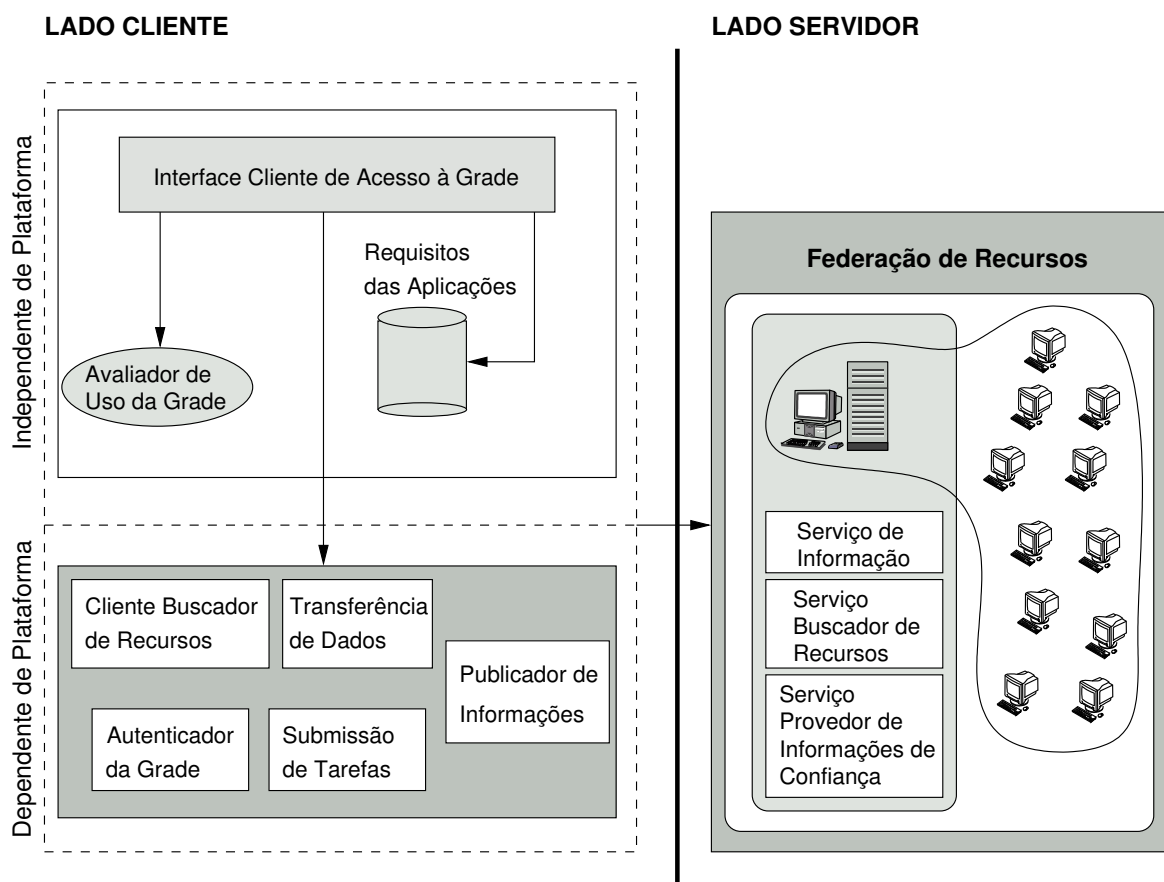


Figura 4.2: Arquitetura proposta

acesso à Grade. Essa análise é feita através da comparação dos requisitos da aplicação e o estado corrente do computador do usuário.

4. Tendo-se o resultado da avaliação o usuário pode então decidir submeter ou não a aplicação a um recurso disponível na Grade;
5. Caso seja decidido por executar localmente, a execução se dá de modo normal;
6. Decidindo-se por usar a Grade, é feito um acesso ao *Serviço de Informação* onde se encontram registrados todos os nós pertencentes a Grade (que formam a *Federação de Recursos*) em busca do recurso que atenda aos requisitos da aplicação. A busca é ativada pelo *Cliente Buscador de Recursos* que invoca o *Serviço Buscador de Recursos*. Além das questões de desempenho, relacionadas a avaliação de *hardware*, dois outros quesitos são considerados: a confiança e o custo de se utilizar o recurso. Para isso, é necessário que usuário defina qual a prioridade para seleção do recurso: desempenho, confiança ou custo. De acordo com essa definição, a seleção é feita e então é retornado o endereço do recurso que melhor se adequa à sua aplicação;
7. Após a seleção do recurso, o usuário informa o(s) arquivo(s) com os dados de entrada, a ser(em) utilizado(s) pela aplicação, para que este(s) seja(m) enviado(s) para o nó

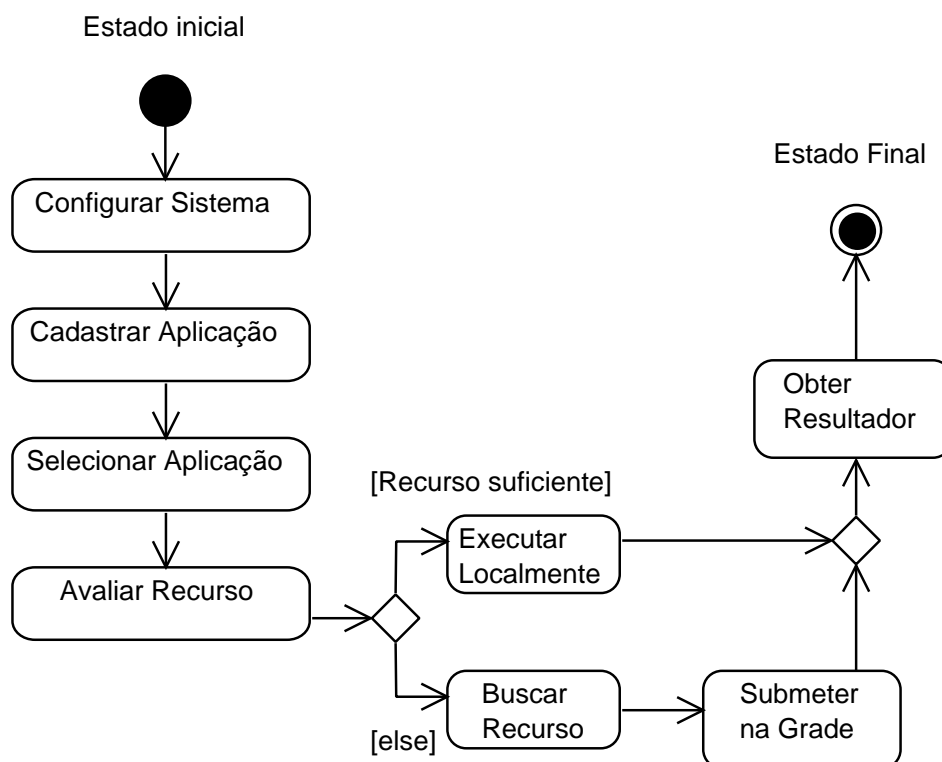


Figura 4.3: Fluxo de execução

remoto pelo módulo *Transferência de Arquivos*;

8. Como passo final, no momento em que é feita a submissão da aplicação, o usuário deve autenticar-se com um *nome de usuário* e *senha*, através do módulo *Autenticador na Grade*. Sendo feito isso, através do módulo *Submissão de Tarefas*, a submissão é efetivada de fato, enviando-se a aplicação e os dados para o(s) nó(s) remoto(s) selecionado(s);
9. Obtendo sucesso na submissão, o resultado da execução é então retornada para o usuário. Caso contrário, mensagens de erro são retornadas ao usuário.

Estes passos serão concretamente vistos no capítulo 5, onde a arquitetura é instanciada e avaliada na forma de um protótipo.

Para que um recurso seja localizado ele precisa estar registrado. Isto significa que caso o proprietário do recurso queira torná-lo disponível como um recurso acessível na Grade, ele deve publicar suas informações em um servidor central, já pré-definido com um *Serviço de Informação* (ver seção 4.3.4).

As seções seguintes serão dedicadas a descrever cada um dos módulos, componentes e serviços presentes na arquitetura.

4.3.1 Federação de Recursos

A chamada **Federação de Recursos** é a rede lógica e temporariamente formada que é composta por todos os recursos computacionais que os participantes da OV desejam compartilhar e assim ter igualmente acesso a todos eles, interconectados. Esta federação é considerada como a OV em Grade.

De forma geral, esse compartilhamento e acesso implicam dizer que o recurso deve ser, de algum modo, publicado para que possa ser encontrado, e cada usuário participante deve ser autenticado com um certificado de segurança para que os outros tenham a garantia de que seus recursos não serão utilizados por algum outro usuário não autorizado.

4.3.2 Interface Cliente de Acesso à Grade

O módulo de **Interface Cliente** é responsável por toda abstração no que se refere ao uso dos serviços fornecidos pela Grade pelo usuário/aplicação-cliente, além de toda a etapa de configuração necessária para utilização da Grade. Através dele é que é feito o acesso aos demais módulos da arquitetura.

Provê as seguintes funcionalidades:

- Requisição certificados de segurança, e de autenticação dos mesmo;
- Definição dos requisitos da aplicação;
- Publicação as informações do recurso, além de informações pertinentes à OV;
- Seleção da aplicação, que será sujeita à avaliação para uso ou não da Grade;
- Definição da prioridade para a busca e seleção do recurso;
- Acesso simplificado aos serviços da Grade;
- Avaliação automática da necessidade de se utilizar a Grade;

Alguns das funcionalidades estão voltadas a uma parte que pode ser considerada mais de adminis-tração e configuração. Apesar dessas funcionalidades serem apresentadas da forma mais simples possível, podem haver casos em que seja necessário um usuário com conhecimentos mais específicos, como por exemplo na definição dos requisitos da aplicação, onde o usuário deve especificar detalhes como a quantidade de memória e de disco rígido necessário para executar a aplicação (ver seção 4.3.3).

Este módulo provê ao usuário uma camada que abstrai a complexidade inerente ao acesso a recursos distribuídos, utilizando tecnologias de Grades Computacionais.

4.3.3 Requisitos das Aplicações

Toda aplicação que for candidata a fazer uso da Grade deve ter os seus requisitos especificados. No modelo proposto esses requisitos são utilizados para três fins:

- Para submissão da aplicação;
- Para avaliação do recurso local, comparando-se o que o recurso oferece e o que a aplicação necessita;
- Para busca de recursos que se adequem às necessidades da aplicação.

A partir da definição dos requisitos, duas estruturas de dados são geradas, como arquivos XML, cada um com sua função específica.

RSL

Conforme foi descrito na seção 3.7.4, a *Globus Alliance* propõe um formato padrão para especificação de recursos, o *Resource Specification Language* (RSL), definido na linguagem XML. Ele é voltado essencialmente para a submissão remota de aplicações e engloba as seguintes informações (ver mais detalhes na seção 3.7.4), necessárias para tal: caminho para o executável, argumentos, variáveis de ambiente, quantidade de execução, arquivos de entrada e saída de dados, quantidade mínima e máxima de memória, tempo máximo de utilização da CPU. Logo, considera-se que gerenciadores de recursos que sigam esse padrão, interpretem esse modelo de especificação.

Portanto, esse arquivo será utilizado exclusivamente pelo módulo **Submissão de Tarefas** sempre que se detectar a necessidade de se usar os recursos da Grade.

RSL Cliente

O **RSL Cliente** foi definido com a proposta de ser uma extensão cliente ao RSL. Além de algumas informações (quantidade mínima de memória e de disco rígido) presentes no RSL, citadas acima, o RSL Cliente reúne informações utilizadas para avaliação local do recurso, e para busca de recursos remotos. A criação/definição desta especificação surgiu da necessidade de se estruturar, assim como no RSL, as informações relativas à aplicação, só que, como dito anteriormente, com propósitos de consulta e avaliação de recursos. Esta especificação está relacionada com o tipo de recurso que aplicação necessita, para que seja executada de forma correta, selecionando os recursos mais adequados. Na tabela 4.1 encontram-se esses parâmetros adicionais, e uma breve descrição de cada um deles:

Seguindo o levantamento bibliográfico feito, relacionado à utilização de semântica em Grades (seção 3.9.2), o presente trabalho aposta na sua aplicação para descrição de recursos

Parâmetro	Descrição
Sistema Operacional	Tipo de versão de sistema operacional necessário para executar a aplicação
Arquitetura	Tipo de arquitetura computacional necessária para executar a aplicação
Processador	Modelo de processador necessário para executar a aplicação
Frequência do Processador (<i>clock</i>)	Frequência mínima do processador necessária para executar a aplicação
Memória	Quantidade mínima de memória necessária para executar a aplicação
Sistema de Arquivos	Sistema de arquivos necessário para executar a aplicação
Disco Rígido	Quantidade mínima de espaço em disco rígido necessária para executar a aplicação
Largura de Banda	Largura de banda mínima necessária para transferir a aplicação e os respectivos dados de entrada

Tabela 4.1: Parâmetros de um RSL Cliente

da Grade. Desta forma, o usuário beneficia-se no momento de definir os requisitos da sua aplicação, como também, para propósitos da seleção de recurso, a função do *serviço buscador de recursos*, torna-se mais eficiente.

De forma a facilitar e padronizar a especificação dos requisitos, este módulo define uma ontologia simples, na forma de um vocabulário com termos comum ao domínio (recursos computacionais). Baseado em (Fleischman, 2004), foi definida uma árvore com a classificação de um recurso computacional, representada na figura 4.4. A definição do vocabulário surgiu como idéia de prover ao usuário uma gama de informações relacionadas com especificações de recursos computacionais, livrando este de se preocupar em fornecer corretamente os detalhes sobre o seu recurso/requisitos da aplicação. Além disso, a partir do momento em que todos os integrantes da OV passam utilizar a mesma especificação, tem-se as informações numa forma e padronizada.

Tendo-se este vocabulário de termos pré-definido, o usuário fica restrito a utilização de valores padronizados, facilitando a busca de recursos que sirvam aos seus requisitos.

4.3.4 Serviço de Informação

De acordo com o que foi explicado na seção 3.7.5, o **Serviço de Informação** funciona com estrutura agregadora de informações, implementada na forma de um serviço de indexação.

Esse serviço foi escolhido como um meio para a publicação de informações por possuir suporte à agregação dos dados (na forma de *Service Data Elements* (SDEs)), permitindo

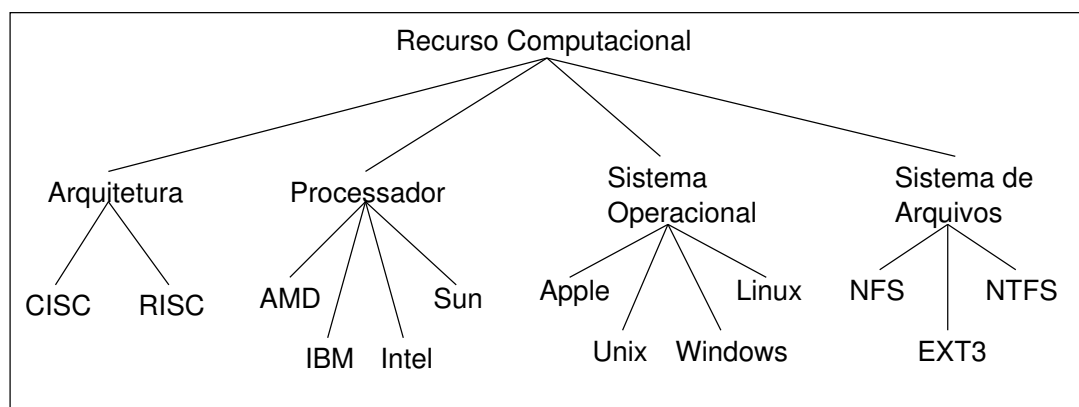


Figura 4.4: Árvore de classificação de recursos (adaptada de (Fleischman, 2004))

que as informações sobre os recursos disponíveis na Grade possam ser acessados de forma centralizada. Dessa forma, a cada requisição de um usuário em busca de recursos, será feita uma consulta nesses dados.

Devido à dinamicidade inerente a um ambiente de Grades, é essencial dispor de um serviço como esse, que possibilite a manipulação de dados estáticos (e.g. modelo do processador e arquitetura do recurso) e dinâmicos (e.g. estado atual de um recurso). A medida que um recurso entra ou sai da Grade, as informações a respeito dos recursos disponíveis devem ser automaticamente atualizadas.

Dois módulos, em especial, farão uso/acesso direto a esse serviço: são os módulo de publicação de serviço, e o serviço buscador de recursos, que serão detalhados nas próximas seções.

4.3.5 Provedor de Informações de Confiança

De acordo com o que foi dito na seção 2.1 sobre Ambiente Virtual de Incubação (AVI), a confiança em OVs é uma estrutura gerida pela administração do próprio AVI, a qual é a única com acessos privilegiados (i.e. escrita e leitura). Os demais integrantes desse AVI devem ter acesso a essas informações, mas somente como forma de consulta (i.e. privilégio de leitura). Neste trabalho é tomada como forma de apresentação destes indicadores que os valores que representam a confiança dos integrantes são fixos.

Como descrito na seção 2.2.2 sobre confiança em AVIs, estas informações são consideradas como valores fixos, gerados a partir do histórico do integrante, de participações em OVs anteriores. Contudo, apesar de frequentemente fixos, estes valores podem ser periodicamente atualizados pelo administrador do AVI.

Na arquitetura proposta neste trabalho, o nível de confiança de um integrante da OV é utilizado como critério de avaliação na busca e seleção de recursos (ver seção 4.3.7), e

em questões de segurança, restringindo o acesso ao recurso a usuários considerados “não confiáveis” 4.3.10. Com isso surgiu a necessidade de prover um modo de fácil acesso a essas informações. Portanto, a proposta deste módulo é de definir um serviço que permita que usuários, aplicações, e outros serviços, tenham acesso a essas informações.

As informações utilizadas para o propósito citado acima são:

- Identificador do usuário e o domínio a qual pertence (e.g. *gsigma.ufsc.br*);
- Identificador da empresa ou organização da qual o usuário faz parte;
- Conceito dado a cada integrante do AVI, que indica a confiança que ele oferece perante os demais integrantes.

Esse serviço é utilizado em duas situações em específico. Primeiro, no módulo de Segurança, para que o usuário possa definir quais parceiros poderão fazer uso do seu recurso. E em segundo, pelo Serviço Buscador de Recursos, que utilizará essas informações para selecionar apenas os recursos que o usuário tiver permissão de acesso, de acordo com a confiança exigida pelos donos dos recursos.

Maiores detalhes de como as informações de confiança são utilizadas pelos módulos citados acima, serão dadas nas seções dos respectivos módulos.

4.3.6 Publicador de Informações

Este módulo é utilizado pelos nós da rede (voluntários) que se tornarão disponíveis na Grade como um recurso computacional em potencial. Tendo essa iniciativa, o usuário deve registrar o seu recurso no **Serviço de Informação**, para que esse possa ser localizado pelo serviço buscador.

A publicação é dividida em dois grupos de informações:

- O primeiro grupo diz respeito às informações de *hardware* e *software*. No caso do *Globus*, são fornecidos serviços onde obtém-se este tipo de informação, como: sistema operacional, versão do *kernel* e do SO, arquitetura do sistema, memória total, disponível e virtual, tamanho total e disponível de disco rígido, sistema de arquivos, carga do processador, modelo do processador, etc.
- O segundo grupo de informações está relacionado com a OV. São informações sobre o custo de acesso ao recurso, o nível de confiança do próprio usuário (C_p), e a confiança mínima (C_m) requerida a um participante para que esse acesse o recurso. Essas informações são armazenadas nos dados de serviço do serviço Informações da OV. O valor do

custo e da confiança mínima exigida são definidos pelo usuário. Já o seu nível de confiança C_p é uma informação que está diretamente ligada aos aspectos da OV, gerada ao longo da vida dessa OV, e é obtida através do serviço Provedor de Informações de Confiança. A necessidade e a forma como estas informações são utilizadas, são explicitadas na seção 4.3.7.

Ambos os grupos de informações são publicados como estruturas de dados de serviços (*service data*) através do serviço Publicador de Informações, diretamente para o Serviço de Informação central, como apresentado na figura 4.5. Esta forma de estrutura - dados de serviços - foi escolhida por ser a sugerida na especificação OGIS. À medida que essas informações vão se alterando, o Serviço de Informação é atualizado, ficando a critério de desenvolvimento/configuração se estas atualizações serão automáticas e/ou manuais.

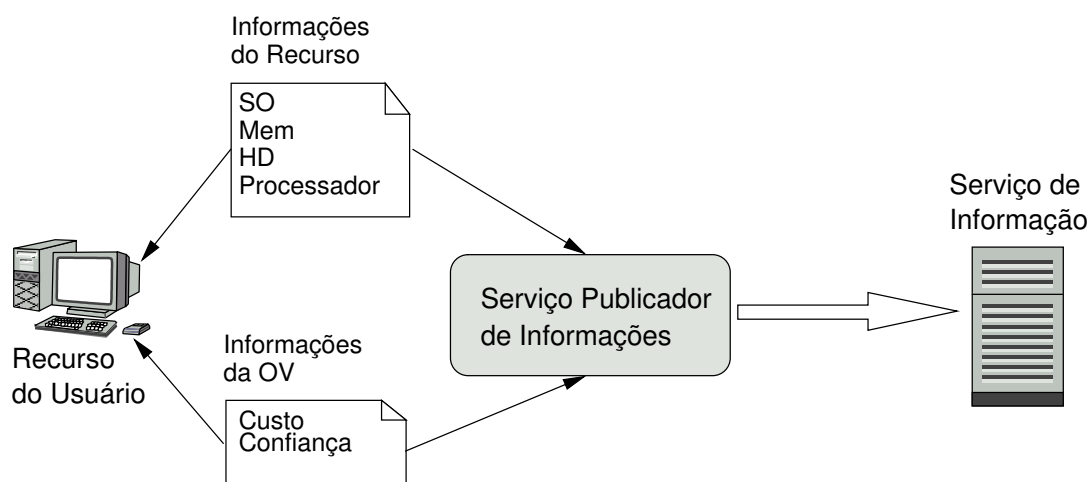


Figura 4.5: Publicação das informações do usuário, no Sistema de Informação

4.3.7 Serviço Buscador de Recursos

A partir dos conceitos dados na seção 3.8.2, definiu-se este módulo, que tem como função a busca e seleção de recursos disponíveis na Grade. O cliente faz um acesso ao serviço buscador informando-lhe os seus requisitos e tem como retorno a localização do(s) recurso(s) que se adequam às suas necessidades.

A abordagem tradicional para busca e seleção de recursos da maioria dos mediadores existentes está focado apenas na comparação de valores relacionados às características de *hardware* e *software* dos recursos, ou seja, **desempenho** é o critério chave de avaliação. O propósito desse módulo é de avaliar, além de desempenho, outros dois critérios: **custo** e **confiança**.

Tomou-se a decisão de adicionar estes critérios na busca/seleção de recursos por estes serem questões relevantes dentro de um ambiente de OVs. O custo é utilizado quando a

organização resolve cobrar para fornecer os seus recursos computacionais a outros. Sendo assim, fica a cargo da organização quanto e como será cobrado pela utilização dos seus recursos, ficando isso mais como questões de implementação. Já o critério de confiança é utilizado em duas perspectivas. A primeira perspectiva é em relação ao usuário que vai em busca de recursos confiáveis para executar a sua aplicação. Logo, o usuário deve informar ao buscador o quão confiável deve ser o integrante que lhe proverá recurso. A outra perspectiva é do provedor do recurso. Este pode querer restringir o acesso ao seu recurso aos parceiros que ele não considere confiável. Para isto ele define qual mínimo de confiança exigida para que outros acessem o seu recurso. Desta forma, ao receber uma requisição, o serviço buscará apenas aos recursos que o usuário tiver permissão de acesso.

Conforme será descrito no módulo a seguir, o usuário deve definir qual a sua prioridade no momento da busca pelo recurso. Com isso, o serviço buscador seleciona o recurso que melhor sirva aos requisitos definidos pelo usuário, dando prioridade a um dos três critérios citados: desempenho, custo ou confiança. Logo:

- Se a prioridade é **desempenho**, o buscador seleciona o recurso que apresenta as melhores características de *hardware*;
- Se a prioridade é **custo**, o buscador seleciona o recurso de menor custo;
- Se a prioridade é **confiança**, o buscador seleciona o recurso que apresente o índice de confiança (C_p) maior ou igual ao requisito de confiança (C_r) da aplicação;

Além de utilizar na busca os critérios citados acima, o buscador faz uso das informações do serviço Provedor de Informações de Confiança para selecionar apenas os recursos que o usuário tem permissão de acesso. Tendo o índice de confiança do usuário (C_p), o serviço busca por recursos que exijam o requisito mínimo de confiança (C_m) no máximo igual a C_p . Ou seja, os recursos selecionados devem atender à seguinte restrição: $C_m \leq C_p$. Com isso, o usuário tem a garantia que o recurso selecionado permite o seu uso.

A grande diversidade de recursos em uma Grade com as mais variadas descrições torna bastante complexa a tarefa de busca e seleção desses recursos. Para tentar amenizar esse problema, este serviço faz uso da padronização definida neste trabalho (ver seção 4.3.3) para especificação dos requisitos da aplicação. O fato de o usuário especificar os requisitos baseando-se em um vocabulário pré-definido, de certa forma obriga este usuário a definir as necessidades das suas aplicações de uma forma padronizada. Isso permite que o buscador trabalhe dentro de um vocabulário comum ao domínio, proporcionando uma busca e seleção de recursos mais eficiente.

Seguindo a linha de padronização que tem ocorrido com Grades, o buscador é definido na arquitetura como um serviço. Com isso, tendo-se o *Web Services Description Language*

(WSDL) que o descreve, o cliente pode facilmente acessá-lo. Este serviço age diretamente com o Serviço de Informação, onde estão armazenados todos os detalhes à respeito dos recursos. A cada requisição feita pelo Cliente Buscador de Recursos, imediatamente o buscador realiza uma busca no Serviço de Informação. Tendo o retorno, a busca é então refinada de acordo com os requisitos e prioridades, como ilustrado na figura 4.6 abaixo.

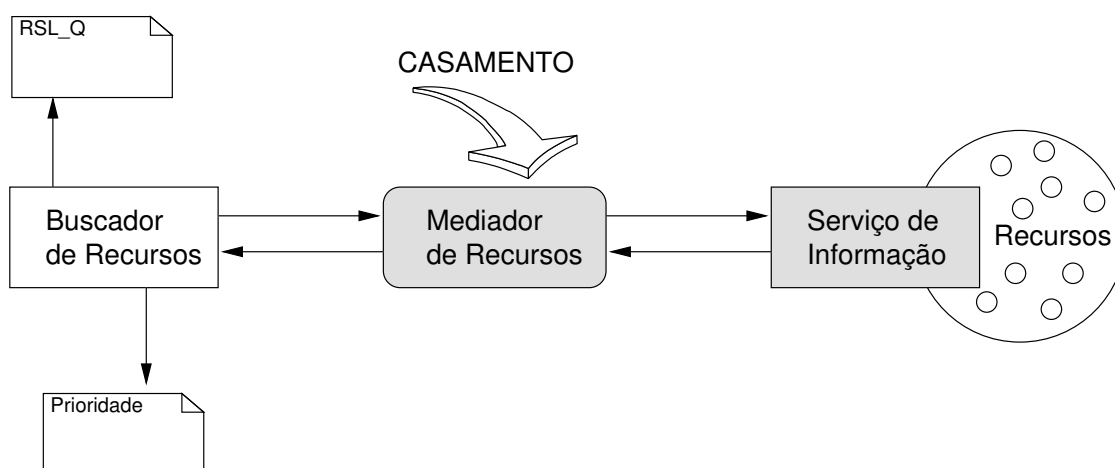


Figura 4.6: Processo de busca e seleção de recursos

4.3.8 Cliente Buscador de Recursos

Antes que a aplicação possa ser submetida para uma execução remota é necessário que seja definido para qual recurso a aplicação - e seus respectivos dados de entrada - será enviada. O módulo Cliente Buscador de Recursos tem a função de fazer essa busca, atuando como a interface entre o usuário e o serviço provido pelo serviço buscador de recursos.

Os parâmetros utilizados na busca, assim como no avaliador, são obtidos no estrutura RSL Cliente. Além dos valores relacionados a *hardware* e *software*, utilizados na busca, o usuário define qual a sua prioridade para a seleção do recurso. A prioridade é definida em cima dos critérios utilizados pelo Serviço Buscador de Recursos em sua avaliação. Os critérios avaliados são os seguintes:

- **Custo:** este critério deve ser escolhido quando usuário decidir que o mais importante na seleção do recurso é o quanto o uso deste recurso vai lhe custar;
- **Confiança:** este critério deve ser escolhido quando usuário decidir que o mais importante na seleção do recurso é a confiança que cada parceiro oferece dentro da OV, dando uma maior garantia (em relação a um parceiro ser mais confiável que outro) que uma determinada tarefa será completada. Para isso, o usuário define o requisito de confiança (C_r) que ele requer que o recurso a ser selecionado tenha;

- Desempenho: este critério deve ser escolhido quando usuário decidir que o mais importante na seleção do recurso é o desempenho destes recursos. Neste caso, apenas serão considerados os requisitos de *hardware* e *software* da aplicação.

Questões de como o critério de “melhor desempenho” vai ser avaliado, fica sob a decisão do programador/analista. Por exemplo, pode-se decidir por priorizar o poder de processamento ou a quantidade de memória.

4.3.9 Avaliador de Uso da Grade

Esse módulo tem a função de avaliar se há a necessidade ou não de fazer uso da Grade. No momento em que o usuário executa uma das aplicações registradas o avaliador é ativado. A avaliação é feita através da comparação dos requisitos da aplicação e o estado atual do computador local, como é representado na figura 4.7. Os valores dos requisitos da aplicação são obtidos através do arquivo RSL Cliente, definido previamente.

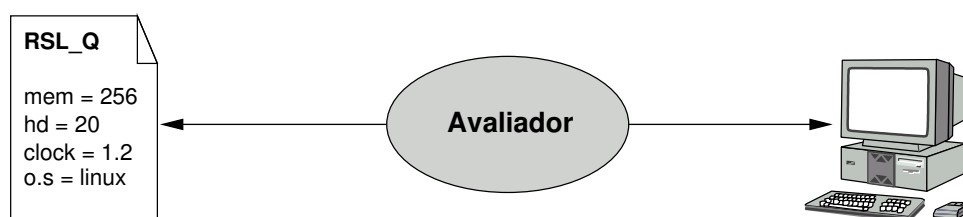


Figura 4.7: Avaliador de recurso

Os três parâmetros considerados na avaliação, são:

- Carga do processador: é avaliada a porcentagem de uso do processador. É definida uma porcentagem de uso limite ul onde, caso a porcentagem de uso corrente uc for maior ou igual - $uc \geq ul$ -, o processador é considerado indisponível;
- Memória: verifica se a quantidade de memória livre é maior ou igual a requisitada pela aplicação;
- Disco rígido: verifica se a quantidade de espaço no disco rígido é maior ou igual a requisitada pela aplicação;

Além dos citados acima, outros parâmetros também são considerados: Sistema Operacional, Arquitetura do Sistema e Frequência do Processador (*clock*). Isso se deve ao fato de o usuário decidir registrar aplicações que não são suportadas pelo seu recurso computacional. Por exemplo, o usuário pode registrar uma aplicação que seja própria para ambientes Unix, mas o seu sistema operacional pode ser um Windows. Assim, o usuário tem a flexibilidade de executar aplicações que o seu recurso nunca suportaria, seja por restrições de *hardware* (processador, arquitetura), ou de *software* (sistema operacional).

4.3.10 Segurança

Este módulo é responsável por atender às questões de segurança, tanto para obter acesso aos recursos compartilhados, como também para definir as permissões de acesso ao seu próprio recurso.

Uma função desse módulo é de possibilitar que o usuário autentique um certificado, feito através do fornecimento de um *nome de usuário* e de uma *senha*, que são definidos quando o certificado é emitido. A autenticação gera um *proxy* localmente, que tem a validade pré-definida e configurável. A proposta da *Globus Alliance* ao criar o conceito de *proxy* é de prover um modo de o usuário se autenticar na Grade uma única vez (*single sign-on*) durante um determinado período de tempo.

Algumas medidas prévias devem ser tomadas antes que esse módulo possa ser utilizado de fato. O certificado, citado acima, deve ser gerado e assinado. A geração desse certificado é feita nesse mesmo módulo pelo usuário. No entanto, a assinatura deste é feita por um *Certificate Authority (CA)*, que age como um administrador central de segurança, responsável por assinar os certificados emitidos pelos usuários. A assinatura é feita e então o certificado é devolvido ao usuário.

A outra função permite que o usuário defina quais participantes da OV terão o direito de efetuar submissões remotas em seu recurso. De início, o usuário especifica qual nível de confiança mínima C_m que um participante deve ter para que ele aceite o compartilhamento. A partir disso é feito um mapeamento entre as informações de confiança (disponíveis no serviço) de todos os participantes C_p e a confiança mínima C_m . Por fim é gerada uma estrutura com todos os participantes que atendem à restrição $C_p \geq C_m$. Sendo assim, o usuário tem a flexibilidade de escolher quem ele quer que faça uso da sua máquina, a partir da confiança que esse integrante apresente.

4.3.11 Submissão de Tarefas

O módulo **Submissão de Tarefas** funciona como uma interface para o Gerenciador de Recursos. Tendo-se definido o(s) recurso(s), que executará(ão) a aplicação, o módulo é ativado fazendo uma requisição ao serviço que será responsável pelo gerenciamento do recurso remoto. Este serviço, por sua vez, faz o interfaceamento com o escalonador que fará de fato, o controle da execução da aplicação.

De acordo com o recurso selecionado (i.e., um simples computador pessoal ou um *cluster*, por exemplo), os tipos de escalonadores utilizados variam. A proposta dessa interface é de abstrair qual tipo de escalonador será encontrado pela aplicação, permitindo à arquitetura interagir com os escalonadores mais utilizados em ambientes de Grades.

O módulo *Grid Resource Allocation and Management* (GRAM) do *Globus* possui um serviço controlador de execução (*Master Fork Managed Job Factory*) que na sua forma original, apenas efetua execuções em ambientes remotos. Mas a partir da instalação de alguns pacotes de interfaces que são disponibilizados no *Globus Toolkit* (GT), é possível utilizar alguns escalonadores, como: PBS, Condor e LSF. Além desses, o GT dá a flexibilidade de se definir novas interfaces. Dessa forma, o módulo Submissão de Tarefas não necessitará fazer acesso direto a nenhum controlador de execução, e sim a um serviço que se preocupará em empilhar a aplicação no respectivo controlador. Assim, a única função do módulo é iniciar a submissão, e aguardar uma resposta vinda do serviço gerenciador.

De acordo com o que foi dito nos módulos anteriores, vários passos têm que ser ultrapassados até que a submissão da aplicação possa ser efetivamente feita. Na figura 4.8 está representado este ciclo, que envolve a publicação dos recursos, e posterior busca por esses, e por fim, a submissão da aplicação por parte do usuário.

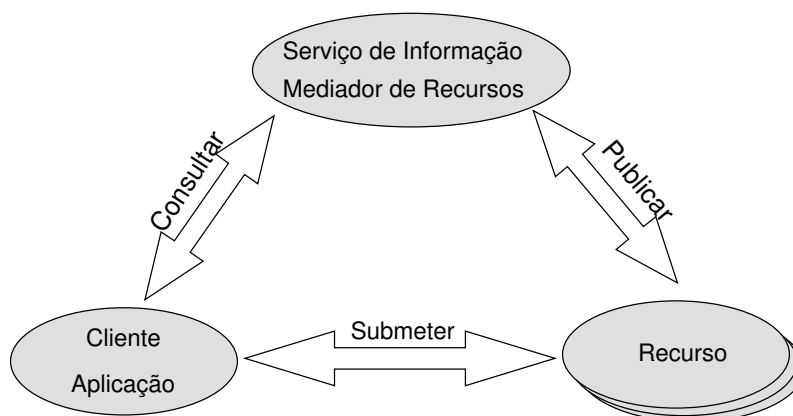


Figura 4.8: Ciclo da submissão

4.3.12 Transferência de Dados

Antes que a execução remota possa ser efetivada, é necessário que tanto a aplicação quanto os dados de entrada sejam transferidos para o(s) recurso(s) onde a aplicação será executada.

Para tal tarefa, se faz necessário um serviço de transferência de arquivos. No caso do *Globus*, tal funcionalidade é dada pelo serviço *GridFTP* (seção 3.7.3), que é uma extensão do protocolo padrão de transferência de arquivos na Internet, o FTP.

Este módulo provê um interface de acesso ao serviço *GridFTP*, que é acessado no momento da submissão. Assim como o serviço de execução remota, o *GridFTP* exige a autenticação do certificado por parte do usuário. Com isso, tem-se a garantia de que apenas os integrantes da federação terão acesso ao serviço.

4.4 Conclusão

Durante o elaboração e desenvolvimento do modelo proposto houve uma preocupação fixa em torná-lo o mais abrangente e flexível possível, considerando-se que seja aplicável a mais diversas plataformas de Grade. Foi visto que para isto, o modelo foi modularizado, como também dividido em camadas, estas buscando uma separação no tocante a tornar a maior quantidade possível de módulos independentes da plataforma de Grade a ser utilizada. Ao mesmo tempo em que os módulos que são dependentes da plataforma, encontram-se bem definidos de acordo com a função a desempenhar na arquitetura.

Contudo, de acordo com o que foi apresentado no início do capítulo, o objetivo central da arquitetura é de levar em consideração questões de OV. Assim, algumas peculiaridades foram acrescentadas, como os módulos Publicador de Informações e o Serviço Provedor de Informações de Confiança; além do módulo Serviço Buscador de Recursos, que tem a função de um mediador de recursos, mas que sofreu algumas adequações para que considerasse fatores como os de confiança, durante a busca de recursos na Grade.

Capítulo 5

Protótipo

Para fins de avaliação do modelo conceitual proposto, foi implementado um protótipo apresentando as funcionalidades e arquitetura descritas no capítulo 4, fazendo-se uso das tecnologias padrões atuais utilizadas no desenvolvimento de aplicações para Grades computacionais.

O teste do protótipo foi realizado tomando como base um sistema desenvolvido no contexto de uma dissertação de mestrado (Bittencourt, 2005), que visava ajudar gestores de Ambiente Virtual de Incubação (AVI) na seleção das empresas mais apropriadas para constituição de uma Empresa Virtual (EV).

Por fim, são apresentados os resultados, dificuldades e problemas encontrados durante todo o processo de implementação do protótipo. Todas as etapas relacionadas a este protótipo foram completamente realizadas pelo mestrando.

5.1 Implementação

Nesta seção serão apresentados todos os detalhes de implementação: ferramentas, linguagens, APIs e ambiente de desenvolvimento utilizados, além da modelagem e metodologias de desenvolvimento.

5.1.1 Ferramentas

O desenvolvimento do protótipo foi todo baseado no *Globus Toolkit 3.2* (Globus, 2005). Conforme apresentado na seção 3.7 sobre a plataforma Globus, esta vem sendo atualmente considerada como “padrão de facto” para desenvolvimento e suporte a aplicações para computação em Grades. Isso se deve principalmente ao fato dela possuir uma grande quantidade

de serviços e APIs, e pelo seu alto investimento em padronização em conjunto com grandes empresas (e.g., HP, IBM, etc.). Além das APIs fornecidas pela própria ferramenta, um projeto paralelo desenvolvido pela *Globus Alliance*, o *Commodity Grid Kit* (CoG), fornece uma série de APIs nas linguagens de programação Java e Python, que facilitam a criação de aplicações que façam uso de serviços avançados de Grades.

Essas foram consideradas as principais razões pelas quais se optou pelo Globus. Evidentemente que os requisitos do problema/domínio de aplicação de OVs foram levados em consideração; porém, sob esta ótica, outras plataformas (descritas no capítulo 3) poderiam ter sido utilizadas.

Tanto as APIs do Globus como do CoG foram utilizadas na implementação do protótipo, nas suas versões para a linguagem Java. Para isso foi utilizado o *Java Standard Edition* (J2SE), na versão 1.4.2¹. A disponibilidade de uma API mais completa, e a já conhecida portabilidade da linguagem, foram os principais fatores motivadores para a escolha de Java como linguagem de implementação.

É importante ressaltar que atualmente já existe a versão 4.0 do *Globus Toolkit*, o qual possui uma série de novas funcionalidades, o que levou a incompatibilidade com algumas características da versão anterior (3.2). Contudo, esta nova versão só se tornou disponível no mês de Maio deste ano (2005), tornando-se inviável a migração de versões.

Como ambiente de programação, foi utilizado o *Netbeans* 3.6, além de alguns *scripts* disponíveis em (Sotomayor, 2004), que facilitam na construção de projetos.

O processo de desenvolvimento foi dado sob o sistema operacional Gnu/Linux, nas distribuições *SuSE 9.2* e *Mandrake 10.1*.

5.1.2 Cenário de Implementação

Seguindo a proposta do trabalho, para o desenvolvimento do protótipo foi considerado um ambiente de uma Organização Virtual, como é mostrado na figura 5.1. Portanto, os integrantes deste ambiente terão a liberdade/flexibilidade de compartilharem os recursos, guardadas as devidas restrições de confiança especificadas. É válido lembrar que os recursos disponíveis na OV podem pertencer aos mais diversos tipos de integrantes, como: empresas, órgãos públicos, ONGs e profissionais liberais.

Dentro do cenário apresentado, os recursos podem assumir três diferentes papéis:

- **Servidor:** dentre todos os recursos pertencentes à OV, se faz necessário que um deles funcione como um servidor central. As funções de serviço de informação (*Index Service*)

¹A versão 3.2 do Globus é incompatível com a versão mais recente de Java, a 1.5.

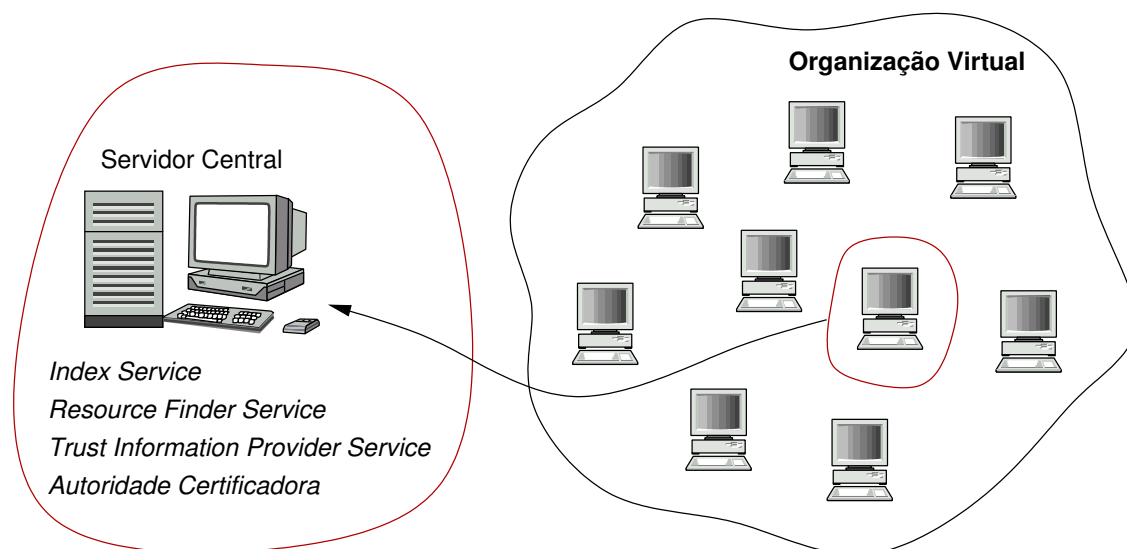


Figura 5.1: Cenário de execução

e autoridade certificadora (*Certificate Authority (CA)*) devem ser únicos em toda a infraestrutura. No caso do serviço de informação, a abordagem centralizada é utilizada pois se tornaria bem mais complexo manter as informações sobre os recursos em mais de uma fonte. Isso dificultaria a função do buscador de recursos². Quanto à autoridade certificadora, isso ocorre por questões de segurança, para que a assinatura de certificados fique sob a responsabilidade de um único administrador. Já os serviços de busca de recursos (*Resource Finder Service*) e de informações de confiança (*Trust Information Provider Service*) não necessariamente precisam estar no mesmo servidor central, já que de qualquer forma eles consultarão um fonte centralizada (maiores detalhes sobre os serviços serão dados na seção 5.1.4). Neste trabalho, todos os serviços citados estão localizados no mesmo servidor.

- **Provedor de recurso:** qualquer computador que faça parte da Grade (e isso requer algumas pré-configurações que serão vistas na seção 5.2.3) tem a opção de ficar disponível como mais um potencial recurso computacional. Contudo, isto não é uma obrigatoriedade dentro da Grade. Para que isto ocorra, é necessário que a informação do recurso seja publicada no servidor central (*Index Service*), além de se ter que definir as permissões de acesso a este recurso. As questões de publicação de informação também serão mais discutidas na seção 5.1.4;
- **Consumidor de recurso:** a partir do momento em que um usuário/recurso se integra à Grade, tendo as devidas permissões de acesso e certificados autenticados, este pode fazer uso de qualquer recurso que esteja disponível e ocioso (conforme será visto mais adiante, por “ocioso” entender-se-á que apenas uma pequena porcentagem do proces-

²Na verdade, à parte as vantagens que um modelo descentralizado teria, optou-se pela implementação mais simples uma vez que este aspecto não era o foco do modelo conceitual/arquitetura proposta nesta dissertação.

sador encontra-se ocupado).

Na seção 5.2 onde é descrito o fluxo de execução do protótipo com um exemplo prático, são detalhadas todas as informações sobre os computadores utilizados nos testes, como também o papel (servidor, consumidor e/ou provedor) que cada um assume.

Na seção que se segue, serão descritas todas as funcionalidades do protótipo, apresentadas na forma de diagramas UML.

5.1.3 Modelo de Implementação

O protótipo desenvolvido visa facilitar as várias etapas necessárias para que se possa fazer uso de recursos ociosos em uma Grade e, sempre que possível, tornar transparente a configuração/execução de alguns destas etapas. Estas facilidades englobam tarefas que não apenas a submissão propriamente dita de aplicações; são tarefas essenciais para que a submissão ocorra. Por exemplo, o usuário deve emitir um certificado de segurança e autenticá-lo, configurar a localização de serviços, etc. O diagrama de caso de uso na figura 5.2 apresenta as interações entre o usuário e o protótipo.

A **configuração do ambiente** engloba uma série de configurações que devem ser fornecidas para se ter o completo funcionamento do protótipo, além de dar ao usuário a flexibilidade de escolher o serviço que melhor lhe convier (isso ocorre somente no caso de se ter disponível mais de um serviço com a mesma função). As configurações são:

- Localização do servidor (onde estão instalado os serviços);
- Nome do serviço de informação (serviço padrão: *Index Service*);
- Nome do serviço de gerenciamento de recursos (serviço padrão: *Master Fork Managed Job Factory*);
- Nome do serviço de busca de recursos (serviço padrão: *Resource Finder*);
- Nome do serviço provedor de informação de segurança (serviço padrão: *Trust Information Provider*);
- E-mail da autoridade certificadora;
- Diretório onde estão localizados os certificados de segurança.

Para que se tenha um maior controle de quem terá permissão de uso ao seu recurso, o usuário pode restringir o uso do seu computador a parceiros que ele considere não ter um

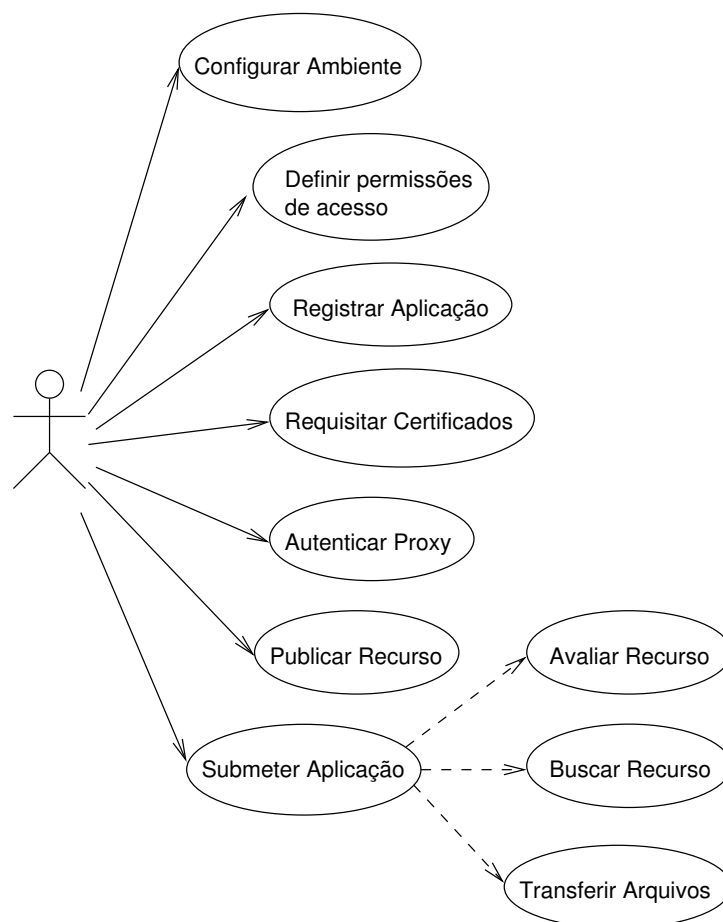


Figura 5.2: Diagrama de Caso de Uso

bom nível de confiança, **definindo as permissões de acesso**. Esse nível de confiança é obtido através de um serviço (*Trust Information Provider Service*). A partir das informações disponíveis neste serviço, é gerado um arquivo de configuração do Globus, onde estão descritos os usuários com permissão.

Também relativo à segurança, o usuário tem as funcionalidades para requisição de certificados e autenticação de *proxy*. Na **requisição de certificados**, algumas informações do usuário/recurso são necessárias para que os outros nós possam identificá-lo e permitir, ou não, o seu acesso. Informações como nome do usuário, nome da máquina, domínio ao qual a máquina pertence e a validade do certificado, são registradas no certificado. No cenário definido neste trabalho, o papel da autoridade certificadora (CA) responsável pela assinatura dos certificados é uma figura central. Após gerado, o certificado deve ser enviado para o CA, que o assina e o devolve para o usuário. Estes certificados são utilizados sempre que os usuários precisam criar um *proxy*, que por sua vez são necessários para que a submissão de aplicações possa ser feitas em nós remotos. Na **autenticação do proxy**, através do fornecimento de uma senha, o usuário prova ser o dono do certificado, e que portanto tem permissão e acesso

aos recursos da OV³.

Qualquer integrante da OV tem a opção de disponibilizar, ou não, o seu recurso na Grade. Caso este tome por opção não disponibilizar, nada precisa ser feito em relação a isso. Caso contrário, o usuário deve **publicar as informações do recurso** para que este possa ser encontrado. Informações pertinentes aos detalhes de *hardware* e *software* do computador são automaticamente obtidas pelo protótipo. Algumas informações ficam em aberto, e portanto devem ser fornecidas pelo próprio usuário, como: custo de acesso ao recurso, e o nível de confiança exigido para se fazer uso do recurso.

A **submissão de aplicações** depende diretamente que o usuário forneça todos os requisitos da aplicação para que esta possa ser executada com sucesso. Das informações fornecidas, as que estão definidas no RSL (ver seção 4.3.3) são utilizadas para execução, e as que são definidas no RSL Cliente (ver seção 4.3.3) são utilizadas na localização do recurso que se adeque às necessidades da aplicação. Os requisitos de execução são enviados juntamente com a aplicação e os dados de entrada para o ambiente remoto, e utilizados na execução no ambiente remoto.

Outras três funções são ativadas como passos prévios pela submissão da aplicação. Primeiro é feita uma detecção da real necessidade de se fazer uso da Grade, através da **avaliação do recurso**, verificando o seu estado atual. Havendo a necessidade, passa-se então à **busca pelo recurso**. A busca faz uso das informações dadas pelo usuário, acessando o serviço de informação (*Index Service*). Localizado o recurso adequado, a submissão pode, enfim, ser realizada. Para isso é feita a **transferência do arquivos** (executável, dados de entrada, bibliotecas utilizadas e arquivos de configuração) necessários para a execução, para o nó remoto.

A descrição dada acima pode ser vista em mais detalhes através de diagramas de classes do protótipo, onde estão definidos os relacionamentos entre as classes, além dos seus métodos e atributos. Nos diagramas apresentados a seguir só estão representados os principais atributos e métodos de cada classe de forma a facilitar a apresentação da figura (nos apêndices, em A.4, é encontrado o diagrama de classes por completo).

A classe *JobManager* é a principal classe da estrutura. Esta classe é responsável por efetivar a submissão de uma aplicação, através do método *submit*. A classe *GramClient* é o cliente de acesso ao serviço de Grade responsável pela execução da aplicação em um ambiente remoto, atuando como intermediário entre a classe *JobManager* e este serviço, como representado na figura 5.3. No Globus, este serviço recebe o nome de *Master Fork Managed Job Factory Service*, e funciona como um escalonador simples. O nome do serviço e a localização (URL) do recurso são dados no atributo *factoryUrl*. Para que a aplicação possa

³Todo o modelo de segurança adotado é oferecido pelo próprio Globus. A proposta desta dissertação apenas faz uso das funcionalidades de segurança oferecidas.

ser executada em um nó remoto, é necessário transferi-la para este nó, assim como os dados utilizados no processamento e possíveis bibliotecas (e.g. um arquivo *jar* de uma biblioteca Java) utilizadas pela aplicação. A classe *GridFTP* provê métodos para transferência para um nó remoto, e do nó remoto para a máquina do usuário.

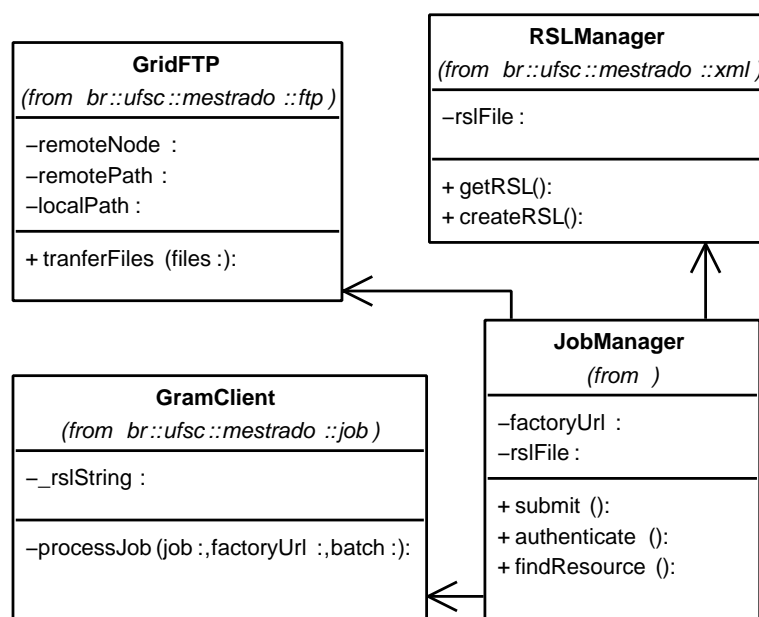


Figura 5.3: Diagrama de Classes

A localização do recurso é acionada pela classe *FindResource*, onde esta funciona como um cliente que acessa o serviço de *Grade Resource Finder Service*, através da interface *ResourceFinderPortType*. O serviço é invocado através do método *find*, que necessita de dois parâmetros. O primeiro, *priority*, informa qual prioridade deve ser tomada na seleção do recurso. As prioridades podem ser: menor custo, maior confiança, ou melhor desempenho (e se esta for a prioridade, deve-se informar se o mais importante para a aplicação é uma maior quantidade de memória ou maior poder de processamento). O segundo parâmetro é o *query*, onde é especificada a consulta com todos os requisitos da aplicação. Pelo padrão definido pelo Globus, a linguagem de consulta utilizada é a XPath (W3C, 1999), uma linguagem definida em XML, que é voltada para acessar determinadas partes de documentos XML. Um exemplo de uma consulta em XPath é mostrada abaixo:

```

//glue:SubCluster/glue:Host[@glue:Name] |
//glue:SubCluster/glue:Host/glue:ProcessorLoad[@glue:Last5Min<20] |
//glue:SubCluster/glue:Host/glue:MainMemory[@glue:RAMSize>=256] |
//glue:SubCluster/glue:Host/glue:Processor[@glue:ClockSpeed>=2000]
  
```

Esta consulta representa a busca por um recurso que tenha estado os últimos 5 minutos com, no máximo, 20% do processador em uso, além de memória de no mínimo 512MB, e processador de no mínimo 2000Mhz de frequência.

As informações obtidas pela classe *FindResource* para montar a consulta são manipuladas pela classe *RSLCManager*. Estas informações são fornecidas pelo usuário no momento em que este especifica os requisitos da aplicação. A associação entre elas é mostrada em 5.4. Esta classe possui métodos para criação e acesso aos atributos que identificam as necessidades da aplicação. Seguindo o modelo definido pelo Globus, com o RSL (Globus, 2000), neste trabalho foi criada uma “extensão” a esta especificação, que representa todos os valores utilizados na busca de recursos, o qual nomeou-se **RSL Cliente**. A figura 5.5 mostra um exemplo de um documento RSL Cliente.

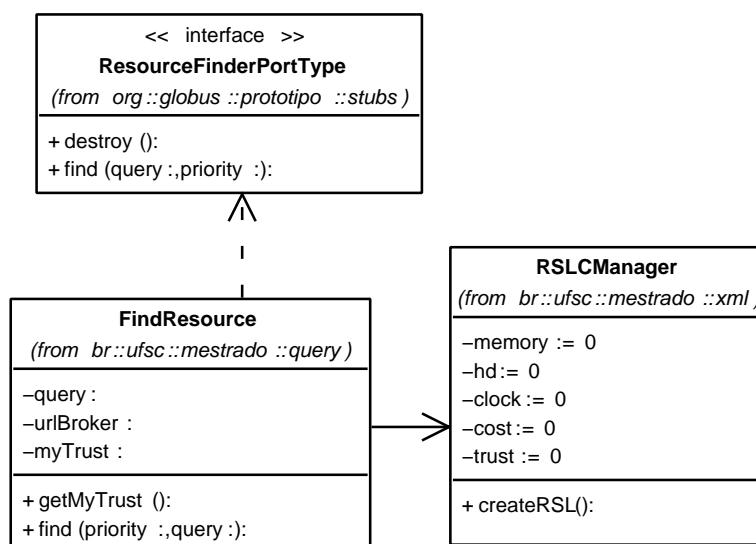


Figura 5.4: Diagrama de Classes

Além das informações de *hardware* e *software*, outros dois critérios devem ser informados, de acordo com a prioridade definida pelo usuário para a aplicação: o custo (*cost*) máximo aceitável para se utilizar um recurso, e a confiança (*trust*) que o requerido recurso deve oferecer.

Assim como a classe *RSLCManager*, que manipula documentos RSLC, também foi desenvolvida a classe *RSLManager*, que manipula os já padronizados documentos RSL. Conforme dito na seção 3.7.4, este tipo de documento especifica detalhes sobre a execução de aplicações pelo módulo GRAM do Globus. Por se tratar de especificações relativamente extensas, esta classe foi desenvolvida para que o usuário facilmente crie/edite documentos RSL.

No modelo conceitual (seção 4.3.3 é sugerida a definição de uma ontologia como forma

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <rsl>
3   <resource>
4     <operatingSystem>
5       <stringElement value="Linux" />
6     </operatingSystem>
7     <architecture>
8       <stringElement value="cisc" />
9     </architecture>
10    <processor>
11      <stringElement value="Intel Pentium 4" />
12    </processor>
13    <clock>
14      <integer value="2000" />
15    </clock>
16    <memory>
17      <integer value="256" />
18    </memory>
19    <hardDisk>
20      <integer value="15" />
21    </hardDisk>
22    <bandWidth>
23      <integer value="300" />
24    </bandWidth>
25    <cost>
26      <integer value="100" />
27    </cost>
28    <trust>
29      <integer value="9" />
30    </trust>
31  </resource>
32 </rsl>
```

Figura 5.5: Exemplo de um documento RSL Cliente

de se ter um vocabulário de termos comuns, que é implementado por ambas as classes, *RSLManager* e *RSLCManager*. Contudo, o modo simplificado como a ontologia foi implementada neste protótipo, tem a sua estrutura aproximada mais a forma de uma taxonomia. O vocabulário utilizado foi baseado no proposto pelo trabalho (Fleischman, 2004), como é apresentada na tabela 5.1. Assim sendo, o usuário limita-se a utilizar os valores definidos neste vocabulário. Com isso, as demais classes e serviços que farão uso destas informações terão a garantia que essas foram especificadas de forma padronizada.

No momento em que a classe *FindResource* monta a consulta, também deve ser informada a confiança do próprio usuário que está realizando a busca. Isto é feito para que o serviço que faz a busca de recursos saiba que só deve procurar por recursos que aceitam usuários com tal nível de confiança. *UserTrust* é a classe cliente responsável por acessar o serviço (seção 5.1.4), através da interface *TrustInformationProviderPortType*, que dispõe destas informações.

A interface *TrustInformationProviderPortType* é também utilizada pela classe *GridMap-FileManager* para gerar/manipular o arquivo *grid-mapfile* do Globus, como mostra a figura 5.6. Este arquivo define os usuários que terão permissão de uso da máquina para submissão

Conceito	Instância
Arquitetura	RISC, CISC
Processador	AMD, AMD64, IBM, Intel, Intel64, Sun
Sistema Operacional	Apple, Linux, Unix, Windows
Linux	Debian, Fedora, FreeBSD, Mandrake, RedHat, SuSE
Unix	AIX, HP-UX, Solaris
Windows	2000, NT, XP
Sistema de Arquivos	EXT, EXT2, EXT3, FAT16, FAT32, NTFS

Tabela 5.1: Dicionário de termos comuns, no domínio de recursos computacionais

de aplicações. Abaixo é apresentado um trecho de um arquivo grid-mapfile (figura 5.7).

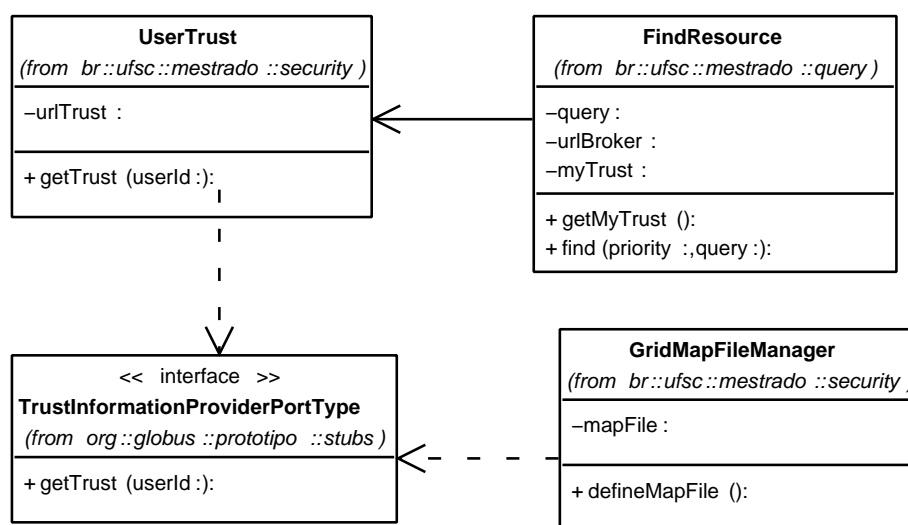


Figura 5.6: Diagrama de Classes

No arquivo está representado que apenas os usuários de conta/máquina terão permissão de uso: *globusadmin/floyd.gsigma.ufsc.br*, *fabiopinheiro/lee.gsigma.ufsc.br* e *josesilva/van-gogh.das.ufsc.br*. Na forma tradicional, este arquivo é preenchido pelo administrador de sistema (no caso, o Globus), que a partir de uma lista já pré-definida, ou de solicitações, define os usuários a serem incluídos. Contudo, neste trabalho foi adotada a abordagem de se definir esta configuração a partir do nível de confiança de cada parceiro da OV. Sendo assim, o usuário deve especificar o nível mínimo de confiança que qualquer usuário que queira

```

1  "/O=Grid/OU=simpleCA-floyd.gsigma.ufsc.br/CN=Globus Admin" globusadmin
2
3  "/O=Grid/OU=simpleCA-lee.gsigma.ufsc.br/CN=Fabio Pinheiro" fabiopinheiro
4
5  "/O=Grid/OU=simpleCA-vangogh.das.ufsc.br/CN=Jose Silva" josesilva

```

Figura 5.7: Exemplo de um arquivo grid-mapfile

usar o seu recurso deve ter. Estes valores são apresentados num formato de acordo com as informações providas pelo serviço *TrustInformationProvider*. Como será mostrado na seção 5.1.4, neste protótipo estes valores são representados como “notas” de 0 (zero) a 10 (dez). Dessa forma, se o usuário define que a confiança mínima é “8”, somente os parceiros com confiança 8, 9 e 10 terão permissões de uso. A classe *GridMapFileManager* oferece métodos para busca em serviços das informações de confiança e para a geração do *grid-mapfile*. Por se tratar de um importante arquivo de configuração, a instalação padrão do Globus localiza o *grid-mapfile* em uma área de acesso restrito ao super usuário (*root*) do sistema.

A classe *InformationPublisher* é utilizada apenas pelos clientes que desejam publicar as informações do seu recurso no serviço de diretórios. O serviço *Index Service*, provido pelo Globus, oferece funcionalidades para publicação e agregação de dados de outros serviços. No caso da classe *InformationPublisher*, através do método *publish*, são publicadas as informações sobre o recurso do cliente, que faz acesso ao *Index Service* através do serviço *SystemInfo*. Isso ocorre com a invocação do método *publishInformation* da interface *SystemInfoPortType*, como mostra o diagrama em 5.8. Mais detalhes sobre esse serviço serão dados na seção 5.1.4.

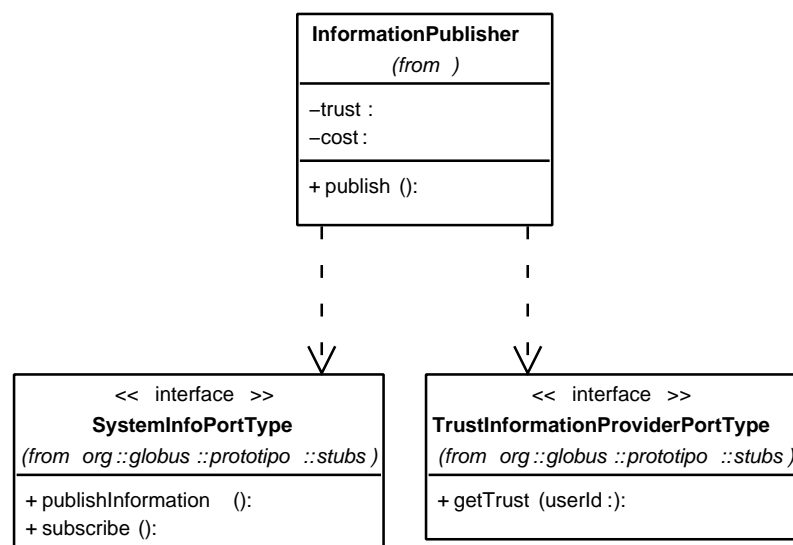


Figura 5.8: Diagrama de Classes

No diagrama em 5.9 é apresentada a classe *GridProxyInit*. Esta classe tem a função de

autenticar o certificado do usuário, gerando o seu *proxy*. Conforme foi dito na seção 3.7.2, cada usuário deve possuir um certificado, que é assinado por uma autoridade certificadora. O usuário deve fornecer a sua senha (criada na geração do certificado) e, através do método *createProxy*, o certificado do usuário é lido (por *default*, o certificado fica localizado no diretório *.globus* no diretório raiz do usuário) e então o *proxy* é criado. A validade do *proxy* é de 24 horas, sendo esse valor configurável. Isso evita que o usuário tenha que se autenticar a cada submissão que ele efetue.

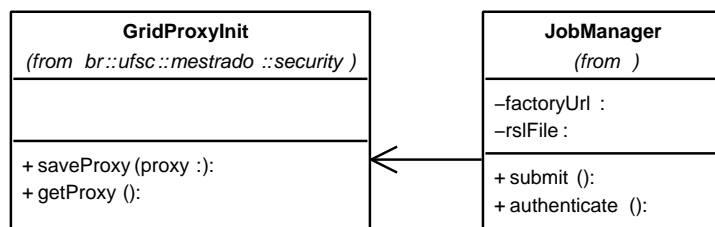


Figura 5.9: Diagrama de Classes

Para detectar a necessidade ou não de se fazer acesso a um recurso da Grade, foi desenvolvida a classe *GridNecessityTester*. Através da classe *SystemStatus*, a classe *GridNecessityTester* obtém o estado corrente da máquina e os compara com os requisitos da aplicação que o usuário quer executar, obtidos pela classe *RSLCManager*, apresentadas no diagrama em 5.10. No quadro 5.11 abaixo é apresentado um trecho de código onde os valores com os requisitos da aplicação são iniciados, assim como um objeto da classe *SystemStatus* é instanciado. No método *gridNecessity*, primeiro é feita uma avaliação se o processador encontra-se ocioso a partir da comparação da carga atual do processador e uma constante (*IDLE_RESOURCE*). Nesta constante é definida qual porcentagem máxima de carga em um processador para que ele seja considerado ocioso (e.g. 0%, 10%). Para efeitos de testes, foi tido como ocioso um processador que tem sua carga ocupada em no máximo 20%⁴. Estando o computador ocioso, os outros valores são então avaliados. Verificando que o computador não dispõe de recursos para executar a aplicação, a classe *JobManager*, que a invocou, inicia o processo para submissão da aplicação em um recurso da Grade.

⁴Este valor foi definido como fixo, por não acarretar em grandes diferenças nos resultados do protótipo. Contudo, de acordo com as necessidades dos usuário, uma adaptação pode ser facilmente feita para que este valor seja configurável.

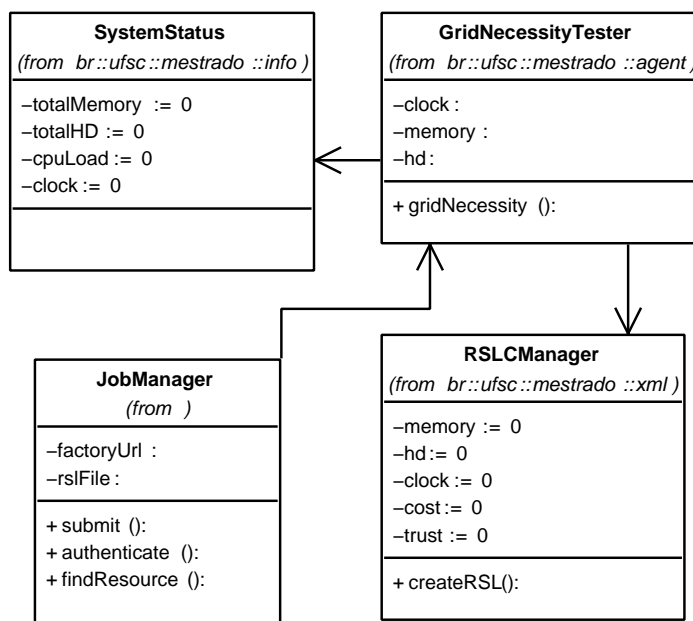


Figura 5.10: Diagrama de Classes

Vale destacar que das classes apresentadas, algumas foram adaptadas de classes já existentes na API do Globus/CoG, outras fizeram uso de algumas classes desta API e as demais foram desenvolvidas totalmente independente das APIs do Globus e/ou *CoG KiT*. Essa classificação é apresentada a seguir:

- Classes Adaptadas do Globus/CoG: *GridProxyInit*, *GramClient* e *GridFTP*;
- Classes que fazem uso da API do Globus/CoG: *JobManager*, *InformationPublisher*, *ResourceFinderPortType*, *SystemInfoPortType* e *TrustInformationProviderPortType*;
- Classes independentes da plataforma: *RSLManager*, *RSLCManager*, *SystemStatus*, *GridNecessityTester*, *UserTrust*, *FindResource* e *GridMapFileManager*.

Por se tratar do ponto central do protótipo, a submissão de aplicações é apresentada no diagrama de seqüência em 5.12. No diagrama são retratadas todas as classes que estão relacionadas a este processo de submissão, e a seqüência em que os passos são executados.

O controle de execução das tarefas/aplicações é de responsabilidade de um escalonador. Contudo, este protótipo foi desenvolvido para executar em ambiente restrito. Portanto, a função de escalonador foi simplificada, deixando à cargo do serviço *Master Fork Managed Job Factory Service* a execução da aplicação. Para submeter a um escalonador PBS 3.8.1, por exemplo, basta utilizar o serviço *Master PBS Managed Job Factory Service*.

```
1 private SystemInformation info;
2 private RSLEManager rsle;
3 private int IDLE_PROCESSOR = 20;
4
5 public GridNecessityTester(String rsleFile) {
6
7     this.info = new SystemStatus();
8     this.rsle = new RSLEManager(rsleFile);
9
10    this.memory = rsle.getMemory();
11    this.clock = rsle.getClock();
12    this.hd = rsle.getHd();
13    this.processor = rsle.getProcessor();
14    this.architecure = rsle.getArchitecture();
15    this.OS = rsle.getOS();
16 }
17
18 public boolean gridNecessity(){
19
20     //Testa se o processador está ocioso
21     if (IDLE_PROCESSOR > info.getProcessorLoad()
22         && this.architecure.equalsIgnoreCase(info.getArchitecture())
23         && this.OS.equalsIgnoreCase(info.getOS())
24         && info.getFreeMemory() >= this.memory
25         && info.getClock() >= this.clock
26         && info.getFreeHD() >= this.hd)
27         return false;
28
29     return true;
30 }
```

Figura 5.11: Trecho do código da classe *GridNecessityTester*

5.1.4 Serviços

O funcionamento do protótipo depende de três serviços de Grade (*Grid Services*), que serão apresentados a seguir.

Serviço de Informações do Sistema (*System Information Service*)

O Serviço de Informações do Sistema é o serviço responsável pela publicação das informações do recurso, para que assim o recurso possa ser localizado e utilizado pelos demais integrantes da OV. Para isso, ele necessita das funcionalidades do *Index Service*, provido pelo Globus. Conforme descrito na seção 3.7.5, o *Index Service* provê métodos para registro/agregação de dados dos serviços de Grade, os *service data*. Este serviço publica dois SDEs (*Service Data Element*) no *Index Service* central. O SDE do serviço *Resource Information Provider* fornece todas as informações de *hardware* e *software* necessárias ao serviço buscador de recursos: sistema operacional, versão do *kernel* e do SO, arquitetura do sistema, memória total, disponível e virtual, tamanho total e disponível de disco rígido, sistema de arquivos, carga do processador e modelo do processador. Uma segunda função deste serviço

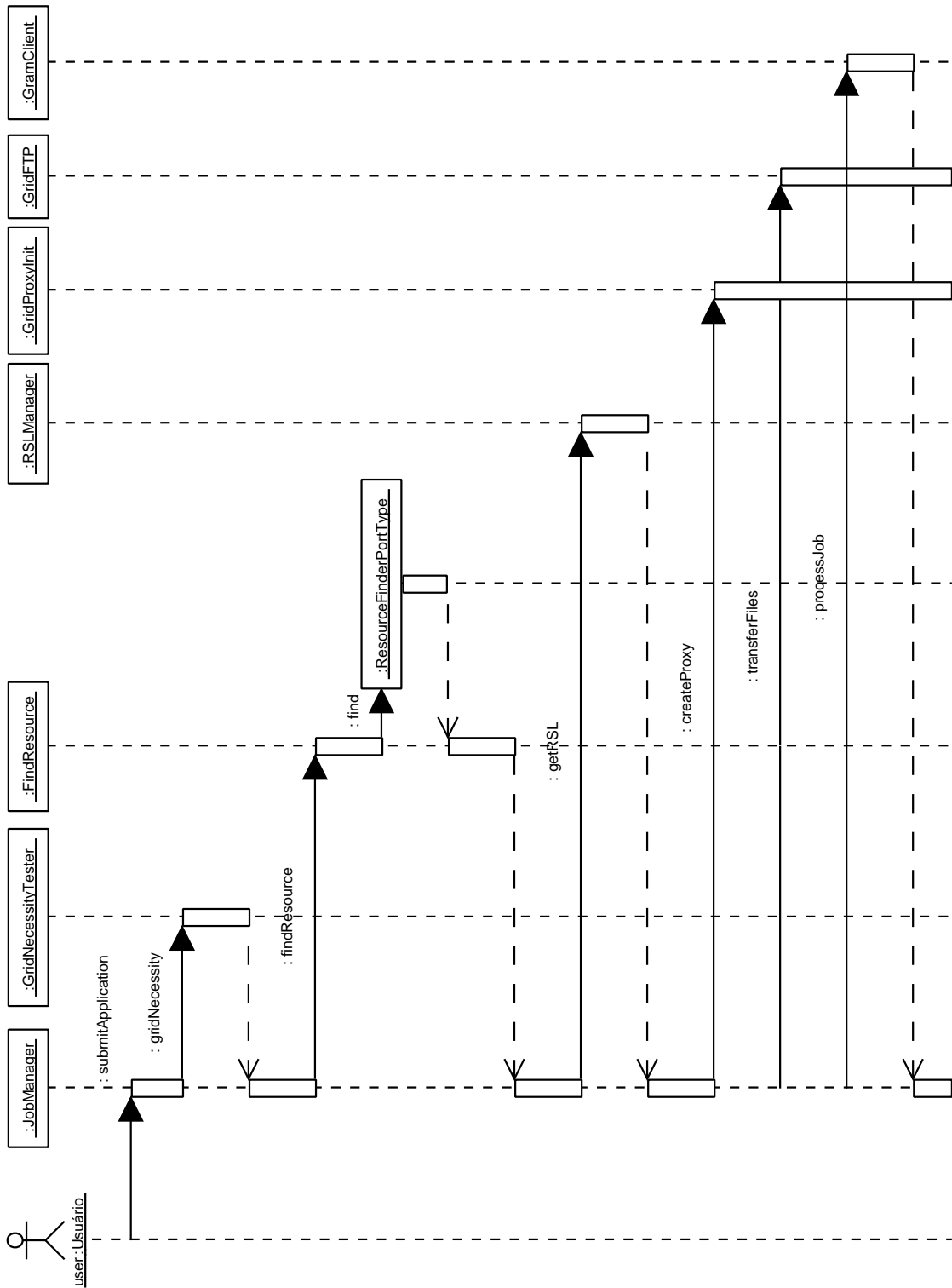


Figura 5.12: Diagrama de seqüência da submissão de aplicações

é de manter o seu SDE atualizado com o estado corrente do computador. Os parâmetros que são constantemente atualizados são: memória disponível, espaço disponível de disco e a carga do processador. Logo, o usuário não precisa se preocupar em fornecer essas informações, ficando esta tarefa completamente transparente. O segundo SDE, de nome *SystemInfoData*, foi criado para fornecer outros tipos de informações, que estão relacionadas com questões de compartilhamentos em OVs. Alguns desses valores são fornecidos pelo próprio usuário, e outros são obtidos através de serviços. São as informações:

- **Custo:** o usuário deve fornecer, caso isto seja previamente definido pela organização, o custo de acesso ao seu recurso. No protótipo, a questão do custo tem o papel apenas como critério de busca/seleção de recursos. Caso seja definida, a cobrança pelo uso do recurso seria feita por um sistema à parte;
- **Nível de Confiança Exigido:** com este parâmetro, o usuário informa o nível mínimo de confiança exigido a um usuário para que esse tenha acesso ao seu recurso;
- **Nível de Confiança Próprio:** além da confiança exigida, o usuário também deve publicar o seu próprio nível de confiança. Isso facilita na busca de recursos já que todas as informações necessárias encontrar-se-ão centralizadas. Este valor não é fornecido pelo usuário mas sim obtido pelo serviço *Trust Information Provider*, que por sua vez só permite a leitura (e não a escrita) destas informações.

Seguindo a especificação OGSII, um *service data* deve ser definido como um XML *Schema*. Uma parte do XML *Schema* que descreve o SDE *SystemInfoData* pode ser visto na figura 5.13. Nele são definidas as três informações a serem publicadas e seus respectivos tipos de dados: *cost* (custo), *trust* (nível de confiança próprio) e *partnerTrust* (nível de confiança exigida ao parceiro).

Este é o único dos serviços desenvolvidos que deve ser instalado localmente em todos os nós que forem ficar disponíveis na Grade. Isso se deve ao fato deste necessitar coletar, através de outro serviço - o *Resource Information Provider* - informações do recurso do usuário. Após coletadas todas as informações, estas são publicadas, agregando-se aos SDEs no *Index Service*, através da classe *DataAggregationType*. O processo de publicação é melhor detalhado através do diagrama de seqüência da figura 5.14.

Portanto, este serviço só precisa ser instalado caso o recurso for ser disponibilizado na Grade.

```

1 <wsdl:definitions name="SystemInfoData" ... >
2 ...
3 <wsdl:types>
4 <xsd:schema attributeFormDefault="qualified"
5   elementFormDefault="qualified"
6   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
7   <xsd:complexType name="SystemInfoDataType">
8     <xsd:sequence>
9       <xsd:element name="cost"><xsd:complexType>
10        <xsd:attribute name="value" type="xsd:int"/>
11        </xsd:complexType></xsd:element>
12        <xsd:element name="trust"><xsd:complexType>
13         <xsd:attribute name="value" type="xsd:int"/>
14         </xsd:complexType></xsd:element>
15        <xsd:element name="partnerTrust"><xsd:complexType>
16         <xsd:attribute name="value" type="xsd:int"/>
17         </xsd:complexType></xsd:element>
18      </xsd:sequence>
19    </xsd:complexType>
20 </xsd:schema>
21 </wsdl:types>
22 </wsdl:definitions>

```

Figura 5.13: Trecho do XML *Schema* que define o *SystemInfoData*

Serviço Provedor de Informações de Confiança (*Trust Information Provider Service*)

Por serem dados gerenciados pela administração do VBE/PVC, as informações sobre a confiança dos integrantes de uma OV estão disponíveis apenas com permissão de leitura. A função deste serviço é de facilitar o acesso as estas informações.

Em um ambiente real de OVs, informações deste tipo são geradas por ferramentas específicas, como por exemplo, através do histórico de participações de um determinado parceiro em OVs anteriores (Afsarmanesh, 2004). O armazenamento pode ser feito em bancos de dados, onde se pode ter um maior controle quanto aos tipos de permissões aos dados.

Neste trabalho foi gerado um arquivo “fictício” com as informações de confiança necessárias, simulando um caso real, já que não foi possível criar este ambiente real. No caso, as informações utilizadas foram a identificação do participante; a organização a qual ele pertence, o domínio ao qual ele pertence, o nível de confiança.

O serviço possibilita que estas informações sejam resgatadas de duas formas:

- Fornecendo o identificador do usuário/seu domínio, obtendo apenas o seu nível de confiança;
- Requisitando o nível de confiança de todos os integrantes da OV. Esta opção é utilizada para a geração do *grid-mapfile*, que define os usuários com permissão de acesso.

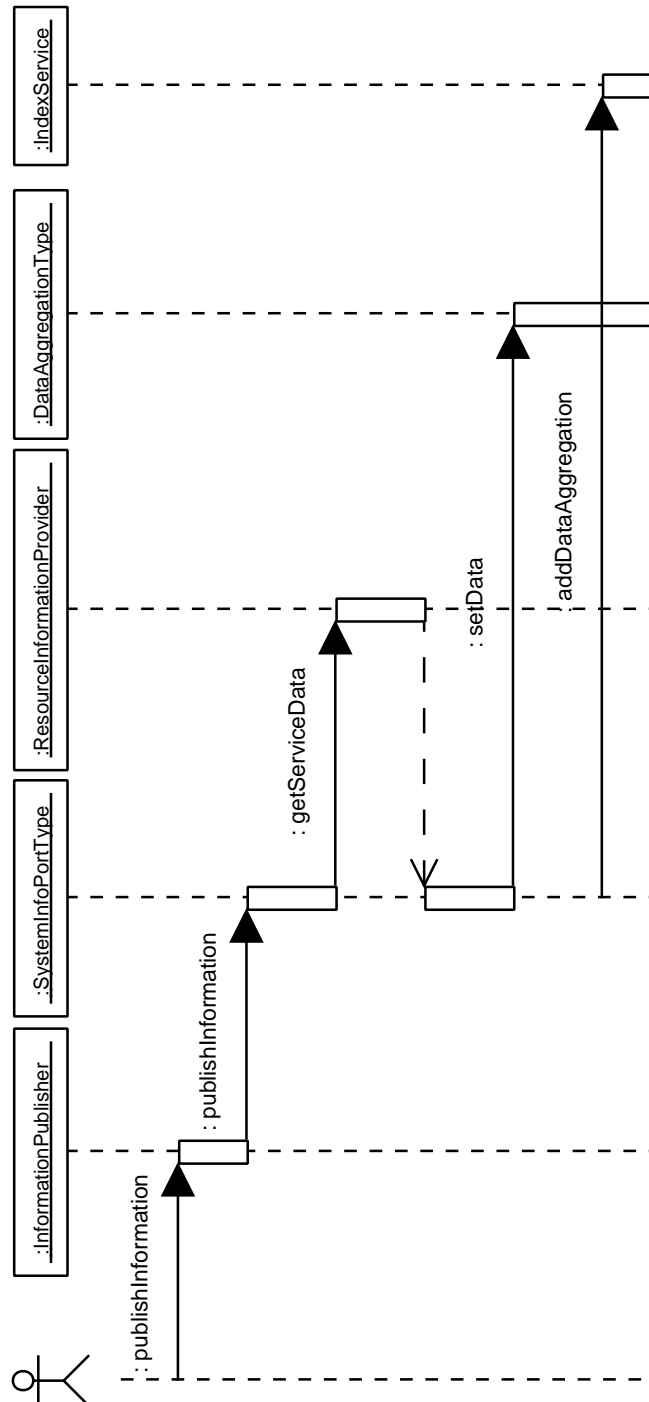


Figura 5.14: Diagrama de seqüência da publicação de informações

Serviço Buscador de Recursos (*Resource Finder Service*)

Como já mencionado anteriormente, este serviço é responsável pela busca de recursos que supram as necessidades das aplicações. Através das informações providas por outro serviço - o *Index Service* - o buscador seleciona o recurso que melhor se adequa aos requisitos da aplicação.

Na descrição da classe cliente *FindResource* foi dito que esta acessa o serviço buscador através da interface *ResourceFinderPortType*, passando como parâmetros a consulta e a prioridade de seleção (atributos *query* e *priority*).

No parâmetro de consulta são descritos apenas os detalhes relacionados aos requisitos da aplicação. Além deste, outros parâmetros são necessários para se efetivar a consulta no *Index Service*, como o nome do *Service Data Element* e do *name space* que identifica este SDE. Por exemplo, consultas relativas ao desempenho, que foram providas pelo serviços *Resource Information Provider*, possuem SDE de nome *mds:Cluster* e *name space* igual a <http://glue.base.ogsa.globus.org/ce/1.1>. Já consultas sobre custo ou confiança, possuem SDE de nome *SystemInfoData* e *name space* igual a http://www.globus.org/namespaces/2004/02/prototipo/SystemInfoService_sd/SystemInfoSDE. Todos estes parâmetros são utilizados pelo método *getXPathQuery* da classe *QueryHelper*, para busca de recursos.

Caso ocorra do retorno da consulta trazer mais de um recurso como opção, o serviço faz a seleção do melhor dentre eles através da prioridade que lhe foi passada. Sendo assim, pode ser selecionado o recurso de melhor desempenho, de maior confiança ou de mais baixo custo. No caso de empate nos valores de prioridade, o desempate é feito com uma segunda prioridade.

Como estas informações de retorno são definidas por um XML *Schema*, a extração das informações se torna mais simples. Para trabalhar com documentos XML, utilizou-se a API JDOM (JDOM, 2000). O retorno para a consulta dada anteriormente que busca um recurso com memória, por exemplo, maior de 256MB e processador de frequência maior que 2000Mhz, é dado na figura 5.15.

Por não se tratar do foco do trabalho, os algoritmos para a seleção de recurso não foram desenvolvidos baseados em nenhum estudo mais aprofundado sobre o tema. De acordo com o que foi descrito no capítulo 4, do modelo conceitual, o diferencial buscado neste serviço encontra-se na preocupação que se teve ao definir um vocabulário de termos comuns, evitando assim que o usuário não fornecesse valores que fugissem a um padrão pré-definido. Na implementação do protótipo, este vocabulário foi definido na forma de um arquivo texto, baseado nos valores descritos na tabela 5.1. Desta forma, o serviço buscador de recurso pode se valer da garantia de grande diminuição de problemas semânticos.


```

1 <ce:Host ce:Name="floyd.gsigma.ufsc.br" ce:UniqueID="floyd.gsigma.ufsc.br" xmlns:ce="
  http://glue.base.ogsa.globus.org/ce/1.1">
2 <ce:OperatingSystem ce:Name="Linux" ce:Release="2.6.8.1-10mdk" ce:Version="#1 Wed
  Sep 8 17:00:52 CEST 2004" xmlns:ce="http://glue.base.ogsa.globus.org/ce/1.1"/>
3
4 <ce:MainMemory ce:RAMAvailable="139" ce:RAMSize="495" ce:VirtualAvailable="373"
  ce:VirtualSize="509" xmlns:ce="http://glue.base.ogsa.globus.org/ce/1.1"/>
5
6 <ce:Processor ce:CacheL2="512" ce:ClockSpeed="2400" ce:Model=" Intel(R) Pentium(R) 4
  CPU 2" ce:OtherProcessorDescription="fpu vme de pse tsc msr pae mce cx8 apic
  sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe
  cid" ce:Vendor=" GenuineIntel" ce:Version="15.2.7" xmlns:ce="http://glue.base.
  ogsa.globus.org/ce/1.1"/>
7
8 <ce:ProcessorLoad ce:Last15Min="00" ce:Last1Min="00" ce:Last5Min="00" xmlns:ce="
  http://glue.base.ogsa.globus.org/ce/1.1"/>
9 <ce:FileSystem ce:AvailableSpace="12425" ce:Name="/" ce:ReadOnly="false" ce:Root="/"
  ce:Size="24606" ce:Type="unavailable" xmlns:ce="http://glue.base.ogsa.globus.
  org/ce/1.1"/>
10
11 <ce:FileSystem ce:AvailableSpace="3230" ce:Name="/home" ce:ReadOnly="false" ce:Root=
  "/home" ce:Size="35246" ce:Type="unavailable" xmlns:ce="http://glue.base.ogsa.
  globus.org/ce/1.1"/>
12
13 <ce:FileSystem ce:AvailableSpace="13282" ce:Name="/usr/pessoal" ce:ReadOnly="false"
  ce:Root="/usr/pessoal" ce:Size="14762" ce:Type="unavailable" xmlns:ce="http://
  glue.base.ogsa.globus.org/ce/1.1"/>
14 </ce:Host>

```

Figura 5.15: Retorno de uma consulta que busca um recurso de bom desempenho

5.2 Execução

5.2.1 Contexto

Como explicado, o protótipo fez uso de um sistema já desenvolvido (em (Bittencourt, 2005)) para ajudar os gerentes na seleção das empresas mais apropriadas para a constituição de uma EV com base em um conjunto de possibilidades de escalonamentos.

5.2.2 Instalação do Protótipo

O protótipo consta de um conjunto de janelas gráficas desenvolvidas utilizando classes da biblioteca *javax.swing* de Java. Para o compilação/execução do protótipo, além das APIs padrões de Java, foram também necessárias as seguintes bibliotecas:

- Pacotes da API do Globus/CoG: *ogsa.jar*, *cog-jglobus.jar*, *gram-rips.jar*, *mds-aggregator.jar*, *mds-index.jar*, *mds-providers.jar*, *mjs.jar*, *mmjfs.jar* e *wsdl4.jar*.
- API JDOM com classes para manipulação de documentos XML, o *jdom.jar*

A instalação do *Globus Toolkit* (GT) só é necessária nos recursos que serão disponibilizados na Grade, já que a aceitação de submissão de aplicações depende de serviços locais, sendo estes suportados pelo container Globus. O serviços locais utilizados são os já citados: *Resource Information Provider Service*, *Master Fork Managed Job Factory Service* e *System Information Service*. Portanto, estes serviços devem ser instalados em cada computador que oferecer os seus recursos.

No caso dos usuários que não vão disponibilizar os seus recursos, não há a necessidade de se instalar o GT. A única coisa que estes usuário necessitarão - além do protótipo - são os pacotes citados acima. Tendo-se isso, o protótipo já está pronto para ser executado, como apresentado na figura 5.16, restando apenas a configuração para o seu pleno funcionamento.

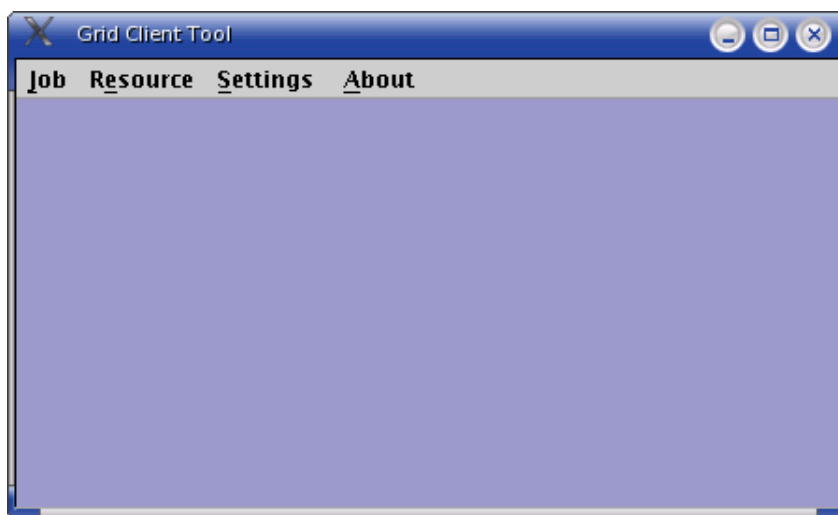


Figura 5.16: Tela principal

Para realizar os testes, o protótipo, assim como o GT 3.2, foram instalados em três computadores, descritos na tabela 5.2, cada uma com as seguintes diferentes configurações, usadas na rede local do GSIGMA⁵. A submissão de aplicações é suportada pelo módulo *Grid Resource Allocation and Management* (GRAM), que está presente apenas na versão completa do *Globus Toolkit*. Esta versão completa só está disponível para a plataforma Unix. Portanto, apesar do protótipo ser portátil (por ser escrito em Java), existe a dependência do sistema operacional devido aos requisitos da plataforma GT.

Toda a configuração/execução será tomada pela visão do usuário “Fábio Pinheiro”, a partir da máquina *chingling*.

⁵No período dos estudos iniciais com o Globus, foram realizados teste de submissão de tarefas entre diferentes redes. No caso, da rede do LCMI (Laboratório de Controle e Microinformática) para a rede do GSIGMA

Nome da máquina	<i>lee</i>	<i>floyd</i>	<i>chingling</i>
Função	Servidor, provedor e consumidor de recursos	provedor e consumidor de recursos	provedor e consumidor de recursos
Usuário	Globus Admin	José da Silva	Fábio Pinheiro
Processador	AMD Athlon XP 1600+	Intel Pentium IV 2.4Ghz	AMD Athlon XP 1600+
Sistema Operacional	Linux SuSE 9.1	Linux Mandrake 10.1	Linux Mandrake 10.1
Memória	256mb	512mb	128mb
Disco Rígido	40gb	40gb	80gb
Sistema de arquivos	EXT3	EXT3	EXT3

Tabela 5.2: Computadores utilizados nos testes

5.2.3 Configuração do Protótipo

A configuração do protótipo engloba passos para definição de URLs, nomes de serviços, e-mail do CA, diretório dos certificados de segurança e nível de confiança exigido. A tela 5.17 apresenta todas essas configurações a serem feitas. Para os testes, a configuração utilizada foi a seguinte:

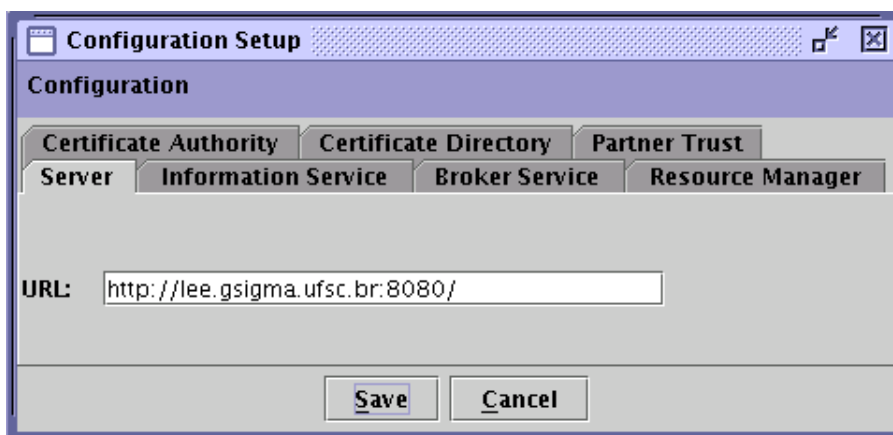


Figura 5.17: Tela de Configuração

- *Server*: `http://lee.gsigma.ufsc.br:8080/`
- *Information Service*: `ogsa/services/base/index/IndexService`
- *Broker Service*: `ogsa/services/prototipo/finder/ResourceFinderService`
- *Resource Manager Service*: `ogsa/services/base/gram/MasterForkManagedJobFactory-Service`
- *Certificate Authority*: `ca@gsigma.ufsc.br`

- *Certificate Directory*: /home/fabiopinheiro/.globus/
- *Partner Trust*: 8

Além destas configurações, o usuário necessita fazer a requisição do certificado de segurança, para que este possa ser assinado pelo CA. A requisição é feita através da tela apresentada na figura 5.18, através do fornecimento do nome do usuário e uma senha, que será utilizada na autenticação do *proxy*.

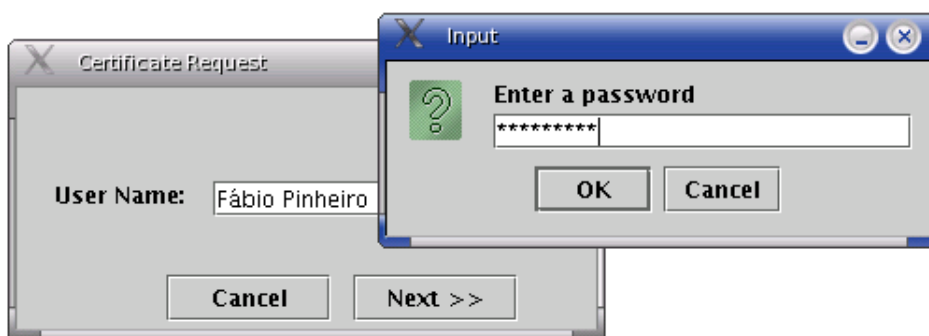


Figura 5.18: Requisição do certificado de segurança

Como resultado da operação, são gerados dois arquivos no diretório (*certificate directory*) configurado pelo usuário:

- *usercert_request.pm*: arquivo com as informações sobre o usuário, computador e domínio.
- *userkey.pm*: arquivo com a chave privada do usuário, criptografada.

Também como etapa de configuração de segurança, o usuário deve definir as permissões de acesso através da geração do *grid-mapfile*. A tela em 5.19 representa uma requisição ao serviço *Trust Information Provider*, em busca de todos os parceiros da OV que possuem nível de confiança maior que “8” (oito), valor definido pelo usuário durante as configurações. O retorno do serviço é então passado por um *parser* que gera as informações no formato do *grid-mapfile* (apresentado na figura 5.7). De acordo com a vontade do usuário, este arquivo pode ser editado e então salvo. O arquivo *grid-mapfile* fica localizado em um diretório de acesso restrito, portanto só pode ser movido para este diretório por um usuário com privilégios de administrador (super-usuário ou *root*).

O último passo de configuração do protótipo é o único opcional. Como dito em seções anteriores, o usuário não é obrigado a disponibilizar o seu recurso na grade. Contudo, caso decida disponibilizá-lo, é necessário que suas informações sejam publicadas no *Index Service*. A figura 5.20 apresenta a tela onde o usuário fornece estas informações: o custo; o seu nível de confiança, obtido através do serviço *Trust Information Provider* (não editável); nível mínimo exigido ao parceiro, obtido das configurações prévias.

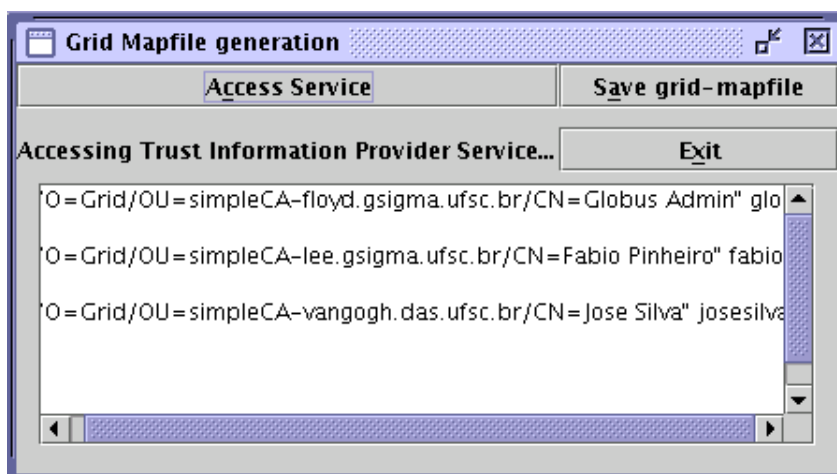
Figura 5.19: Definição do arquivo *grid-mapfile*

Figura 5.20: Publicação das informações

As informações relativas a *hardware* e *software* são omitidas do usuário. Como já detalhado no diagrama de seqüência 5.14, clicando no botão “*publish*”, o usuário invoca o serviço local *System Information*, que coleta as informações fornecidas pelo próprio usuário e as informações de *hardware* e *software* obtidas através do serviço local *Resource Information Provider*, e as publica no serviço de informação, definido nas configurações como *ogsa/services/base/index/IndexService*. A partir deste momento, qualquer consulta feita em busca de recursos incluirá esse como possível candidato.

Além das informações de *hardware* e *software* dos computadores utilizados nos testes, apresentados na tabela 5.2, as informações de custo e confiança são publicadas com os seguintes valores, como mostrado na tabela 5.3:

Nome da máquina	<i>lee</i>	<i>floyd</i>	<i>chingling</i>
Custo	200	500	250
Confiança do usuário	10	6	9
Confiança exigida a parceiros	9	10	8

Tabela 5.3: Informações publicadas

5.2.4 Fluxo de Execução

Tendo-se o protótipo devidamente configurado, o usuário pode usar de fato a ferramenta para submeter suas aplicações.

A aplicação desenvolvida em (Bittencourt, 2005) consiste de um sistema multiagente que, como tal, tem inúmeras funcionalidades que, no todo, visam dar como resultado final a mais adequada combinação de empresas de um AVI para compor o grupo que formará uma Organização Virtual (OV) para um dado negócio.

No que toca a funcionalidade “geração de escalonamento” (por um agente chamado *Ag-BEManager*), era executado um algoritmo “pesado”, típico em escalonamento, especialmente quando há vários escalonamentos possíveis de serem gerados. Na prática, sem uma infraestrutura de Grades, mesmo que existam computadores ociosos, o usuário é obrigado a possuir um computador potente para que se consiga executar tais aplicações “pesadas”, sem poder fazer uso dos recursos da OV. Portanto, exigindo deste, tanto investimentos em recursos computacionais, como na tomada mais eficiente de partido de estar envolvido numa relação de colaboração com outras empresas.

Portanto, o cenário tratado foi a verificação de o usuário, de forma transparente, poder exportar uma restrição de recursos computacionais ora existente em seu ambiente, para outros recursos que no momento passam executar a aplicação de geração de escalonamento de OVs.

O primeiro passo para submissão é o cadastro da aplicação, fornecendo todos os parâmetros para a sua execução, e os seus requisitos, como mostra a figura 5.21. Além disso é definida a linha de comando (*execution command*) que deve ser executada no nó remoto, como também os arquivos (APIs, arquivos de configuração, etc.) que devem ser transferidos para este nó.

Após salvar o dado cadastro, o usuário⁶ fornece um nome que o identifique. Neste caso, foi dado o nome de “aplicacao01”, sendo gerado dois arquivos XML: o *aplicacao01.rsl* e o *aplicacao01.rlse*, com os detalhes de execução e de consulta, respectivamente.

A partir deste momento, a aplicação está pronta para ser executada na Grade. Para iniciar o processo, o usuário seleciona uma das aplicações cadastradas - no caso a “aplicacao01” - o que ativa a execução do avaliador do recurso (classe *GridNecessityTester*). O avaliador verifica que o estado atual do computador (*chिंगling*) do usuário, e conclui que, no momento, não há recursos suficientes para executar a aplicação; o computador encontrava-se com o seu processador ocupado em certa de 45%, e apenas 60MB de memória disponível.

Tomando-se a decisão de usar a Grade, o usuário deve primeiro autenticar o certificado de segurança, através do fornecimento de sua senha, como é apresentado na figura 5.22. A

⁶Caso o usuário não possua conhecimentos suficientes para fazer a especificação dos requisitos da aplicação, este serviço pode ser solicitado a uma pessoa com conhecimentos da área

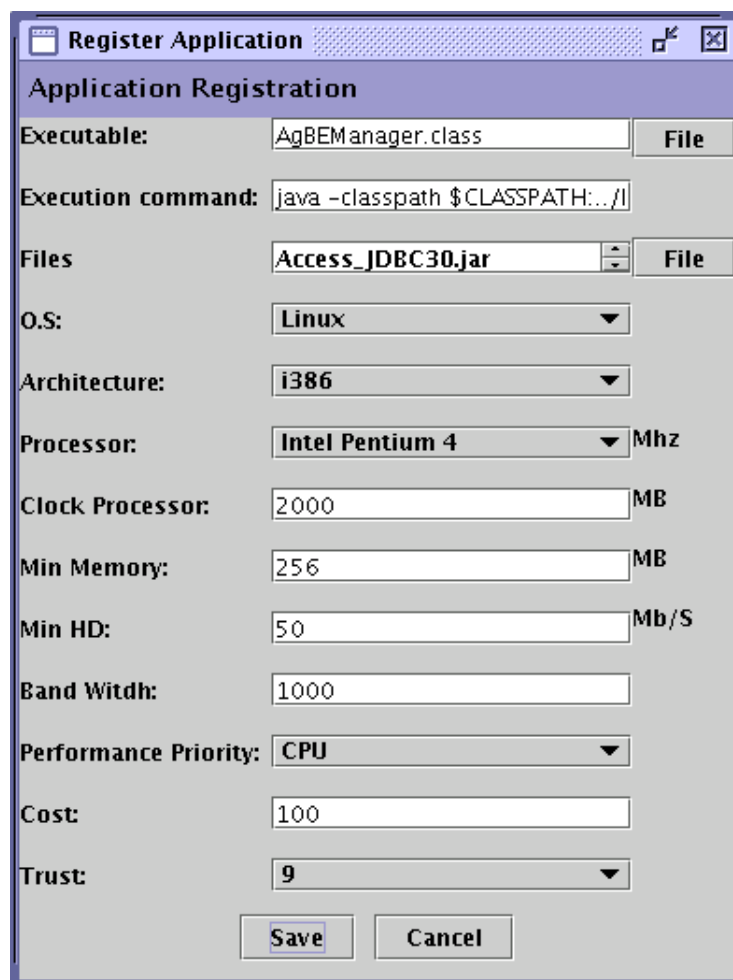
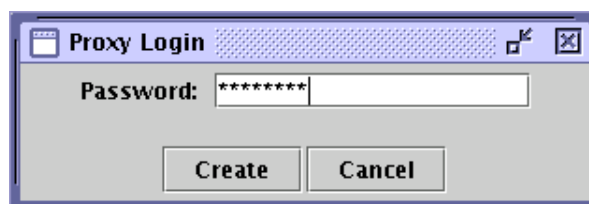


Figura 5.21: Requisitos da aplicação

autenticação leva à criação do *proxy*, que por configuração *default*, tem validade de 24 horas.

Figura 5.22: Criação do *proxy*

Após criado o *proxy*, é apresentada ao usuário a tela para submissão de sua aplicação 5.23. O usuário deve informar o(s) arquivo(s) com os dados de entrada utilizados no processamento da aplicação, caso seja necessário. Para esta aplicação, foi selecionado o arquivo *EV.xml*, o qual possui várias configurações de OVs a serem avaliadas pela aplicação, que selecionará a que obtiver o maior conceito. No caso, optar-se-ia pela composição e escalonamento de OV cuja função de qualidade foi a maior de todas: 20.519. Além disso, este arquivo contém as métricas pelas quais as empresas devem ser avaliadas, assim como os pesos de cada métrica, os pesos dos níveis de escala de cada métrica, e o peso de cada ítem para o sucesso da OV.

O usuário deve também selecionar qual a prioridade na busca de recursos. Para um primeiro teste, o prioridade selecionada foi o **desempenho** (*performance*). Ao iniciar a execução (botão *execute*), o protótipo faz uma busca pelo recurso de melhor desempenho ao serviço de busca de recursos (*Resource Finder Service*), localizado na máquina *lee*. Selecionado o recurso, a aplicação é imediatamente submetida. Após a execução, o usuário tem o resultado apresentado na área de texto na parte inferior da tela. Uma opção também possível, caso seja(m) gerado(s) arquivo(s) como o resultado do processamento, é que esse(s) seja(m) baixado(s) na máquina do usuário via ftp (classe *GridFTP*).

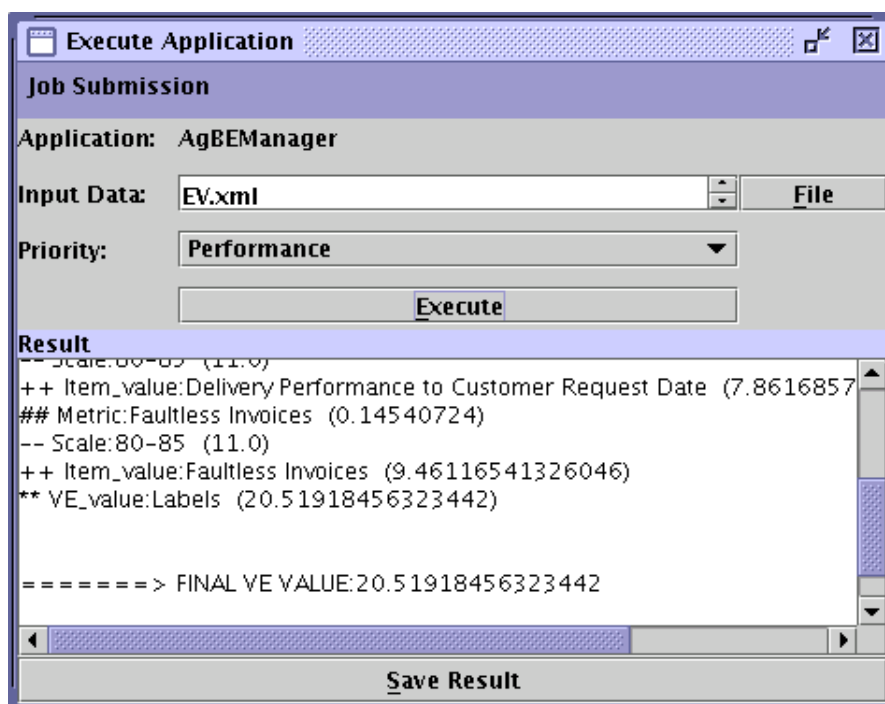


Figura 5.23: Submissão da aplicação

Para se ter idéia de como funcionam os diferentes critérios de seleção, a execução da mesma aplicação foi feita considerando-se as três diferentes prioridades. Como pode ser observado na tabela 5.4 que resume o resultado das buscas. Quando considerado o critério **custo**, nenhum recurso foi selecionado. Isso ocorreu pois o requisito da aplicação - custo menor ou igual a 100,00 - não foi atendido. Já no critério de confiança, a máquina *lee* foi a única a atender aos requisitos da aplicação (confiança maior ou igual a 9).

Nome da máquina	<i>lee</i>	<i>floyd</i>	<i>chingling</i>
Custo	_____	_____	_____
Confiança	selecionado	_____	_____
Desempenho	_____	selecionado	_____

Tabela 5.4: Informações publicadas

5.3 Resultados, Dificuldades e Problemas Encontrados

Vários problemas/dificuldades foram encontrados no processo de instalação, configuração e utilização da plataforma Globus. Os primeiros meses deste trabalho foram realizados na versão 3.0 do *Globus Toolkit*, que era a no momento disponível. No entanto, esta versão apresentava uma série de problemas e instabilidades, o que dificultou um melhor rendimento. A sua instalação apresentou incompatibilidades em algumas distribuições Linux, como a Mandrake, por exemplo. A inicialização do *container* frequentemente apresentava alguns erros, assim como também a submissão. Em abril de 2004 foi lançada a versão 3.2, mais estável e mais completa. No início também apresentou alguns problemas, sendo as suas correções reportadas no *site* ou na lista de discussão do projeto. Contudo, ainda não foi dada uma solução a uma importante falha no serviço de informação, e dificilmente será dada, pois os atuais esforços passaram a se concentrar na versão 4.0 da plataforma. O erro ocorre no momento da agregação de alguns SDEs ao *Index Service*. Neste trabalho, o erro foi considerado como da plataforma pois, além de ser reportado na lista de discussão oficial por diversos usuários, este não se manifestava constantemente. De forma geral, em questão de robustez da plataforma, este foi o maior problema enfrentado. Aproximadamente seis meses foram necessários para se ter um domínio do Globus (incluindo estudos, instalação, configuração e testes iniciais), para que então se pudesse começar a desenvolver algo de fato.

Quanto à implementação, apesar de dispor de uma vasta API (Globus e CoG), a documentação é bastante deficiente. Tutoriais, como o disponível em (Sotomayor, 2004)⁷ e pela área de desenvolvimento da IBM (IBM, 2005), foram de grande utilidade nos primeiros passos de implementação. Nas versões mais recentes do GT e CoG, foi despendido um maior investimento no tocante à documentação.

Os resultados se mostraram satisfatórios do ponto de vista de prover facilidades e transparência ao usuário. A ferramenta apresentou uma grande praticidade, principalmente quanto a etapa de configuração, já que essa envolve uma maior quantidade de passos. O fato de abstrair do usuário do grande número de configurações, concentrando-as em uma única ferramenta, permite ao usuário uma adesão mais rápida e simples ao, ainda complexo, ambiente de Grade. Na submissão, o usuário pode rapidamente modificar os requisitos e prioridades de suas aplicações, obtendo os recursos que mais lhe forem adequados. A restrição de acesso ao recurso a partir da avaliação do confiança oferecida tornou-se uma tarefa simples para uma funcionalidade de grande importância no domínio das OVs.

Contudo, o protótipo ainda apresenta algumas limitações, como por exemplo: a busca de recursos poderia ser mais flexível, permitindo que o usuário possa adicionar e/ou remover parâmetros na consulta (por exemplo, o usuário poderia decidir que o modelo do processador

⁷Este tutorial, criado por um pesquisador da área de Computação em Grades, acabou por se tornar oficial na *Globus Alliance*

não é importante e remover este parâmetro da consulta); o fornecimento dos requisitos da aplicação poderiam ser feitos de forma mais detalhada, assim como um dicionário de termos (relacionados a recursos computacionais) mais completo. Aspectos mais relacionados à concorrência de execução e à segurança, apesar de poderem trazer maiores ganhos globais, não depende do que foi proposto. O primeiro porque cabe às aplicações propriamente ditas serem projetadas para fazer uso/tirar vantagem da Computação em Grade. O segundo porque é a própria plataforma de Grade que oferece serviços de segurança. Evidentemente que outros serviços de segurança poderiam ser desenvolvidos, mas isto não é o foco da proposta desta dissertação.

A avaliação mais relevante sobre o protótipo desenvolvido não deve ser pautada fundamentalmente em parâmetros quantitativos, clássicos na engenharia de *software*. Tendo em vista os objetivos propostos e foco/motivação essencial deste trabalho, o fato de propiciar a “qualquer” usuário a possibilidade fazer uso de computação em Grade, fácil e transparentemente, tende a contribuir significativamente para a melhoria dos níveis de competitividade das empresas. Também de melhor usufruir da colaboração *per se* existente quando empresas fazem parte de uma Organização Virtual em termos de recursos computacionais ociosos e, assim, sem necessitar de investimentos desnecessários em infraestruturas para poder executar as cada vez mais complexas e “pesadas” aplicações empresariais.

Capítulo 6

Conclusões

O trabalho propôs uma abordagem para permitir uma utilização flexível e transparente de recursos computacionais que estão ociosos para se sobrepor a restrições de *hardware* e *software* existentes ao se desejar executar aplicações. Esta execução é feita remotamente através do suporte de uma plataforma de Grade (Globus) e de uma interface cliente, que tanto deixa isto transparente para o usuário, como também permite fazer um uso mais racional dos recursos disponíveis. A arquitetura definida reuniu as funcionalidades essenciais para efetivamente se fazer uso de Grades computacionais. A consideração aos problemas semânticos existentes em tais ambientes, apesar de tratado ainda de forma simples, permitiu uma busca/seleção de recursos mais eficiente. Observou-se que membros de Organizações Virtuais (OVs) podem se beneficiar tornando os seus recursos disponíveis para os demais parceiros, e que métricas qualitativas mais ricas podem ser acrescentadas para uma seleção mais inteligente de recursos computacionais. A realçar a utilização da questão da *confiança*, crítica em OVs.

O protótipo desenvolvido ofereceu elementos para conclusões preliminares de que a Computação em Grade é uma tecnologia praticável a ser aplicada no problema da crescente complexidade dos sistemas empresariais e que pode ser útil a micro, pequenas e médias empresas na diminuição dos seus investimentos em tecnologia da informação. Apesar da dificuldade para se compreender completamente a plataforma Globus, enfrentando problemas de instabilidade e por várias vezes a falta de uma documentação mais completa por parte desta, o protótipo pode ser desenvolvido com sucesso, disponibilizando todas as funcionalidades apresentadas. A sua usabilidade mostrou-se simples, permitindo uma rápida instalação/configuração da parte cliente da infraestrutura. Isto é claramente observado quando ao invés da criação/edição de alguns arquivos e da execução de uma sequência de extensas linhas de comando, o usuário lida apenas com uma interface gráfica, onde pouco há o que se fazer.

Devido a uma infraestrutura reduzida de computadores (pela indisponibilidade de uma quantidade maior de máquinas e também pela complexidade de se configurá-las/mantê-las),

não foi possível realizar testes com aplicações (paralelas) que fizessem uso de grande número de recursos computacionais, que deixasse ainda mais claro toda a potencialidade da tecnologia de Grades. Pelo mesmo motivo, nos testes não foram utilizados escalonadores comuns em uma infraestrutura de Grades, como Condor-G ou o PBS. No entanto, pelo fato do Globus prover interfaces para uso destes escalonadores através de serviços, em teoria, o protótipo adequar-se-ia com uma simples configuração na URL para o *gerenciador de recursos* e acrescentando-se alguns parâmetros nos requisitos da aplicação. Por outro lado, o objetivo maior da proposta não era uma melhoria de desempenho, e sim a possibilidade de um usuário/empresa poder fazer uso de recursos computacionais ociosos existentes, sem o qual não teria (ou com muitas dificuldades) como resolver o seu problema. Ainda, visou tirar maior proveito de uma colaboração existente dentro de uma OV.

Apesar de ser um limitante vigente do Globus, a arquitetura/protótipo desenvolvidos requerem que os provedores de recursos computacionais sejam baseados em sistemas operacionais Unix (e seus derivados), fazendo com que recursos de ambientes Windows não possam ser contemplados. No entanto, clientes Windows são aceitos sem maiores problemas.

Contudo, apesar das grandes promessas, é importante ter em mente que Grade não é a solução para todos os problemas de desempenho, e nem uma “bala de prata” que faz com que qualquer aplicação execute sensivelmente mais rápida, sem a necessidade de maiores esforços, ou até mesmo em um investimento em *software* e *hardware*. Nem toda aplicação é adequada para ser executada em uma Grade. E alguns tipos, nem mesmo, podem ser paralelizados. A configuração de uma Grade pode afetar fortemente o desempenho, a confiança e a segurança da infraestrutura computacional de uma empresa (Foster e Kesselman, 2004). Como isso, a suas utilização/aplicação dentro de uma organização deve ser bastante ponderada antes de ser realmente efetivada, avaliando-se a real necessidade em tal investimento.

Também, como demonstração de resultados efetivos desta dissertação, foram obtidas duas publicações em congressos internacionais. A primeira, ocorreu no *38th CIRP Internal Seminar on Manufacturing Systems* em maio deste ano em Florianópolis-SC. A segunda, no *6th IFIP Working Conference on Virtual Enterprises*, a ser apresentada no final de setembro deste ano em Valência na Espanha.

6.0.1 Sugestões para Trabalhos Futuros

Para trabalhos futuros propõe-se um maior investimento na solução dos problemas de semântica, ainda tão forte em ambientes de Grades, como por exemplo através do desenvolvimento de uma ontologia mais completa. Conforme apresentado nos trabalhos correlatos (seção 3.9), várias pesquisas neste sentido vêm sendo feitas. A arquitetura aqui proposta poderia fazer uso das idéias apresentadas em (Heine et al., 2004), que propõe a utilização de uma ontologia “descentralizada”, onde os catálogos de recursos são consultados e distribuídos via

rede P2P. Desta forma, cada usuário não precisaria ter necessariamente a mesma ontologia, sendo estas ontologias completada/melhoradas a medida que o conhecimento fosse distribuído pela rede.

Também como trabalho futuro é interessante a aplicação da arquitetura-cliente proposta em outras plataformas de Grade (e.g Condor, MyGrid, etc.). Na medida que estas tentativas de integração fossem sendo feitas, a arquitetura sofreria ajustes no sentido de torná-la o mais abrangente possível no tocante a suportar outras plataformas. Até o momento, pouco vem sendo desenvolvido em Grades para ambientes Windows. No entanto, caso estes sejam desenvolvidos baseados nos padrões OGSA/OGSI (ou OGSA/WSRF), é esperada uma fácil integração com plataformas que também os sigam, como o Globus. Assim sendo, seria de grande valia investir numa adaptação da arquitetura/protótipo para tais plataformas, possibilitando ainda mais uma grande quantidade de usuário a fazerem uso de recursos ociosos na rede.

Apêndice A

A.1 GWSDL do Serviço *SystemInformation*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <definitions name="SystemInfoService"
3   targetNamespace="http://www.globus.org/namespaces/2004/02/prototipo/
4     SystemInfoService_sd"
5   xmlns:tns="http://www.globus.org/namespaces/2004/02/prototipo/SystemInfoService_sd"
6   xmlns:data="http://www.globus.org/namespaces/2004/02/prototipo/SystemInfoService_sd/
7     SystemInfoSDE"
8   xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
9   xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
10  xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
11  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
12  xmlns="http://schemas.xmlsoap.org/wsdl/">
13 <import location="../../ogsi/ogsi.gwsdl"
14   namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
15 <import location="SystemInfoSDE.xsd"
16   namespace="http://www.globus.org/namespaces/2004/02/prototipo/SystemInfoService_sd/
17     SystemInfoSDE"/>
18 <types>
19 <xsd:schema targetNamespace="http://www.globus.org/namespaces/2004/02/prototipo/
20   SystemInfoService_sd"
21   attributeFormDefault="qualified"
22   elementFormDefault="qualified"
23   xmlns="http://www.w3.org/2001/XMLSchema">
24 <xsd:element name="publishInformation">
25   <xsd:complexType/>
26 </xsd:element>
27 <xsd:element name="publishInformationResponse" type="tns:publishInformationResponse">
28   <xsd:complexType name="publishInformationResponse"/>
29 </xsd:element>
30 <xsd:element name="setBandWidth" type="tns:setBandWidth">
31   <xsd:complexType name="setBandWidth">
32     <xsd:sequence>
33       <xsd:element name="value" type="xsd:int"/>

```

```
33     </xsd:sequence>
34   </xsd:complexType>
35 </xsd:element>
36 <xsd:element name="setBandWidthResponse" type="tns:setBandWidthResponse">
37   <xsd:complexType name="setBandWidthResponse"/>
38 </xsd:element>
39 <xsd:element name="setCost" type="tns:setCost">
40   <xsd:complexType name="setCost">
41     <xsd:sequence>
42       <xsd:element name="value" type="xsd:int"/>
43     </xsd:sequence>
44   </xsd:complexType>
45 </xsd:element>
46 <xsd:element name="setCostResponse" type="tns:setCostResponse">
47   <xsd:complexType name="setCostResponse"/>
48 </xsd:element>
49 <xsd:element name="setTrust" type="tns:setTrust">
50   <xsd:complexType name="setTrust">
51     <xsd:sequence>
52       <xsd:element name="value" type="xsd:int"/>
53     </xsd:sequence>
54   </xsd:complexType>
55 </xsd:element>
56 <xsd:element name="setTrustResponse" type="tns:setTrustResponse">
57   <xsd:complexType name="setTrustResponse"/>
58 </xsd:element>
59 <xsd:element name="setPartnerTrust" type="tns:setPartnerTrust">
60   <xsd:complexType name="setPartnerTrust">
61     <xsd:sequence>
62       <xsd:element name="value" type="xsd:int"/>
63     </xsd:sequence>
64   </xsd:complexType>
65 </xsd:element>
66 <xsd:element name="setPartnerTrustResponse" type="tns:setPartnerTrustResponse">
67   <xsd:complexType name="setPartnerTrustResponse"/>
68 </xsd:element>
69 <xsd:element name="setUrl" type="tns:setUrl">
70   <xsd:complexType name="setUrl">
71     <xsd:sequence>
72       <xsd:element name="value" type="xsd:string"/>
73     </xsd:sequence>
74   </xsd:complexType>
75 </xsd:element>
76 <xsd:element name="setUrlResponse" type="tns:setUrlResponse">
77   <xsd:complexType name="setUrlResponse"/>
78 </xsd:element>
79 </xsd:schema></types>
80 <gwsdl:portType name="SystemInfoPortType" extends="ogsi:GridService
81   ogsi:NotificationSource">
82   <operation name="publishInformation">
83     <input message="tns:PublishInformationInputMessage"/>
84     <output message="tns:PublishInformationOutputMessage"/>
```

```

84     <fault name="Fault" message="ogsi:FaultMessage"/>
85 </operation>
86 <operation name="setBandWidth">
87     <input message="tns:SetBandWidthInputMessage"/>
88     <output message="tns:SetBandWidthOutputMessage"/>
89     <fault name="Fault" message="ogsi:FaultMessage"/>
90 </operation>
91 <operation name="setCost">
92     <input message="tns:SetCostInputMessage"/>
93     <output message="tns:SetCostOutputMessage"/>
94     <fault name="Fault" message="ogsi:FaultMessage"/>
95 </operation>
96 <operation name="setTrust">
97     <input message="tns:SetTrustInputMessage"/>
98     <output message="tns:SetTrustOutputMessage"/>
99     <fault name="Fault" message="ogsi:FaultMessage"/>
100 </operation>
101 <operation name="setPartnerTrust">
102     <input message="tns:SetPartnerTrustInputMessage"/>
103     <output message="tns:SetPartnerTrustOutputMessage"/>
104     <fault name="Fault" message="ogsi:FaultMessage"/>
105 </operation>
106 <operation name="setUrl">
107     <input message="tns:SetUrlInputMessage"/>
108     <output message="tns:SetUrlOutputMessage"/>
109     <fault name="Fault" message="ogsi:FaultMessage"/>
110 </operation>
111 <sd:serviceData name="SystemInfoData"
112     type="data:SystemInfoDataType"
113     minOccurs="1"
114     maxOccurs="1"
115     mutability="mutable"
116     modifiable="true"
117     nillable="false">
118 </sd:serviceData>
119 </gwsdl:portType>
120 </definitions>

```

A.2 GWSDL do Serviço *TrustInformationProvide*

```

1
2 <?xml version="1.0" encoding="UTF-8"?>
3 <definitions name="TrustInformationProviderService"
4     targetNamespace="http://www.globus.org/namespaces/2004/02/prototipo/
5         TrustInformationProviderService"
6     xmlns:tns="http://www.globus.org/namespaces/2004/02/prototipo/
7         TrustInformationProviderService"
8     xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
9     xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"

```



```

8   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9   xmlns="http://schemas.xmlsoap.org/wsdl/"
10
11  <import location="../../ogsi/ogsi.gwsdl"
12    namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
13
14  <types>
15  <xsd:schema targetNamespace="http://www.globus.org/namespaces/2004/02/prototipo/"
16    TrustInformationProviderService"
17    attributeFormDefault="qualified"
18    elementFormDefault="qualified"
19    xmlns="http://www.w3.org/2001/XMLSchema">
20    <xsd:element name="getTrust">
21      <xsd:complexType>
22        <xsd:sequence>
23          <xsd:element name="userId" type="xsd:string"/>
24        </xsd:sequence>
25      </xsd:complexType>
26    </xsd:element>
27    <xsd:element name="getTrustResponse">
28      <xsd:complexType>
29        <xsd:sequence>
30          <xsd:element name="trust" type="xsd:int"/>
31        </xsd:sequence>
32      </xsd:complexType>
33    </xsd:element>
34  </xsd:schema></types>
35  <gwsdl:portType name="TrustInformationProviderPortType" extends="ogsi:GridService">
36    <operation name="getTrust">
37      <input message="tns:GetTrustInputMessage"/>
38      <output message="tns:GetTrustOutputMessage"/>
39      <fault name="Fault" message="ogsi:FaultMessage"/>
40    </operation>
41  </gwsdl:portType>
42 </definitions>

```

A.3 GWSDL do Serviço *ResourceFinder*

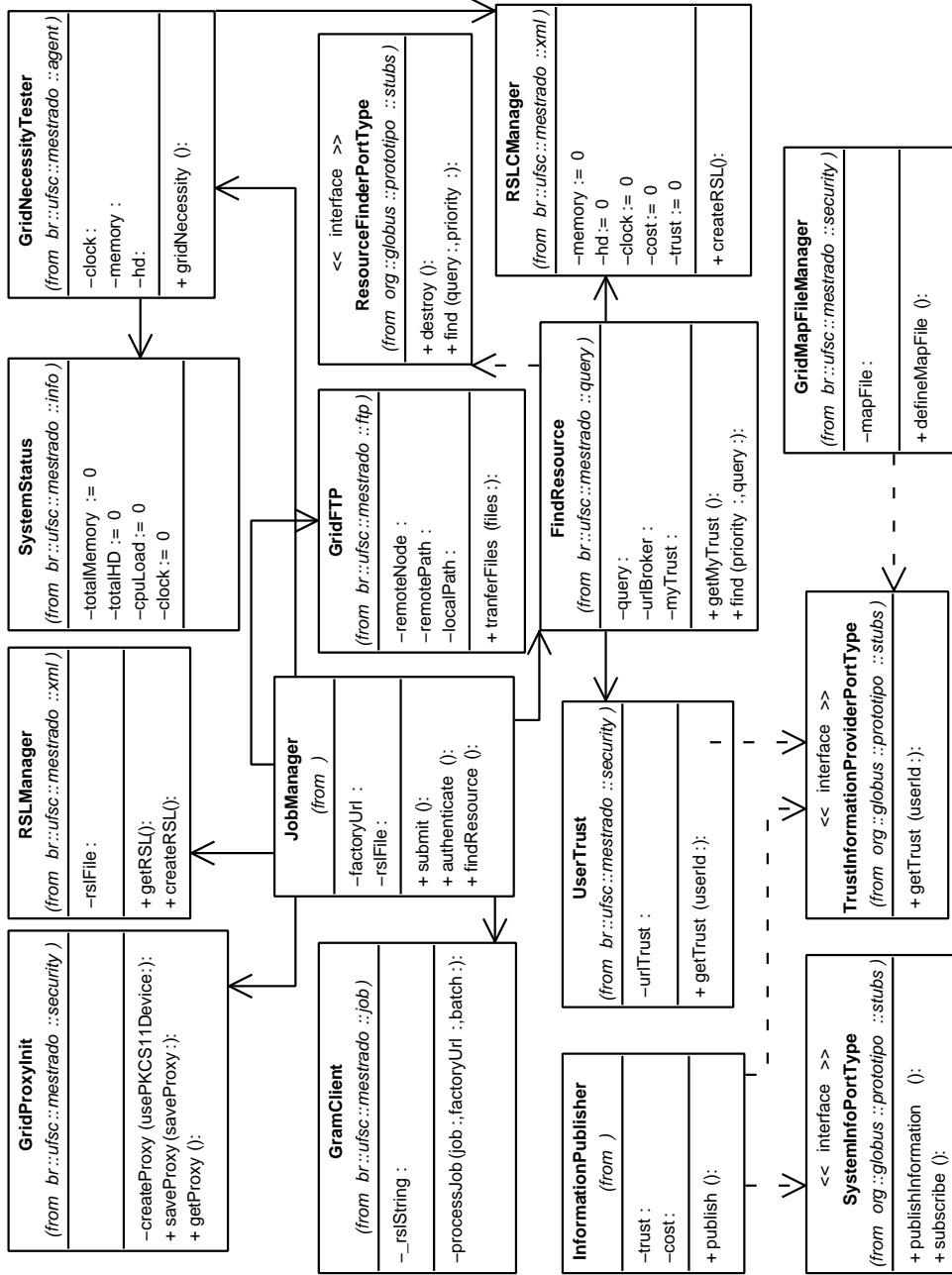
```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <definitions name="ResourceFinderService"
3    targetNamespace="http://www.globus.org/namespaces/2004/02/prototipo/"
4    ResourceFinderService"
5    xmlns:tns="http://www.globus.org/namespaces/2004/02/prototipo/ResourceFinderService"
6    xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
7    xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
8    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9    xmlns="http://schemas.xmlsoap.org/wsdl/"
10  <import location="../../ogsi/ogsi.gwsdl"
11    namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>

```

```
11 <types><xsd:schema targetNamespace="http://www.globus.org/namespaces/2004/02/prototipo/  
12 ResourceFinderService"  
13 attributeFormDefault="qualified"  
14 elementFormDefault="qualified"  
15 xmlns="http://www.w3.org/2001/XMLSchema">  
16 <xsd:element name="find">  
17 <xsd:complexType>  
18 <xsd:sequence>  
19 <xsd:element name="query" type="xsd:string"/>  
20 </xsd:sequence>  
21 <xsd:sequence>  
22 <xsd:element name="priority" type="xsd:string"/>  
23 </xsd:sequence>  
24 <xsd:sequence>  
25 <xsd:element name="performancePriority" type="xsd:string"/>  
26 </xsd:sequence>  
27 </xsd:complexType>  
28 </xsd:element>  
29 <xsd:element name="findResponse">  
30 <xsd:complexType><xsd:sequence>  
31 <xsd:element name="result" type="xsd:string"/>  
32 </xsd:sequence></xsd:complexType>  
33 </xsd:element>  
34 </xsd:schema></types>  
35 <gwsdl:portType name="ResourceFinderPortType" extends="ogsi:GridService">  
36 <operation name="find">  
37 <input message="tns:FindInputMessage"/>  
38 <output message="tns:FindOutputMessage"/>  
39 <fault name="Fault" message="ogsi:FaultMessage"/>  
40 </operation>  
41 </gwsdl:portType>  
</definitions>
```

A.4 Diagrama de Classes



Referências Bibliográficas

- Abramson, D., Buyya, R., e Giddy, J. (2002). A computational economy for grid computing and its implementation in the nimrod-g resource broker. *Future Gener. Comput. Syst.* v. 18, n. 8, p. 1061–1074.
- Afgan, E. (2004). Role of resource broker in the grid. Em *Proceedings of the 42nd annual Southeast regional conference*, página 299.
- Afsarmanesh, H. (2004). D21.1: Characterization of Key Components, Features, and Operating Principles of the VBE. Em *ECOLEAD - European Collaborative Networked Organizations Leadership Initiative*, páginas 37–63. UNINOVA, Portugal.
- Afsarmanesh, H. e Camarinha-Matos, L. M. (1997). Federated information management for cooperative virtual organizations. Em *DEXA '97: Proceedings of the 8th International Conference on Database and Expert Systems Applications*, páginas 561–572, London, UK. Springer-Verlag.
- Aktas, M. S., Pierce, M., , e Fox, G. C. (2004). Designing ontologies and distributed resource discovery services for an earthquake simulation grid. Em *GGF11 Semantic Grid Applications Workshop*, Honolulu.
- Andrade, N., Cirne, W., Brasileiro, F., e Roisenberg, P. (2003). Ourgrid: An approach to easily assemble grid with equitable resource sharing. Em *9th Workshop on Job Scheduling Strategies for Parallel Processing*, Seattle, USA.
- Argonne, I. F., Jennings, N. R., e Kesselman, C. (2004). Brain meets brawn: Why grid and agents need each other. Em *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, página 8.
- Assunção, M. (2004). *Implementação e Análise de uma Arquitetura de Grids de Agentes para a Gerência de Redes e Sistemas*. Dissertação (mestrado em ciência da computação), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.
- Baker, M. e Buyya, R. (1988). Cluster computing: The commodity supercomputing. *Journal of Software - Practice & Experience*. v. 1, n. 1.

- Berman, F., Fox, G., e Hey, T., editors (2003). *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons.
- Berstis, V. e Ferreira, L. (2002). Fundamentals of Grid Computing - IBM RedPaper.
- Bifulco, A. (2005). D41.2: PVC Objectives and Life Cycle Elements. Em *ECOLEAD - European Collaborative Networked Organizations Leadership Initiative*, páginas 47–54. UNINOVA, Portugal.
- Bittencourt, F. (2005). *A Systematic Approach For Partners Search And Selection In Virtual Enterprises Using The Ahp Method And The Scor Model*. Dissertação (mestrado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.
- Brooke, J., Fellows, D., Garwood, K., e Goble, C. (2004). Semantic matching of grid resource descriptions. Em *2nd European Across-Grids Conference (AxGrids 2004)*, Cyprus.
- Brooke, J. e Garwood, K. (2003). Interoperability of grid resource descriptions. Em *GGF9 Semantic Grid Workshop*, Chicago, USA.
- Bruck, J., Dolev, D., Ho, C.-T., e Strong, R. (1995). Efficient message passing interface (mpi) for parallel computing on clusters of workstations. Em *SPAA '95: Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, páginas 64–73, New York, NY, USA. ACM Press.
- Bultje, R. e Wilk, J. (1998). Taxonomy of Virtual Organisations, based on definitions, characteristics and typology. v. 2, n. 3, p. 431–447.
- Buyya, R. (1999). *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Buyya, R. (2002). *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. Ph.d. thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia.
- Buyya, R., Abramson, D., e Giddy, J. (2000). Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. Em *HPC ASIA '2000*, China. IEEE CS Press, USA.
- Calder, B., Chien, A. A., Wang, J., e Yang, D. (2005). The entropia virtual machine for desktop grids. Em *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, páginas 186–196, New York, NY, USA. ACM Press.
- Camarinha-Matos, L. M. e Afsarmanesh, H. (1999). The virtual enterprise concept. Em *PRO-VE '99: Proceedings of the IFIP TC5 WG5.3 / PRODNET Working Conference*

- on Infrastructures for Virtual Enterprises*, páginas 3–14, Deventer, The Netherlands, The Netherlands. Kluwer, B.V.
- Camarinha-Matos, L. M. e Afsarmanesh, H. (2004a). *Collaborative Networked Organizations: A research agenda for emerging business models*. Kluwer Academic, Norwell, MA, USA.
- Camarinha-Matos, L. M. e Afsarmanesh, H. (2004b). *On Emerging Technologies for VO*. In: *Collaborative Networked Organizations: A research agenda for emerging business models*, páginas 207–224. Kluwer Academic.
- Camarinha-Matos, L. M. e Afsarmanesh, H. (2004c). *Support Infrastructure For New Collaborative Forms*. In: *Collaborative Networked Organizations: A research agenda for emerging business models*, páginas 175–192. Kluwer Academic.
- Camarinha-Matos, L. M. e Afsarmanesh, H. (2004d). The Emerging Discipline of Collaborative NetWorks. Em *PRO-VE'2004 - 5th IFIP Working Conference on Infrastructures for Virtual Enterprises, 2004, Toulouse. Virtual Enterprises and Collaborative Networks*, páginas 3–16, Toulouse. Kluwer Academic.
- Cancer@Home (2005). Compute Against Cancer. <http://www.computeagainstcancer.org/>, último acesso em : Abril de 2005.
- Cannataro, M. (2003). Knowledge discovery and ontology-based services on the grid. Em *GGF9 Semantic Grid Workshop*, Chicago, USA.
- Catlett, C. (2003). Standards for grid computing: Global grid forum. *Journal of Grid Computing*. v. 1, n. 1, p. 3–7.
- Chetty, M. e Buyya, R. (2002). Weaving computational grids: How analogous are they with electrical grids? *Computing in Science and Engg*. v. 4, n. 4, p. 61–71.
- Chien, A. A. (2002). Architecture of the entropia distributed computing system. Em *IPDPS '02: Proceedings of the 16th International Symposium on Parallel and Distributed Processing*, página 55.2, Washington, DC, USA. IEEE Computer Society.
- Cirne, W. (2002). Grids computacionais: Arquiteturas, tecnologias e aplicações. Em *Anais do Terceiro Workshop em Sistemas Computacionais de Alto Desempenho*, Espírito Santo, Brasil.
- Cirne, W. e Santos-Neto, E. (2005). Grids computacionais: Da computação de alto desempenho a serviços sob demanda. Em *Mini-curso no 23º Simpósio Brasileiro de Redes de Computadores*, Ceará, Brasil.
- CoG (2005). Commodity Grid Kits. <http://www.cogkit.org/>, último acesso em : Maio de 2005.

- Condor (2005). The Condor Project: High Throughput Computing. <http://www.cs.wisc.edu/condor/>, último acesso em: Abril de 2005.
- Corcho, O. e Gomez-Perez, A. (2000). A roadmap to ontology specification languages. Em *EKAW 2000: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, páginas 80–96, London, UK. Springer-Verlag.
- Costa, L. B., Feitosa, L., Araújo, E., Mendes, G., Coelho, R., Cirne, W., e Fireman, D. (2004). Mygrid: A complete solution for running bag-of-tasks applications. Em *22º Simpósio Brasileiro de Redes de Computadores (SBRC 2004)*, Rio Grande do Sul, Brasil.
- Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Maguire, T., Snelling, D., e Tuecke, S. (2004). From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution.
- Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., e Tuecke, S.. v. .
- Dahan, M. e Boisseau, J. R. (2001). The gridport toolkit: A system for building grid portals. Em *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01)*, página 216, Washington, DC, USA. IEEE Computer Society.
- Dantas, M. A. R., Allemand, J. N. C., e Passos, L. B. C. (2002). An evaluation of Globus and Legion software environments. Em *Workshop on Grid Computing of the Scholl the High Energy Physics*, Rio de Janeiro, Brazil.
- Dantas, M. R. (2003). *Grids Computacionais - Fundamentos, Ambientes e Experiências, em Computação em Cluster*. Brasport.
- DataSynapse (2005). GridServer - Virtualization: The Transformation of Enterprise IT.
- De Roure, D. e Hendler, J. (2004). E-science: the grid and the semantic web. *IEEE Intelligent Systems*. v. 19, n. 1, p. 65–71.
- De Roure, D., Jennings, N., e Shadbolt, N. (2003). The semantic grid: A future e-science infrastructure. Em Berman, F., Fox, G., e Hey, A. J. G., editors, *Grid Computing - Making the Global Infrastructure a Reality*, páginas 437–470. John Wiley and Sons Ltd.
- Decker, S., Erdmann, M., Fensel, D., e Studer, R. (1998). Ontobroker: Ontology based access to distributed and semi-structured information. Em *DS-8: Proceedings of the IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics- Semantic Issues in Multimedia Systems*, páginas 351–369, Deventer, The Netherlands, The Netherlands. Kluwer, B.V.

- Digipede (2005). Digipede Network . <http://www.digipede.net/>, último acesso em : Maio de 2005.
- ECOLEAD (2004). ECOLEAD - European Collaborative Networked Organizations Leadership Initiative. <http://www.ecolead.org/>, último acesso em : Julho de 2005.
- Erwin, D. W. e Snelling, D. F. (2001). Unicore: A grid computing environment. Em *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, páginas 825–834, London, UK. Springer-Verlag.
- Farquhar, A., Fikes, R., e Rice, J. (1997). The ontolingua server: a tool for collaborative ontology construction. *Int. J. Hum.-Comput. Stud.* v. 46, n. 6, p. 707–727.
- Fensel, D., Horrocks, I., Harmelen, F., McGuinness, D., e Patel-Schneider, D. (2001). Oil: Ontology infrastructure to enable the semantic web. v. 16, n. 2.
- FigthAIDS@Home (2005). Folding@Home Distributed Computing. <http://www.stanford.edu/group/pandegroup/folding/>, último acesso em : Abril de 2005.
- Filos, E. e Devine, M. (2000). Virtual teams and the organizational grapevine. Em *Second Working Conference On Infrastructure For Virtual Organizations*, páginas 413–424, Florianópolis, Brasil.
- Fitzgerald, S. (2001). Grid information services for distributed resource sharing. Em *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01)*, página 181, Washington, DC, USA. IEEE Computer Society.
- Fleischman, A. M. P. (2004). *Ontologias Aplicadas À Descrição De Recursos Em Grids Computacionais*. Dissertação (mestrado em ciência da computação), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.
- Foster, I. e Iamnitchi, A. (2003). On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. Em *2nd Intl Workshop on Peer-to-Peer Systems (IPTPS '03)*, páginas 118–128, Berkeley, CA. Springer-Verlag.
- Foster, I. e Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*. v. 11, n. 2, p. 115–128.
- Foster, I. e Kesselman, C., editors (1999). *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Foster, I. e Kesselman, C., editors (2004). *The grid 2: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

- Foster, I., Kesselman, C., Nick, J., e Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration - OGSA. Em *Global Grid Forum*.
- Foster, I., Kesselman, C., e Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*. v. , páginas 200–222.
- Freitas, F. (2002). *Sistemas multiagentes cognitivos para recuperação e extração integradas de informação na web*. Tese (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.
- Frey, J., Tannenbaum, T., Foster, I., Livny, M., e Tuecke, S. (2001). Condor-G: A computation management agent for multi-institutional grids. Em *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC)*, páginas 7–9, San Francisco, California.
- Gava, T. e Menezes, C. (2003). Especificação de software baseada em ontologias. Em *III Escola de Informática*, páginas 167–205.
- Genome@Home (2005). Genome@Home. <http://www.stanford.edu/group/pandegroup/genome/>, último acesso em : Abril de 2005.
- GGF (2005). Global Grid Forum. <http://www.gridforum.org>, último acesso em: Março de 2005.
- Gil, Y., Deelman, E., Blythe, J., Kesselman, C., e Tangmunarunkit, H. (2004). Intelligence and grids: Workflow planning and beyond. *IEEE Intelligent Systems*. v. 19, n. 1, p. 26–33.
- Globus (2000). Resource Specification Language (RSL). http://www-unix.globus.org/toolkit/docs/3.2/gram/ws/developer/mjs_rsl_s%chema.html, último acesso em : Maio de 2004.
- Globus (2005). Globus Alliance. <http://www.globus.org>, último acesso em: Março de 2005.
- Goble, C. (2000). Supporting Web based Biology with Ontologies. Em *Third IEEE ITAB00*, páginas 384–390, Arlington, VA.
- Goble, C. e De Roure, D. (2002a). The grid: an application of the semantic web. *ACM SIGMOD Record*. v. 31, n. 4, p. 65–70.
- Goble, C. e De Roure, D. (2002b). The semantic web and grid computing. Em Kashyap, V. e Shklar, L., editors, *Real World Semantic Web Applications*, volume 92 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

- Graham, S., Karmarkar, A., Mischkinsky, J., Robinson, I., e Sedukhin, I. (2005). Web Services Resource 1.2 (WS-Resource).
- Graham, S. e Treadwell, J. (2004). Web Services Resource Properties 1.2 (WS-ResourceProperties).
- Grid, A. (2005). Access Grid Project: a global community. <http://www.accessgrid.org/>, último acesso em : Junho de 2005.
- GridBus (2004). The GridBus Project - Grid Computing and Distributed Systems. <http://www.gridbus.org/broker/>, último acesso em : Janeiro de 2005.
- GridSphere (2004). GridSphere Project - A GridSphere Portal Framework. <http://www.gridsphere.org/>, último acesso em : Maio de 2005.
- Grimshaw, A. S., Humphrey, M. A., e Natrajan, A. (2004). A philosophical and technical comparison of legion and globus. *IBM J. Res. Dev.* v. 48, n. 2, p. 233–254.
- Grimshaw, A. S. e Wulf, W. A. (1996). Legion: flexible support for wide-area computing. Em *EW 7: Proceedings of the 7th workshop on ACM SIGOPS European workshop*, páginas 205–212, New York, NY, USA. ACM Press.
- Grimshaw, A. S., Wulf, W. A., e Team, C. T. L. (1997). The legion vision of a worldwide virtual computer. *Commun. ACM*. v. 40, n. 1, p. 39–45.
- GRIP (2005). Grid Interoperability Project. <http://www.grid-interoperability.org/>, último acesso em : Maio de 2005.
- Gruber, T. R. (1993). Towards Principles for the Design of Ontologies Used for Knowledge Sharing. Em Guarino, N. e Poli, R., editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers.
- Guarino, N., Masolo, C., e Vetere, G. (1999). Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*. v. 14, n. 3, p. 70–80.
- Harth, A., Decker, S., He, Y., Tangmunarunkit, H., e Kesselman, C. (2004). A semantic matchmaker service on the grid. Em *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, páginas 326–327, New York, NY, USA. ACM Press.
- Heine, F., Hovestadt, M., e Kao, O. (2004). Towards ontology-driven p2p grid resource discovery. Em *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, páginas 76–83, Washington, DC, USA. IEEE Computer Society.

- Horrocks, I., Patel-Schneider, P. F., e van Harmelen, F. (2002). Reviewing the design of daml+oil: an ontology language for the semantic web. Em *Eighteenth national conference on Artificial intelligence*, páginas 792–797, Menlo Park, CA, USA. American Association for Artificial Intelligence.
- Huedo, E., Montero, R. S., e Llorente, I. M. (2004). The GridWay Framework for Adaptive Scheduling and Execution on Grids. *Journal of Software - Practice and Experience*. v. , 34, p. 631–651.
- IBM (2005). Grid computing zone at IBM developerWorks. <http://www-130.ibm.com/developerworks/grid/>, último acesso em : Abril de 2005.
- Infoware, G. (2005). Grid Computing Info Centre (GRID Infoware). <http://www.gridcomputing.com/>, último acesso em : Abril de 2005.
- JDOM (2000). JDOM v1.0 API Specification. <http://www.jdom.org/docs/apidocs/>, último acesso em : Novembro de 2004.
- Jiang, G. e Cybenko, G. (2004). Functional validation in grid computing. *Autonomous Agents and Multi-Agent Systems*. v. 8, n. 2, p. 119–130.
- Joseph, J. e Fellenstein, C., editors (2003). *Grid Computing*. Prentice Hall PTR.
- Kent, R. (1999). Conceptual Knowledge Markup Language: The Central Core. Em *Twelfth Workshop on Knowledge Acquisition, Modeling and Management*, Banff.
- Legion (2004). Legion: A Worldwide Virtual Computer. <http://legion.virginia.edu/>, último acesso em : Abril de 2004.
- Litzkow, M., Livny, M., e Mutka, M. (1988). Condor - a hunter of idle workstations. Em *Proceedings of the 8th International Conference of Distributed Computing Systems*.
- Liu, L. e Meder, S. (2005). Web Services Base Faults 1.2 (WS-BaseFaults).
- Ludwig, S. e van Santen, P. (2002). A grid service discovery matchmaker based on ontology.
- MacGregor, R. M. (1991). Inside the loom description classifier. *SIGART Bulletin*. v. 2, n. 3, p. 88–92.
- Maguire, T. e Snelling, D. (2005). Web Services Service Group 1.2 (WS-ServiceGroup).
- McGuinness, D. L. e van Harmelen, F. (2004). OWL Web Ontology Language Overview. W3C Recommendation, February, 2004. <http://www.w3.org/TR/owl-features/>, último acesso em : Junho de 2005.
- Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., e Xu, Z. (2002). Peer-to-peer computing.

- Moreau, L., Miles, S., Papay, J., Decker, K., e Payne, T. (2003). Publishing semantic descriptions of services. Em *GGF9 Semantic Grid Workshop*, Chicago, USA.
- Motta, E. (1998). An Overview of the OCML Modelling Language. Em *8th Workshop on Knowledge Engineering Methods and Languages (KEML '98)*, Karlsruhe, Germany.
- Mulder, W. e Meijer, G. (2004). Squads: Software development and maintenance on the Grid by means of mobile virtual organizations using adaptatives information services. Em *Proceedings of PRO-VE2004*, France.
- Nimrod (2005). Nimrod. <http://www.csse.monash.edu.au/~nimrod/nimrodg/>, último acesso em : Maio de 2005.
- OASIS (2005). Organization for the Advancement of Structured Information Standards. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf, último acesso em : Abril de 2005.
- O'Brien, P. D. e Nicol, R. C. (1998). Fipa towards a standard for software agents. *BT Technology Journal*. v. 16, n. 3, p. 51–59.
- OGCE (2004). Open Grid Computing Environments Collaboratory. <http://www.ogce.org/>, último acesso em : Maio de 2005.
- OurGrid (2004). OurGrid Project. <http://www.ourgrid.org>, último acesso em: Outubro de 2004.
- Ouzounis, E. K. (2001). *An Agent-Based Platform for the Management of Dynamic Virtual Enterprises*. Ph.d. thesis, Departamento de Eletrônica e Informática, Universidade de Berlin, Berlin, Alemanha.
- Parameswaran, M., Susarla, A., e Whinston, A. B. (2001). P2P Networking: An Information-Sharing Alternative. *Computer*. v. 34, n. 7, p. 31–38.
- Rabelo, R. J. e Camarinha-Matos, L. M. (1996). Towards agile scheduling in extended enterprise. Em *BASYS'96: Proceedings of Balanced Automation Systems II*, páginas 413–422. Chapman & Hall.
- Rabelo, R. J., Klen, A. A. P., e Klen, E. R. (2004). Effective Management of Dynamic and Multiple Supply Chains. *Int. Journal of Networking and Virtual Organizations*. v. , páginas 193–208.
- Rocha, J., Domingues, M., Callado, A., Souto, E., Silvestre, G., Kamienski, C., e Sadok, D. (2004). *Peer-to-Peer: Computação colaborativa na internet*. Em *22º Simpósio Brasileiro de Redes de Computadores*.

- Ruscic, B., Amin, K., Wagstrom, P., Krishnan, S., e Nijssure, S. (2003). A Framework for Building Scientific Knowledge Grids Applied to Thermochemical Tables. *International Journal of High Performance Computing Applications*. v. 17, n. 4, p. 431–447.
- Sampaio, T. (2003). *Extração de Informação Para Busca Semântica Na Web Baseada Em Ontologias*. Dissertação (mestrado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.
- Sandholm, T. e Gawor, J. (2003). Globus Toolkit 3 Core: A Grid Service Container Framework.
- SETI@home (2005). The Search for Extraterrestrial Intelligence. <http://setiathome.ssl.berkeley.edu/>, último acesso em : Abril de 2005.
- Singh, M. P. e Huhns, M. N. (2005). *Service-oriented computing: Semantics, Processes, Agents*, páginas 87–118. John Wiley & Sons, England.
- Sotomayor, B. (2004). The Globus Toolkit 3 Programmer's Tutorial. <http://gdp.globus.org/gt3-tutorial/>, último acesso em : Abril de 2005.
- Srinivasan, L. e Banks, T. (2005). Web Services Resource Lifetime 1.2 (WS-ResourceLifetime).
- Tangmunarunkit, H., Decker, S., e Kesselman, C. (2003). Ontology-based resource matching in the grid - the grid meets the semantic web. Em *1th Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid) at the Twelfth International World Wide Web Conference*.
- Tao, F., Chen, L., Cox, S. J., Shadbolt, N. R., Puleston, C., e Goble, C. A. (2003). Semantic support for grid-enabled design search in engineering. Em *GGF9 Semantic Grid Workshop*, Chicago, USA.
- Thain, D., Tannenbaum, T., e Livny, M. (2002). Condor and the Grid. Em Berman, F., Fox, G., e Hey, T., editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc.
- Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., e Kesselman, C. (2002). Grid service specification.
- Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Maguire, T., Sandholm, T., Vanderbilt, P., e Snelling, D. (2003). Open Grid Services Infrastructure (OGSI) Version 1.0. Em *Global Grid Forum*.
- Venkatakrishnan, S., Kuchhal, M., e Kumar, S. Grid Application Framework for Java (GAF4J). IBM AlphaWorks Technology - Technical White Paper.

- von Laszewski, G., Foster, I., Gawor, J., e Lane, P. (2001). A java commodity grid kit. *Concurrency and Computation: Practice and Experience*. v. 13, n. 8-9, p. 643–662.
- von Laszewski, G., Foster, I. T., e Gawor, J. (2000). Cog kits: a bridge between commodity distributed computing and high-performance grids. Em *Java Grande*, páginas 97–106.
- von Laszewski, G. e Hategan, M. (2004). Grid workflow - an integrated approach.
- von Laszewski, G. e Hategan, M. (2005). Karajan workflow. http://www.cogkit.org/release/4_0_a1/manual/workflow/workflow.html, último acesso em : Maio de 2005.
- W3C (1994). World Wide Web Consortium. <http://www.w3.org/>, último acesso em : Agosto de 2004.
- W3C (1999). XML Path Language (XPath) Version 1.0, W3C Recommendation. <http://www.w3.org/TR/xpath>, último acesso em : Março de 2005.
- W3C (2001a). Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation. <http://www.w3.org/TR/REC-xml>, último acesso em : Agosto de 2004.
- W3C (2001b). XML Schema Part 0: Primer, W3C Recommendation. <http://www.w3.org/TR/xmlschema-0/>, último acesso em : Agosto de 2004.
- Welch, V. (2004). X.509 proxy certificates for dynamic delegation. Em *3rd Annual PKI R&D Workshop*, Gaithersburg.
- WSRF (2005). Web Service - Resource Framework. <http://www.globus.org/wsrf/>, último acesso em : Abril de 2005.