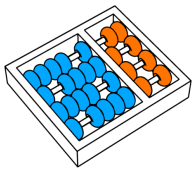
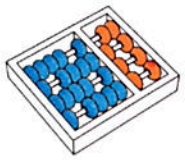


Geraldo Magela da Silva

"MidGov: Middleware para Governo Eletrônico baseado em Grades Computacionais"

CAMPINAS
2013



Universidade Estadual de Campinas
Instituto de Computação

Geraldo Magela da Silva

"MidGov: Middleware para Governo Eletrônico baseado em Grades Computacionais"

Orientador(a): **Prof. Dr. Edmundo Roberto Mauro Madeira**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Geraldo Magela da Silva, sob orientação de Prof. Dr. Edmundo Roberto Mauro Madeira.

Edmundo Madeira

Assinatura do Orientador(a)

CAMPINAS
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

Si38m

Silva, Geraldo Magela, 1985-
MidGov: middleware para governo eletrônico baseado em grades computacionais / Geraldo Magela da Silva. – Campinas, SP : [s.n.], 2013.

Orientador: Edmundo Roberto Mauro Madeira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Governo eletrônico. 2. Grades computacionais (Sistemas de computador). 3. Serviços na Web - Semântica. 4. Arquitetura orientada a serviços (Computação). 5. Middleware. I. Madeira, Edmundo Roberto Mauro,1958-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: MidGov: middleware for electronic government based on grid computing
Palavras-chave em inglês:
Electronic government
Grid computing (Computer systems)
Semantic Web
Service-oriented architecture (Computer science)
Middleware
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Jó Ueyama
Islene Calciolari Garcia
Data de defesa: 03-10-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 03 de outubro de 2013, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Jó Ueyama
ICMC / USP

Profª. Drª. Islene Calciolari Garcia
IC / UNICAMP

Prof. Dr. Edmundo Roberto Mauro Madeira
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

MidGov: Middleware para Governo Eletrônico baseado em Grades Computacionais

Geraldo Magela da Silva[1]

3 de outubro de 2013

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (*Orientador*)
- Prof. Dr. Jó Ueyama
  Instituto de Ciências Matemáticas e de Computação - USP
- Profª. Drª. Islene Calciolari Garcia
  Instituto de Computação - UNICAMP
- Prof. Dr. Bruno Richard Schulze (Suplente)
  Laboratório Nacional de Computação Científica - LNCC
- Prof. Dr. Luiz Fernando Bittencourt (Suplente)
  Instituto de Computação - UNICAMP

[1]O aluno recebeu recursos financeiros do CNPq (processo 134416/2008-2) entre 06/2008 e 03/2010.

Resumo

Agências governamentais ao redor do mundo estão realizando grandes investimentos na utilização de Tecnologia da Informação e Comunicação em suas atividades. Essa tendência, conhecida como Governo eletrônico, impulsiona grande demanda por pesquisas cujo foco principal é o desenvolvimento de aplicações destinadas a um governo mais transparente e colaborativo. Aplicações para esse tipo de cenário introduzem uma série de desafios que precisam ser enfrentados, incluindo maior interoperabilidade entre sistemas, escalabilidade, questões de segurança, entre outros. Nesse sentido, o paradigma de Arquitetura Orientada a Serviços (SOA) apresenta-se como uma interessante proposta para mitigar a heterogeneidade dos serviços prestados pelas diversas entidades envolvidas. Além disso, computação em grade pode ser considerada uma solução promissora para aplicações de *middleware* em Governo eletrônico, graças à sua alta capacidade de armazenamento e processamento, além de sua recente orientação a serviços, tornando-a uma poderosa ferramenta para aplicações intra-domínio. Considerando esses desafios, este trabalho propõe uma plataforma para aplicações de Governo eletrônico em sistemas em grades utilizando serviços de suporte fornecidos pelo Globus Toolkit 4 (GT4) no contexto da *Web* Semântica. O trabalho inclui a implementação de um protótipo do *middleware* e sua validação através de um cenário de aplicação.

Abstract

Government agencies around the world are making large investments in the use of Information and Communication Technology in their activities. This trend, known as electronic government, drives a demand for research focused on development of applications aimed at a more transparent and collaborative government. Applications for this type of scenario pose a series of challenges to be faced, including greater interoperability between systems, scalability, and security issues, among others. In this sense, the paradigm of Service-Oriented Architecture (SOA) presents itself as an interesting proposal to mitigate the heterogeneity of services provided by various involved entities. Furthermore, grid computing can be considered a promising solution for middleware applications in e-Government due to its high storage and processing capacity, and also its recent service orientation, making it a powerful tool for intra-domain applications. Considering these challenges, this dissertation proposes a platform for e-Government applications on grid computing, using the support services provided by the Globus Toolkit 4 (GT4) in the context of the Semantic Web. The work includes the implementation of a middleware prototype and its validation through an application scenario.

Ao meu pai Afonso Ferreira, à minha mãe Maria Neuza (em memória)
e aos meus irmãos Maria José, Nazareth e Afonso Ermelindo.

Agradecimentos

Primeiramente a **Deus**, e a toda minha familia, principalmente minha mãe, **Maria Neuza** (em memória), e meu pai, **Afonso Ferreira**. São os verdadeiros responsáveis por tudo.

Um agradecimento muito especial a minha esposa **Rita Elena**. Muito obrigado pelo carinho, motivação, apoio e por sempre acreditar que daria tudo certo.

Muito obrigado ao meu orientador, Prof. **Edmundo Madeira**, pelas palavras, críticas, sugestões, pela paciência, e claro, pelo apoio incondicional. Sem você não seria possível!

Muitíssimo obrigado também aos amigos que se fizeram presente durante toda a trajetória, como o Luciano, Milton, Andrei Braga, Cesar Chaves, Neumar Malheiros, Flávio Kubota, Pedro Henrique, Rafael Curi, Rafael, Thiago Genez, Thiago Pedroso, Gilberto, Fabrício, Ana Rézio entre outros. Obrigado pela amizade. Muito obrigado também ao Sr. Gerônimo e ao Carlos Froldi (o que inclui o Prof. Edmundo) pelas divertidas conversas sobre futebol na sala do café.

Obrigado ao Carlos Senna, Luiz Bittencourt, Daniel Batista, Ivo Santos e Neumar Malheiros, pela força, suporte e orientação.

Muito obrigado à galera da república CaiPinto, que substituíram minha familia durante boa parte dessa trajetória.

Muito obrigado a todos os amigos do LRC (Laboratório de Redes de Computadores).

Não poderia deixar de agradecer ao Instituto de Computação da Unicamp e ao CNPq que permitiu total foco no trabalho através da Bolsa de Mestrado oferecida.

Enfim, assim fiz boas e consistentes amizades. Muito obrigado por tudo!

Sumário

Lista de Acrônimos

URI *Uniform Resource Identifier* 16

W3C *World Wide Web Consortium* 8

WSDL *Web Services Description Language* 9

WSRF *Web Services Resource Framework* 2

WS-BPEL *Web Service Business Process Execution Language* 11

XML *eXtensible Markup Language* 9

Lista de Códigos

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Nas últimas décadas, grandes evoluções tecnológicas são perceptíveis principalmente no campo de processamento e distribuição de dados, onde máquinas totalmente centralizadas foram substituídas por redes de computadores interconectadas, nas quais a carga de trabalho pode ser processada em conjunto [Tanenbaum, 2002]. Como resultado desse progresso tecnológico, principalmente em conectividade, a demanda por Tecnologias da Informação e Comunicação (TIC) voltada a ambientes integrados e eficientes vem crescendo nos últimos anos.

Nesse contexto, entidades governamentais estão realizando grandes esforços para a utilização e desenvolvimento de tecnologias de informação para uso em suas atividades, processo conhecido como Governo Eletrônico, ou e-Governo. De modo geral, e-Governo é a chave para encurtar a distância entre cidadãos e seus políticos, permitindo sua participação em decisões governamentais e reforçando a democracia [Garcia et al., 2004]. Embora seja um fenômeno recente, já se transformou em requisito básico no setor público, obtendo crescentes investimentos em diversos países ao redor do mundo [UNPAN, 2010]. Sua principal demanda está vinculada a aplicações [Zhang and Wang, 2008; Agarwal et al., 2010; Ma, 2010] visando ambientes mais eficientes, transparentes e colaborativos ao governo.

Aplicações para esse tipo de cenário introduzem alguns desafios que devem ser enfrentados do ponto de vista computacional, incluindo maior interoperabilidade, escalabilidade e questões de segurança. Nesse sentido, o emergente paradigma de Arquitetura Orientada a Serviços (SOA) apresenta-se como uma forte proposta para mitigar a heterogeneidade, cujo princípio fundamental é a divisão de funcionalidades em módulos básicos, acessíveis através de tecnologias de Serviços *Web* [Curbera et al., 2003].

A abordagem orientada a serviços ganhou grande aceitação em práticas corporativas, sendo vista como uma forma adequada para sobrepor problemas de interação entre suas unidades funcionais, que podem resultar em longo tempo de espera, baixa produtividade

e redundância no cumprimento das tarefas [Muehlen, 2004].

Grades computacionais fornecem alto poder de processamento e armazenamento em ambientes heterogêneos. Sua construção pode ser realizada através do *Globus Toolkit* desenvolvido pela Globus Alliance, o qual inclui um conjunto de serviços que endereçam questões de segurança, descoberta de informações, gestão de recursos, gestão de dados, comunicação, detecção de falhas e portabilidade. Em sua quarta versão o *toolkit* da Globus implementa a especificação *Open Grid Services Architecture* (OGSA) [Grimshaw et al., 2009] com *Web Services Resource Framework* (WSRF) [Graham et al., 2006], promovendo uma completa convergência entre sistemas em grades e SOA. Tal convergência tem transformado grades computacionais em uma poderosa e interessante ferramenta para aplicações que envolvam serviços intra-domínios [Sotomayor and Childers, 2006].

1.1 Motivação

Ambientes colaborativos apresentam-se em várias tarefas administrativas do setor público, o qual inclui interações entre os diferentes órgãos públicos e até mesmo serviços de terceiros, como empresas privadas. No entanto, nesse contexto os ambientes colaborativos são ainda pouco explorados do ponto de vista eletrônico [KBSt, 2008]. Portanto, características encontradas em sistemas em grade, aliado às facilidades disponíveis no *Globus Toolkit*, tornam esses sistemas um caminho promissor na construção de serviços para e-Governo. Todavia, para que uma grade construída através do *Globus Toolkit* versão 4 possa suportar aplicações de e-Governo, alguns desafios precisam ser sobrepostos.

Considerando aspectos tecnológicos, a especificação SAGA (*Standards and Architectures for eGovernment Applications*) cita alguns requisitos chave para aplicações de e-Governo: interoperabilidade, reusabilidade, abertura, escalabilidade e segurança. Para mais detalhes veja Subseção 2.1.2. Baseado nesses requisitos, o trabalho desenvolvido por [Santos and Madeira, 2007] mostra que o *Globus Toolkit 4.0* (GT4) endereça requisitos como interoperabilidade em nível técnico, reusabilidade e abertura. Todavia questões relacionadas a interoperabilidade em nível semântico, escalabilidade e segurança precisam ser consideradas para que grade computacional possa suportar adequadamente aplicações de governo eletrônico.

Embora não evidente, os desafios tecnológicos que um *middleware* para Governo Eletrônico precisa enfrentar vão estritamente além daqueles encontrados em *middlewares* genéricos e voltados a organizações privadas, principalmente em aspectos relacionados a interoperabilidade, transparência e o trato das informações, conforme ilustrado por *Davies et. al.* em [Davies et al., 2007]. Dessa forma, para mapearmos essas peculiaridades e tornar sistemas em grade mais adequados às aplicações de e-Governo, propomos uma plataforma idealizada como um *middleware*.

1.2 Objetivos e contribuições

Sendo assim, este trabalho tem como objetivo propor um *middleware* para aplicações de Governo eletrônico sobre o ambiente fornecido pelas grades computacionais. A plataforma utiliza um orquestrador para a execução de *workflows*, chamado *Grid Process Orchestration* (GPO) [Senna and Madeira, 2007], e facilidades da infraestrutura fornecida pelo *Globus Toolkit* 4, além de recursos da *Web* Semântica.

Utilizamos a arquitetura do *middleware* apresentado em [Santos and Madeira, 2010] como base para nosso trabalho. Esta dissertação apresenta as seguintes contribuições:

- Revisão bibliográfica referente a pesquisas na área de e-Governo, grades computacionais e propostas de e-Governo utilizando sistemas em grades;
- A infraestrutura do *middleware* proposto aliado a um motor de orquestração sobre grades computacionais;
- A estratégia de utilização de serviços do *Globus Toolkit* 4 para suprir requisitos de segurança, transparência entre as entidades envolvidas; e
- Implementação de um protótipo da infraestrutura e sua avaliação através da aplicação de dois cenários presentes em atividades do governo.

1.3 Estrutura da dissertação

Os capítulos desta dissertação estão organizados da seguinte forma:

- O Capítulo 2 contém uma descrição dos conceitos necessários ao entendimento desta dissertação. Definimos os paradigmas de Governo eletrônico, SOA e *Web* Semântica. Características sobre grades computacionais e algumas de suas tecnologias são também apresentadas.
- O Capítulo 3 apresenta uma ampla revisão bibliográfica sobre pesquisas em Governo eletrônico, grades computacionais e a combinação entre esses dois paradigmas.
- O Capítulo 4 detalha a solução proposta, sua infraestrutura e componentes.
- O Capítulo 5 apresenta uma avaliação da solução proposta a partir da implementação de dois cenários de aplicação para e-Governo.
- O Capítulo 6 conclui este trabalho, destacando as potenciais extensões em trabalhos futuros.

Capítulo 2

Fundamentos Básicos

Neste capítulo abordamos conceitos básicos sobre as tecnologias e paradigmas envolvidos nesta dissertação. Inicialmente, a Seção 2.1 apresenta o Governo eletrônico e seus aspectos. Em seguida, discutimos o paradigma de Arquitetura Orientada a Serviços (SOA), suas abordagens tecnológicas e definimos conceitualmente o termo *middleware* na Seção 2.2. Características da *Web* Semântica, sua aplicação em Serviços *Web* e padrões utilizados são assuntos da Seção 2.3. Na Seção 2.4, apresentamos uma definição sobre a tecnologia de Grade computacional, além de uma breve discussão sobre grades e Computação em nuvem mais voltado ao contexto de aplicações para Governo eletrônico. A Seção 2.5 apresenta a infraestrutura e facilidades do *Globus Toolkit* 4 e GPO. Resumo e considerações finais sobre o capítulo são o assunto da Seção 2.6.

2.1 Governo Eletrônico

Governo eletrônico, ou e-Governo (do inglês, *e-Government*), tem como objetivo a prestação e/ou obtenção de informações, serviços ou produtos através de meios eletrônicos. Engloba serviços não apenas para o cidadão, mas também para si mesmo [Zweers and Planqué, 2001]. Em outras palavras, e-Governo busca otimizar atividades do governo através das vantagens observadas na utilização das TICs.

Durante a década de 90, práticas de e-Governo começaram a surgir em países industrializados da época, motivadas principalmente pelo contínuo crescimento nas dimensões do setor público e pela maior interação com o cidadão. Atualmente seus benefícios vão muito além do aumento em produtividade, transformando-se em uma significativa solução para ampliar a comunicação entre políticos e cidadão, rotulada por muitos como poderosa ferramenta para fortalecer a democracia. Alguns de seus benefícios incluem a redução burocrática, aumento da credibilidade dos administradores, maior transparência e participação do cidadão em decisões do setor público, além da redução de custos e a eliminação

de fronteiras geográficas [UNPAN, 2010].

Sua evolução pode ser equiparada aos diferentes níveis de sofisticação em TIC, a qual tem motivado uma contínua demanda por novos sistemas em e-Governo. No que se refere a entrega de serviços *online*, observam-se estágios bem definidos, porém não totalmente disjuntos, no avanço de aplicações em e-Governo, que incluem [Hiller and Bélanger, 2001; UNPAN, 2010]:

- **Informação:** nesse estágio, páginas *Web* do governo fornecem informações estáticas referentes a serviços oferecidos ao cidadão, departamentos, leis, e demais informações de interesse do cidadão;
- **Comunicação bidirecional:** maior interação com o cidadão é alcançada com a utilização de serviços *online* conectados a banco de dados, com facilidades para buscas e a utilização de correio eletrônico pela administração pública;
- **Transações:** amplamente utilizadas na atualidade, o governo passa a disponibilizar vasta quantidade de serviços mais complexos e completos, dispensando a presença do cidadão frente aos departamentos públicos. Inclui transações financeiras e não-financeiras (pagamento de multas, certidão de nascimento, entre outras); e
- **Serviços Conectados:** em um estágio mais avançado, e-Governo atende às necessidades do cidadão através de um portal único, com serviços integrados (interoperáveis), personalizados, envolvendo o cidadão em decisões governamentais com a utilização da Web 2.0 [Lewis, 2006], da descrição semântica de informações via Web 3.0 [Hendler, 2009], e seguindo uma abordagem de governo centrada no cidadão, que por sua vez se transforma em um cliente do governo.

2.1.1 Aplicações do Governo Eletrônico

De forma geral, e-Governo basicamente se relaciona dos seguintes modos: interações entre governo e cidadão, G2C (*Government-to-Citizens*); negócios entre Governo e o setor empresarial, G2B (*Government-to-Business*); e relações envolvendo agências quaisquer do governo, G2G (*Government-to-Government*). Modelos G2C e G2B fazem parte do grupo de serviços com interações externas ao governo (*front-office*), enquanto G2G engloba atividades com interações internas do governo (*back-office*) [KBSt, 2008]. Tais relações são mapeadas na prestação e oferecimento de serviços que podem ser classificados nas seguintes categorias:

- Serviço eletrônico (*e-Service*): abrange serviços *online* aos cidadãos (G2C). Alguns exemplos incluem cadastro de pessoa física e serviços de busca;

- Arquivamento eletrônico (*e-Archiving*): todos os serviços voltados à gestão de informações, como arquivos digitais, documentos públicos e dados do cidadão;
- Negócios eletrônicos (*e-Business*): inclui serviços que interagem no modelo G2B, como por exemplo, imposto sobre pessoa jurídica, contratos eletrônicos (*e-procurement*); e
- Democracia eletrônica (*e-Democracy*): voltada principalmente ao modelo G2C, engloba serviços para prover uma participação mais ampla e mais ativa ao cidadão em decisões do governo, como votação eletrônica e fóruns de discussão.

2.1.2 Requisitos do Governo Eletrônico

Durante o desenvolvimento de aplicações para e-Governo vários requisitos devem ser levados em consideração. A concreta apuração de requisitos varia de acordo com as características individuais das aplicações. Por exemplo, caso uma aplicação de e-Governo tenha uma alta taxa de acesso, características como escalabilidade e disponibilidade podem ser requisitos alvos. Entretanto, alguns requisitos técnicos são fundamentais para o sucesso de qualquer aplicação em e-Governo [KBSt, 2008]. Abaixo descrevemos cada um deles.

- **Interoperabilidade** – refere-se à "*habilidade de duas ou mais entidades se comunicarem e cooperarem independente das diferenças na linguagem de implementação, ambiente de execução ou modelo de abstração utilizado*" [Wegner, 1996]. O objetivo é alcançar o compartilhamento de informações e o trabalho em conjunto entre agências do governo, cidadãos, e parceiros. Sobretudo, a interoperabilidade deve ser garantida em três níveis diferentes:

  1. **Técnico**. Relaciona-se ao simples compartilhamento de informações. Uma linguagem comum para descrição de dados e a definição de rotas de transmissão e protocolos são requisitos para a interoperabilidade técnica;
  2. **Semântico**. Determina que dois sistemas quaisquer derivem as mesmas inferências a partir das mesmas informações. Associado normalmente com o significado dos dados, indica que há um entendimento comum a respeito da forma e conteúdo dos dados sendo transmitidos entre os parceiros na comunicação. Seu sucesso depende do sucesso da camada técnica; e
  3. **Organizacional**. Nesse nível de interoperabilidade, políticas de acesso às informações são utilizadas, as quais determinam quando e por que dados são compartilhados. Em outras palavras, o compartilhamento de dados é coordenado através dos regulamentos da organização. Interoperabilidade organizacional apoia-se no sucesso de interoperação técnica e semântica.

- **Reusabilidade** – requisito que permite o reuso de modelos de dados, experiência entre agências, componentes ou serviços em projetos de e-Governo. Reutilização evita a redundância e reduz o tempo de desenvolvimento;

- **Abertura** – representa a adoção de padrões abertos em aplicações de e-Governo para prover seu sucesso e sua usabilidade a longo prazo. Suas aplicações podem definir e documentar claramente suas interfaces ou serem encapsuladas de forma que permitam a integração de novos sistemas.

- **Escalabilidade** – requisito desejado em alguns sistemas e extremamente importante em outros, como sistemas *Web*. Permite alcançar a eficiência desejada na usabilidade de aplicações mesmo quando o grau de utilização aumenta;

- **Segurança** – devido ao tipo de informação que o governo manipula, questões de segurança devem ser fortemente consideradas. Aplicações de e-Governo devem garantir que informações sejam acessadas, modificadas ou publicadas em acordo com as políticas de segurança predefinidas.

Assim como em ambientes empresariais, diferentes níveis de implementação de TIC são realizados nos vários órgãos do governo de forma independente. A falta de um arcabouço de orientação no desenvolvimento de sistemas, atrelado à utilização de diferentes plataformas e linguagens de programação, torna a integração entre os mesmos uma tarefa extremamente complexa. Dentre as principais estratégias para aumentar o grau de interoperabilidade entre sistemas encontra-se o paradigma SOA.

2.2 Arquitetura orientada a serviços e Middleware

SOA emergiu principalmente em resposta à mudança nos negócios empresariais, na qual empresas totalmente integradas tornaram-se redes empresariais. Trata-se de uma estratégia de TIC baseada em princípios de computação distribuída, onde as funcionalidades implementadas pelas aplicações são constituídas por combinações de serviços. Dessa forma, serviços tornam-se módulos básicos, e a composição destes passa a ser a principal preocupação no desenvolvimento de aplicações [Curbera et al., 2003]. Uma outra definição é fornecida pela *World Wide Web Consortium* (W3C), onde SOA constitui-se de "*um conjunto de componentes que podem ser invocados, e cuja descrição de interface possa ser publicada e descoberta*" [W3C, 2004c].

No cenário atual de aplicações distribuídas, SOA emergiu como a principal estratégia para aumentar o grau de interoperabilidade entre os diversos serviços rodando em sistemas heterogêneos. Nesse contexto, características (como baixo acoplamento, padrões

amplamente utilizados e abertos, entre outras) encontradas em tecnologias de Serviços *Web*, aliadas ao forte suporte fornecido a partir de grandes empresas como IBM, Microsoft e Oracle, motivaram sua rápida adoção em SOA. Assim, blocos básicos (ou serviços) em aplicações SOA são geralmente envolvidos pelas tecnologias presentes na pilha de protocolos dos Serviços *Web*.

Dessa forma, uma aplicação em SOA estende a execução de diferentes unidades funcionais através de interfaces em termos de protocolos e funcionalidades, as quais possibilitam a independência de plataforma e linguagem utilizada. Alguns aspectos em suas soluções incluem: baixo acoplamento, abstração, reusabilidade, autonomia, capacidade de composição e descoberta [Keen et al., 2004; Bichler and Lin, 2006].

2.2.1 Serviços *Web*

Serviços *Web* (do inglês, *Web Services*) fornecem a possibilidade de que novas aplicações possam interagir com aplicações já existentes, sem conhecimento prévio, e que aplicações desenvolvidas em diferentes plataformas sejam compatíveis. Suas especificações são baseadas em padrões abertos, permitem fraco acoplamento, interoperabilidade universal, e possuem mecanismos para descrever, acessar, localizar e identificar componentes de *software* [Russell et al., 2005].

A W3C define Serviço *Web* como "*um sistema de software projetado para suportar interações máquina a máquina sobre a rede, com interface descrita em formato processável, utilizando Web Services Description Language (WSDL), na qual outros sistemas possam interagir de forma prescrita usando mensagens Simple Object Access Protocol (SOAP), normalmente transmitida via Hyper Text Transfer Protocol (HTTP) e serializadas em eXtensible Markup Language (XML)*" [W3C, 2004c].

A Figura 2.1 apresenta a arquitetura de Serviços *Web* e as três especificações iniciais. De forma geral, a descrição da interface é feita em WSDL [Chinnici et al., 2007], que inclui informações sobre como utilizar o serviço, suas funcionalidades, quais requisições e respostas podem ser trocadas, além de sua localização. As interações entre clientes e serviços ocorrem via mensagens SOAP [Mitra and Lafon, 2007] com conteúdo XML. Tais interações fazem parte do passo seguinte à descoberta da respectiva descrição do serviço em WSDL, previamente inserido no *Universal Description Discovery and Integration* (UDDI) [Clement et al., 2004], e que por sua vez possibilita a publicação, descoberta e integração do serviço [Papazoglou and Georgakopoulos, 2003; Curbera et al., 2002].

Para o suporte a arquitetura SOA, a tecnologia de Serviços *Web* fornece uma pilha de protocolos (exposta na Figura 2.2) modular, o que permite a seleção de determinadas tecnologias para compor uma configuração particular. Seus protocolos incluem mecanismos de comunicação, descrição, descoberta e composição de serviços, bem como um conjunto

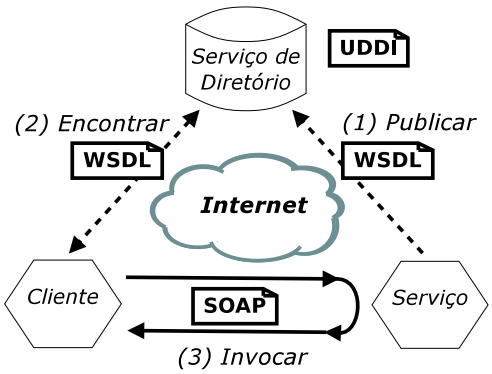


Figura 2.1: Arquitetura de Serviços *Web*

básico de protocolos para qualidade de serviço (QoS) [Curbera et al., 2003].

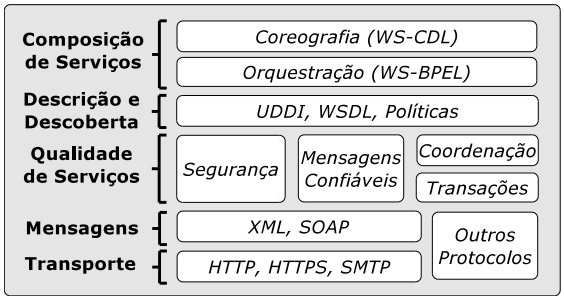


Figura 2.2: Pilha de protocolos dos Serviços *Web*

A crescente aceitação e adoção a SOA usando Serviços *Web* remodelou a forma como empresas conduziam trabalhos em conjunto, representando ganhos em desempenho e redução de gastos [Bichler and Lin, 2006]. Nesse sentido, o modelo inicial de Serviços *Web* retratado na Figura 2.1 se moveu para uma nova fase, onde aplicações distribuídas de larga escala fazem parte do processo [Curbera et al., 2003]. Nesse contexto, enquanto determinado nível de maturidade tem sido alcançado pela adoção de padrões de Serviços *Web*, ainda existem grandes desafios relacionados à sua composição.

2.2.2 Composição de serviços

Mecanismos para composição de serviços visam combinar dois ou mais serviços para que possam atender a requisitos que estão além de suas capacidades individuais [Papazoglou and Georgakopoulos, 2003]. O resultado da composição de serviços pode ser utilizado por uma composição adicional (de modo recursivo) ou mesmo como solução ao consumidor de serviços. Dois modelos de execução são frequentemente aplicados: Orquestração e

Coreografia. Embora possam se sobrepor, definem características específicas quanto a composição de serviços [Peltz, 2003].

- **Orquestração**: descreve as interações em um processo de negócio, incluindo a lógica de negócios, a coordenação e o gerenciamento de serviços (internos e externos), geralmente coordenados por um Orquestrador. Em outras palavras, o Orquestrador trata o fluxo do processo de negócio como um todo, invocando serviços em uma ordem predefinida em nível de mensagens. Neste tipo de composição, cada serviço participante desconhece o processo de negócio, exceto o Orquestrador. Uma linguagem amplamente utilizada para orquestração é a *Web Service Business Process Execution Language* (WS-BPEL) [Jordan and Evdemon, 2007]; e

- **Coreografia**: abordagem na qual os processos são autônomos, tendo responsabilidades iguais no processo de troca de mensagens. Trata-se de uma abordagem onde o controle é distribuído. Comparada à orquestração, a coreografia é mais colaborativa, não existe a figura de um orquestrador e cada serviço envolvido possui conhecimento do processo de negócio. Duas linguagens frequentemente adotadas para especificação de coreografia são a *Web Services Choreography Description Language* (WS-CDL) [Kavantzas et al., 2005] e a *Web Service Choreography Interface* (WSCI) [Arkin et al., 2002].

Uma variedade de especificações tem sido introduzidas para tratar mecanismos de composição de serviços. Algumas destas propostas foram interrompidas, como por exemplo: *Web Services Flow Language* (WSFL) [Leymann, 2001] e XLang da Microsoft, enquanto outras alcançaram maior aceitação na indústria, o que provocou uma convergência (no que se refere à adoção) para tais especificações, como por exemplo WS-CDL, WSCI e WS-BPEL.

WS-CDL

Recentemente proposta e com intuito de substituir a especificação WSCI, WS-CDL [Ross-Talbot and Fletcher, 2006] é uma linguagem desenvolvida pela W3C, baseada em XML, que descreve colaborações par-a-par (*peer-to-peer*), onde cada parceiro é autônomo e não existe uma relação mestre e escravo entre os participantes, ou seja, é totalmente descentralizado. Embora seja uma iniciativa pela *W3C* voltada à padronização em definições de coreografia para Serviços *Web*, WS-CDL ainda não alcançou de fato o *status* de padrão e ampla adoção para esse propósito [Cambronero et al., 2009].

Suas descrições definem o comportamento dos serviços e a troca de mensagens para a realização de um objetivo, independente da plataforma de suporte, modelo ou linguagem de programação utilizada. Assim como WSCI, WS-CDL não abrange a descrição e

execução do *workflow* correspondente a cada composição, que por sua vez, pode utilizar diferentes mecanismos para cada um destes serviços, como por exemplo WS-BPEL.

WS-BPEL

WS-BPEL, ou simplesmente BPEL [Jordan and Evdemon, 2007], foi inicialmente projetada pela IBM, BEA e Microsoft. Foi inspirada a partir de dois padrões anteriores, WSFL (*Web Services Flow Language*) da IBM e Xlang da Microsoft. Atualmente é padronizada pela OASIS (*Organization for the Advancement of Structured Information Standards*). BPEL fornece uma linguagem baseada em XML para descrever o comportamento de um "*Processo de Negócio*" (*workflow*). Sua especificação define a lógica de controle e o fluxo de dados entre Serviços *Web* em um fluxo de trabalho, além de englobar mecanismos para lidar com exceções e falhas de processamento.

No contexto de BPEL, uma composição de serviços é chamada de processo e os serviços participantes são geralmente nomeados de parceiros. As interações entre o processo e seus parceiros são geralmente comunicações convencionais par-a-par (*peer-to-peer*), onde parceiros externos expõem suas operações (pontos de entrada/saída e tipos de dados exigidos através de definições em WSDL) e BPEL define a sequência de trocas de mensagens entre eles (em uma camada acima à WSDL) [Peltz, 2003].

BPEL fornece suporte a descrição de processos executáveis e abstratos. Essa característica permite a separação de aspectos públicos a partir do comportamento interno do processo de negócio. Processos abstratos são explicitamente declarados como abstratos, e descrevem o seu comportamento sem expor detalhes sobre a lógica interna do processo (detalhes da execução). Processos executáveis, por outro lado, definem o comportamento completo do processo de negócio.

Uma vez que os parceiros são identificados, BPEL fornece um conjunto de elementos usados para definir toda a lógica do processo. A estrutura de um processo em BPEL é formada basicamente por três seções: definição dos parceiros que interagem durante o processo de negócio (<*partnerLink*>), definição de variáveis de dados usadas (<*variables*>) e descrição do comportamento do processo (<*process*>). Uma vez que iremos trabalhar com orquestração, BPEL se torna uma tecnologia interessante no contexto do trabalho.

2.2.3 Sistemas de *workflow*

Geralmente utilizado como sinônimo de "processo de negócio", *workflow* (ou fluxo de trabalho) está relacionado com a realização de determinado objetivo através da coordenação entre atividades em concordância com regras já definidas [Hollingsworth, 1995]. Devido principalmente à quantidade de tecnologias relacionadas a *workflow* e às diferentes heranças conceituais, foram evidenciadas divergências em relação às definições de conceitos

e consenso sobre a forma com que os mesmos podem ser utilizados. Uma iniciativa para resolver esse problema e unificar as definições presentes em sistemas de *workflow* foi iniciada pela *Workflow Management Coalition* (WfMC) – organização sem fins lucrativos com o objetivo de desenvolver uma terminologia aliada a padrões para maiores avanços e oportunidades em tecnologias de *workflow* [Muehlen, 2004].

A WfMC define *workflow* como "*a automação do processo de negócio, no todo ou em parte, durante o qual documentos, informações ou tarefas são passadas de um participante para outro, de acordo com um conjunto de regras processuais*" [WfMC, 1999]. Um *workflow* é uma típica representação de processo, onde quaisquer serviços participantes podem ser orquestrados por um sistema, normalmente conhecido por Sistema de Gestão de *Workflows*.

Um sistema de gestão de *workflows* é tipicamente composto por componentes para criação do modelo de *workflow*, funcionalidades para geração de suas instâncias e mecanismos para a execução das mesmas [Hollingsworth, 1995]. Segundo a WfMC um sistema de gestão de *workflows* pode ser definido como "*um sistema que define, gerencia e cria a execução de sistemas de workflows através do uso de software, rodando em um ou mais motores de workflow, que é capaz de interpretar a definição do processo, interagir com os participantes e, quando necessário, recorrer ao uso de ferramentas de TIC e aplicações*" [WfMC, 1999].

De modo geral, sistemas de *workflows* não são triviais, e tipicamente utilizam diferentes tecnologias/paradigmas (como SOA, entre outras), para o seu funcionamento. Por motivos relacionados à abstração, sistemas de gestão de *workflow* são amplamente implementados em uma camada entre as aplicações e a infraestrutura disponível, geralmente conhecida por *middleware*.

2.2.4 Middleware

Em resposta à conectividade, diferentes empresas ou departamentos em organizações estão realizando enorme esforço na entrega integrada de serviços ao consumidor. Na prática, a integração dos diferentes sistemas pode englobar incompatibilidades referentes a protocolos de transporte, sincronização da comunicação entre componentes distribuídos que executam em paralelo, e a heterogeneidade do *hardware*, *software* e dados em diferentes nós [Emmerich et al., 2007]. *Middleware* é uma forma extremamente explorada para simplificar estas tarefas e fornecer determinada abstração durante a programação de aplicações.

Middleware pode ser definido como um *software* localizado em uma camada intermediária (camada do meio, *middle*) entre as aplicações e a plataforma, com o principal objetivo de abstrair a heterogeneidade e a complexidade inerente encontrada, por exemplo, em sistemas distribuídos [Bernstein, 1996]. A Figura 2.3 ilustra esse tipo de ambiente,

onde três nós executando uma aplicação distribuída acessam plataformas heterogêneas conectadas através de um *middleware*.

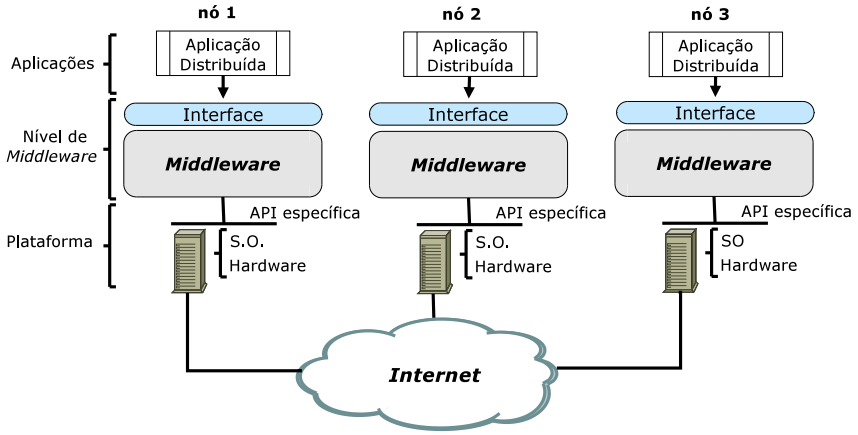


Figura 2.3: Contexto do *middleware*

Embora uma camada adicional penalize o desempenho da aplicação, diversos benefícios são evidenciados a partir da abstração fornecida pelos sistemas de *middleware*. Alguns deles incluem a redução do custo e tempo em desenvolvimento de aplicações, sua fácil evolução, melhor qualidade, portabilidade e interoperabilidade, ocultação de níveis mais baixos, independência de plataforma e linguagem, e reuso de código. De forma geral, *middleware* é uma solução elegante e amplamente adotada na construção de sistemas distribuídos [Emmerich et al., 2007; Bernstein, 1996].

Middleware orientado a serviços

Recentemente, SOA tem se tornado o principal paradigma utilizado em tradicionais aplicações cliente/servidor. Uma de suas principais vantagens é fornecer às aplicações a possibilidade de integrar vários serviços disponíveis e utilizá-los conforme necessário ao invés de implementar todo o serviço desde o início. Quando bem definido e baseado em padrões, SOA pode fortalecer sistemas distribuídos com um sistema flexível e ambiente de fácil desenvolvimento de aplicações [Bichler and Lin, 2006].

Middleware também oferece capacidades similares para integrar e reusar componentes de software. Contudo, ele não suporta a integração de serviços *online* e sob demanda com facilidade. Diversos pesquisadores veem SOA como substituto para *middleware* uma vez que o desenvolvimento de aplicações não depende da implementação detalhada, mas sim principalmente da integração com os componentes disponíveis. Todavia, SOA tem apresentado um drástico crescimento e questões de *design* de projeto e desenvolvimento bem planejado tornaram-se importantes. Como resultado, abordagens utilizando *middleware*

são frequentemente utilizadas para facilitar o desenvolvimento de serviços usando SOA, comumente conhecido como *middleware* orientado a serviços [Qilin and Mintian, 2010; Al-Jaroodi et al., 2010].

Aplicações em *middleware* evoluíram a partir de simples aplicações, como a ocultação de detalhes do banco de dados, para sofisticados sistemas onde importantes informações são manuseadas em sistemas distribuídos [Bernstein, 1996]. Nesse contexto, diversos esforços foram conduzidos com o intuito de endereçar deficiências na construção de arquiteturas de *software* mais flexíveis em diferentes domínios de aplicação [Emmerich, 2000]. Algumas das linhas de pesquisa em que *middleware* é aplicado incluem Computação adaptativa [Sadjadi and McKinley, 2003], Computação Móvel [Mascolo et al., 2002], Computação pervasiva [Schiele et al., 2010], entre outras.

Se por um lado a interoperabilidade tem sido aperfeiçoada com a utilização das tecnologias até então apresentadas, por outro, mecanismos de composição e descoberta de Serviços *Web* são fundamentalmente baseados em combinação sintática, utilizando descrições em WSDL, a qual limita o sucesso dos mesmos. Nesse contexto, *Web* Semântica desenvolve uma importante função, atribuindo significado (sentido) a conteúdos publicados na Internet de modo que sejam perceptíveis ao computador.

2.3 Web Semântica

De forma geral, atualmente a *Internet* é interoperável apenas sintaticamente, com conteúdo projetado apenas para a leitura humana, impossibilitando por exemplo, a discriminação entre informações comerciais e acadêmicas ou mesmo simples relações entre as mesmas. Nesse contexto, a *Web* Semântica apontada como o próximo passo na evolução da *Web* oferece uma pilha de tecnologias para atribuir significado a conteúdos da *Internet*. O seu principal objetivo é permitir que máquinas possam inferir novas informações com o propósito de realizar escolhas mais inteligentes e reduzir a intervenção humana [Berners-Lee et al., 2001; Fensel et al., 2006].

Tim Berners-Lee, inventor da *World Wide Web* (WWW) introduziu a *Web* Semântica em [Berners-Lee et al., 2001], onde a define como "*uma extensão da atual Web, na qual informações recebem um significado bem definido, permitindo que computadores e pessoas trabalhem em cooperação*". Em outras palavras, o entendimento humano é codificado de tal forma que máquinas possam processar e chegar às mesmas conclusões que um ser humano, através do raciocínio lógico [Fensel et al., 2006]. Uma completa visão da *Web* Semântica proposta por Tim Berners-Lee, também conhecida como "bolo em camadas", é apresentada na Figura 2.4 [Berners-Lee, 2007]. Em sua hierarquia de tecnologias, cada camada utiliza capacidades da camada inferior, com intuito de fornecer uma descrição formal de conceitos, termos e relacionamentos em um dado domínio.

A parte inferior no "bolo em camadas" fornece uma base sintática, formada por padrões existentes na *Web*. Conceitos e termos utilizam a sintaxe fornecida pelo XML para estrutura de conteúdos, codificados pelo Unicode, e são identificados de forma única e compartilhada na *Web* através de uma *Uniform Resource Identifier* (URI). As camadas do topo, prova e confiança, ainda não possuem padrões definidos e estão relacionadas principalmente à aplicação [Fensel et al., 2006]. Em camadas intermediárias encontram-se tecnologias relacionadas às anotações semânticas, que serão apresentadas adiante e possuem maior relevância no presente trabalho.

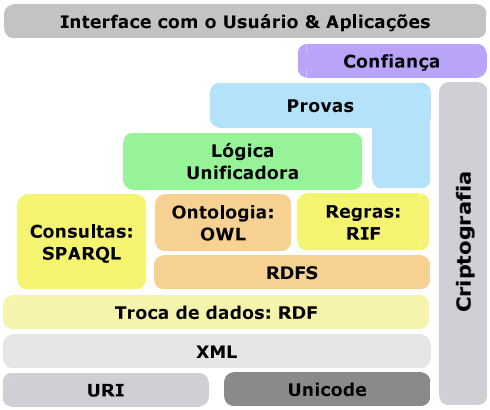


Figura 2.4: Pilha de tecnologias da Web Semântica

No ambiente fornecido pela *Web* Semântica, as ontologias realizam papel importante, facilitando a interoperação entre informações de recursos através de ligações entre si. Sua utilização sobre a *Web* apresenta grande potencial para sobrepor problemas relacionados ao compartilhamento, reuso de conhecimento e a integração de informação.

2.3.1 Ontologias

O termo ontologia é originário da filosofia. Em uma perspectiva mais moderna, a partir da década de 70, a noção de ontologia transformou-se em tópicos de subáreas da computação [Jepsen, 2009]. Assim, diversas definições podem ser encontradas na literatura, todavia, uma delas tem prevalecido diante as demais. Introduzida por Thomas Gruber em [Gruber, 1993], onde uma ontologia é uma "*especificação formal e explícita de uma conceitualização compartilhada*". Em outras palavras, a descrição de Gruber define ontologia como uma especificação clara e precisa, entendível por máquinas (através de linguagem) e cooperativa em um determinado domínio.

Ontologias permitem o compartilhamento do conhecimento de um domínio a partir de sua comunidade. Uma típica ontologia possui uma taxonomia (ciência da classificação), a qual define axiomas de conceitos (relevantes ao domínio), e relacionamentos entre os

mesmos [Gruber, 1993]. Em [Fensel et al., 2006], é apresentado um exemplo de taxonomia retratada na Figura 2.5, onde conceitos descrevem objetos do mundo real, seus relacionamentos definem sub-conceitos, referidos pelo termo "*é um*" , e a captura do conceito pelo objeto é realizada através do termo "*instância de*". Dessa forma, considerando o exemplo, uma máquina pode inferir que Luiz é uma pessoa, sabendo que Luiz é um estudante e que todo estudante é uma pessoa.

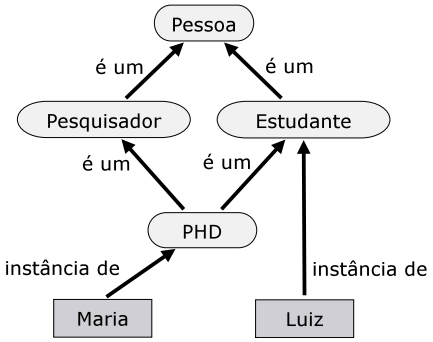


Figura 2.5: Exemplo de Taxonomia

O significado para humanos é perceptível através das definições dos termos, normalmente descritos em linguagem natural. Já suas relações formalmente definidas são voltadas principalmente ao entendimento das máquinas. A utilização de ontologias estende a capacidade das funcionalidades na *Web*, permitindo que agentes incluam regras de inferência, associação de estruturas de conhecimento, buscas mais precisas, reduzindo sua ambiguidade, tornando suas decisões mais inteligentes e ampliando a automação em seus processos [Berners-Lee et al., 2001; Fensel et al., 2006].

Para realizar a definição de uma ontologia é preciso uma linguagem formal que a torne legível para as máquinas. Nesse sentido, diversas linguagens de ontologias para a *Web* foram propostas, muitas delas com determinadas restrições e alto poder de expressividade para a descrição do conhecimento. Dentre as linguagens de ontologias, *Resource Description Framework* (RDF) [Manola and Miller, 2004] e *Web Ontology Language* (OWL) [W3C, 2004b] são definidas como padrão pela W3C, incluídas no núcleo das tecnologias da *Web* Semântica (veja a Figura 2.4) e possuem grande influência sobre as demais linguagens.

2.3.2 Serviços na *Web* Semântica

Relativamente recente, Serviços na *Web* Semântica (*Semantic Web Service*) originam-se da convergência entre a *Web* Semântica e o paradigma SOA. A ideia principal é mapear facilidades da *Web* Semântica no contexto dos Serviços *Web*, tecnologia utilizada com

frequência em aplicações orientadas a serviços. Serviços *Web* possuem um conjunto de tecnologias amplamente aceito para suportar a interação de serviços sobre a *Web*, porém, sem especificar o que o serviço faz ou como interagir com o mesmo de forma legível para a máquina.

Para sobrepor a essa limitação, *Semantic Web Service* (SWS) permite a utilização de descrições sobre o que o serviço faz e como ele faz, além de outras informações, as quais permitem tratar desafios na automação e interoperação de serviços na *Web* [Preist, 2007; Klusch, 2008]. Seus benefícios têm motivado o surgimento de diferentes iniciativas. Uma das principais especificações nessa área, documentada em recentes submissões da W3C, é a OWL-S [W3C, 2004a].

OWL-S

Semantic Markup for Web Services (OWL-S) [W3C, 2004a] é basicamente a precursora da ideia de SWS. Originária a partir da especificação DAML-S (*DARPA Agent Markup Language for Services*), OWL-S é construída no topo de OWL e combina um conjunto de ontologias OWL inter-relacionadas voltadas à definição de termos em aplicações orientadas a serviços. OWL-S adiciona uma camada aos padrões de Serviços *Web* para suportar sua descoberta, invocação, composição, interoperação e monitoramento da execução através de descrições semânticas dos serviços.

A estrutura de ontologia em OWL-S é dividida em três sub-linguagens as quais oferecem três tipos de conhecimentos básicos a respeito dos serviços, são elas [Fensel et al., 2006]:

- ***Service Profile*:** Descreve "*o que o serviço faz*" e oferece um meio pelo qual o serviço pode ser anunciado. Inclui descrições sobre a funcionalidade e outras propriedades não funcionais do serviço;
- ***Service Model*:** Contém definições sobre "*como o serviço trabalha*". O seu principal objetivo é permitir a invocação, composição, monitoramento e a recuperação de serviços. As informações incluem entradas, saídas, pré-condições (circunstâncias precedentes ao uso do serviço), e efeitos (modificações causadas pelo serviço); e
- ***Service Grounding*:** Descrições sobre "*como acessar o serviço*" são encontradas nesse elemento. Inclui detalhes sobre qual é o formato da mensagem, protocolos de comunicação, entre outros.

A primitiva ***Service*** realiza uma ligação entre o perfil (*ServiceProfile*), o modelo (*ServiceModel*), e a fundamentação (*ServiceGrounding*) do serviço, em uma única unidade que pode ser publicada e invocada, como apresentado na Figura 2.6. Uma instância de

Service irá existir para cada um dos serviços publicados. OWL-S ganhou um considerável impulso pelo fato de fazer o reuso de OWL (padrão recomendado pela W3C), o que possibilitou sua maior adoção na literatura. Por esses motivos, escolhemos OWL-S para descrições em SWS.

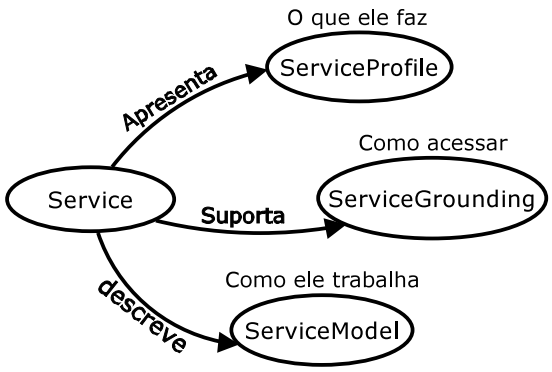


Figura 2.6: Visão geral da ontologia OWL-S

2.4 Grades computacionais

O conceito de Grades computacionais foi criado na década de 90 por Ian Foster e Carl Kesselman no laboratório de Argonne (*Argonne National Laboratory*) nos Estados Unidos [Foster and Kesselman, 1998]. O termo grade foi adotado a partir da analogia entre grades computacionais e a rede elétrica, devido à sua transparência para com o usuário e suas tecnologias de transmissão e distribuição. Tal conceito foi criado na ideia de que recursos em uma dada organização podem ser melhor explorados. No ambiente fornecido pelos sistemas em grades, recursos (seja computacionais, armazenamento, entre outros) podem ser compartilhados entre organizações, o que constitui o que denominamos de Organização Virtual (OV). A seguir iremos explorar melhor esses conceitos.

2.4.1 Definição

Computação em grade envolve a integração, gestão de serviços e recursos computacionais fracamente acoplados, heterogêneos, e geograficamente distribuídos. Abrange recursos em múltiplos domínios, com objetivo de alcançar alta capacidade de computação [Foster and Kesselman, 1998; Zhu, 2003]. Ian Foster fornece outra descrição presente em [Foster, 2002], na qual, tecnologias em grades são compostas por:

- Recursos coordenados que não estão sujeitos a controles centralizados;

- Padrões abertos de interfaces e protocolos de propósito geral; e
- Qualidade de serviço não trivial.

Requisitos presentes em estratégias de colaboração na indústria, e-Governo, ciência e engenharia são satisfatoriamente tratados através das funcionalidades de sistemas em grade. O ambiente resultante é formado por OVs, que por sua vez, são compostas por membros (indivíduos e instituições), um conjunto de recursos e regras de compartilhamento, definindo o que é compartilhado, quem é permitido, e sob quais condições ocorre o compartilhamento. Novos membros e regras são incluídos em OVs de forma dinâmica [Foster et al., 2001]. A Figura 2.7 mostra um exemplo desse ambiente.

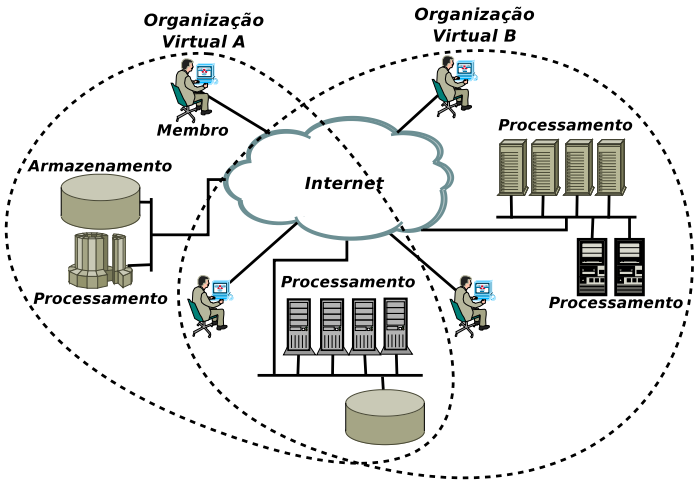


Figura 2.7: Ambiente em sistemas de grade

Grades computacionais foram idealizadas em ambientes heterogêneos. Dessa forma, a interoperabilidade é uma questão central para possibilitar aos diversos participantes relações de compartilhamento entre si. Assim como em um ambiente de rede, a interoperabilidade entre nós da grade é alcançada através da definição de protocolos. Portanto, uma grade é também uma arquitetura de protocolos [Foster et al., 2001].

Características como fraco acoplamento, heterogeneidade e sua ampla abrangência, distinguem tecnologias de grades de sistemas tradicionais de alto desempenho, como computação em *cluster*. A infraestrutura da grade abrange *hardware* para alcançar interconexão, armazenamento, processamento, etc; e *software* para controlar e monitorar seus resultados. Utiliza protocolos, serviços e ferramentas voltados à gestão, localização e transporte de dados; serviços e protocolos de consulta de informação sobre o estado de recursos; soluções de segurança; políticas de compartilhamento; e protocolos de gestão de recursos [Foster and Kesselman, 1998].

A complexidade de uma grade é proporcional ao número de organizações que a mesma suporta e suas limitações geográficas. O aumento do número de organizações faz com que se aumente a complexidade de determinados mecanismos, como por exemplo segurança, disponibilidade e desempenho. Outro fator importante, é a interoperabilidade, acomodando participantes em diferentes plataformas, linguagens e ambientes de programação.

2.4.2 Arquitetura

Grades computacionais têm como característica a execução paralela de suas tarefas, que permite reduzir o tempo de processamento. Para que os diversos processos possam se comunicar, é preciso que haja alguns componentes relacionados a conectividade, segurança, compartilhamento, entre outros. Todos os componentes da grade são categorizados de acordo com seu propósito em um diagrama de camadas que inclui [Foster et al., 2001]:

- **Camada de aplicação:** trata-se da camada que inclui aplicações do usuário, em uma organização virtual.
- **Camada de cooperação:** esta camada busca prover interações entre coleções de recursos, como serviço de diretório, serviços de monitoração, serviços de replicação de dados, entre outros.
- **Camada de conectividade de recursos:** esta camada define protocolos para comunicação e autenticação para transações pela rede, bem como protocolos para negociação, monitoramento, e controle de operações compartilhadas em recursos individuais.
- **Camada de construção ou fábrica:** trata-se da interface de controle local. Os componentes dessa camada implementam operações específicas de cada recurso, seja físico ou lógico, como: recursos computacionais, sistemas de armazenamento, suporte de rede, entre outras.

2.4.3 Aplicações

Tecnologias de grade podem ser aplicadas em uma gama de diferentes propósitos. Suas características tornam-na mais apropriadas a problemas de grandes dimensões. Descreveremos brevemente a seguir as cinco principais classes de problemas tratados por sistemas em grade [Foster and Kesselman, 1998]:

- **Computação Distribuída**. Sistemas em grade podem ser utilizados para combinar recursos dispersos, que normalmente não são encontrados em um único sistema. Nesse ambiente, questões como escalonamento de tarefas, escalabilidade de

protocolos e tolerância a falhas são de extrema importância para desempenho das aplicações;

- **Computação de alto desempenho**. Aplicações que se enquadram nesta classe geralmente distribuem aplicações fracamente acopladas ou mesmo independentes sobre os diversos recursos da grade, com foco em desempenho e maior capacidade de processamento (frequentemente sobre recursos ociosos);

- **Computação sob demanda**. Essa classe de aplicações demanda capacidades para utilização de recursos especiais, não locais, por motivos de conveniência ou relação custo-benefício. Nesse ambiente, questões de reserva e agendamento são consideradas para o restrito conjunto de recursos, os quais podem incluir impressoras, *software* especializado, sensores, entre outros;

- **Uso intensivo de dados**. O foco de problemas mapeados nessa classe tratam enorme quantidade de processamento de dados e comunicação entre repositórios distribuídos geograficamente; e

- **Colaboração**. Aplicações presentes nessa classe estão interessadas em uma infraestrutura virtual de cooperação e compartilhamento, permitindo a troca de informações e a integração de tarefas a partir de diferentes membros.

2.4.4 Computação em Nuvem

Atualmente, computação em nuvem (*Cloud computing*) tem sido comumente rotulada como a tecnologia que revolucionará a indústria de TIC. Características como fraco acoplamento, orientação a serviço, alta escalabilidade, recursos sob demanda e seu fácil manuseio estão a transformando em umas das principais tecnologias na área de computação distribuída [Gong et al., 2010]. Trata-se de uma tecnologia baseada na Internet em fase de amadurecimento, e inúmeros desafios são encontrados em sua adoção [Dillon et al., 2010].

Uma definição amplamente adotada na literatura é apresentada em [Mell and Grance, 2009], onde computação em nuvem é um "*modelo para permitir acesso conveniente e sob demanda a um conjunto de recursos de computação, configuráveis e compartilhados (por exemplo, hardware, plataformas de desenvolvimento e serviços), rapidamente fornecidos e lançados com o mínimo de esforço de gestão ou interações com o provedor do serviço*".

Como apontado por Tyrone Grandison em [Grandison et al., 2010], computação em nuvem é fruto da evolução em cadeia a partir das tecnologias de Computação em Grade, passando por *Utility Computing* (1990) e *Software as a Service* (2001). Para Trevor Doerksen [Doerksen, 2008], computação em nuvem é um sistema em grade amigável ao

usuário com uma progressão a partir de grades mais voltado ao meio comercial. De forma geral, a ideia básica é que dados e aplicações sejam hospedados de forma centralizada e acessível independente do lugar, tempo e dispositivo utilizado, necessitando apenas acesso à Internet [Wyld, 2009].

Frequentemente confundida com tecnologias de computação em grade, as tecnologias em nuvem ainda não possuem uma definição unificada. Entretanto, esse novo paradigma tem recebido grande atenção, como demonstra a Figura 2.8. A imagem exibe um gráfico referente aos termos computação em grade (*grid computing*) e computação em nuvem (*cloud computing*), e representa a média de tráfego no volume de indexação em pesquisas através do mecanismo de busca do Google (*Google Search*). O gráfico foi gerado pelo *Google Trends*, também do Google, e baseia-se em dados coletados ao redor do mundo no período de 2005 até a metade de 2013, sendo que o valor 100 representa o interesse máximo das pesquisas.

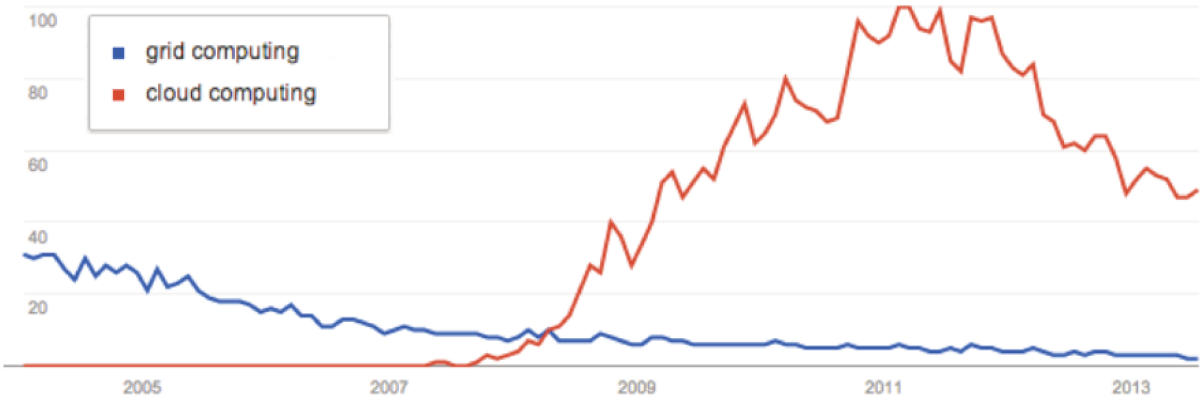


Figura 2.8: Grades vs Nuvem (Fonte: *Google Trends*)

Embora não seja a real taxa de utilização, o gráfico representa um esboço do grau de popularidade das respectivas tecnologias. Sendo assim, percebe-se o crescente interesse em computação em nuvem a partir de sua origem, porém, coloca-se em evidência que computação em nuvem não é um substituto imediato dos sistemas em grade, o que pode ser observado pelo declínio normal da tecnologia de grade, mesmo a partir do surgimento e do acentuado crescimento de computação em nuvem. Sobretudo, embora essas tecnologias sejam bem semelhantes, elas possuem diferenças significativas. Geralmente tecnologias de sistemas em grade e demais tecnologias, como por exemplo virtualização [Li et al., 2010], desenvolvem um importante papel em computação em nuvem.

Se comparado a sistemas em grades, computação em nuvem apresenta consideráveis vantagens. Em resumo, tecnologias em nuvem são fáceis de utilizar, permitindo rápido desenvolvimento e pouca customização. Por outro lado, sistemas em grade precisam adap-

tar aplicações com camadas adicionais em seu ambiente ("*gridified*"). Nuvem geralmente incorpora a tecnologia de virtualização, o que gera grandes vantagens, como migração e escalabilidade em nível de *hardware*. Grade comumente utiliza o princípio no qual uma única instância serve a múltiplos clientes (*Multitenancy*). Em nuvem, uma relativa economia de gastos na infraestrutura e suporte é observada, devido principalmente à centralização dos recursos, economia esta nem sempre alcançável em sistemas em grade. Qualidade de Serviço (QoS) é outra vantagem inerente de nuvem, pois sistemas em grade apresentam apenas a abordagem de melhor esforço [Vaquero et al., 2008; Dillon et al., 2010].

Entretanto, alguns dos desafios em nuvem, como monitoramento, padronização e integração de diferentes organizações, são naturalmente tratados por grades. Sistemas em grade fornecem grande quantidade de serviços de alto nível para gestão de execução, gestão de dados, e infraestrutura voltada à segurança, que são mapeados como grandes desafios em computação em nuvem [Zhang et al., 2010]. Em [Wyld, 2009; Dillon et al., 2010] são destacados demais desafios quanto à adoção de tecnologias em nuvem. A Tabela 2.1 resume algumas características importantes de ambas as tecnologias evidenciando pequenas similaridades e diferenças [Vaquero et al., 2008; Gong et al., 2010].

Tabela 2.1: Grade vs Nuvem

Características	Computação em nuvem	Grades computacionais
Usabilidade	Sim	Parcial
Escalabilidade	Sim	Sim
Padronização	Parcial	Sim
QoS	Sim	Parcial
Redução de custos	Sim	Parcial
Virtualização	Sim	Parcial
Heterogeneidade	Sim	Sim
Interoperabilidade	Parcial	Sim
Segurança	Parcial	Sim
Orientação a Serviços	Sim	Sim
Orientação a *Workflow*	Parcial	Parcial

Quando comparado à computação em grade, tecnologia em nuvem adiciona características promissoras tanto para aplicações empresariais quanto para e-Governo. Por outro lado, a sua falta de maturidade, ausência de padrões e preocupações tradicionais de computação corporativa (como a segurança de dados), ainda impõem grandes barreiras em sua adoção. Todavia, várias dessas barreiras são objeto de estudo de diversas pesquisas o que permite que alguns desses problemas possam ser resolvidos [Wyld, 2009; Dillon et al., 2010]. Dessa forma, durante a realização do trabalho optamos pela utilização

de tecnologias em grade devido à imaturidade observada em tecnologias de computação em nuvem, no momento do desenvolvimento desta dissertação e pelo fato de que grade computacional endereça de forma satisfatória diversos desafios presentes em ambientes multi-organizacionais.

2.5 Globus Toolkit e GPO

Globus Toolkit disponibiliza um conjunto de serviços para o desenvolvimento de aplicativos baseados em grades e para a construção de sistemas em grade. Esses componentes consistem em serviços de alto nível, de propósito geral, bibliotecas de programação e ferramentas de desenvolvimento. Implementado como um serviço da grade sobre o Globus Toolkit, o GPO permite a orquestração de serviços em um determinado fluxo de trabalho previamente definido. A seguir daremos maiores detalhes sobres ambas tecnologias.

2.5.1 Globus Toolkit

O Globus Toolkit [Globus Alliance, 2010b; Foster and Kesselman, 1997] é fruto de uma iniciativa que consistiu em estabelecer ligações entre 11 redes de alta velocidade para a criação de uma grade de rede nacional, I-WAY [Foster et al., 1996]. Essa experiência foi utilizada para a execução de diversas aplicações nas vésperas e durante a conferência *Supercomputing* de 1995. O sucesso dessa iniciativa incentivou o surgimento da primeira versão do Globus.

Ao longo dos anos, o *Globus Toolkit* (GT) se transformou em uma das principais tecnologias na área de sistemas em grade. Seus benefícios foram reconhecidos pelo meio industrial, tornando-se a tecnologia adotada em diversas aplicações empresariais. A partir de sua primeira versão em 1998, o GT continuou evoluindo, aderindo a padrões e tecnologias, preservando seu código aberto (*open source*) e sem fins lucrativos (filosofia adotada em suas origens). Durante a escrita do presente trabalho o GT encontrava-se em sua quinta versão (estável). A seguir, a partir da segunda versão, descreveremos brevemente as principais características de cada uma de suas versões, com maior ênfase na quarta versão, utilizada neste trabalho.

Globus Toolkit 2

A segunda versão do GT alcançou prestígio no meio industrial e rápida adoção em ambientes de computação em grade. Sua infraestrutura é composta por serviços de gestão de execução, serviços de informação e gestão de dados, além de uma infraestrutura de segurança (*Grid Security Infrastructure* (GSI)) e um conjunto de APIs (C *Common Libraries*) para desenvolvimento de aplicações na grade (veja Figura 2.9).

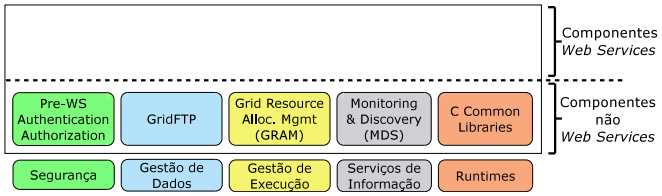


Figura 2.9: Componentes do GT2

A gestão de execução no Globus [Czajkowski et al., 1998] possui um único serviço chamado *Globus Resource Allocation Manager* (GRAM), o qual permite o gerenciamento de execução de tarefas sobre recursos da grade, inclusive remotamente. Sua arquitetura é baseada em camadas, potencialmente interagindo com diversos gerenciadores de recursos locais. O GRAM não realiza escalonamento de tarefas.

O serviço de informação do Globus é baseado no *Monitoring and Discovery Service* (MDS) [Czajkowski et al., 2001], o qual simplifica a interação entre clientes/aplicações e informações de recursos dinâmicos e estáticos em uma grade computacional.

A gestão de dados no Globus é responsável pelo acesso e gerenciamento de dados no ambiente de computação em grade. Sua infraestrutura é composta basicamente pelo componente GridFTP, protocolo de transferência confiável.

A infraestrutura de segurança do Globus, comum a todos os componentes, é chamada de GSI [Foster et al., 1998]. Suas funcionalidades tratam: delegação de permissões, comunicação segura entre processos e autenticação de usuário. O mecanismo de autenticação utiliza certificados X.509 assinados através de um *Certificate Authority* (CA).

Em relação à versão inicial, o GT versão 2 expandiu sua documentação e apresentou consideráveis melhorias em seus componentes. As melhorias englobam: uma tecnologia de empacotamento; instalação customizada, na qual apenas componentes necessários ao usuário são instalados; novos serviços de alto nível para replicação de dados; maior desempenho e suporte a autenticação aos serviços de informação [Globus Alliance, 2010a].

Globus Toolkit 3

O Globus Toolkit continuou evoluindo como resultado do *feedback* de desenvolvedores e do progresso em seu desenvolvimento. Em sua terceira versão, o *toolkit* da Globus apresentou uma profunda mudança em sua infraestrutura. Diferentemente das versões anteriores, o GT3 iniciou a migração para padrões com adicional adoção das tecnologias de Serviços *Web*, introduzindo o termo Serviços da Grade (*Grid Services*), que são Serviços *Web* adaptados ao ambiente de Grade.

Nesse estágio, o *toolkit* iniciou a incorporação de especificações, como a OGSA [Foster et al., 2002; Grimshaw et al., 2009] (ver a seguir), que é um arcabouço para definição

de arquiteturas voltadas a ambientes de computação em grade orientada a serviços, e a *Open Grid Services Infrastruture* (OGSI) [OGF, 2004], que é uma camada de extensão sobre OGSA responsável por definir convenções e extensões sobre Serviços *Web*, de forma a adequá-los a requisitos da OGSA. OGSI inclui componentes para definir abordagens para interfaces padrão, criação, gerência e troca de informações [Czajkowski et al., 2004].

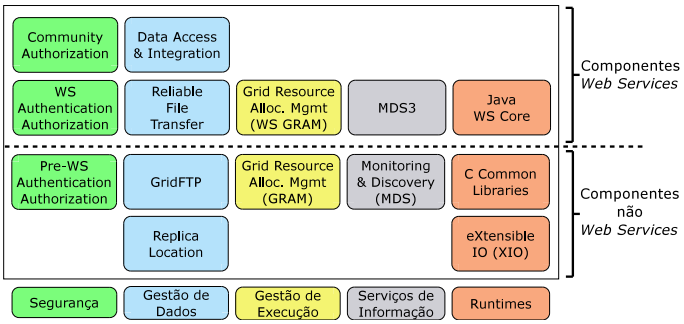


Figura 2.10: Componentes do GT3

Nessa versão o *toolkit* embutiu a API para o desenvolvimento de serviços na grade usando linguagem Java, chamado *Java WS Core*, antes realizado apenas em linguagem C. Como mostra a Figura 2.10, outros serviços para facilitar a utilização de recursos na grade também foram incluídos. Alguns desses serviços são construídos no topo de OGSI, chamados de componentes *web services*, e outros serviços, não *web services*, foram mantidos principalmente por questões de compatibilidade com sua versão anterior.

Globus Toolkit 4

A estratégica de manter o código aberto possibilitou ao GT manter-se em constante evolução. Em sua quarta versão, adicionou melhorias em desempenho, usabilidade, demais facilidades em seu núcleo, além de estender seu ambiente de desenvolvimento à linguagem de programação Python (*Python WS Core*), como apresentado na Figura 2.11. A mudança realizada no *toolkit* destacou-se pela ênfase em componentes *Web Services*, todavia, com suporte a serviços não *Web Services* com o propósito de manter compatibilidade com suas versões anteriores.

Motivado pelas vantagens do desenvolvimento padronizado, a partir da terceira versão, o GT passou a adotar padrões de desenvolvimento. Em sua quarta versão, o *toolkit* incorporou a especificação OGSA com WSRF, a qual provocou uma completa convergência entre grades e o paradigma SOA.

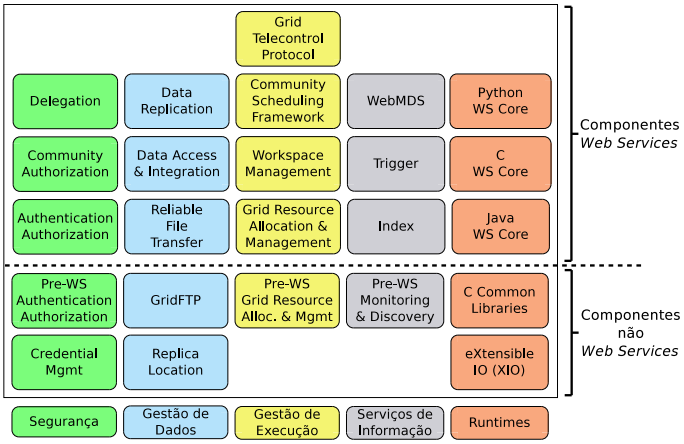


Figura 2.11: Componentes do GT4

OGSA

A OGSA [Grimshaw et al., 2009], desenvolvida pela *Open Grid Forum* (OGF), é originária a partir da publicação *"The Physiology of the Grid"* em [Foster et al., 2002]. Seu objetivo é o desenvolvimento de uma arquitetura aberta e padronizada para sistemas em grade. A OGSA trata desafios complexos em ambientes de grade através da definição de um conjunto de padrões, protocolos, e interfaces específicas para seus serviços.

A especificação OGSA é construída sobre tecnologias de Serviços *Web*. Os recursos são apresentados como Serviços da Grade (*Grid Services*), que na realidade são Serviços *Web* com um conjunto adicional de convenções determinadas pela OGSA [Foster et al., 2002]. Dessa forma, Serviços da Grade (SGs) fazem uso de várias tecnologias presentes na pilha de protocolos utilizadas em Serviços *Web* (veja Subseção 2.2.1).

Em OGSA, serviços são criados e destruídos dinamicamente (transientes). Eles permitem o mapeamento de múltiplas instâncias de recursos lógicos em um mesmo recurso físico, e mantêm o estado de uma invocação para outra (*stateful*). Por outro lado, Serviços *Web* são persistentes e não mantêm o estado (*stateless*) de uma transação para outra. Todavia, para adequar tecnologias de Serviços *Web* a requisitos da OGSA, uma nova especificação foi proposta, conhecida como WSRF [Foster et al., 2002; Sotomayor and Childers, 2006].

WSRF

A adoção da especificação WSRF [Graham et al., 2006] proporcionou uma completa convergência entre sistemas de grade e o paradigma SOA, e representa uma extensão às capacidades dos Serviços *Web*, tornando-os adequados a requisitos impostos pela OGSA no ambiente de grade. WSRF é padronizado pela OASIS, e compreende um conjunto de especificações na arquitetura de Serviços *Web* que fornece uma forma mais simples,

familiar e incremental para o seu manuseio.

Uma característica importante em WSRF é a forma como mantém informações durante as interações dos Serviços *Web*. Ao invés de transformar Serviços *Web* em entidades que mantém estado (*stateful*), o estado é mantido em uma entidade separada, chamada recurso. Assim, a composição de um Serviço *Web* com uma entidade que mantém estado (recurso) é chamado *WS-Resource*, e utiliza a especificação *WS-Addressing* para formalizar o relacionamento entre as entidades. *WS-Addressing* fornece uma forma eficiente para localizar um *WS-Resource*, utilizando um *Endpoint Reference* (EPR). Para a identificação do *WS-Resource*, o EPR é composto por uma URI (*Uniform Resource Identifier*) associada ao serviço e um identificador associado ao recurso, como mostra a Figura 2.12. O Listing 2.1 apresenta a representação em XML do EPR, composta de uma URI (*Address*) e os recursos associados (*resourceID*).

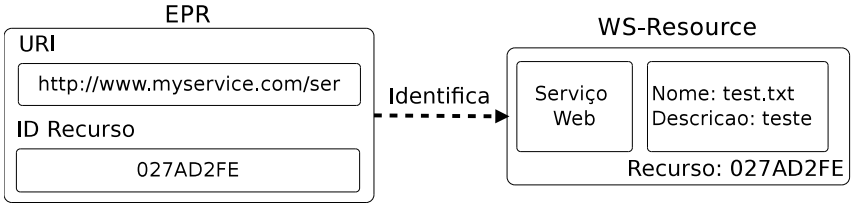


Figura 2.12: Relação EPR e *WS-Resource*

```
1 <wsa:EndpointReference>
2     <wsa:Address>
3         http://www.myservice.com/service
4     </wsa:Address>
5     <wsa:ReferenceProperties>
6         <tns:resourceID> 027AD2FE </tns:resourceID>
7     </wsa:ReferenceProperties>
8 </wsa:EndpointReference>
```

Listing 2.1: Representação em XML do EPR

WSRF [Graham et al., 2006] possui funcionalidades equivalentes se comparado com OGSI (utilizado no GT3), porém é composto por uma família de especificações desacopladas, que permitem maior flexibilidade. Diferentemente de OGSI, WSRF torna explícita a distinção entre o serviço e as entidades que mantém o estado (*stateful*), introduzindo o conceito de *WS-Resource*. Resumidamente, o conjunto de especificações WSRF é o seguinte [Graham et al., 2006; Czajkowski et al., 2004]:

- ***WS-ResourceProperties***: essa especificação fornece um conjunto de interfaces que nos permite acessar, modificar e consultar propriedades de recursos;

- ***WS-ResourceLifetime***: mecanismos básicos para a gestão do ciclo de vida de um *WS-Resource*. Em outras palavras, os recursos podem ser criados e destruídos em qualquer momento;
- ***WS-RenewableReferences***: mecanismos para recuperação de EPR quando o mesmo se torna inválido;
- ***WS-ServiceGroup***: permite a gestão de grupos para *WS-Resource* ou de serviços. Embora muito básico, essa especificação é a base para serviços de descoberta como o *Index Service* do *GT*4, que nos permitem agrupar diferentes serviços juntos e descobrí-los por meio de um ponto central de entrada (o grupo de serviço); e
- ***WS-BaseFaults***: suporte a falhas durante a invocação de um serviço.

Associado a WSRF e também relevante, a família de especificações *WS-Notification* fornece um conjunto de interfaces para a utilização do *design pattern* de notificações em Serviços Web. Esta especificação permite que uma notificação seja produzida quando ocorrem modificações em um *WS-Resource* no ambiente fornecido por WSRF [Sotomayor and Childers, 2006; Czajkowski et al., 2004]. Internamente é dividida em três especificações:

- ***WS-Topics***: fornece a funcionalidade para a definição de tópicos de interesse para notificação, utilizada pelas outras duas especificações;
- ***WS-BaseNotification***: esta especificação define a interface padrão de notificação para clientes e servidores; e
- ***WS-BrokeredNotification***: define a interface padrão para notificações utilizando uma entidade intermediária (*broker*).

WS-Notification é comumente utilizada para monitoramento em tempo real da mudança de estados dos serviços no sistema em grade, utilizando, por exemplo, a subscrição de propriedades de recursos. A seguir apresentamos alguns dos componentes do GT4 relevantes a esta dissertação.

Java WS Core

Java WS Core fornece APIs e ferramentas para hospedar serviços existentes e para o desenvolvimento de novos serviços em linguagem Java para sistemas em grade. Fornece suporte para notificação, descoberta, infra-estrutura de segurança, entre outras facilidades. No GT4, Java WS Core implementa padrões WSRF e *WS-Notification* [Sotomayor and Childers, 2006].

GSI

Uma vez que sistemas em grade possuem interações inter-organizacionais, recursos podem ser acessados por diferentes organizações, o que torna a segurança um requisito ainda mais importante. No GT4, os componentes de segurança são referidos coletivamente como *Grid Security Infrastructure* (GSI) [Welch et al., 2003], que fornece serviços fundamentais de segurança necessários para suportar a grade. GSI fornece bibliotecas e ferramentas para autenticação e proteção de mensagens que usam criptografia de chave pública (também conhecida como criptografia assimétrica) como base para sua funcionalidade. Suas características incluem segurança em nível de mensagem e de transporte; autenticação através de certificados digitais X.509; vários esquemas de autorização; delegação de credenciais; e diferentes níveis de segurança: contêiner, serviço e recurso. Os seus componentes incluem:

- ***Authentication and Authorization***: bibliotecas e ferramentas para controle de acesso a serviços e recursos, juntamente com o uso de diferentes métodos de autorização;
- ***Delegation***: conjunto de serviços que delega credencias para um contêiner;
- ***Community Authorization***: serviço para gestão de políticas de autorização de recursos da OV; e
- ***Credential Management***: incluem um componente para autorização de certificados (SimpleCA) e um repositório de credenciais (Myproxy).

Globus Toolkit 5

Em sua última versão lançada em 2010, a Globus Alliance modificou rigorosamente o *toolkit* e adicionou melhorias e novas características em suas principais facilidades. Sua nova versão é retratada pela Figura 2.13.

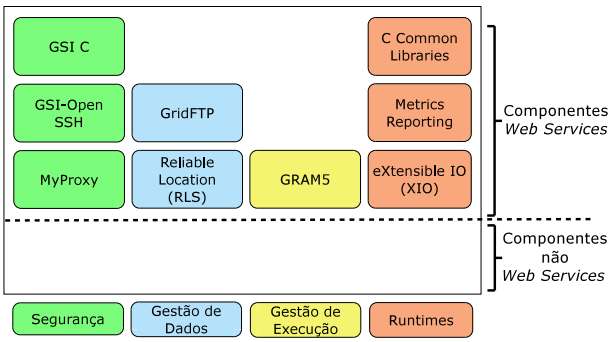


Figura 2.13: Componentes do GT5

Para manter suas funcionalidades através de Serviços Web e atacar algumas limitações da implementação utilizando GT4 Java WS Core e WSRF, um novo projeto chamado Globus Crux [Globus Alliance, 2011] foi iniciado. Globus Crux fornece uma coleção de ferramentas e serviços projetados para simplificar o processo de desenvolvimento SOA. Os módulos de serviços de informação do GT4 serão substituídos pelo *Integrated Information Services* (IIS) baseado em Crux. Seu toolkit agora inclui o Globus.org [Foster, 2011], um serviço online para transferência confiável de dados que substituirá o *Reliable File Transfer* (RFT). A Tabela 2.2 resume as tecnologias e características presentes no *toolkit* durante sua evolução.

Tabela 2.2: Resumo da evolução do GT

Versão	Ano	Tecnologias Utilizadas	Orientação a serviços	Serviços *Web*	Serviços não *Web*
GT 1	1998	HTTP	–	–	√
GT 2	2002	WSDL, WS-*	√	–	√
GT 3	2003	OGSA e OGSI	√	√	√
GT 4	2005	OGSA e WSRF	√	√	√
GT 5	2010	Globus Crux	√	√	–

Durante a fase final do desenvolvimento do presente projeto de pesquisa o GT5 foi lançado pela Globus Alliance. Não atualizamos para essa última versão do Globus *Toolkit* por que o GT5 não inclui mais a API Java, o que impossibilitaria a utilização de todo o código desenvolvido, pelo curto espaço de tempo, e pelo fato de que o GT4 atende as nossas expectativas.

2.5.2 GPO

Embora o *toolkit* da *Globus* forneça serviços para a gestão e execução de serviços na grade, ele não possui ferramentas para gestão de fluxos de atividades fortemente acopladas na grade. Para mapear os diversos fluxos de tarefas colaborativas no ambiente de e-Governo adotamos uma ferramenta chamada GPO, proposta em [Senna and Madeira, 2007].

GPO é um *middleware* para criação e gerência de fluxos de aplicações (Serviços da Grade) fortemente acopladas (como *workflows*) que interagem no ambiente de computação em grade. O GPO recebe um arquivo contendo descrições de um *workflow* em *Grid Process Orchestration Language* (GPOL), linguagem para submissão de *workflows*, e possui três componentes principais em sua infraestrutura para a gestão do mesmo (veja Figura 2.14): *GPO Run*, *GPO Maestro Factory Service* e *GPO Maestro Service Instance*.

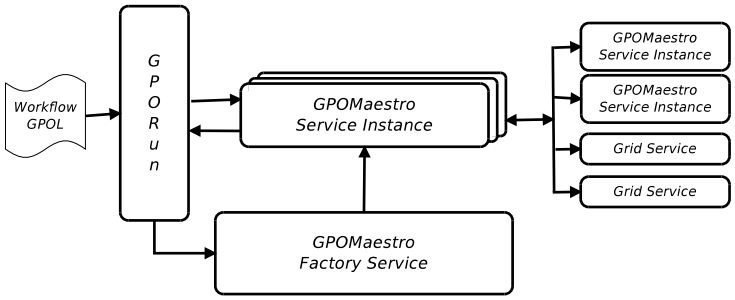


Figura 2.14: Arquitetura do GPO

O GPO *Run* permite a submissão de *workflows* ao GPO. Após a submissão, o *GPO Run* usa o GPO *Maestro Factory Service* (fábrica de serviços) para criar uma nova instância, chamada GPO *Maestro Service Instance*, que fica responsável pela execução do *workflow* na grade. Após a sua criação, a instância passa a interagir diretamente com o *GPO Run*. Após a execução do processo, o GPO *Run* termina a instância entregando o resultado à aplicação. GPO *Maestro Factory Service* e o *GPO Maestro Service Instance* são serviços da grade e trabalham sob o conceito de fábrica/instância. Dessa forma, cada *workflow* submetido é executado por uma dada instância do GPO *Maestro Service Instance*.

O GPO utiliza o padrão de fábrica da especificação *WS-Resource*, na qual separa serviço e recurso (persistência de estado). Ao receber a requisição para execução de um *workflow*, o *GPO Maestro Factory Service* solicita a criação de uma nova instância de recurso e recebe como retorno um identificador para essa instância. Em seguida, utilizando a identificação do recurso e a referência à instância do serviço (*GPO Maestro Service Instance*) é gerado o EPR (ou *WS-Resource*), que por sua vez é entregue ao cliente.

Cada *workflow* executado possui um recurso associado. Utilizando o EPR correspondente é possível acessar as variáveis mantidas pelo recurso. Essas variáveis ("*variáveis de processo*") acompanham passo a passo a execução do *workflow*, o que permite o seu monitoramento [Senna and Madeira, 2007]. As variáveis são:

- RP VALUE: contém o último valor retornado ao GPO Run;
- RP LASTOP: contém a descrição da última operação executada;
- RP EPRFILE: é o EPR da instância criada;
- RP WRFFILE: é o nome do arquivo GPOL em execução;
- RP WRFJOBID: identificação do workflow;
- RP STATUS: contém o estado da execução (uso interno);

- RP STEPEXEC: indica a atividade em execução;
- RP FAULTVALUE: código da falha ocorrida; e
- RP FAULTDESC: descrição da falha ocorrida.

GPOL

A GPOL, introduzida em [Senna and Madeira, 2007], é uma linguagem voltada à descrição de *workflows*. GPOL é baseada em XML, utiliza conceitos de orquestração presentes em WS-BPEL e características voltadas a sistemas de grade, como gestão de ciclo de vida e conceitos de fábrica/instância.

A estrutura de GPOL pode ser dividida em duas partes: definições (<*definitions*>) e processo (<*process*>). A primeira declara as variáveis e identifica todos os processos presentes no fluxo de trabalho, a segunda define a ordem do fluxo de execução, ou seja, a ordem em que os serviços são invocados no "*processo de negócio*". Algumas das funcionalidades presentes em GPOL são:

- Definição das variáveis e serviços utilizados: <*gpo:definitions*>;
- Definição da ordem de execução dos serviços: <*gpo:process*>;
- Controle lógico: <*gpo:if*>, <*gpo:while*>;
- Execução sequencial e paralela: <*gpo:sequence*> e <*gpo:flow*>;
- Invocação de um serviço da grade: <*gpo:invoke*>.

O Listing 2.2, apresenta um exemplo em GPOL. O elemento raiz da estrutura em GPOL inicia-se com o elemento <gpo:job>, o qual especifica o nome de "*ExecMerge*", e cujo objetivo é concatenar o conteúdo de dois arquivos em um terceiro, chamado *file.txt*. A seção de definições, delimitada pelo elemento <*gpo:definitions*>, possui a declaração das variáveis (*merge*, *file1* e *file2*) e seus respectivos valores mapeados pelo atributo *value*.

Em seguida definimos o fluxo do processo, delimitado pelo elemento <*gpo:process*>. Nesse exemplo o processo possui um elemento <*gpo:execute*>, que executa o comando "*cat*" (*/bin/cat*) (comando do sistema Linux) utilizando a variável *merge* como argumento; e dois elementos <*gpo:execute*> para a execução em paralelo (através do elemento <*gpo:flow*>) do comando "*rm*" (*/bin/rm*) (comando do sistema Linux), utilizando como parâmetro as variáveis *file1* e *fil2*, respectivamente.

```
<gpo:job name="ExecMerge">

    <!-- definitions section-->
    <gpo:definitions name="job_Definitions">
        <gpo:variables>
            <variable name="merge" type="string" value="file1.txt file2.txt > file.
                txt"/>
        </gpo:variables>
        <gpo:variables>
            <variable name="file1" type="string" value="file1.txt"/>
        </gpo:variables>
        <gpo:variables>
            <variable name="file2" type="string" value="file2.txt"/>
        </gpo:variables>
    </gpo:definitions>

    <!-- process section-->
    <gpo:process name="job_Process">
        <gpo:execute>
            executable="/bin/cat"
            <gpo:argument variable="merge"/>
        </gpo:execute>

        <gpo:flow>
            <gpo:execute>
                executable="/bin/rm"
                <gpo:arg variable="file1"/>
            </gpo:execute>
            <gpo:execute>
                executable="/bin/rm"
                <gpo:arg variable="file2"/>
            </gpo:execute>
        </gpo:flow>
    </gpo:process>
</gpo:job>
```

Listing 2.2: Exemplo de *Workflow* em GPOL

2.6 Resumo conclusivo

Neste capítulo apresentamos uma revisão dos conceitos necessários para o entendimento do conteúdo deste trabalho. Inicialmente definimos e-Governo, descrevemos seus principais requisitos e sua importância no setor público. Em seguida introduzimos o paradigma SOA, suas características e definimos o termo *Middleware*, além de tecnologias de Serviços *Web* e *Web* Semântica. Destacamos também características, benefícios, e tecnologias de Grades computacionais e incluímos as tecnologias em nuvem no âmbito de e-Governo, além de apresentar os desafios e as estratégias seguidas no trabalho.

Percebe-se que a infraestrutura fornecida pelas Grades Computacionais através do Globus Toolkit trata vários requisitos presentes não só em e-Governo, mas também em

várias aplicações empresariais. Todavia, a inclusão de outras tecnologias como, *Web* Semântica e composição de serviços (como *workflows*), apresenta grande potencial no mapeamento dos demais requisitos impostos por aplicações de e-Governo. Outro fato importante é a contextualização das tecnologias em nuvem em aplicações de e-Governo, que embora seja uma escolha promissora, apresenta diversos desafios em sua adoção.

Capítulo 3

Trabalhos Relacionados

A possibilidade de utilizar TIC em processos do governo possibilita o alcance de diversos benefícios, os quais representam grandes passos na desburocratização, eficiência e colaboração em suas atividades. Tais benefícios são requisitos emergentes em e-Governo, e têm motivado crescente demanda por pesquisas e desenvolvimento na área.

Este capítulo apresenta uma revisão bibliográfica do assunto abordado. A Seção 3.1 apresenta trabalhos voltados ao e-Governo na literatura, relatando suas características e as principais tecnologias abordadas. A Seção 3.2 compreende propostas no sentido de facilitar a utilização de Grades Computacionais. Governo eletrônico sobre Grades Computacionais é o assunto da Seção 3.3. Para finalizar o capítulo, a Seção 3.4 destaca as considerações finais a respeito da revisão bibliográfica apresentada.

3.1 Middleware para Governo Eletrônico

Aplicações para e-Governo apresentam uma demanda por requisitos complexos e multi-organizacionais. Nesse contexto, diversas propostas são encontradas na literatura com intuito de fornecer plataformas para aplicações nessa área. Entretanto, como argumenta Shvaiko em [Shvaiko et al., 2009], tais plataformas são voltadas às necessidades regionais ou de um país/estado em específico. Apresentamos abaixo alguns desses trabalhos.

Um projeto selecionado como caso de boa prática de interoperabilidade em níveis regionais e locais da administração pública da Itália é apresentado em [Marcucci et al., 2006]. A proposta, *Interoperabilità e Cooperazione Applicativa tra le Regioni e le Province Autonome* (**ICAR**), é parte de um plano italiano para desenvolvimento de e-Governo em nível inter-regional. Seu objetivo engloba interligação segura entre redes, o intercâmbio de dados (em formato padrão) e a cooperação de aplicações entre as diferentes autoridades regionais, abrangendo diretamente dezesseis regiões da Itália e províncias autônomas de Trento.

O ICAR é subdividido em dez subprojetos, coordenados cada um por uma região principal e um grupo de regiões parceiras. Três deles tratam projetos infraestruturais, enquanto os sete restantes são basicamente estudos de caso, destinados ao desenvolvimento de aplicações de cooperação entre serviços. O projeto segue o padrão nacional para desenvolvimento de sistema público de cooperação e conectividade, *Sistema Pubblico di Connettività e Cooperazione* – SPC, permitindo a entrega de serviços em acordo com abordagens *Event-Driven Arquitecture* (EDA) e SOA utilizando tecnologias de Serviços *Web* e seus padrões. Na padronização das informações são utilizados XML (para codificação de conteúdo de mensagem) e XML *schemas*, que permitem a definição da estrutura do conteúdo de mensagens trocadas e o vocabulário compartilhado.

Em [Drigas and Koukianakis, 2009], uma infraestrutura para a disseminação de informação e para a colaboração de serviços é apresentada. O autores introduzem um ambiente baseado em alta interatividade, mapeando níveis de usuários presentes no ambiente do setor público. A plataforma consiste em um portal *Web*, que oferece ferramentas (via Serviços *Web*) que permitem a comunicação entre usuários (como fórum de discussão, *chat* e caixa de mensagem), e serviços de informação, que compõem serviços responsáveis pela apresentação de funções do governo (como anúncios e perguntas frequentes), ambos gerenciáveis de acordo com permissões configuráveis a cada nível de usuário. A infraestrutura aborda um sistema de e-aplicação/e-petição que possui serviços responsáveis pela gestão e categorização de aplicações/petições.

Em [Shvaiko et al., 2009], propõe-se um *framework* para interoperabilidade no setor público, *e-Government Interoperability Framework for Mozambique* (**eGIF4M**), no contexto de Moçambique. A abordagem propõe uma arquitetura de entrega de serviços (via barramento de serviços), ciclo de vida para padrões (acomodando a evolução dos projetos de e-Governo), definição de um modelo de maturidade (disponibilizando a habilidade de medir o nível de adoção e difusão da proposta) e suporte de ações para assegurar sua sustentabilidade.

A arquitetura do *framework* segue os paradigmas SOA e EDA, e baseia-se na identificação e alocação de padrões para os variados componentes arquiteturais, através de um mecanismo chamado ciclo de padrões, onde estados representam os estágios de vida natural dos padrões (*Emerging*, *Current* e *Fading*). A infraestrutura baseia-se em um modelo em níveis e um conjunto de iniciativas para a utilização da proposta a longo prazo, que inclui: uma estrutura organizacional, sua disseminação, e o chamado *living lab*, paradigma em que novos produtos são originados de pesquisas e inovações orientadas ao usuário e à vida real.

***Citizen-oriented e-Government Platform* (CogPlat)**, idealizado em [Santos and Madeira, 2010], é uma arquitetura orientada a serviços, voltada às aplicações de e-Governo, a qual fornece um conjunto de facilidades para a integração e colaboração entre

entidades no âmbito da administração pública. A plataforma (implementada em tecnologia Microsoft .NET) inclui um mecanismo de composição semiautomática de serviços e introduz um conjunto de políticas para regular a colaboração entre as diferentes entidades (diferentes níveis de autonomia, privacidade, rastreabilidade e gestão de identidade). Sua infraestrutura é ilustrada na Figura 3.1.

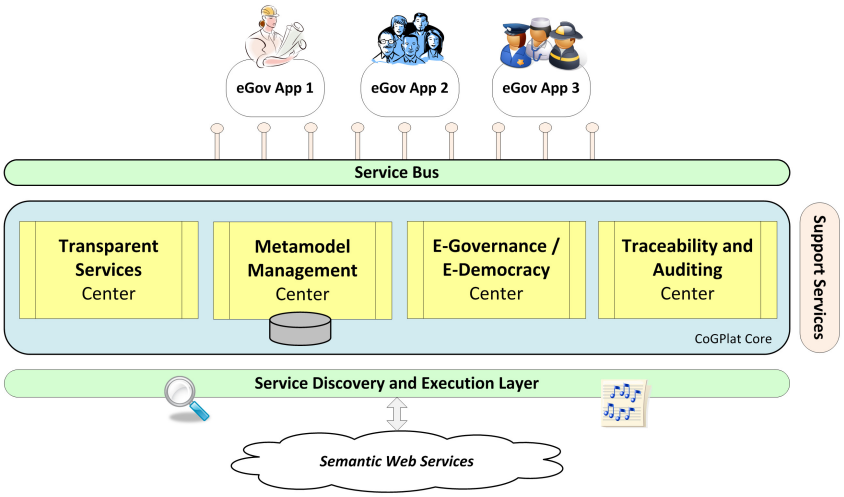


Figura 3.1: Infraestrutura do CogPlat.

CogPlat possui um Barramento de Serviços, o qual exporta suas funcionalidades (via Serviços *Web*) às aplicações externas. Seu núcleo engloba quatro módulos com funções específicas, como: composição de serviços (Centro de Serviços Transparentes); gestão de ontologias e perfis semânticos (Centro de Gestão de Metamodelos); ferramentas de colaboração (Centro de e-Democracia e e-Governança); e monitoramento e auditoria de serviços (Centro de Rastreabilidade e Auditoria). A infraestrutura inclui uma camada de descoberta de serviços semanticamente anotados, sua execução (através do motor de execução .NET 3.0 *Windows Workflow Foundation*) e suporte a mecanismos de segurança.

Uma plataforma para integração de serviços e dados em aplicações de e-Governo, **SoGoSP**, é proposta em [Ma, 2010]. A abordagem implementa o paradigma SOA através de Serviços *Web* em tecnologias de *workflows*. Sua infraestrutura divide suas funcionalidades em quatro camadas, que incluem: gestão de aplicações (camada de aplicação), facilidades para suporte visual de modelagem, administração e monitoramento de aplicações (camada de serviço comum), gestão dos serviços (camada de suporte de serviços), e integração de dados e serviços a partir de sistemas externos (camada de integração). A interação entre provedor e consumidor é realizada através do motor de coreografia, SoWFE (*Service-oriented Workflow Engine*), sobre *workflows* descritos em XPDL (XML *Process Definition Language*).

Uma arquitetura de e-governo unificada (*one-stop*) que endereça principalmente requisitos de integração e interoperabilidade na administração pública é apresentada em [Sedek et al., 2012]. A arquitetura é baseada no padrão SOA, utiliza WS-BPEL para o controle de fluxo de serviços e se divide em três camadas: Portal Unificado, que permite aos usuários acesso a serviços disponíveis; Provedor de Aplicação, responsável por assegurar a integração e a interoperabilidade; e Provedor de Serviços, que fornece serviços através de tecnologias de serviços *web*.

3.2 Tecnologias de Grades Computacionais

Diversos esforços relacionados a Grades têm proporcionado sua fácil adoção e a transformado em uma excelente ferramenta em vários tipos de aplicações. Nesta seção apresentamos algumas iniciativas bem estruturadas voltadas à construção de sistemas em grade e realizamos uma breve comparação entre essas tecnologias e o GT4 (apresentado na Seção 2.5.1).

3.2.1 Integrade

O Integrade [Integrade, 2011] é um projeto multi-institucional que consiste no desenvolvimento de uma infraestrutura de *software* voltada à construção de um ambiente computacional, utilizando recursos primordialmente ociosos (Grades oportunistas). Sua arquitetura implementa um *Middleware*, de código aberto, orientado a objetos, que utiliza o *Common Object Request Broker Architecture* (CORBA). O Integrade fornece suporte a aplicações como *Bag-of-Tasks* (aplicações paralelas com tarefas independentes) e paralelas (tarefas dependentes) [Goldchleger et al., 2004].

O ambiente fornecido pelo Integrade é estruturado em *clusters*, e composto por nós com papéis específicos (veja Figura 3.2). O nó *Gerenciador* de *cluster* possui como principal objetivo a coleta de informações para definição do escalonamento de aplicações (utilizando previsão probabilística); o nó *Usuário* realiza submissões de aplicações na grade; o nó *Dedicado* compõe máquinas cujos recursos são dedicadas ao ambiente de computação; e por último, o nó *Compartilhado*, que é composto por máquinas cujos recursos pertencem a um dado usuário e são compartilhados no ambiente computacional. A plataforma permite estender o ambiente a uma hierarquia de *clusters* e endereça requisitos de qualidade de serviços de usuários locais, *checkpointing* (para assegurar contra falhas) e segurança [Goldchleger et al., 2004].

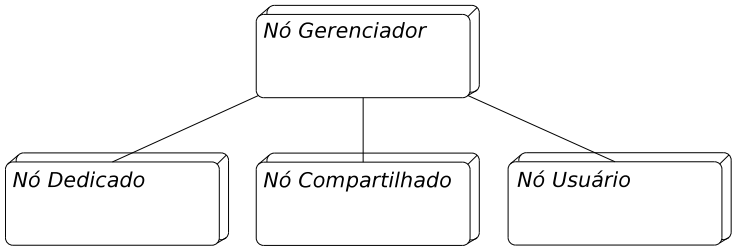


Figura 3.2: Arquitetura da *cluster* na Integrade

3.2.2 OurGrid

OurGrid [Andrade et al., 2003] fornece um ambiente de compartilhamento de recursos em uma comunidade sobre arquiteturas de rede par-a-par (*peer-to-peer*). O projeto implementa um *Middleware* de código aberto baseado em "*favores*", ou seja, os participantes prestam favores, que eventualmente serão recompensados em caso de necessidade, tendo prioridade de acordo com a quantidade de favores (não há negociação).

De forma geral, o ambiente é composto por Clientes e Brokers. Os clientes são *softwares* utilizados por usuários para acessar recursos da comunidade, que precisam encapsular pelo menos uma aplicação de escalonamento de atividades (exemplo MyGrid [Cirne et al., 2003]). O *Broker* realiza o gerenciamento de todos os recursos, agindo como fornecedor e consumidor ao acesso a recursos na comunidade OurGrid. OurGrid é voltado basicamente a aplicações *Bag-of-Tasks* e trata questões de segurança durante a execução.

3.2.3 Condor

Condor [Condor, 2011] é um projeto de pesquisa da Universidade de Wisconsin em Madison, inicialmente projetado em 1984, e alcançou grande interesse comercial e acadêmico. Condor é um sistema de computação distribuída de código aberto para gestão otimizada da carga de trabalho voltado à alta capacidade de computação para tarefas de longa duração. Pode ser utilizado em uma infraestrutura dedicada ou trabalhar em computadores (*desktops*) inativos com suporte para aplicações paralelas e sequenciais [Litzkow et al., 1988].

Os usuários submetem tarefas ao Condor, que são incluídas em uma fila, o escalonador define quando e onde executar as tarefas com base em políticas de escalonamento. O progresso é monitorado, e o usuário é informado após sua conclusão. Condor considera processos de migração e *checkpointing* para garantir continuidade da execução de tarefas, acesso justo entre usuários "*leves*" e "*pesados*", com adição de qualidade de serviços a usuários locais [Litzkow et al., 1988]. Condor pode ser utilizado em conjunto com tecnologias de grades. O Condor-G [Frey et al., 2002] é uma parte do Condor responsável

pela gestão de tarefas e é completamente interoperável com recursos do GT.

3.2.4 GridKit

Gridkit [Grace et al., 2004] é um middleware reflectivo que suporta diferentes paradigmas de comunicação. Seu principal objetivo é suportar a deficiência no desenvolvimento de aplicações em grade em frente à diversidade de tipos de dispositivos (por exemplo, dispositivos móveis, sensores, *clusters*) e redes utilizadas (como exemplo, rede de área local, redes sem fio, redes ad hoc). O GridKit é baseado no conceito de redes sobrepostas (*overlay network*) plugáveis e que suporta um conjunto extensível de serviços de alto nível.

Como apresentado na Figura 3.3, o Gridkit possui uma camada de serviços da grade (*Grid Services*), apresentados como serviços *web*, no topo de quatro domínios de suporte genérico ao *middleware*. Esses, por sua vez, são suportados pela camada *Open Overlay* que abstrai a diversidade de mecanismos de comunicações subjacentes. Os quatro domínios endereçados pelo Gridkit são:

- *Interaction services*: Esse domínio fornece serviços de comunicação plugáveis e uma genérica API para que os desenvolvedores utilizem instâncias dos serviços de comunicação;

- *Resource discovery*: Esse domínio contém serviços para descoberta de recursos. Ele suporta o uso de múltiplas tecnologias de descoberta de serviços plugáveis, o que permite maximizar a flexibilidade disponível para aplicativos.

- *Resource management*: Possui serviços plugáveis voltados à gestão de recursos distribuídos e suporte para escalonamento de aplicações da grade.

- *Grid security*: Esse componente possui serviços que suportam comunicação segura entre participantes considerando o tipo de comunicação em uso.

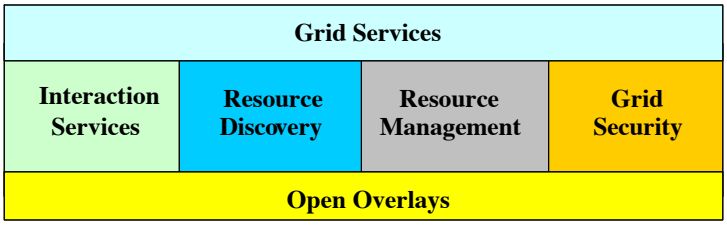


Figura 3.3: Arquitetura do Gridkit [Grace et al., 2004]

3.2.5 Comparação

A Tabela 3.1 realiza um comparativo entre as tecnologias para construção de grades computacionais mencionadas acima, voltado a uma visão inter-organizacional. Quando comparado às tecnologias de grade computacional, fica evidente a visão do GT4 (veja Seção 2.5.1) quanto a problemas inter-organizacionais, que demandam principalmente heterogeneidade, componentes de segurança, escalabilidade, e composição entre serviços. O GT4 incorporou padrões Web e o paradigma SOA, artefatos voltados a esse tipo de ambiente. Integrade [Integrade, 2011], OurGrid [Andrade et al., 2003] e Condor [Condor, 2011] são claramente voltados à construção de grades computacionais para o fornecimento de poder computacional.

O GridKit é uma interessante abordagem para adição de suporte a tipos de dispositivos, tais como dispositivos móveis, e tecnologias de comunicação à grade computacional. Está localizado em um nível acima das demais tecnologias de grade computacional citadas anteriormente, e por esse motivo, não iremos relacioná-la na Tabela 3.1.

Tabela 3.1: Resumo das tecnologias para grades, voltado a visão inter-organizacional

Software	Infraestrutura de Segurança	Heterogeneidade	Escalabilidade	OGSA	Orientados a Serviços
GT 4	√	√	√	√	√
Integrade	Parcial	√	√	–	–
OurGrid	Parcial	√	√	–	–
Condor	Parcial	√	√	–	–

3.3 Governo Eletrônico sobre Grades Computacionais

A adoção a padrões abertos, orientação a serviços, além de suas características naturais está provocando uma demanda por pesquisas e plataformas com foco na utilização de grades em aplicações de e-Governo, além de recentes pesquisas utilizando tecnologias em nuvem. Apresentamos algumas dessas iniciativas a seguir.

Uma investigação sobre o potencial de grades computacionais em ambientes de e-governo é apresentado em [Ahsan and Sobhan, 2012]. O trabalho estabelece uma comparação evidenciando os possíveis potenciais, desafios e obstáculos entre as características de grade computacional, padrões de desenvolvimento e paradigmas de programação com os cinco "pontos de vista" a partir dos quais ambientes de e-governo podem ser observa-

dos. Esses "pontos de vista" incluem: *enterprise*, *information*, *engineering*, *technology* e *computational*.

Uma plataforma para integração de informações dos cidadãos no setor público utilizando tecnologias em grades é proposto em [Alfonso et al., 2007]. O **gCitizen**, como é chamado, é composto por facilidades oferecidas pelo GT4 e propõe novos serviços como requisitos à integração, que incluem: sistema de endereçamento, GAC (*General Addressing Convention*), o qual fornece um identificador de serviços baseado principalmente em sua funcionalidade; uma arquitetura para descoberta de serviços, DSDA (*Distributed Service Discovery Architecture*), sua abordagem inclui um diretório distribuído de serviços IS (Index Services) através do FADA (*Federated Advanced Directory Architecture*) utilizados para consulta via DDS (*Distributed Discovery Services*); e um *framework* de *log* distribuído, DGLF (*Distributed General Logging Framework*), disperso na organização virtual. Propriedades de serviços (via XML e ontologias) permitem sua utilização sem conhecimento prévio do mesmo. O compartilhamento de informações apoia-se em um modelo de dados definidos em XML e XML Schema, voltado às necessidades particulares da implementação.

Yang *et al.* [Yang et al., 2007] propõem uma plataforma (**eGovGrid**) para a integração dos vários departamentos da administração pública através de um centro de recursos virtuais compartilhado. O *framework* implementa processos de negócios, através de Serviços *Web*, fornecidos a partir de uma infraestrutura de serviços em grade, a qual combina tecnologias de computação em grade e o paradigma SOA em um ambiente fornecido pelo Legion [Grimshaw et al., 1997].

Em [Zhang and Wang, 2008] é apresentado o suporte distribuído a aplicações de e-Governo através de uma arquitetura em multicamadas baseada no paradigma de Grade Semântica (*Semantic Grid*). Esta plataforma consiste em cinco camadas, a saber: camada de Recursos, de Ontologias, de *Middleware*, de Aplicação e de Portal. Sua camada de ontologia inclui descrições formais de conceitos (ontologias) e relações entre os mesmos, através de tecnologias como OWL e *Semantic Web Rule Language* (SWRL). Serviços do GT4 são utilizados como suporte. A abordagem semântica da proposta consiste em um repositório de serviços, via UDDI, para registro de capacidades sintáticas, repositório para registro das capacidades semânticas, e um mecanismo de mapeamento entre repositórios.

Uma infraestrutura baseada em sistemas de grade para integração de informações geradas através dos vários departamentos no setor público é proposta em [Kook et al., 2009]. O sistema suporta o ambiente da grade através de um *Middleware* baseado em sistemas multiagentes, que implementa o papel do cliente e servidor. O trabalho propõe também o GMDR (*Government Metadata Registry*) para a interoperabilidade capaz de sobrepor a heterogeneidade dos dados e sistemas através de ontologias e relações semânticas. Sua arquitetura consiste em vários módulos de gestão divididos em camadas.

Em [Agarwal et al., 2010], é apresentada uma proposta para habilitar processamento de dados no projeto SAFORAH (*System of Agents for Forest Observation Research with Advanced Hierarchies*) através de grades computacionais baseada no *Globus Toolkit*. SAFORAH foi criado com objetivo de coordenar e simplificar o arquivamento e compartilhamento de grande conjunto de dados geoespaciais entre grupos de pesquisa com o Serviço Florestal Canadense, e vários outros parceiros acadêmicos e governamentais.

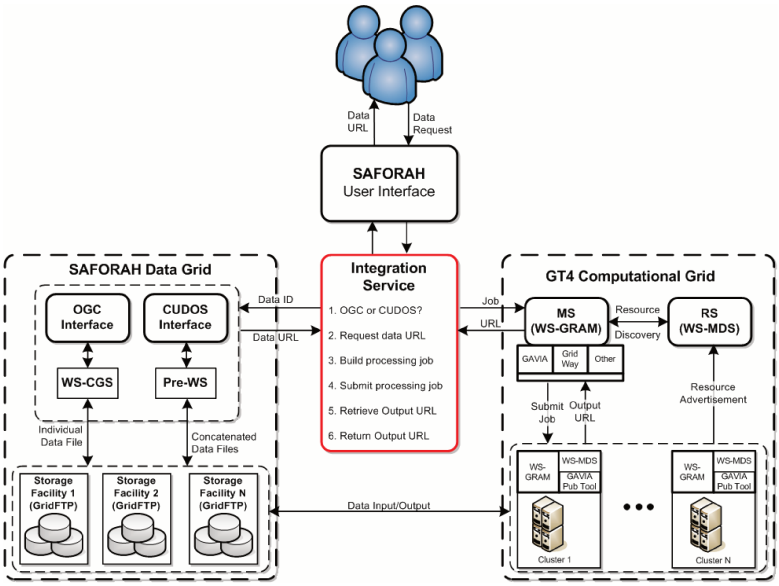


Figura 3.4: Infraestrutura do SAFORAH [Agarwal et al., 2010].

A infraestrutura é composta por uma grade de dados e a grade computacional (Figura 3.4). A integração é realizada via um serviço da grade, *Integration Service*. A grade de dados conta com uma interface baseada no padrão OGC (*Open Geospatial Consortium*), a qual utiliza serviços da grade (*Grid Services*) para acesso a dados individuais e uma interface para acesso a dados concatenados. A grade computacional possui dois componentes principais: um meta-escalonador e um serviço de registro. Cada *cluster* publica seu *status* no registro, que é usado pelo meta-escalonador para escalonar tarefas. A abordagem utiliza o WS-MDS (*Web Services Monitoring and Discovery Service*) para implementação do serviço de registro e o meta-escalonador Gavia, desenvolvido usando o Condor-G para a tomada de decisão para alocação.

O projeto para um *framework* de e-governo sobre grade computacional é proposto em [Lu et al., 2011]. O *framework* engloba a ideia de um portal unificado (*one-stop*), e alinhado às características de grade computacional e procura endereçar uma lista de problemas chave em ambientes da administração pública, como por exemplo, a falta de padronização na utilização de tecnologias, compartilhamento de informações, cooperação

entre departamentos e questões de segurança.

Sistemas em grade têm sido utilizados em diversos projetos empresariais e também em atividades de e-Governo. Entretanto, recentemente, grande parte de pesquisas e desenvolvimento tem direcionado seu foco para a migração [Kundra, 2010; Chu et al., 2012] e a construção de mecanismos eletrônicos sobre computação em nuvem.

Em [Liang, 2012], uma investigação apresenta as principais deficiências do e-governo na atualidade, os benefícios da utilização de computação em nuvem, e explora a combinação entre computação em nuvem e e-governo. O trabalho sugere uma estratégia para a seleção de modelos de implementação e modelos de serviços presentes em computação em nuvem para departamentos e processos de negócios na administração pública.

Na Indonésia, a adoção de uma infraestrutura de e-governo com base em computação em nuvem voltada à padronização, ao compartilhamento de recursos e à consolidação de dados no governo é apresentada em [Ahmad and Hasibuan, 2012]. A arquitetura proposta consta de seis camadas (usuário, acesso, serviço, gestão, virtualização e infraestrutura). A camada de serviço (*Cloud Services*) interage com o usuário e oferece quatro modelos de serviços, que são: *Software as a Service* (oferece aplicações como email, aplicações G2G, G2B, G2C, entre outros); *Government as a Service* (centrada no serviço do governo para o público, ex.: serviços de impostos, etc.); *Infrastructure as a Service* (fornece infraestrutura física), e *Data as a Service* (informações fornecidas pelo governo).

Em [Zhang and Chen, 2010] é apresentada uma solução para arquiteturas de computação em nuvem para e-Governo. O trabalho introduz o termo *C-Government* (*Cloud Government*), e realiza um mapeamento entre funções pretendidas pelo governo (produtividade, transparência, participação e colaboração) em camadas presentes em tecnologias de nuvem (*Software as a Service* - SaaS; *Platform as a Service* - PaaS; e *Infrastructure as a Service* - IaaS). Uma infraestrutura de serviços para computação escalável com adicional centro de dados é proposta no projeto Nebula [Nebula, 2011]. Nebula possui código aberto e utiliza computação em nuvem voltada inicialmente às necessidades de cientistas da NASA.

Como parte de uma iniciativa para a utilização de nuvem em e-Governo, o NIST (*National Institute of Standards and Technology*), em colaboração com demais agências e especialistas, está desenvolvendo importantes pesquisas para identificar, priorizar, e alcançar um consenso com respeito à padronização em nuvem (*Federal Cloud Computing Initiative*). Dessa forma, vários projetos voltados à adoção de tecnologias em nuvens são apresentados em [Kundra, 2010].

3.4 Resumo conclusivo

Neste capítulo foi feita uma revisão sobre projetos em e-Governo, sistemas em grade e a relação dos dois conceitos. Como apresentado, existem várias pesquisas na área de e-Governo, área que demanda novos projetos com o desenvolvimento de novas tecnologias, em sua maioria voltados para as necessidades regionais ou de um país em específico.

O *toolkit* da Globus tem se tornado um padrão para construção de sistemas em grade e, diferentemente da maioria das tecnologias de grade computacional, possui um foco voltado a problemas inter-organizacionais. Outro ponto importante é a forte presença de computação em nuvem em recentes pesquisas voltadas a ambientes governamentais e empresariais, revelando assim um futuro promissor para a exploração dessa recente tecnologia.

Capítulo 4

Middleware proposto

Nesse capítulo apresentamos a proposta de um *middleware* para e-Governo sobre grades computacionais. Inicialmente, na Seção 4.1, abordamos os principais desafios enfrentados no projeto. Em seguida, a Seção 4.2 explora a arquitetura de forma geral, destacando as comunicações entre diferentes módulos e camadas. Na Seção 4.3 destacamos os módulos internos do *middleware*, suas classes e suas funcionalidades. A integração do *middleware* com o GT4, onde são endereçados detalhes sobre a execução de serviços, monitoramento e a utilização de segurança do *toolkit*, está presente na Seção 4.4. A Seção 4.5 trata aspectos de implementação do *middleware*. O resumo do capítulo e as considerações finais sobre o trabalho são o assunto da Seção 4.6.

4.1 Desafios

Este trabalho foi inicialmente inspirado na pesquisa apresentada em [Santos and Madeira, 2007]. Nessa pesquisa os autores apresentam uma investigação relacionada aos principais desafios a serem vencidos para que grades computacionais, através do GT4, possam atender aos principais requisitos das aplicações de e-Governo, requisitos apresentados na Subseção 2.1.2. A pesquisa destaca o suporte a processos baseados em *workflows*, requisitos específicos de segurança, e o suporte à *Web* Semântica como alguns dos principais desafios para transformar tecnologias de sistemas em grade em uma sólida e poderosa solução em *middleware* para aplicações de e-Governo. Com foco nos desafios citados, este trabalho propõe um *middleware* sobre grade computacional para aplicações de e-Governo.

Aplicações de e-Governo apresentam requisitos mais rigorosos e com elevado grau de granularidade se relacionado aos requisitos normalmente encontrados em aplicações do setor privado [Davies et al., 2007]. Na Subseção 2.1.2 descrevemos os requisitos chave presentes em aplicações de e-Governo. Associado a esse ambiente, o GT4 apresentou grande evolução com relação ao tratamento da interoperabilidade organizacional, aderindo

a padrões e a paradigmas emergentes. Entretanto, alguns desafios são evidenciados para que sistemas em grade possam atender aos requisitos chave presentes em e-Governo. A Tabela 4.1 relaciona quais requisitos de e-Governo (veja Subseção 2.1.2) são suportados e quais carecem de suporte nesse ambiente, tendo como foco a utilização do GT4, como colocado em evidência em [Santos and Madeira, 2007].

Tabela 4.1: E-Governo sobre grades computacionais (GT4)

Requisito	Suporte	Não tem suporte
Interoperabilidade	Técnico	Semântico
Reusabilidade	√	–
Abertura	√	–
Escalabilidade	Processamento e armazenamento	Controle descentralizado confiável
Segurança	Autenticação, anonimato e proteção de mensagens	Requisitos de segurança específicos para e-Governo

- Interoperabilidade: esse requisito é satisfeito pela grade principalmente em nível técnico, devido à utilização de abordagens como SOA e tecnologias como Serviços *Web*. Porém sem suporte semântico.

- Reusabilidade: esse requisito está mais relacionado com o modelo de serviço do que com a infraestrutura do *middleware*. Todavia, devido ao ambiente aberto e cooperativo suportado pelo GT4 a reutilização de serviços é facilitada.

- Abertura: esse requisito está relacionado à utilização de padrões abertos e interfaces bem definidas, suportados pelo GT4.

- Escalabilidade: é considerada definitivamente a grande vantagem de sistemas em grade. Essa escalabilidade, contudo, é voltada apenas ao poder de processamento e armazenamento. Porém, em ambientes de e-Governo, é necessário um controle confiável dos processos devido à descentralização presente no ambiente da grade, ainda sem suporte no GT4.

- Segurança: possui suporte adequado a partir da grade computacional se considerarmos autenticação do usuário, anonimato e proteção de mensagens. Requisitos de segurança mais específicos ao ambiente de e-Governo não são suportados.

Na Tabela 4.1 relacionamos requisitos de e-Governo com a infraestrutura fornecida pela grade computacional, relatando quais requisitos são suportados e quais carecem de

suporte. A seguir iremos apresentar uma descrição mais detalhada sobre os principais desafios em interoperabilidade, escalabilidade e seguraça para que grades computacionais, utilizando o GT4, possam suportar ambientes de e-Governo [Santos and Madeira, 2007]. Os desafios são exibidos na Tabela 4.2.

Tabela 4.2: Desafios para suporte de e-Governo sobre grades computacionais (GT4)

Requisito	Desafios
Interoperabilidade	Descrição semântica + suporte a ontologias Suporte a *Workflows* + Mecanismos de tolerância a falhas Acessibilidade
Escalabilidade	Mecanismos de Controle decentralizado Distintas OV no ambiente Rastreabilidade e auditoria
Segurança	Políticas de acesso a informações Autonomia e Proteção contra roubo de identidade

Interoperabilidade

Web semântica (veja Seção 2.3) é um poderoso mecanismo para aumentar a interoperabilidade entre serviços. As atuais implementações de sistemas em grade não possuem suporte a descrições semânticas e ontologias, e apresentam-se como um interessante desafio para a comunidade científica.

Além disso, aplicações de e-Governo normalmente são descritas como *workflows* ou DAGs (*Direct Acyclic Graphs*) e muitas tarefas são interdependentes e necessitam de eficientes mecanismos de tolerância a falha, ambos ainda pouco explorados no ambiente fornecido pelo GT4. Outro desafio que precisa ser enfrentado está relacionado à acessibilidade, permitindo suporte em nível de *middleware* a dispositivos móveis e mecanismos de acessibilidade a deficientes e idosos.

Escalabilidade

Considerando a descentralização presente nesse ambiente, questões relacionadas à falta de controle sobre tarefas precisam ser endereçadas na grade computacional. Além disso, uma importante questão na utilização de aplicações de e-Governo sobre grades computacionais está relacionada à correta modelagem das organizações virtuais, em que OVs devem ser formadas apenas com os membros envolvidos na tomada de decisão.

Finalmente, a habilidade de monitorar os estados de diferentes processos em execução e a possibilidade de realizar uma auditoria no final das execuções são também requisitos

em diversas aplicações de e-Governo, o que permite maior transparência da administração pública.

Segurança

Devido ao tipo de informações e processos envolvidos, muitas aplicações de e-Governo possuem requisitos específicos de segurança. A implementação da grade utilizando GT4 possui satisfatórios mecanismos de segurança, mas o suporte para demandas relacionadas à autonomia (nível de autonomia da plataforma e aplicações sobre operações internas) e à proteção contra roubo de identidade ainda necessitam de melhorias. Outro ponto importante refere-se às políticas de acesso a informações (nível de privacidade de informações entre duas entidades) integradas aos serviços do ambiente, pois constituem mecanismos essenciais em ambientes de e-Governo e ainda não são suportados pelo GT4.

Apresentados os desafios, o objetivo do presente trabalho é propor uma plataforma, através de um *middleware*, que viabilize o ambiente computacional em grade (utilizando o *GT*4) às aplicações de e-Governo. Na Tabela 4.3, destacamos os requisitos que exploramos no trabalho e suas respectivas estratégias de abordagens. Neste trabalho não exploramos os desafios voltados à segurança e acessibilidade por questões de escopo do projeto. Iremos utilizar a infra-estrutura de segurança presente no GT4 que abrange os principais requisitos para o contexto.

Tabela 4.3: Requisitos vs estratégias

Requisito	Estratégia
Interoperabilidade	- Composição de serviços (como *workflow*) utilizando o GPO - Adição de descrições semânticas em OWL-S
Escalabilidade	- Distintas OV no ambiente - Construção de um *Middleware* para controle centralizado - Rastreabilidade e auditoria através de especificações WSRF

Como relatamos em capítulos anteriores, a administração pública pode se beneficiar de diversas formas com a adoção de mecanismos eletrônicos; todavia, as aplicações voltadas a esse segmento apresentam grandes desafios do ponto de vista tecnológico. Nesse contexto apresentamos o MidGov, um *Middleware* para aplicações de e-Governo, orientado a serviços, e que permite a redução burocrática em atividades do setor público.

O objetivo do MidGov é propor um ambiente sobre grades computacionais para superar obstáculos de comunicação presentes em redes empresariais, mais voltado às aplicações do setor público. Sua infraestrutura permite aumentar a interação e colaboração entre as entidades no ambiente governamental. A Figura 4.1 apresenta uma visão geral do projeto,

em que o MidGov reside em uma das máquinas da grade construída através do *Globus Toolkit* 4. Dessa forma, as tarefas em cada um dos diferentes órgãos do governo são encapsuladas como serviços da grade e entregues sobre o ambiente de forma transparente.

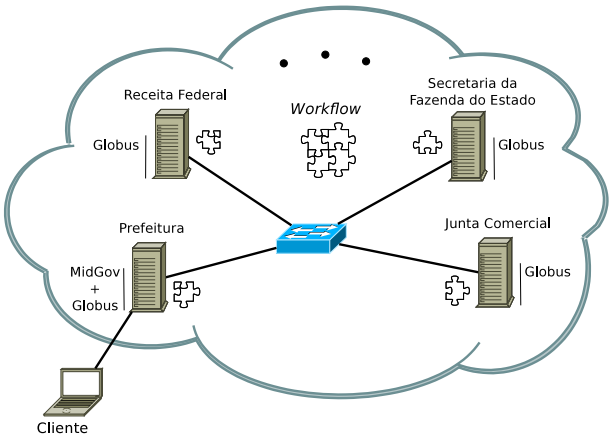


Figura 4.1: Visão geral do ambiente

A infraestrutura proposta, MidGov (***Mid****dleware for* ***Gov****ernment*), é baseada no ***C****itizen-****o****riented e-****G****overnment* ***Plat****form (CogPlat)*, apresentado em [Santos and Madeira, 2010], com adicional suporte de serviços fornecidos pelo *Globus Toolkit 4* (GT4) [Globus Alliance, 2010b] para suporte e adequação sobre o sistema de grade.

4.2 Arquitetura do MidGov

A Figura 4.2 mostra a arquitetura completa do projeto, que engloba: um cliente (Centro de Administração *Web*) como uma aplicação *Web*; o servidor, que fornece serviços através da grade computacional; e uma camada de *middleware* responsável pela interação entre ambas as partes.

Na camada superior, o Centro de Administração *Web* fornece uma interface amigável ao usuário para utilização e gerência do sistema. Funcionalidades como buscas semânticas, execução de serviços e *workflows* na grade, e gestão de metamodelos (perfis OWL-S e ontologias) são acessíveis a partir do sistema *Web*.

O MidGov exporta suas funcionalidades como Serviços *Web* através de um barramento de serviços (*Enterprise Service Bus*). As funcionalidades são agrupadas em módulos, que são: Centro de e-Governança/e-Democracia, Centro de Gestão de Metamodelos, Centro de Gestão de Serviços e Centro de Rastreabilidade e Auditoria. Esses módulos acessam serviços disponíveis na grade computacional e informações na base de dados contendo registros (log do sistema), metamodelos e informações sobre serviços e workflows.

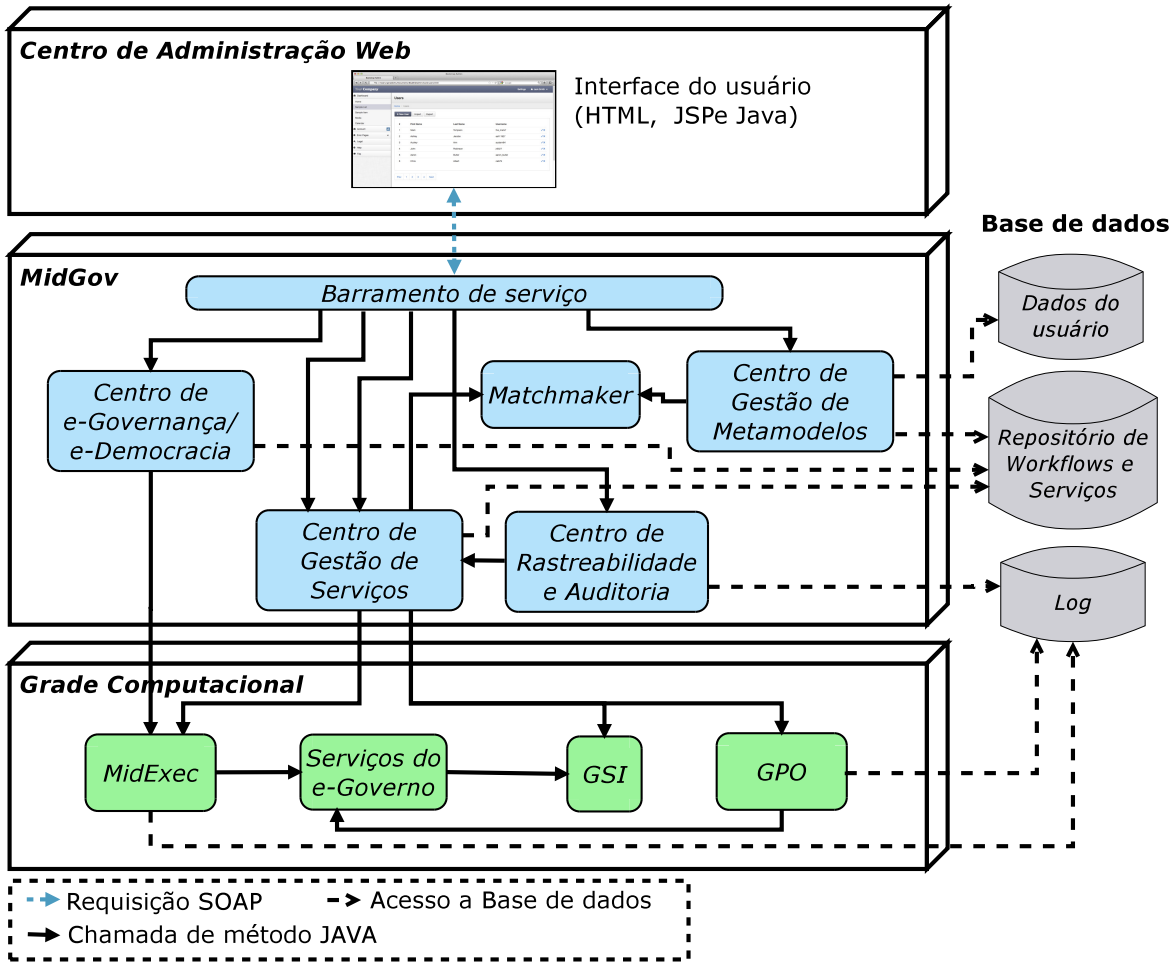


Figura 4.2: Arquitetura do MidGov.

Na camada inferior estão os serviços disponíveis na grade computacional. Nessa camada encontram-se os serviços do e-Governo, que representam atividades da administração pública; o MidExec, responsável pela execução de serviços simples na grade (contido em um único Serviço da Grade); e o GPO, que realiza a gerência de fluxos de execução de serviços na grade. Esses serviços são desenvolvidos como serviços da grade (termo apresentado na Seção 2.5.1), o que permite a utilização de diversas especificações fornecidas pela WSRF, a infraestrutura de segurança GSI (explorada na Subseção 2.5.1), e diversas outras facilidades presentes no GT4.

4.3 Componentes do MidGov

De forma geral, o MidGov é composto por um barramento de serviços que permite o acesso às suas funcionalidades; quatro módulos com conjuntos predefinidos de facilidades, e serviços fornecidos a partir da grade computacional construída através do GT4.

Como mencionado acima, a infraestrutura proposta é baseada no *CogPlat*, presente em [Santos and Madeira, 2010]. A Tabela 4.4 apresenta as semelhanças e as principais diferenças entre as duas plataformas.

Tabela 4.4: *CogPlat* vs *MidGov*

Características	*CogPlat*	*MidGov*
Composição de serviços	Semiautomático	Estático
Fluxo de Serviços	*Windows Workflow Foundation*	GPO
Políticas de Interação	Sim	Parcial
Grade Computacional	Não	Sim
Centro de Gestão de Serviços	Sim	Sim
Centro de Gestão de Metamodelos	Sim	Sim
Centro de e-Democracia e e-Governança	Sim	Sim
Centro de Rastreabilidade e Auditoria	Sim	Sim

No que se refere à automatização na composição de serviços, três abordagens podem ser identificadas na literatura, que são: composição de serviços estática, semiautomática e automática.

Por questões de escopo utilizamos uma abordagem de composição de serviços estática, onde o modelo de processo é criado manualmente e os serviços são vinculados em tempo de projeto. Abordagens com alto grau de dinamismo (menor intervenção humana) são dificilmente encontradas na literatura e possuem grande complexidade [Fluegge et al., 2006]. Para a implementação de uma abordagem semiautomática encontramos algumas barreiras relacionadas, como por exemplo, a utilização da linguagem GPOL, que não oferece suporte para a criação de *workflows* abstratos. Então optamos por na primeira versão do Midgov implementar a composição de serviços estática. Considerando a arquitetura do Midgov, uma possível proposta para a implementação de um mecanismo semiautomático para a composição de serviços [Fluegge et al., 2006; Preist, 2007; Ngan and Kanagasabai, 2012] segue os seguintes passos:

1. Especificação de modelos de processos abstratos (*abstract workflows*). Ao invés de descrições sintáticas, disponibiliza-se a descrição semântica da funcionalidade do serviço. BPEL^{Light} [Nitzsche et al., 2007] permite a descrição da lógica do processo;

2. Geração de um mecanismo (no Centro de Gestão de Metamodelos) para funcionar como um serviço de composição (*Service Composer*). Sua função é mapear as atividades descritas no processo abstrato em serviços concretos. Essa atividade irá interagir com o OWLS-MX para procurar serviços utilizando a respectiva descrição semântica;

3. Em um último estágio, o processo abstrato precisa ser traduzido, bem como vinculado a serviços concretos, em uma linguagem que permita a execução na grade computacional. A conversão de um processo abstrato, utilizando BPEL^{Light}, para um processo concreto, utilizando GPOL, pode representar uma abordagem interessante uma vez que GPOL é baseada em WS-BPEL e há várias semelhanças entre as duas linguagens.

Para a orquestração dos serviços utilizamos o GPO, o qual permite a orquestração do fluxo de execução de serviços na grade computacional. A plataforma *CogPlat* introduz a utilização de políticas de interação entre os parceiros que colaboram no processo (políticas de autonomia, privacidade, rastreabilidade e gestão de identidade). Construímos funcionalidades baseadas nas políticas citadas acima voltadas à rastreabilidade, privacidade e gestão de identidade na grade computacional através da GSI. A tabela Tabela 4.5 resume a implementação de políticas a partir do CogPlat.

Tabela 4.5: Políticas de Interação

Politicas de Interação	*CogPlat*	*MidGov*
Autonomia de entidades	√	–
Privacidade de dados	√	√
Rastreabilidade de serviços	√	√
Gestão de identidade	√	√

A Figura 4.3 apresenta um relacionamento entre a arquitetura do MidGov e CogPlat. Como mencionado anteriormente, a arquitetura do MidGov é baseada no CogPlat. O MidGov herda o barramento de serviços, os quatro módulos internos, a função macro de cada módulo e as políticas de interação, não evidentes na figura. Todavia, a implementação dos módulos no *Middleware for Government* (MidGov) é voltada à utilização dos serviços e às facilidades da grade computacional através do *Java WS Core*. Sendo assim, a codificação interna é completamente diferente, permitindo que os serviços do *middleware* acessem serviços da grade computacional. Se comparado ao *CogPlat*, o MidGov adicionou características inerentes dos sistemas de grades computacionais, tais como escalabilidade, interoperabilidade, além de diversas facilidades a partir do *Globus Toolkit*.

A seguir apresentaremos mais detalhes a respeito de cada um dos componentes presentes no *middleware*.

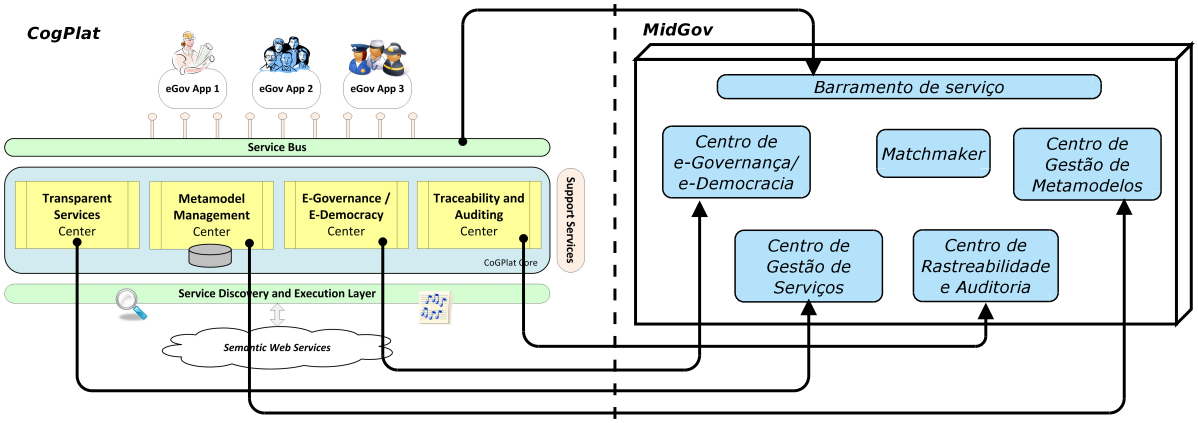


Figura 4.3: Relacionamento entre CogPlat e MidGov

4.3.1 Centro de Gestão de Metamodelos

Este módulo oferece serviços para a gestão de metamodelos, serviços da grade e *workflows* em GPOL. Os seus componentes (como apresentado pela Figura 4.4) são os seguintes:

- ***OntologyManager***: possui serviços que acessam métodos do *Owls-MX Matchmaker* para a gestão de perfis OWL-S (que são armazenados em sua base de conhecimento) e serviços para a gestão de Ontologias;
- ***ServiceManager***: inclui facilidades para a gestão de *workflows* e serviços, que no *middleware* são registrados em um repositório;
- ***Owls-MX Matchmaker***: oferece uma consulta semântica baseada no mapeamento de entrada/saída (*Input, Output, Preconditions and Effects*) de perfis escritos em OWL-S, juntamente com demais serviços para a gestão dos perfis em sua base de conhecimento [Klusch et al., 2006].

O *middleware* permite a adição de serviços simples (executa um único serviço da grade), ou compostos (executa um fluxo de serviços como *workflows*). Como precondições, o serviço/*workflow* a ser inserido no *middleware* deve possuir seus respectivos serviços da grade disponíveis na grade computacional (para a implementação e adição de um serviço na grade veja [Sotomayor and Childers, 2006]); e a base de ontologias preestabelecida no repositório para a busca semântica.

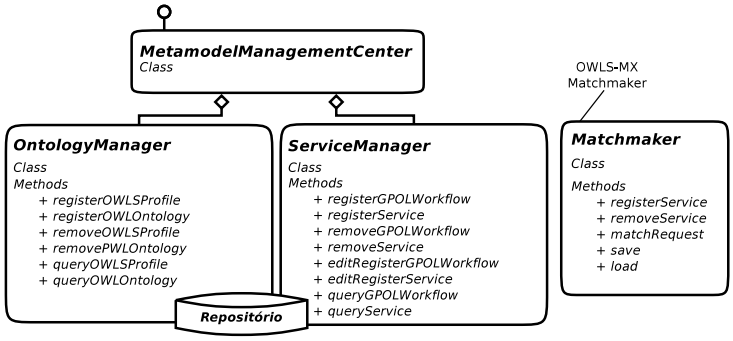


Figura 4.4: Diagrama de classe: Centro de Gestão de Metamodelos

Durante a inclusão de novos serviços/*workflows* no ambiente, o usuário deve adicionar informações sobre a sua descrição semântica (em OWL-S). Para *workflows*, além de informações, é necessário indicar o caminho do arquivo em GPOL contendo a descrição do fluxo da execução. Uma dificuldade nesse processo está relacionada com a utilização de *workflows* descritos em GPOL, já que não existem ferramentas gráficas para a sua construção, transforma-a em uma atividade complexa para usuários leigos.

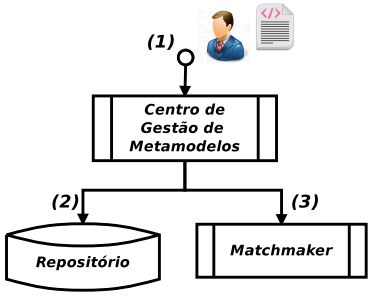


Figura 4.5: Inclusão de serviços

O fluxo é apresentado na Figura 4.5, onde o novo serviço é incluído através do Centro de Gestão de Metamodelos (1). O processo interno persiste as informações em um repositório (2), e em seguida armazena a descrição semântica no repositório do *Matchmaker* (3). Para manter um vínculo entre o repositório do *Matchmaker* e o repositório do *middleware*, o armazenamento do perfil semântico utiliza o identificador a partir do repositório do *middleware*.

No repositório de *workflows* e serviços (implementado como um banco de dados) encontram-se informações utilizadas para a apresentação ao usuário e para a execução na grade. A Figura 4.6 apresenta a modelagem de dados para essas informações.

Durante a consulta de serviços simples apenas o nome (*name*), a descrição (*description*) e os parâmetros (*parameters*) de entrada do serviço são exibidos ao usuário. As demais

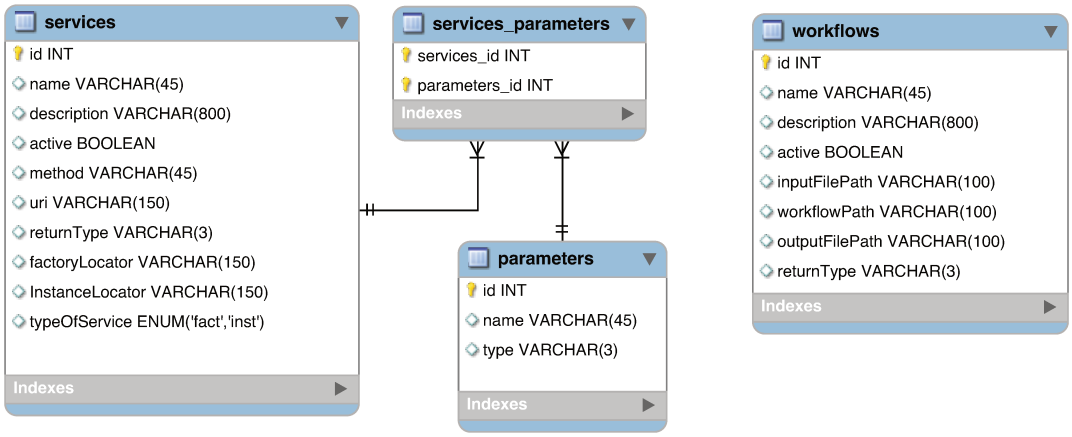


Figura 4.6: Modelagem de dados para Serviços e *Workflows*

informações são utilizadas internamente para a sua execução. O atributo *active* indica se o serviço está ativo ou inativo e apenas serviços ativos são retornados ao usuário. O atributo *method* indica o nome do método a ser executado no serviço, com localização no atributo *uri* (*Uniform resource identifier*). O tipo de retorno do serviço da grade é armazendo em *returnType* (por exemplo *integer*, *string*).

Os elementos *factoryLocator* (*Factory Addressing Locator*) e *instanceLocator* (*Service Addresing Locator*) indicam a localização dos *stubs* para o serviço alvo (serviço de fábrica e de instância, respectivamente). No caso da implementação do padrão fábrica/instância haverá um serviço a mais (o serviço de fábrica) e será necessário a localização de seus *stubs*. Por outro lado, caso o serviço alvo implemente apenas o serviço de instância (*Service Instance*), é necessário indicar apenas o atributo *instanceLocator*. O atributo *typeOfService* indica o tipo do serviço, que pode ser *factory* ou *instance*.

Para serviços compostos, apenas os atributos nome (*name*), descrição (*description*) e o caminho do arquivo contendo os valores de entrada para o *workflow* (*inputFilePath*) são exibidos ao usuário. As informações restantes são utilizadas internamente. Os atributos *active* e *returnType*, novamente, indicam se o *workflow* está ativo ou não e o tipo de retorno, respectivamente; o atributo *workflowPath* mantém o caminho do arquivo em GPOL (no repositório interno) contendo a descrição do *workflow*; e o atributo *outputFilePath* indica o caminho para o arquivo de saída (o resultado) da execução. O arquivo em GPOL contém todas as informações sobre quais serviços fazem parte do fluxo e como devem ser acessados.

4.3.2 Centro de Gestão de Serviços

Este módulo inclui serviços para a descoberta de serviços e *workflows*, além de sua gestão durante a execução na grade computacional. Como demostra a Figura 4.7, sua estrutura interna possui os seguintes componentes:

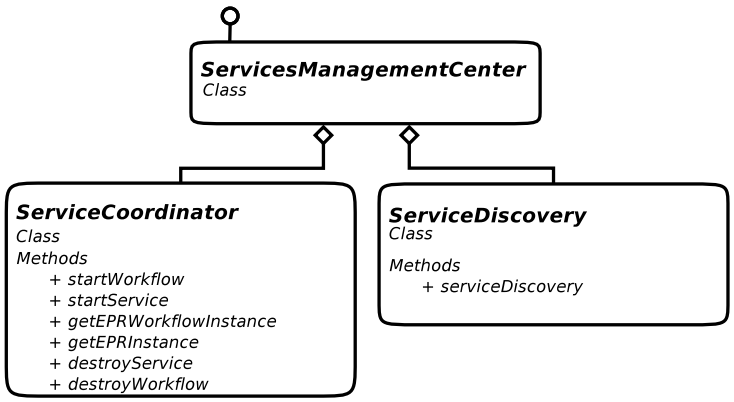


Figura 4.7: Diagrama de classe: Centro de gestão de Serviços

- ***Service Coordinator***: fornece métodos para controle da execução de serviços e *workflows*.
- ***Service Discovery***: realiza a descoberta de *workflows*/serviços. O método utiliza o *Owls-MX Matchmaker* [Klusch et al., 2006] para fornecer um mecanismo de busca semântica.

Neste trabalho utilizamos descrições semânticas para descoberta de serviços e *workflows*. Como exibe a Figura 4.8, o processo de descoberta se inicia a partir do módulo de gestão de serviços (1). Esse módulo possui o serviço *Service Discovery* (2), o qual repassa a requisição ao mecanismo do *Matchmaker* no Centro de Gestão de Metamodelos (3). Internamente, a busca semântica é baseada no casamento entre IOPEs ((*Input, Output, Preconditions and Effects*)), bem como em parâmetros não funcionais, tais como nome, descrição e categoria do perfil semântico enviado e perfis semânticos previamente armazenados no repositório do *Matchmaker*. Associado ao perfil semântico existe um identificador (ID) utilizado pelo *Service Discovery* para realizar o mapeamento no repositório (4). O resultado da busca retorna uma lista de serviços/*workflows* à aplicação em (5) ou (6).

4.3.3 Centro de e-Democracia e e-Governança

Possui serviços voltados à participação do cidadão na administração pública, com o objetivo de aumentar a interação entre cidadãos e seus políticos. Sua infraestrutura inclui os

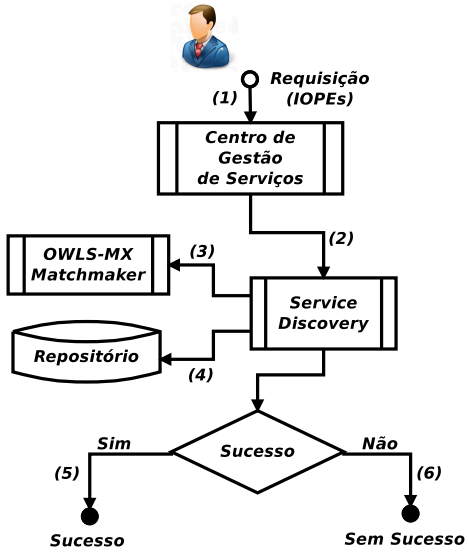


Figura 4.8: Descoberta de serviços

seguintes componentes (veja a Figura 4.9):

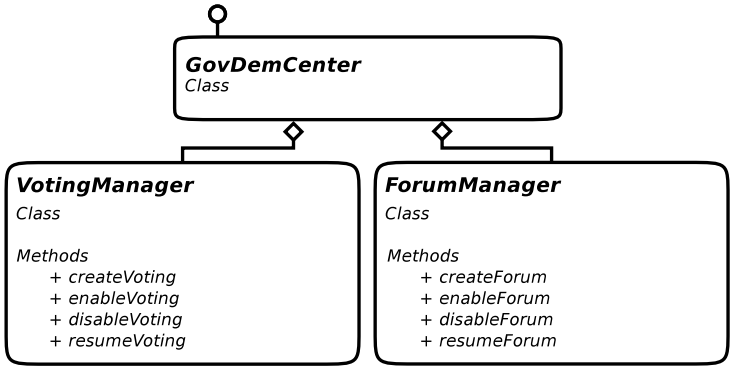


Figura 4.9: Diagrama de classe: Centro de e-Democracia e e-Governança

- ***VotingManager***: fornece funcionalidades para o gereciamento de uma votação eletrônica;
- ***ForumManager***: o cidadão poderá introduzir sua opinião ou mesmo realizar discussões sobre determinado assunto através de um Fórum.

4.3.4 Centro de Rastreabilidade e Auditoria

Inclui serviços com facilidades para rastrear os estágios de determinada execução e sua futura auditoria (após o processo ser completado). Como mostra a Figura 4.10, este

módulo é composto pelos elementos a seguir:

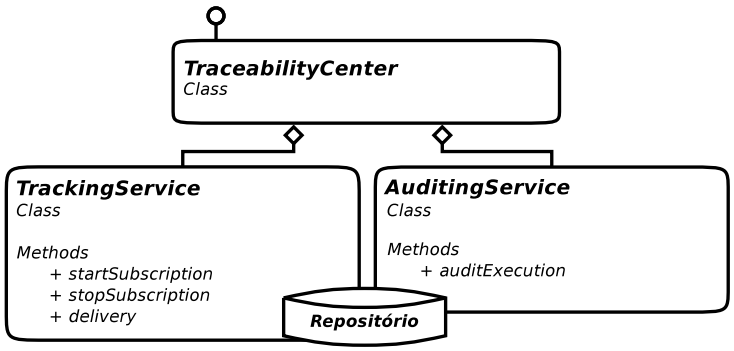


Figura 4.10: Diagrama de classe: Centro de Rastreabilidade e Auditoria

- ***TrackingService***: possui facilidades para realizar o rastreamento das execuções. Internamente é implementado o padrão *WS-Notification* (como cliente) para realizar o monitoramento dos serviços. Para obter informações do serviço é realizada a subscrição dos respectivos recursos (*Resources Properties*) de monitoramento que são exibidos em tempo real.

- ***AuditingService***: realiza a auditoria de serviços concluídos (acessando registros do sistema).

4.4 Integração com o GT4

Em nosso contexto, cada serviço proveniente do setor público é disponibilizado como um serviço da grade. Nesse ambiente, o *middleware* acessa os serviços disponíveis na grade como um cliente. Para isso, utilizamos o *Java WS Core* (veja Seção 2.5.1), que fornece mecanismos para gestão de ciclo de vida dos serviços e recursos, facilidade para execução de tarefas, mecanismos de segurança, entre outros benefícios.

Para a construção dos serviços da grade utilizamos o padrão de implementação fábrica/instância, recomendado pela WSRF, padrão utilizado também pelo GPO. Esse padrão simplifica a gestão de múltiplos recursos, utilizando para isso um serviço de fábrica (*Factory Service*), responsável pela criação de recursos; e um serviço de instância (*Instance Service*), responsável por acessar as informações contidas no recurso, como demostra a Figura 4.11. O cliente solicita a criação de um recurso através do serviço de fábrica, que por sua vez invoca o *Resource Home* para a criação um novo recurso. Este recurso é identificado por um ID (*Identifier*), retornado ao serviço de fábrica e utilizado, em conjunto com a referência para o serviço de instância, para a geração do EPR. De posse do EPR,

o cliente poderá utilizar as facilidades do WSRF para acessar o serviço de instância e controlar seu ciclo de vida.

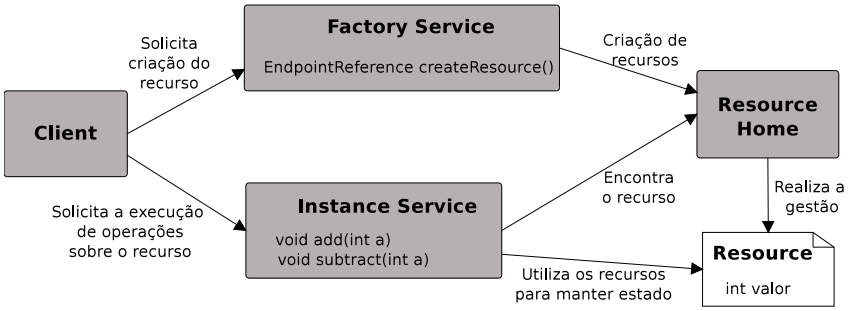


Figura 4.11: Padrão de implementação fábrica/instância

Na camada da grade computacional, camada mais abaixo na infraestrutura, dois serviços foram utilizados para intermediar as execuções de serviços compostos e simples, que são respectivamente: o GPO (apresentado na Subseção 2.5.2), e o MidExec.

O MidExec é acessível pelo *middleware* na grade computacional e seu principal objetivo é a execução de serviços simples. Uma das motivações que levaram a construção de um serviço para intermediar a execução de serviços da grade inclui a utilização de especificações padrão para o monitoramento, gestão do ciclo de vida, geração de registros e determinados aspectos de segurança. Dessa forma o parceiro não precisa se preocupar com a completa implementação desses requisitos.

O GT4 implementa o padrão da especificação *WS-Resource*, onde serviço e recurso são entidades distintas. Nesse padrão o recurso faz a persistência do estado (valores e variáveis). Nesse contexto, o serviço MidExec cria um recurso para cada serviço de instância alvo contendo variáveis para o acompanhamento de sua execução, também utilizadas como base para o seu monitoramento. Essas variáveis são semelhantes às encontradas no GPO para acompanhamento da execução de *workflows*. As variáveis contidas em recursos do MidExec são:

- RP_VALUE: contém o último valor retornado;
- RP_LASTOP: contém a última operação realizada;
- RP_STEPEXEC: apresenta a atividade em execução;
- RP_FAULTVALUE: código da falha ocorrida; e
- RP_FAULTDESC: descrição da falha ocorrida.

Assim como o GPO, o MidExec é um serviço da grade implementado através do padrão fábrica/instância apresentado na Figura 4.11. Internamente o MidExec é composto por um serviço de fábrica (*MidExec Factory Service*), um serviço de instância (*MidExec Service Instance*) e mais dois componentes de uso interno para o gerenciamento de recursos, *MidExec Resource Home* e *MidExec Resource*. O serviço de fábrica possui um método chamado *createResource*, o qual permite a criação de um EPR composto por: uma instância do serviço *MidExec Service Instance*; e um recurso *MidExec Resource*, que inclui as variáveis de processo citadas acima. O gerenciamento dos recursos criados é realizado pelo *MidExec Resource Home* (através dos métodos *create* e *remove*). Após a geração do EPR, a instância do *MidExec Service Instance* permite iniciar a execução do serviço alvo utilizando o método *execService* e finalizar a execução usando o método *destroy*. A Figura 4.12 mostra o diagrama de classes do MidExec.

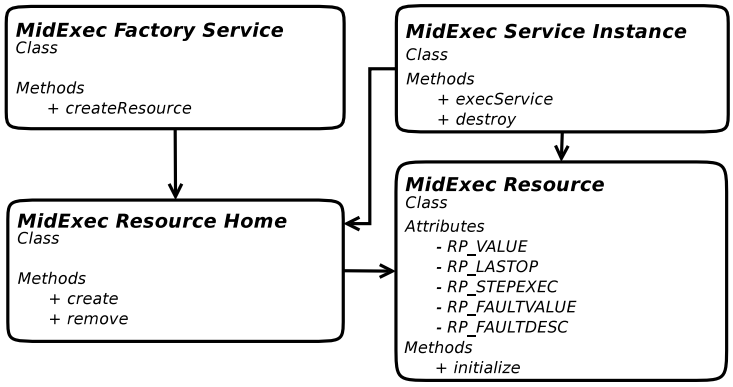


Figura 4.12: Diagrama de classes do MidExec

4.4.1 Execução na grade

O *middleware* possibilita a execução de serviços na grade a partir de requisições de usuários através do barramento. Esses serviços podem ser simples, ou compostos.

Para executar um serviço/*workflow* na grade, o usuário inicialmente realizará uma busca semântica no repositório do *Matchmaker*. O resultado retorna uma lista de, no máximo, quatro serviços/*workflows* e seus respectivos IDs (*identifier*). Utilizando os IDs retornados, uma busca no repositório do *middleware* é realizada para que informações (como nome e descrição) sobre os serviços/*workflows* sejam retornadas ao usuário. Com o resultado, o usuário poderá iniciar uma execução na grade ou realizar uma nova busca no repositório.

O MidExec e o GPO são invocados de forma semelhante pelo Centro de Gestão de Serviços (através do *ServiceCordinator*) no *middleware*. Internamente o *middleware* armazena a URI do serviço de fábrica de ambos. Quando é realizada uma execução na

grade, o *middleware* recupera o *portType* do serviço de fábrica (usando a localização dos *stubs*) e invoca o método *createResource*. Essa ação gera um EPR contendo o novo recurso (que inclui as variáveis de processo para acompanhamento da execução) e um serviço de instância, responsável pela execução na grade. Após esse processo, o EPR retornado é utilizado para recuperar o *portType* do serviço de instância, o que permitirá o acesso aos seus métodos.

Para a execução de um serviço simples, o *middleware* invoca a fábrica do MidExec (Figura 4.13). Após a chamada de seu serviço de fábrica (*MidExec Factory Service*), o *middleware* interage (através do *portType*) diretamente com o serviço de instância gerado (*MidExec Service Instance*). Internamente o método *execService(infoService)* é utilizado para execução do serviço alvo, que recebe como parâmetro as informações do serviço (como nome do método, URI, entre outros) e realiza sua execução na grade. Ao término do processo, o *middleware* destrói a instância e retorna o valor resultante ao usuário. Durante a execução, a instância do MidExec atualiza suas variáveis de processo permitindo o seu monitoramento.

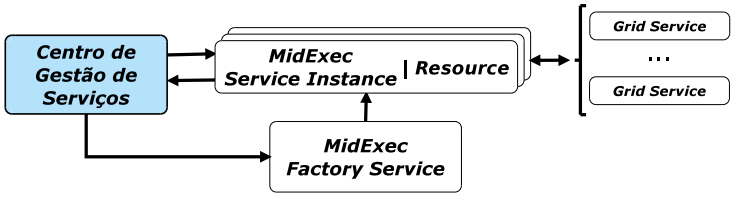


Figura 4.13: Execução de serviços simples

Para a execução de um *workflow*, o *middleware* invoca o serviço de fábrica do GPO (*GPOMaestro Factory Service*) (Figura 4.14). Após a geração do EPR, o *middleware* acessa (através do *portType*) o serviço de instância do GPO (*GPOMaestro Instance Service*) e inicia a execução ao invocar o método *submit(workflow_file)*. Esse método recebe como parâmetro o caminho de um arquivo GPOL contendo a descrição do *workflow* a ser executado. Internamente, a instância do GPO verifica sua correção sintática e, caso não haja erros, inicia a execução na grade. Com a execução concluída o resultado é retornado ao *middleware*. Durante toda execução a instância do GPO atualiza suas variáveis de processo, o que permite o seu monitoramento.

O GPO pode ser utilizado dentro de um *workflow*, permitindo níveis de *workflows*, uma vez que o GPO é um serviço da grade e pode ser invocado no fluxo de execução de um *workflow* como qualquer outro serviço.

A Figura 4.15 apresenta o diagrama de sequência e seus respectivos componentes presentes na arquitetura do *middleware*, para a execução de um *workflow* na grade através do GPO. A aplicação requisita a execução na grade junto ao *middleware* utilizando o serviço *Service Cordinator* presente no Centro de Gestão de Serviços (CGS), como apresentado

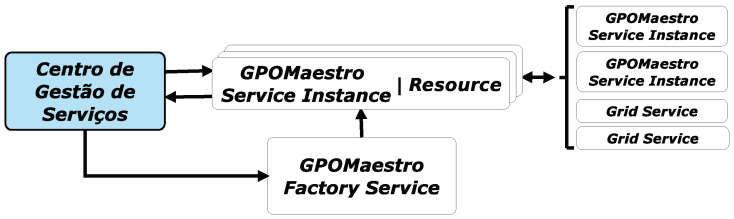


Figura 4.14: Execução de workflows

na Figura 4.2. Antes de iniciar a execução, note que o método *getEPRWorkflowInstance* (1) é responsável pela execução do serviço de fábrica do GPO e por retornar o EPR de sua instância para a aplicação (2 e 3). Esse passo é realizado de forma transparente para o usuário e torna-se necessário para que a aplicação possa manter o EPR e utilizá-lo em outras funções. No passo seguinte, a aplicação invoca o método *startWorkflow* (4) que iniciará a execução na grade através de dois passos: recuperação do *portType* do serviço de instância (5) e submissão do *workflow* (6) ao serviço de instância do GPO.

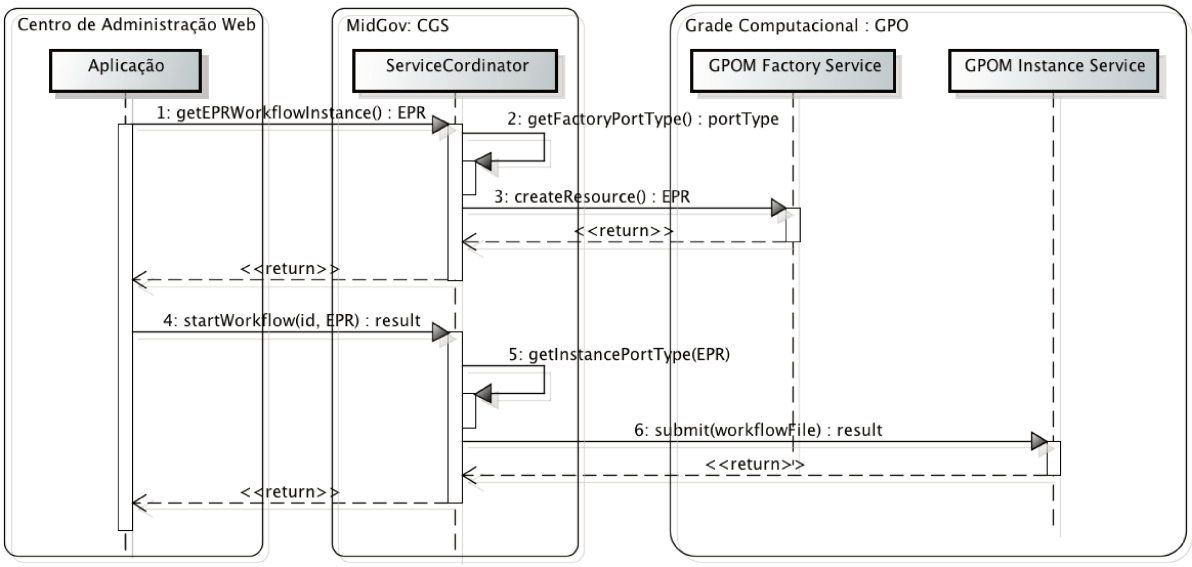


Figura 4.15: Diagrama de sequência para a execução na grade

A execução de serviços simples através do MidExec é idéntica, com a diferença de que é invocado o método *getEPRInstance* do serviço *ServiceCondinator*, responsável por acessar a fábrica do MidExec e gerar o EPR; e o método da instância do MidExec se chama *execService(infoService)*, responsável pela execução do serviço alvo na grade.

Para terminar o processo manualmente utilizamos a facilidade de gestão do ciclo de vida oferecido pela especificação *WS-ResourceLifetime*, que é implementada também pelo

GPO. Para utilizar esse recurso basta estender o *portType* do MidExec com a especificação *ImmediateResourceTermination* do padrão WSRF. Essa especificação adiciona uma operação *destroy* ao *portType* do MidExec que permite realizar o término de sua instância imediatamente. A Figura 4.16 apresenta o diagrama de sequência para esse processo, onde o EPR (1) é passado pela aplicação ao *middleware* que, após recuperar o *portType* (2) do serviço de instância, a destrói (3), terminando o processo.

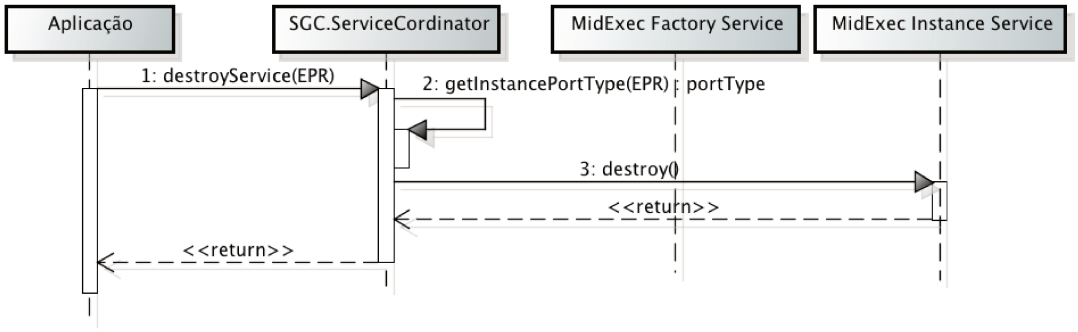


Figura 4.16: Término da Instância do Serviço MidExec

4.4.2 Monitoramento

Para manter o monitoramento sobre as execução na grade utilizamos a especificação *WS-Notification*, a qual permite que o cliente seja notificado quando eventos acontecem no servidor. A abordagem é a seguinte:

1. O cliente solicita ao servidor que o notifique assim que acontecer determinado evento. Esse passo é chamado de *subscription*;

2. O cliente e o servidor voltam para as suas atividades e, quando acontecer o evento especificado, o servidor envia uma notificação ao cliente.

A familia da especificação *WS-Notifications* fornece um conjunto de interfaces padrões para a utilização de notificações em Serviços Web. O GT4, embora implementado de forma parcial, a especificação permite que as variáveis do recurso (*Resource Properties*) sejam definidas como tópicos (*topic*). Dessa forma, sempre que houver mudança no valor das variáveis especificadas, uma notificação será enviada ao cliente.

Em nosso contexto o cliente é o *middleware* e a subscrição (*subscription*) ocorre nas variáveis de processo do MidExec e do GPO, variáveis a serem monitoradas. Para implementar a especificação *WS-Notification* é necessário estender o *portType* do serviço de

instância do MidExec e do GPO com o padrão *NotificationProducer* e *SubscriptionManager*, que expõe a operação *Subscribe* utilizada para subscrever as variáveis de processo; e a operação *destroy* para destruir a subscrição. No lado do cliente, o *middleware* implementa (através do Centro de Rastreabilidade e Auditoria) a interface *NotifyCallback*, que requer que seja implementado um método *deliver*, responsável por tratar as notificações recebidas e encaminhá-las à aplicação. No início da execução, o usuário poderá escolher se deve haver o monitoramento ou não.

A Figura 4.17 exibe o diagrama de sequência para a execução na grade utilizando *WS-Notification*. Inicialmente a aplicação invoca o método *startworkflow* (1) para a execução de um *workflow* passando o EPR do serviço de instância. Utilizando o EPR, o *middleware* obtém o *portType* (2) do serviço de instância e em seguida realiza a subscrição (3) das variáveis de processo. No passo seguinte, o *middleware* realiza a submissão do *workflow* (4) e à medida em que o mesmo é executado e suas variáveis de processo são atualizadas, notificações são entregues à aplicação (5, 6, 7 e 8). Após o término do processo, o método *destroy* (9) finaliza a subscrição.

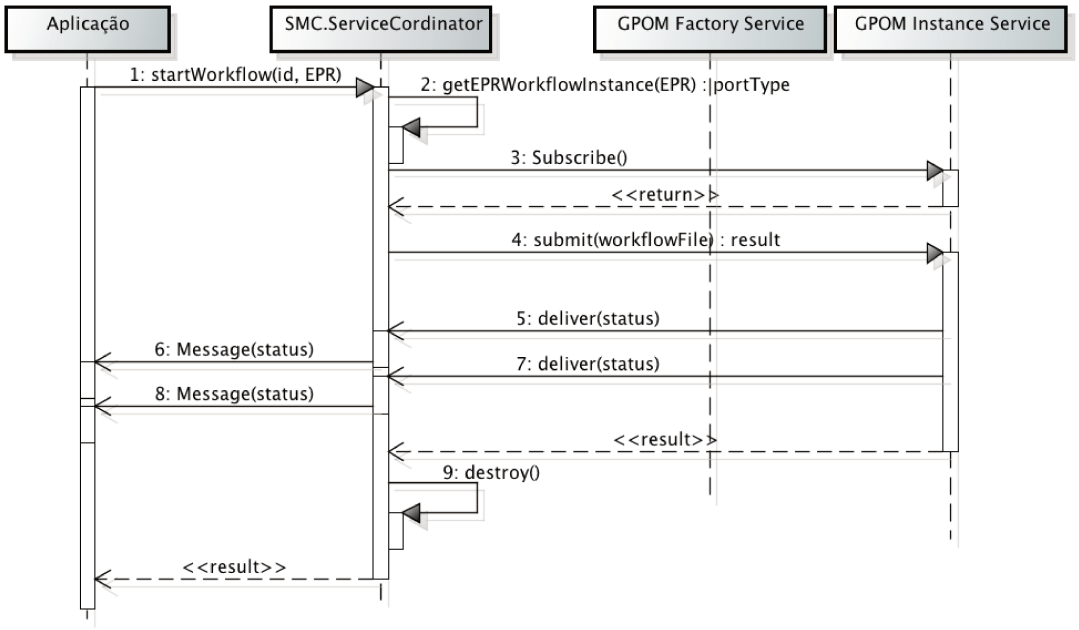


Figura 4.17: Diagrama de sequência para o processo de monitoramento

4.4.3 Auditoria

O registro de *logs* é uma efetiva ferramenta para encontrar problemas no funcionamento do sistema. Para a auditoria, registros de *logs* foram realizados para as principais ações no

middlware, no serviço MidExec e no GPO, que por sua vez, são armazenados no sistema de arquivos. Realizamos o registro de *logs* das seguintes ações:

- Na realização de registro, edição e remoção de serviços, *workflows* e ontologias;
- Na invocação do MidExec ou o GPO, e na geração de exceções para conexões remotas (*RemoteException*);
- Na execução do serviço da grade pelo MidExec/GPO (gerando registros de *logs* para casos de exceções);
- No término da execução (com sucesso ou não);

4.4.4 Integração com a infraestrutura de segurança

Segurança é uma das principais preocupações no contexto de grade computacional e de e-Governo [KBSt, 2008; Sotomayor and Childers, 2006]. Esse cenário envolve membros em múltiplos domínios administrativos compartilhando recursos. O GT4 fornece uma infraestrutura de segurança, GSI, que cobre diversos aspectos em ambientes de computação em grade. Todavia, para ambientes de e-Governo é preciso um trabalho mais detalhado nesse sentido para que todas as requisições do ambiente sejam atendidas. Neste trabalho, decidimos utilizar os mecanismos de segurança presentes no GT4.

As comunicações são sempre mutualmente autenticadas (exceto para autenticações anônimas) utilizando certificados (X.509) no GT4. Ele suporta o método de autenticação *GSI Transport* (mecanismo padrão), o qual constrói um canal de segurança sobre a Internet com maior desempenho (se relacionado aos outros métodos suportados) evitando a escuta e a violação do conteúdo. Em nossa pesquisa realizamos a implementação desse mecanismo.

Em relação à privacidade e integridade dos dados, a GSI permite estabelecer a comunicação de duas maneiras: utilizando assinaturas digitais, o que garante a integridade e não a privacidade; ou por meio da encriptação, que garante a integridade e privacidade. Optamos por utilizar comunicações com encriptação, dado o caráter crítico do conteúdo transmitido.

Em relação à autorização dos membros, a GSI permite que os parceiros possam customizar um grupo de autorizações, tornando o seu serviço acessível apenas para os usuários e/ou serviços especificados. Outro aspecto da GSI é a autenticação anônima, cujo princípio é permitir ao usuário o envio de requisições anônimas, não autenticadas. Esse tipo de autenticação é importante para sistemas administrativos uma vez que permite manter requisitos referentes ao sigilo durante, por exemplo, o processo de votação de uma enquete.

No GT4, a segurança do cliente e do servidor é configurada separadamente na grade. A segurança do servidor é configurada através do descritor de segurança, especificando como e quem tem autoridade para acessar o serviço. O cliente, por outro lado, pode fazer essa especificação de segurança programaticamente, o que permite que o usuário especifique os modos de segurança sobre o *middleware*. Entretanto, a segurança do *toolkit* está associada apenas ao ambiente da grade. Entre a plataforma e o portal de acesso ao sistema estão inseridos outros artefatos para a segurança.

4.5 Aspectos de Implementação

Um protótipo do *middleware* proposto com as principais funcionalidades foi implementado utilizando a linguagem Java versão 1.6. A Tabela 4.6 apresenta o estado da implementação.

Tabela 4.6: Estado da Implementação

Módulo do *Middleware*	Estado
Centro de Gestão de Serviços	Completa
Centro de Gestão de Metamodelos	Completa
Centro de e-Democracia e-Governância	Parcial (20%)
Centro de Gestão Rastreabilidade e Auditoria	Parcial (70%)
Integração com a Segurança da grade	Completa
Execução na grade	Completa
Monitoramento	Parcial (90%)
Auditoria	Parcial (70%)

Para possibilitar maior integração junto ao *Java WS Core*, a linguagem Java foi escolhida para a implementação. *Java WS Core* fornece métodos para a criação de recursos, geração do EPR, entre outras funcionalidades. O barramento de serviços exporta Serviços *Web* utilizando Apache CXF sobre o servidor Tomcat (versão 6.0.20). Tais Serviços *Web* são acessíveis através de um portal construído utilizando JSP (*JavaServer Pages*), HTML (*HyperText Markup Language*) e Java *Actions*. Para geração de registro de *log* utilizamos o *Log4J*.

Buscas semânticas foram realizadas utilizando as facilidades do matchmaker proposto em [Klusch et al., 2006], implementado em Java. O Mathmaker armazena os perfis semânticos (em OWL-S) em seu repositório interno utilizando a tecnologia Jena. As descrições de *workflows* (em GPOL) e ontologias (em OWL) são armazenadas em subdiretórios da plataforma enquanto os demais repositórios são implementados com o banco de dados MySQL, onde ficam informações de serviços e *workflows*.

4.6 Resumo conclusivo

Neste capítulo foi apresentado o MidGov, um *middlware* para aumentar a integração e colaboração no ambiente governamental. Descrevemos sua infraestrutura interna, seus módulos e comunicações. Apresentamos como é realizada a integração com serviços na grade computacional, construída através do GT4, e o mapeamento das necessidades de execução, monitoramento, auditoria e requisitos de segurança. Embora a arquitetura do MidGov seja centralizada, ela permite a execução em um ambiente descentralizado e heterogêneo através dos serviços MidExec e GPO.

A implementação do *middleware* demandou esforços significativos. O código foi construído utilizando as facilidades da *Java WS Core* que implementa o padrão WSRF somado ao padrão *WS-Notification*. A integração com serviços implementados e facilidades incluídas do *toolkit* GT4, como a GSI, apresentaram-se de forma simples, todavia, com considerável quantidade de informações para acesso e configuração.

Uma ponto crítico foi a realização de registro de *logs* internos aos serviços executados nos diferentes domínios. Dessa forma é necessário um mecanismo de captura de *logs* distribuído para que o *middleware* obtenha registro de *log* do sistema de forma completa, o que o torna mais transparente, requisito com forte presença em aplicação de e-Governo.

Para maior interoperabilidade do *middleware* com o GPO ainda é necessário mais trabalho. O GPO não separa a definição de *workflows* abstratos de *workflows* concretos e não permite a atribuição de valores aos parâmetros de forma programável. Dessa forma, maior esforço deve ser realizado para que o GPO possa ser utilizado em um ambiente automatizado e com menor interferência humana.

Capítulo 5

Validação do Middleware

Para a validação da proposta do *middleware* construímos dois cenários de teste. O primeiro cenário utiliza o *middleware* para a execução de um serviço simples. Esse serviço implementa a validação do CPF (Cadastro de Pessoa Física), apresentado na Seção 5.1. O segundo cenário, presente na Seção 5.2, realiza a execução de um serviço composto (como *workflow*) na grade. O resumo e algumas considerações sobre o capítulo são o assunto da Seção 5.3.

5.1 Cenário 1: validação do número de CPF

Nesse primeiro cenário realizamos a validação de um serviço simples (execução de um único serviço) na grade através do MidGov. O principal motivo da escolha do serviço é que se trata de um serviço real e a sua implementação pode ser realizada através de um único serviço, o que permitirá validar todo o processo de execução de serviços simples, desde a sua descoberta através de um perfil semântico até o resultado final do processo. Serviços simples estão presentes com frequência em aplicações de e-Governo (como exemplo, consulta de CPF, consulta ao Imposto de renda, entre outros).

O cenário consiste na implementação do algoritmo de validação numérica do CPF (Cadastro de Pessoa Física). Esse documento é único e registra o cidadão na Receita Federal brasileira. Contém informações sobre o cidadão e é necessário em diversas ocasiões. Ao ser emitido, o CPF gera um número de onze algarismos, sendo os dois últimos dígitos números que possibilitam a verificação automática e a prevenção de erros de digitação.

O algoritmo de verificação do CPF calcula o primeiro dígito verificador a partir dos nove primeiros dígitos do CPF, e em seguida calcula o segundo dígito verificador utilizando os nove primeiros dígitos mais o primeiro dígito verificador.

Vamos utilizar como exemplo o seguinte CPF fictício: 222.124.248-35.

Para calcular o primeiro dígito verificador utilizamos os nove primeiros dígitos do

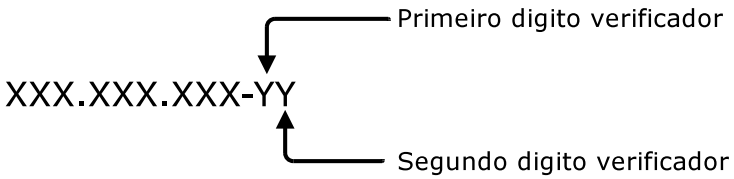


Figura 5.1: Cadastro de Pessoa Física

CPF (222.124.248) e multiplicamos cada dígito, da direita para a esquerda, por um valor crescente a partir do número 2. Dessa forma temos:

2	2	2	1	2	4	2	4	8
10	9	8	7	6	5	4	3	2
20	18	16	7	12	20	8	12	16

A soma dos resultados é igual a: $20 + 18 + 16 + 7 + 12 + 20 + 8 + 12 + 16 = 129$. De posse do resultado, realizamos a divisão por uma constante 11. $129 \div 11 = 11$ com resto 8. Caso o resto da divisão seja menor que 2, então o dígito verificador é zero (0); caso seja maior ou igual a 2 então o dígito verificador é igual a: 11 menos o resto da divisão ($11 - 8 = 3$). Como o primeiro dígíto verificador está correto, precisamos realizar o cálculo do segundo dígito verificador.

Para calcular o segundo dígito verificador iremos realizar o mesmo processo, porém utilizando também o primeiro dígito verificador (222.124.248-3). Dessa forma temos:

2	2	2	1	2	4	2	4	8	3
11	10	9	8	7	6	5	4	3	2
22	20	18	8	14	24	10	16	24	6

Novamente, a soma é igual a: $22 + 20 + 18 + 8 + 14 + 24 + 10 + 16 + 24 + 6 = 162$. A divisão por 11 será: $162 \div 11 = 14$ com resto 8. Após esse processo teremos que o segundo dígito verificador é $11 - 8 = 3$. Portanto o número de CPF de entrada é inválido.

5.1.1 Construção do serviço

Implementamos o algoritmo de validação de CPF como um serviço da grade chamado *FactCCPFService* (*Factory Check CPF Service*). Sua função é receber o CPF como entrada, realizar os cálculos predefinidos e retornar uma saída contendo (uma cadeia de caracteres) "*Valid CPF!*" se o CPF é válido e "*Invalid CPF!*" caso contrário. Utilizamos dois recursos computacionais: Nix (Intel Core 2 Quad, 2.6 Ghz, 8GB RAM) e Zeus (Intel Xeon, 2.66Ghz, 12 GB RAM). Nesse contexto ambos os recursos contém o GT4

instalado, sendo que o *middleware* está localizado em Nix (IP 10.3.77.15) e o serviço *FactCCPFService* está disponível em Zeus (IP 10.3.77.5).

Inicialmente, é necessário adicionar informações do serviço ao *middleware*, além do perfil semântico, para que o mesmo fique disponível através de seu repositório. As informações inseridas foram:

- *Name*: *Check CPF*;
- *Description*: *CPF Numerical Validation (Natural Persons Register)*;
- *Method*: *check*;
- *OWL-S File*: */Users/geraldoms/checkCPF.owls*;
- *URI*: *http://10.3.77.15:8080/wsrf/services/examples/core/ccpf/FactCCPFService*;
- *Parameter Name*: *CPF*;
- *Parameter Type*: *String*;
- *Return Type*: *String*;
- *Factory Addressing Locator*: *org.globus.examples.stubs.CCPF.service.FactCCPFServiceAddressingLocator*;
- *Service Addresing Locator*: *org.globus.examples.stubs.CCPFService_instance.service.CCPFServiceAddressing Locator*; e
- *Type of Service*: *fact*.

Essas informações indicam inclusive a localização dos *stubs* (do servico de fábrica e de instância) e se o serviço é de fábrica ou instância (*Type of Service*), necessários no momento da execução do serviço na grade. A busca semântica é realizada utilizando o nome, descrição e as definições dos termos para *Input*, *Output*, *Preconditions and Effects* (IOPE) para o serviço em questão. Em seguida, a execução na grade é realizada através do *middleware* a partir de uma requisição da aplicação.

5.1.2 Descoberta e execução do serviço

O processo de descoberta e execução do serviço é apresentado na Figura 5.2. A decoberta inicia com uma requisição da aplicação (passando um perfil semântico) para o Centro de Gestão de Serviços, solicitando uma busca por serviços/*workflows* através do método *ServiceDiscovery* (*CGS.ServiceDiscovery*) (**1**). Nesse ponto, o método *matchRequest* do *Matchmaker* (*CGM.Owls-Mx.matchRequest*) é invocado (**2**). O resultado da consulta é retornado à aplicação. Como passo inicial à execução, é gerado um serviço de instância (em **3** e **4**) do MidExec (*MidExec Instance Service*) através do método *createResource* do serviço de fábrica (*MidExec Factory Service*) e retornado à aplicação (**5**). Em seguida o *middleware* recupera o *portyType* do serviço de instância do MidExec e solicita a execução do serviço alvo (em **6** e **7**) passando as informações necessárias e retornando o resultado à aplicação (**8**).

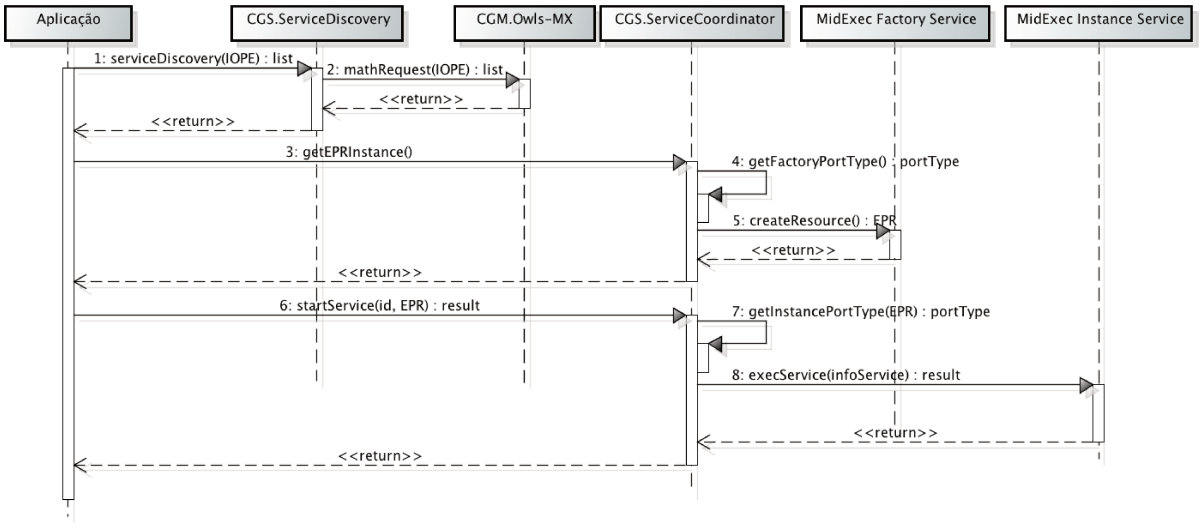


Figura 5.2: Diagrama de sequência para a descoberta e execução do Serviço

O *middleware* inicia a execução na grade através do serviço de instância do MidExec. Esse serviço cria uma instância do serviço *FactCCPFService* e acessa o seu método *check*, responsável pela verificação. Internamente as variáveis de processo são atualizadas para que a aplicação esteja ciente do andamento do processo. Ao final, o resultado é entregue ao *middleware* que realiza a destruição da instância do MidExec responsável pela execução.

5.2 Cenário 2: Cadastro Sincronizado Nacional

No segundo cenário realizamos a validação de um serviço composto (execução de um *workflow*) na grade através do MidGov. Escolhemos o serviço de Cadastro Sincroni-

zado Nacional (CadSinc) por tratar-se de um serviço real, multi-organizacional e que executa diversas atividades internamente, como um *workflow*. Aplicações de e-Governo como *workflows* estão frequentemente presentes na administração pública e representam a interação entre os diversos órgãos/departamentos para fornecer determinado serviço.

O CadSinc [Receita Federal, 2003] teve início com a Emenda Constitucional nº 42/2003 e representa um esforço da administração pública para a simplificação de processos cadastrais, com a consequente redução de custos e prazos. O CadSinc representa a integração dos procedimentos cadastrais relativos às Pessoas Jurídicas nas três esferas do Governo. Não se trata de um cadastro único e sim uma sincronização entre os diversos cadastros existentes, mantendo-se a necessidade de informações específicas de cada um. Em resumo, as etapas internas são [Receita Federal, 2003]:

1. **Consultas Prévias**: o empreendedor fornecerá informações sobre o futuro negócio para verificar sua adequação junto às exigências dos mais diversos órgãos governamentais. Essas informações são repassadas aos órgãos responsáveis por licenças e alvarás de funcionamento. Caso não haja restrições, o cidadão receberá eletronicamente a autorização para utilização do nome empresarial escolhido e do endereço pretendido.

2. **Registro Público e Inscrição Tributária**: De posse da autorização (concebida no passo anterior), o cidadão solicitará o registro do negócio no órgão apropriado (Junta, Cartório, etc). Caso tudo corra bem, o cidadão receberá o seu ato constitutivo já devidamente registrado, bem como as respectivas inscrições (CNPJ e inscrição(ões) estadual e/ou municipal, além do alvará de funcionamento).

A Figura 5.3 apresenta uma representação gráfica do fluxo de atividades do CadSinc (simplificado), passando pelos diversos órgãos administrativos, para a realização do cadastro de pessoa jurídica.

Após o cidadão ter escolhido o nome empresarial e a atividade econômica, o fluxo do cadastro jurídico segue as seguintes etapas: **(a)** O cidadão utiliza o PGD (Programa Gerador de Documentos) CNPJ para criar o DBE (Documento Básico de Entrada), que é enviado ao JUCE (Junta Comercial) e à Prefeitura Municipal. Em seguida, **(b)** a prefeitura realiza uma consulta prévia quanto à possibilidade de exercer a atividade no local pretendido com base em regras de zoneamento. O DBE é enviado ao JUCE **(c)**, possivelmente com os demais documentos relacionados à liberação da Receita Federal. Caso não seja possível o exercício da atividade **(d)**, a JUCE é notificada. Em caso de aprovação pela Prefeitura **(e)**, é realizado o cadastro em nível federal e estadual. Continuando o fluxo, uma consulta é realizada na Receita Federal **(f)** e caso não existam restrições, uma autorização é enviada à JUCE **(g)**. Após todos esses trâmites, a JUCE poderá ou não autorizar o registro da empresa, de acordo com a documentação recebida.

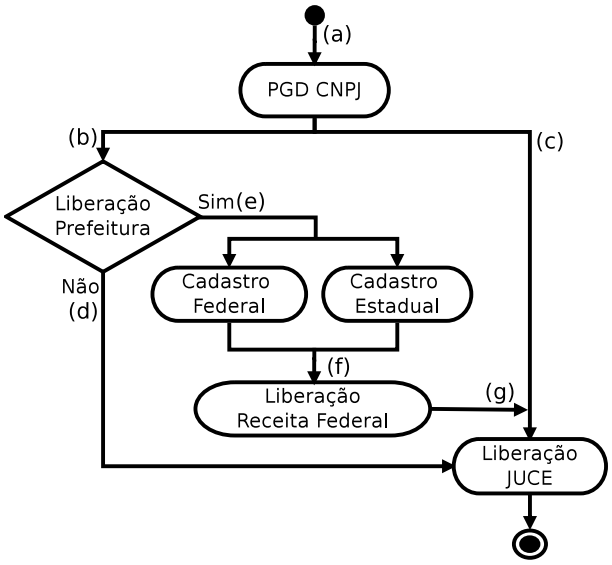


Figura 5.3: Fluxo de atividades para o CadSinc

5.2.1 Construção do workflow

Uma possível solução para o CadSinc é a utilização de um *workflow* (fluxo de trabalho) para orientar a comunicação e execução de atividades entre os diversos órgãos administrativos. O processo pode ser representado como um grafo acíclico direcionado (*directed acyclic graph*), como mostra a Figura 5.4. Nesse grafo os vértices representam serviços (processamento de informações) realizados pelos respectivos órgãos e as arestas a comunicação entre eles.

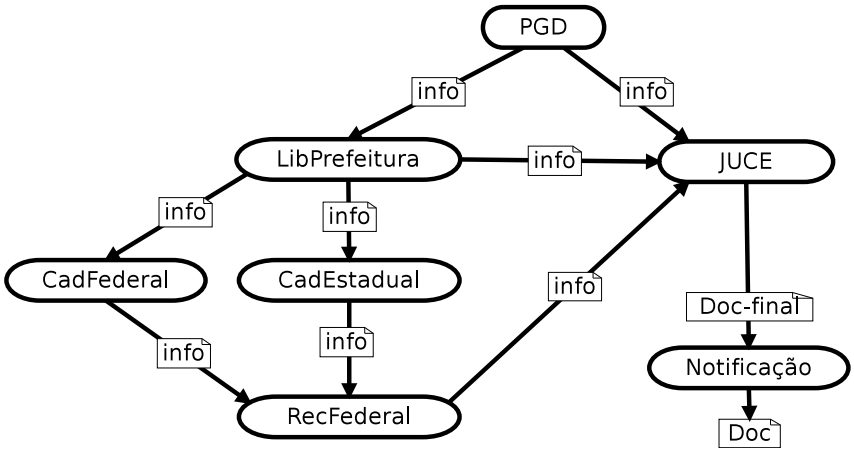


Figura 5.4: Workflow CadSinc

O primeiro passo para a execução de um *workflow* no *middleware* proposto é a cons-

trução dos serviços da grade que farão parte do fluxo de execução. Todavia, a implementação real dos serviços do CadSinc para a validação do sistema não é viável para esta dissertação. Sendo assim, utilizamos o ambiente de emulação descrito em [Senna et al., 2011], com o propósito de facilitar a construção do cenário.

O ambiente proposto em [Senna et al., 2011] permite a construção de um *workflow* emulado com comportamento similar ao *workflow* real. Para esse propósito é introduzido um serviço da grade chamado *FactRCService* (*Factory Random Content Service*), que recebe arquivos de entrada e gera arquivos de saída com conteúdo aleatório. Esse serviço implementa três métodos: *Data Distribution*, *Data Aggregation* e *Process*. O método *Data Distribution* recebe dados de uma única fonte e produz múltiplas saídas. O método *Data Aggregation* recebe dados a partir de múltiplas fontes e produz uma única saída. E finalmente, o método *Process* recebe dados a partir de uma fonte e um tempo mínimo estipulado de execução, e produz apenas uma saída. Durante o tempo estipulado de execução, o método *Process* executa um *benchmark* para emular o processamento de dados.

Em nosso *workflow* emulado, utilizamos o método *Process* seguido do método *Data Distribution* em todos os vértices cuja entrada de dados é gerada por uma única fonte e que produz múltiplas saídas. Para vértices com múltiplas entradas e apenas uma saída, utilizamos o método *Data Aggregation* seguido do método *Process*. Para vértices com apenas uma entrada e uma saída, utilizamos apenas o método *Process*. A comunicação (envio de arquivos) entre vértices é realizada pelo comando *scp* do sistema operacional Linux. Utilizamos cinco recursos computacionais para o cenário: Nix (Intel Core 2 Quad, 2.6 Ghz, 8GB RAM), Apolo (Intel Pentium IV, 3.0Ghz, 2GB RAM), Zeus (Intel Xeon, 2.66Ghz, 12 GB RAM), Cronos e Dionisio (Intel Core 2 Quad, 2.4 Ghz, 4GB RAM). Nesse contexto o *middleware* está localizado em Dionisio e cada um dos recursos contém o GT4 instalado e o serviço *FactRCService* disponível. A Figura 5.5 apresenta esse fluxo de execução.

Como passo inicial à execução é necessário realizar o cadastro do *workflow* no *middleware*. No processo de inserção de informações, o *middleware* requer o caminho da descrição do fluxo em GPOL e seu perfil semântico em OWL-S (ambos presentes no Apêndice A e no Apêndice B, respectivamente). Após a inclusão das informações, o Centro de Gestão de Metamodelos persistirá as informações no repositório do *middleware*. As informações inseridas são (Figura 5.6):

- *Name*: *National Synchronized Register*;
- *Description*: *A National Synchronized Register project (CadSinc), an initiative to simplify the process of registration of legal entities (like companies and stores) in the three spheres of government: municipal, state and federal levels*;

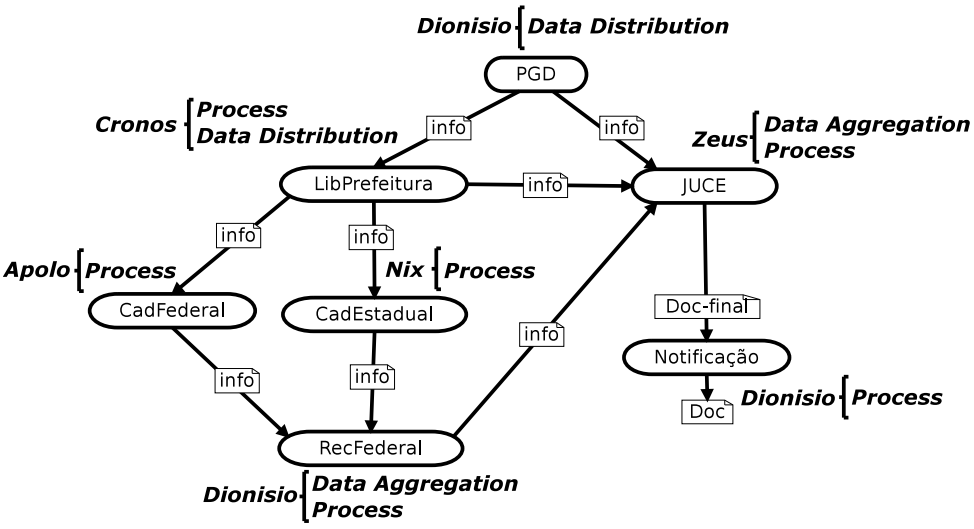


Figura 5.5: Execução do workflow

- *Workflow Path*: */tmp/Workflows/cadSinc.gpol*;
- *OWL-S File*: */Users/geraldoms/cadSinc.owls*;
- *Input File Path*: */tmp/Docs/documento.txt*;
- *Output File Path*: *OutputFile/documento_final.txt*; e
- *Return Type*: *INT*.

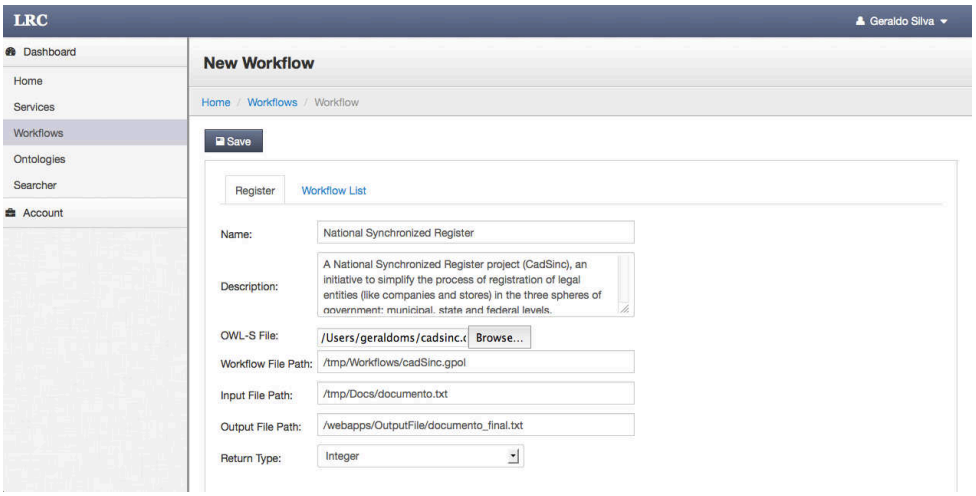


Figura 5.6: Cadastro do workflow

Para a realização de consultas no repositório é necessária a construção de um perfil semântico contendo informações de busca: nome do serviço, descrição e os *Input, Output, Preconditions and Effects* (IOPE) (definições dos conceitos). Como resultado, o *middleware* apresenta a lista de serviços que "casam"(*match*) com a requisição. O resultado para a busca é retornado ao cliente (*a aplicação*) que iniciará a execução do respectivo serviço/*workflow*.

5.2.2 Descoberta e execução do Workflow

O processo de descoberta e execução do *workflow* segue os estágios ilustrados na Figura 5.7. Inicialmente a aplicação envia uma requisição (contendo o perfil semântico) para o Centro de Gestão de Serviços solicitando uma busca por serviços/*workflows* (*CGS.ServiceDiscovery*) (**1**). Em seguida, facilidades do *Matchmaker* são utilizadas para realizar uma busca semântica (*CGM.Owls-Mx.matchRequest*) por perfis em sua base de conhecimento (**2**). O resultado é retornado à aplicação. Como pré-requisito à execução, é criada uma nova instância do GPO (*GPOM Instance Service*) através do seu serviço de fábrica (*GPOM Factory Service*) e o seu EPR é retornado à aplicação (em **3**, **4** e **5**). Continuando o processo, a aplicação solicita ao *middleware* a execução do *workflow* indicado através do método *startWorkflow* (*CGS.ServiceCoodinator.startWorkflow*) (**6**). Com a requisição, o *middleware* recupera o *portType* do serviço de instância do GPO (**7**) e o executa (**8**), retornando o seu resultado para a aplicação.

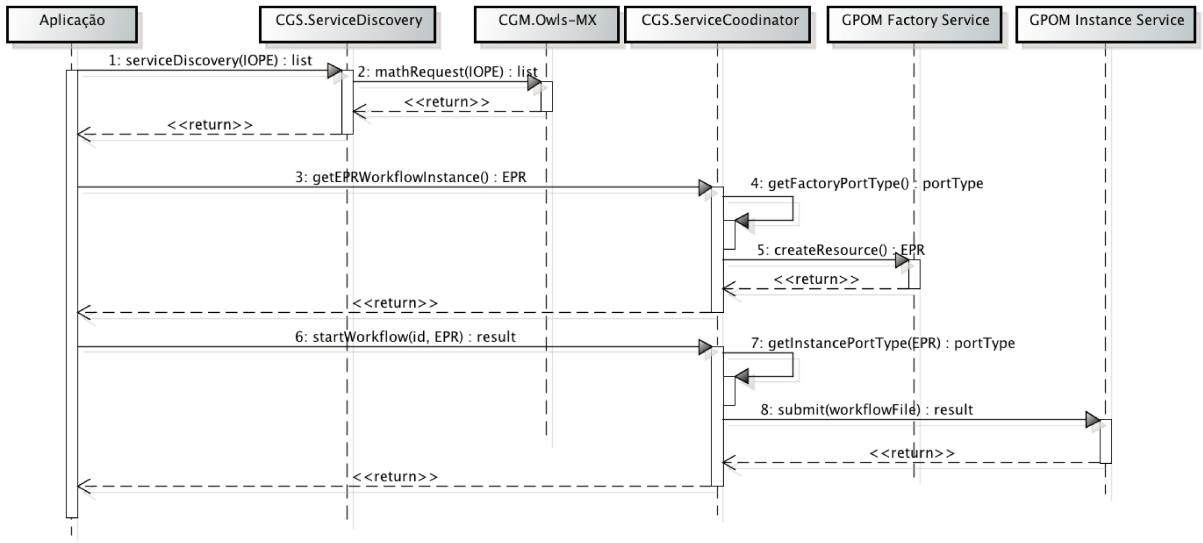


Figura 5.7: Diagrama de sequência para a descoberta e execução do *workflow*

O *middleware* inicia a execução do *workflow* na grade com a criação de uma instância

do GPO (*GPO Maestro Instance Service*). A instância do GPO realiza a criação de uma instância do *FactRCService* em cada um dos respectivos recursos computacionais e inicia o fluxo de execução do *workflow* alvo. Internamente o GPO atualiza suas variáveis de processo para serem utilizadas no monitoramento. No final da execução o GPO gerará um arquivo de saída no caminho indicado em *Output File Path*, e o *middleware* o disponibilizará para a aplicação.

5.2.3 Avaliação Qualitativa

Durante o processo de implementação do cenário do CadSinc, percebemos que os esforços no desenvolvimento concentravam-se, em sua maioria, na inclusão de informações do *workflow* no MidGov e na codificação do *workflow* em GPOL. Todavia, grande complexidade na definição de processos públicos foi transferido para os componentes de *middleware*.

O cenário foi construído através de um emulador, e cada etapa do processo possui um tempo definido. Dessa forma detalhes sobre o *overhead* e demais análises quantitativas do ambiente fornecido pela grade computacional não foram evidenciados. Contudo, requisitos de interoperabilidade técnica e semântica foram satisfeitos com sucesso. O controle centralizado através do MidGov apresentou eficiência, permitindo que as instâncias de serviços do GPO fossem executadas e terminadas prontamente a partir do Centro de administração *web*.

As funcionalidades do MidGov são totalmente expostas como serviços *web* tradicionais para facilitar a interoperabilidade com as aplicações e também com outras plataformas de e-Governo. Internamente o MidGov é modelado e executado de uma forma orientada a objeto, tornando mais simples a sua compreensão e adaptação.

5.3 Resumo conclusivo

Neste capítulo mostramos a implementação de dois cenários de aplicação da plataforma. O primeiro trata-se da implementação do algoritmo de validação do CPF (Cadastro de Pessoa Física). Dessa forma, serviços mais simples, por exemplo, de consultas do e-Governo podem ser acessados como serviços da grade. O segundo cenário é a implementação do Cadastro Sincronizado Nacional (CadSinc), onde vários Órgãos administrativos se comunicam para realizar os processos cadastrais relativos a Pessoas Jurídicas. O CadSinc é um ótimo exemplo para argumentar a necessidade de esforços em TI para aumentar a interação e a interoperabilidade entre os serviços das entidades administrativas.

A preparação dos cenários exigiu um trabalho bem minucioso. O desenvolvimento de serviços da grade através do GT4 necessita da construção de diversos arquivos de configuração para a implementação. Outro aspecto bastante complexo foi a geração do

fluxo de execução através da GPOL, pois não existem ferramentas gráficas para a sua criação, o que dificulta a preparação do ambiente.

Outro ponto importante em relação à implementação refere-se à utilização do GPO, que, embora extremamente funcional e simples de utilizar, ainda carece de características voltadas à construção de *workflows* dinâmicos, onde atributos possam ser configuráveis e haja uma separação entre processos executáveis e abstratos.

Capítulo 6

Conclusão

Nesse trabalho apresentamos o Midgov, um *middleware* para suportar a integração e colaboração entre serviços prestados pela administração pública sobre o ambiente fornecido pelas grades computacionais. Seu principal objetivo é superar obstáculos de comunicação, presentes em redes empresariais, mais voltado às aplicações do setor público.

Em capítulos anteriores apresentamos o objeto de estudo do trabalho, o e-Governo, termo cunhado a partir do esforço para a utilização de soluções em TIC em atividades da adminitração pública. Introduzimos conceitos referentes a diversas tecnologias aplicadas nesse contexto como SOA, Serviços *Web*, *Web* semântica, Grades computacionais, *Globus Toolkit* e o GPO. Além disso, enfatizamos a relação de nuvem computacional e grade computacional nesse contexto e destacamos importantes trabalhos na área de e-Governo, grades computacional e e-Governo sobre grades computacionais.

Nos demais capítulos introduzimos a arquitetura do *middleware* proposto, seus módulos internos e sua integração com a grade computacional. Dois cenários foram aplicados à plataforma: um serviço simples com a implementação do algoritmo de validação numérica do CPF; e um serviço composto (como *workflow*) que simula o processo realizado pelo Cadastro Sincronizado Nacional.

Grades são tecnologias de grande poder de processamento e escalabilidade em ambientes heterogêneos. Com a sua orientação a serviços, apresentam-se como uma interessante solução em ambientes intra-domínios. Sistemas em grade, embora tenham avançado em direção a aplicações intra-domínios, ainda possuem diversas barreiras a serem quebradas, por exemplo, a espera na construção de aplicações, usabilidade, entrega de *Quality of Service*, execução de *workflows*, entre outras.

A implementação do middleware baseia-se em padrões atuais de *design* da indústria, que tornam possível uma fácil gestão do ambiente. A utilização dos recursos de fábrica/instância associada à utilização de *threads* permite que o *middleware* execute diversos serviços/ workflows na grade. A busca implementada permite uma precisão satisfatória,

embora seja preciso um maior esforço na preparação do perfil semântico.

A pesquisa contribui de forma efetiva com um extenso levantamento bibliográfico e com a proposta do middleware, promovendo assim a convergência de aplicações de e-Governo com as tecnologias de grades, agregando atributos como interoperabilidade, escalabilidade, reusabilidade e abertura.

Trabalhos futuros

Uma perspectiva de trabalho já iniciada está relacionada à implementação de mecanismos de busca semântica baseados em funcionalidades e descrições predefinidas dos serviços (Serviços da Grade), voltadas ao usuário final.

Para um melhor e mais abrangente mapeamento de requisitos em aplicações de e-Governo, mais esforços devem ser realizados na construção de mecanismos sobre a grade. Alguns desses mecanismos envolvem a fácil geração de perfis OWL-S a partir de serviços da grade, mecanismos para a construção dinâmica de formulários html relacionados a entradas de dados nos serviços da grade, e a conexão direta de buscadores semânticos com serviços indexadores, como o *Index Service*.

Outros trabalhos incluem:

- Implementação de componentes de segurança para requisitos específicos como níveis de autonomia, mecanismos contra roubo de identidade e políticas de acesso com "ajuste fino" à informação;
- Inclusão do Condor-G na infraestrutura da grade, implementado sobre aplicações de e-votação as quais demandam grande poder de processamento (escalabilidade), característica inerente a sistemas em grade;
- Criação de uma plataforma gráfica para a criação de *workflows* em GPOL. Dessa forma simplificaria o processo de criação de *workflows*;
- Composição dinâmica de serviços (serviços da grade) no ambiente de grade computacional. Permitir que vários serviços da grade sejam compostos de forma dinâmica com o mínimo de intervenção humana.
- Implementação da plataforma em computação em nuvem. Embora ainda em fase inicial, computação em nuvem tem apresentado diversas características desejáveis para aplicações de e-Governo; e
- Adaptação do *middleware* para utilizar o *Globus Toolkit* versão 5. Atualmente o toolkit da *Globus* está em sua versão 5 e várias mudanças, relacionadas principal-

mente a desempenho, foram realizadas. Dessa forma, a utilização do GT5 passa a ser uma solução interessante.

Publicações

Duas publicações foram geradas a partir desse trabalho:

- Silva, G., Santos, I., e Madeira, E. R. M., "*MidGov: Middleware para Governo Eletrônico baseado em Grades Computacionais*". Em WCGE II Workshop de Computação Aplicada em Governo Eletrônico, 2010, Belo Horizonte. Anais do XXX Congresso da SBC - WCGE II Workshop de Computação Aplicada em Governo Eletrônico, 2010; e

- Silva, G., Santos, I., e Madeira, E. R. M., "*MidGov: middleware for electronic government based on grid computing*". Em Middleware '10 Posters and Demos Track, páginas 3:1–3:2, New York, NY, USA. ACM, 2010.

Referências Bibliográficas

Agarwal, A., Armstrong, P., Charbonneau, A., Chen, H., Desmarais, R. J., Gable, I., Goodenough, D. G., Guan, A., Impey, R., Moa, B., Podaima, W., and Sobie, R. J. (2010). "*Service-Oriented Grid Computing for SAFORAH.*". In *High Performance Computing Systems and Applications*, volume 5976 of *Lecture Notes in Computer Science*, pages 283–291. Springer Berlin / Heidelberg.

Ahmad, M. and Hasibuan, Z. A. (2012). "*Government services integration based on cloud technology*". In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, IIWAS '12, pages 303–306, New York, NY, USA. ACM.

Ahsan, K. M. Q. and Sobhan, M. A. (2012). "*Implementing Secure e-Government activities using grid computing in the context of digital bangladesh*". In *National Conference on Communication and Information Security (NCCIS)*, pages 99–106.

Al-Jaroodi, J., Mohamed, N., and Aziz, J. (2010). "*Service Oriented Middleware: Trends and Challenges*". In *Seventh International Conference on Information Technology: New Generations (ITNG)*, pages 974–979.

Alfonso, C. D., Caballer, M., Carrión, J. V., Hernández, V., Alfonso, C. D., Caballer, M., and Carrión, J. V. (2007). "*gCitizen: a grid middleware for a transparent management of the information about Citizens in the public administration*". *Journal of Theoretical and Applied Electronic Commerce Research*, 2(1):18–32.

Andrade, N., Cirne, W., Brasileiro, F., and Roisenberg, P. (2003). "*OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing*". In Feitelson, D., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 61–86. Springer Berlin / Heidelberg.

Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., Trickovic, I., and Zimek, S. (2002).

"*Web Service Choreography Interface (WSCI) 1.0*". World Wide Web Consortium, http://www.w3.org/TR/wsci.

Berners-Lee, T. (2007). "*The Semantic Web 'layercake' diagram*". Online. Disponível em `http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#%2824%29`. Último acesso em 10/08/2012.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). "*The Semantic Web*". *Scientific American*, 284(5):34–43.

Bernstein, P. A. (1996). "*Middleware: a model for distributed system services*". *Commun. ACM*, 39(2):86–98.

Bichler, M. and Lin, K.-J. (2006). "*Service-Oriented Computing*". *Computer*, 39(3):99 – 101.

Cambronero, M. E., Díaz, G., Martínez, E., and Valero, V. (2009). "*A Comparative Study between WSCI, WS-CDL, and OWL-S*". In *ICEBE*, pages 377–382. IEEE Computer Society.

Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). "*Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*". World Wide Web Consortium (W3C), Recommendation, http://www.w3.org/TR/wsdl20.

Chu, W., Chang, C.-H., Lu, C.-W., Chen, J.-N., and Wang, F.-J. (2012). "*The Development of Cloud Computing and Its Challenges for Taiwan*". In *IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, pages 380–386.

Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., Brasileiro, F., Sauve, J., Silva, F. A. B., Barros, C. O., Silveira, C., and Silveira, C. (2003). Running bag-of-tasks applications on computational grids: the mygrid approach. In *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pages 407 –416.

Clement, L., Hately, A., von Riegen, C., and Rogers, T. (2004). "*UDDI Version 3.0.2*". Organization for the Advancement of Structured Information Standards, UDDI Spec Technical Committee Draft, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.

Condor (2011). Disponível em `http://www.cs.wisc.edu/condor/`. Último acesso em março/2011.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). "*Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI*". *Internet Computing, IEEE*, 6(2):86 –93.

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., and Weerawarana, S. (2003). "*The next step in Web services*". *Commun. ACM*, 46(10):29–34.

Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Maguire, T., Snelling, D., and Tuecke, S. (2004). From open grid services infrastructure to ws-resource framework: Refactoring & evolution. *Global Grid Forum Draft Recommendation*, (Ver. 1.0):1–20.

Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C. (2001). "*Grid Information Services for Distributed Resource Sharing*". In *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing*.

Czajkowski, K., Foster, I., Karonis, N. T., Kesselman, C., Martin, S., Smith, W., and Tuecke, S. (1998). "*A Resource Management Architecture for Metacomputing Systems*". In Feitelson, D. G. and Rudolph, L., editors, *JSSPP*, volume 1459 of *Lecture Notes in Computer Science*, pages 62–82. Springer.

Davies, J., Janowski, T., Ojo, A., and Shukla, A. (2007). "*Technological foundations of electronic governance*". In *Proceedings of the 1st international conference on Theory and practice of electronic governance*, ICEGOV '07, pages 5–11, New York, USA. ACM.

Dillon, T., Wu, C., and Chang, E. (2010). "*Cloud Computing: Issues and Challenges*". In *24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010*, pages 27–33.

Doerksen, T. (2008). "*Cloud Computing - The User-Friendly Version of Grid Computing*". Online. Disponível em `http://virtualization.sys-con.com/node/593313`. Último acesso em 02/08/2012.

Drigas, A. and Koukianakis, L. (2009). "*Government Online: An E-Government Platform to Improve Public Administration Operations and Services Delivery to the Citizen*". In *Visioning and Engineering the Knowledge Society. A Web Science Perspective*, pages 523–532. Springer.

Emmerich, W. (2000). "*Software Engineering and Middleware: A Roadmap*". In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 117–129, New York, USA. ACM.

Emmerich, W., Aoyama, M., and Sventek, J. (2007). "*The impact of research on middleware technology*". *ACM SIGSOFT Software Engineering Notes*, 41(1):89–112.

Fensel, D., Lausen, H., Polleres, A., Bruijn, J. d., Stollberg, M., Roman, D., and Domingue, J. (2006). *"Enabling Semantic Web Services: The Web Service Modeling Ontology"*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Fluegge, M., Santos, I. J. G., Tizzo, N. P., and Madeira, E. R. M. (2006). "*Challenges and techniques on the road to dynamically compose web services*". In *Proceedings of the 6th international conference on Web engineering*, ICWE '06, pages 40–47, New York, NY, USA. ACM.

Foster, I. (2002). "*What is the Grid? A Three Point Checklist*". *GRIDtoday*, 1(6).

Foster, I. (2011). "*Globus Online: Accelerating and Democratizing Science through Cloud-Based Services*". *Internet Computing, IEEE*, 15(3):70 –73.

Foster, I., Geisler, J., Nickless, B., Smith, W., and Tuecke, S. (1996). "*Software infrastructure for the I-WAY high-performance distributed computing experiment*". pages 562 –571.

Foster, I. and Kesselman, C. (1997). "*Globus: A metacomputing infrastructure toolkit*". *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128.

Foster, I. and Kesselman, C. (1998). *"The Grid: Blueprint for a New Computing Infrastructure"*. Morgan Kaufmann Publishers.

Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). "*The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*". In *Open Grid Service Infrastructure Working Group, Global Grid Forum*.

Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. (1998). "*A Security Architecture for Computational Grids*". In *ACM Conference on Computer and Communications Security*, pages 83–92.

Foster, I., Kesselman, C., and Tuecke, S. (2001). "*The Anatomy of the Grid: Enabling Scalable Virtual Organizations*". *International Journal of Supercomputer Applications*, 15(3):200–222.

Frey, J., Tannenbaum, T., Foster, I., Livny, M., and Tuecke, S. (2002). "*Condor-G: A Computation Management Agent for Multi-Institutional Grids*". *Cluster Computing*, 5(3):237–246.

Garcia, T. H. B., Pomar, C. D., and Hoeschl, H. C. (2004). "*Democracy in the Electronic Government Era*". In *I3E*, IFIP Conference Proceedings, pages 67–76. Kluwer.

Globus Alliance (2010a). "*Globus Toolkit – Release notes*". Online. Disponível em `http://www.globus.org/toolkit/releasenotes/`. Último acesso em 10/08/2012.

Globus Alliance (2010b). "*The Globus Toolkit*". Online. Disponível em `http://www.globus.org/toolkit`. Último acesso em 10/08/2012.

Globus Alliance (2011). "*Globus Crux Toolkit*". Online. Disponível em `http://confluence.globus.org/display/whi/Globus+Crux+Toolkit`. Último acesso em 25/05/2011.

Goldchleger, A., Kon, F., Goldman, A., Finger, M., and Bezerra, G. C. (2004). "*InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines*". *Concurrency and Computation: Practice and Experience*, 16(5):449–459.

Gong, C., Liu, J., Zhang, Q., Chen, H., and Gong, Z. (2010). "*The Characteristics of Cloud Computing*". In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 275 –279.

Grace, P., Coulson, G., Blair, G., Mathy, L., Yeung, W., Cai, W., Duce, D., and Cooper, C. (2004). "*GRIDKIT: Pluggable Overlay Networks for Grid Computing*". In *The 6th International Symposium on Distributed Objects and Applications (DOA '04)*, volume 3291, pages 1463–1481. Springer.

Graham, S., Karmarkar, A., Mischkinsky, J., Robinson, I., and Sedukhin, I. (2006). "*Web Services Resource 1.2 (WS-Resource)*". *OASIS Standard April*.

Grandison, T., Maximilien, E. M., Thorpe, S., and Alba, A. (2010). "*Towards a Formal Definition of a Computing Cloud*". In *SERVICES*, pages 191–192.

Grimshaw, A., Morgan, M., Merrill, D., Kishimoto, H., Savva, A., Snelling, D., Smith, C., and Berry, D. (2009). "*An Open Grid Services Architecture Primer*". *Computer*, 42(2):27 –34.

Grimshaw, A. S., Wulf, W. A., and The Legion Team, C. (1997). "*The Legion Vision of a Worldwide Virtual Computer*". *Commun. ACM*, 40(1):39–45.

Gruber, T. R. (1993). "*A translation approach to portable ontology specifications*". *Knowl. Acquis.*, 5(2):199–220.

Hendler, J. (2009). "*Web 3.0 Emerging*". *Computer*, 42(1):111–113.

Hiller, J. S. and Bélanger, F. (2001). "*Privacy strategies for Electronic Government*". In *E-Government Series*. E-Government Series, PricewaterhouseCoopers Endowment for Business of Government, Arlington, VA.

Hollingsworth, D. (1995). "*Workflow Management Coalition - The Workflow Reference Model*". Technical report, Workflow Management Coalition.

Integrade (2011). Disponível em `http://www.integrade.org.br/`. Último acesso em março/2011.

Jepsen, T. C. (2009). "*Just What Is an Ontology, Anyway?*". *IT Professional, IEEE Computer Society. Los Alamitos, CA, USA*, 11(5):22–27.

Jordan, D. and Evdemon, J. (2007). "*Web Services Business Process Execution Language Version 2.0*". OASIS Standard, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., and Barreto, C. (2005). "*Web Services Choreography Description Language Version 1.0*". World Wide Web Consortium, Candidate Recommendation, http://www.w3.org/TR/ws-cdl-10.

KBSt (2008). *"SAGA – Standards and Architectures for eGovernment Applications – Version 4.0"*. German Federal Ministry of Interior, http://www.cio.bund.de/saga.

Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J., and Verschueren, P. (2004). "*Patterns: Implementing an SOA Using an Enterprise Service Bus*". In *IBM Comp.*, pages 33–71. IBM Corp.

Klusch, M. (2008). "*Semantic Web Service Description*". In Calisti, M., Walliser, M., Brantschen, S., Herbstritt, M., Schumacher, M., Schuldt, H., and Helin, H., editors, *CASCOM: Intelligent Service Coordination in the Semantic Web*, Whitestein Series in Software Agent Technologies and Autonomic Computing, pages 31–57. Birkhäuser Basel.

Klusch, M., Fries, B., and Sycara, K. (2006). "*Automated semantic web service discovery with OWLS-MX*". In *Autonomous agents and multiagent systems*, pages 915–922. ACM.

Kook, Y.-G., Lee, J., and Kim, J.-S. (2009). "*E-Government Grid System Based on Multi-agent for Interoperability*". *International Conference on Innovation Management*, pages 7–10.

Kundra, V. (2010). State of public sector cloud computing. Technical report, Federal Chief Information Officer.

Lewis, D. (2006). "*What is web 2.0?*". *ACM Crossroads*, 13(1):3.

Leymann, F. (2001). "*Web Services Flow Language (WSFL 1.0)*". Technical report, IBM.

Li, Y., Li, W., and Jiang, C. (2010). "*A Survey of Virtual Macahine System: Current Technology and Future Trends*". In *Third International Symposium on Electronic Commerce and Security (ISECS) 2010*, pages 332 –336.

Liang, J. (2012). "*Government Cloud: Enhancing Efficiency of E-Government and Providing Better Public Services*". In *International Joint Conference on Service Sciences (IJCSS)*, pages 261–265.

Litzkow, M., Livny, M., and Mutka, M. (1988). "*Condor - A Hunter of Idle Workstations*". In *Proceedings of the 8th International Conference of Distributed Computing Systems*.

Lu, T., Lin, Q., and Lv, H. (2011). "*E-government Framework Design Based on Grid Technology*". In *International Conference on Internet Computing Information Services (ICICIS)*, pages 339–342.

Ma, H. (2010). "*A Service-oriented E-government Support Platform for Integration of Application and Data*". In *Second International Conference on Information Technology and Computer Science (ITCS)*, pages 398–401.

Manola, F. and Miller, E., editors (2004). *"RDF Primer"*. W3C Recommendation. World Wide Web Consortium, http://www.w3.org/TR/rdf-primer/.

Marcucci, L., Kluzer, S., and Nicolini, A. (2006). "*ICAR - a System for e-Enabled cooperation among Regional, Local and National Administrations in Italy*". European Commission.

Mascolo, C., Capra, L., and Emmerich, W. (2002). "*Mobile Computing Middleware*". In *Advanced Lectures in Networking*, pages 20–58, New York, USA. Springer-Verlag.

Mell, P. and Grance, T. (2009). "*The NIST Definition of Cloud Computing (v15)*". Technical report, National Institute of Standards and Technology.

Mitra, N. and Lafon, Y. (2007). "*SOAP Version 1.2 Part 0: Primer (Second Edition)*". World Wide Web Consortium (W3C), Recommendation, http://www.w3.org/TR/soap12-part0. W3C Recommendation.

Muehlen, M. Z. (2004). *"Workflow-based Process Controlling: Foundation, Design, and Implementation of Workflow-driven Process Information Systems"*, volume 6 of *Advances in Information Systems and Management Science*. Logos, Berlin.

Nebula (2011). Disponível em `http://nebula.nasa.gov/`. Último acesso em março/2011.

Ngan, L. and Kanagasabai, R. (2012). "*Semantic Web service discovery: state-of-the-art and research challenges*". *Personal and Ubiquitous Computing*, pages 1–12.

Nitzsche, J., Lessen, T., Karastoyanova, D., and Leymann, F. (2007). "$BPEL^{light}$". In *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 214–229. Springer Berlin Heidelberg.

OGF (2004). "*Open Grid Service Infrastructure Primer*". *Relatório Técnico, Open Grid Forum, www.ogf.org/documents/GFD.31.pdf*.

Papazoglou, M. P. and Georgakopoulos, D. (2003). "*Service-Oriented Computing*". *Commun. ACM*, 46(10):24–28.

Peltz, C. (2003). "*Web Services Orchestration and Choreography*". *Computer*, 36(10):46–52.

Preist, C. (2007). "*Goals and Vision: Combining Web Services with Semantic Web Technology*". In *Semantic Web Services: Concepts, Technologies, and Applications*, chapter 6, pages 159–178.

Qilin, L. and Mintian, Z. (2010). "*The State of the Art in Middleware*". In *International Forum on Information Technology and Applications (IFITA)*, volume 1, pages 83–85.

Receita Federal (2003). "*Cadastro Sincronizado Nacional*". www16.receita.fazenda.gov.br. Acessado em 10/08/2012.

Ross-Talbot, S. and Fletcher, T. (2006). "*Web Services Choreography Description Language: Primer*". World Wide Web Consortium, Working Draft, http://www.w3.org/TR/ws-cdl-10-primer.

Russell, N., ter Hofstede, A. H. M., Edmond, D., and van der Aalst, W. M. P. (2005). "*Workflow Data Patterns: Identification, Representation and Tool Support*". In *Conceptual Modeling –Entity Relationship*, pages 353–368. Springer.

Sadjadi, S. M. and McKinley, P. K. (2003). "*A Survey of Adaptive Middleware*". Technical Report MSU-CSE-03-35, Department of Computer Science, Michigan State University, East Lansing, Michigan.

Santos, I. J. G. and Madeira, E. R. M. (2007). "*E-Government and Grid Computing - Potentials and Challenges Towards Citizen-Centric Services*". In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ICEIS'07, Funchal - Portugal*, pages 144–148.

Santos, I. J. G. and Madeira, E. R. M. (2010). "*A Semantic-enabled Middleware for Citizen-centric E-Government Services*". In *International Journal of Intelligent Information Technologies, 6(3): 34-55.*

Schiele, G., Handte, M., and Becker, C. (2010). "*Pervasive Computing Middleware*". In *Handbook of Ambient Intelligence and Smart Environments*, pages 201–227. Springer.

Sedek, K., Omar, M., and Sulaiman, S. (2012). "*Interoperable SOA-based architecture for e-government portal*". In *IEEE Conference on Open Systems (ICOS)*, pages 1–6.

Senna, C. and Madeira, E. R. M. (2007). "*A middleware for instrument and service orchestration in computational grids*". In *Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGrid*, Rio de Janeiro.

Senna, C. R., Bittencourt, L. F., and Madeira, E. R. M. (2011). "*An Environment for Evaluation and Testing of Service Workflow Schedulers in Clouds*". *International Conference on High Performance Computing & Simulation*, pages 301–307.

Shvaiko, P., Villafiorita, A., Zorer, A., Chemane, L., Fumo, T., and Hinkkanen, J. (2009). "*eGIF4M: eGovernment Interoperability Framework for Mozambique*". In *Electronic Government*, volume 5693 of *Lecture Notes in Computer Science*, pages 328–340. Springer Berlin / Heidelberg.

Sotomayor, B. and Childers, L. (2006). *"Globus toolkit 4: programming Java services"*. The Elsevier series in Grid computing. Morgan and Kaufmann, San Francisco, CA.

Tanenbaum, A. S. (2002). *"Computer Networks (4th Edition)"*. Prentice Hall PTR, 4 edition.

UNPAN (2010). *"United Nations e-Government Survey 2010. Leveraging E-government at a Time of Financial and Economic Crisis"*. United Nations Department of Economic and Social Affairs, New York, NY.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). "*A break in the Clouds: Towards a Cloud Definition*". *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55.

W3C (2004a). "*OWL-S: Semantic Markup for Web Services*". W3C Recommendation. Disponível em http://www.w3.org/Submission/OWL-S/.

W3C (2004b). "*OWL –Web Ontology Language*". http://www.w3.org/TR/owl-features. Acessado em 10/08/2012.

W3C (2004c). "*Web Services Glossary*". http://www.w3.org/TR/ws-gloss.

Wegner, P. (1996). "*Interoperability*". *ACM Computing Surveys*, 28(1):285–287.

Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Metier, S., Pearlman, L., and Tuecke, S. (2003). "*Security for Grid Services*". In *In Twelfth International Symposium on High Performance Distributed Computing*, pages 48–57. IEEE Press.

WfMC (1999). *"Workflow Management Coalition Terminology & Glossary"*. The Workflow Management Coalition Specification, http://www.wfmc.org/Glossaries-FAQs.

Wyld, D. C. (2009). *"Moving to the Cloud: An Introduction to Cloud Computing in Government"*. IBM Center for the Business of Government, e-Government Series, Washington, USA.

Yang, D., Han, Y., and Xiong, J. (2007). "*eGovGrid: A Service-Grid-Based Framework for E-Government Interoperability*". In *Integration and Innovation Orient to E-Society Volume 2*, volume 252 of *IFIP Advances in Information and Communication Technology*, pages 364–372. Springer Boston.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). "*Cloud computing: state-of-the-art and research challenges*". *Journal of Internet Services and Applications*, 1(1):7–18.

Zhang, W. and Chen, Q. (2010). "*From E-government to C-government via Cloud Computing*". In *International Conference on E-Business and E-Government (ICEE)*, pages 679 –682.

Zhang, W. and Wang, Y. (2008). "*Towards building a semantic grid for E-government applications*". *World Scientific and Engineering Academy and Society Transactions on Computer Research*, 3(4):273–282.

Zhu, Y. (2003). "*A Survey on Grid Scheduling Systems*". *PhD Qualifying Examination, Department of Computer Science, Hong Kong University of science and Technology*, pages 1–9.

Zweers, K. and Planqué, K. (2001). "*Electronic Government. From an Organization Based Perspective Towards a Client Oriented Approach*". In E-Government, D., editor, *Prins J.E.J. (ed.)*. Kluwer Law International.

Apêndice A

Workflow emulado para o Cadastro Síncrono

```
<?xml version="1.0" encoding="UTF-8"?>
<!--=========================================================================
    cadSinc - Executa servico de Cadastro Sincrono
    @author: Geraldo Magela
    Atualizado em: 22/03/2013

    Executa simulacao de workflow usando:
      GPO executado em Artemis
==========================================================================-->

<gpo:job name="cadSinc">
  <gpo:definitions name="job_Definitions">
    <gpo:variables>
        <!-- Variaveis -->
        <gpo:variable name="retCode"  type="int" value="0"/>
        <gpo:variable name="sRetCode"  type="string"/>
        <gpo:variable name="libPrefeitura"  type="string" value="/tmp/Docs/
            documento_i001.txt /tmp/Docs/documento.txt 7106"/>
        <gpo:variable name="cadFederal"     type="string" value="/tmp/Docs/
            documento_i001.txt /tmp/Docs/documento.txt 7240"/>
        <gpo:variable name="cadEstadual"    type="string" value="/tmp/Docs/
            documento_i001.txt /tmp/Docs/documento.txt 7270"/>
        <gpo:variable name="recFederal"     type="string" value="/tmp/Docs/
            documento.txt /tmp/Docs/documento.txt 7250"/>
        <gpo:variable name="juce"           type="string" value="/tmp/Docs/
            documento_i001.txt /tmp/Docs/documento.txt 7250"/>
    </gpo:variables>
        <gpo:gservices>
        <!-- Duas instancias criadas em Dionisio: permitir paralelismo no envio-->
        <gpo:gsh name="Inst_Dionisio-mf1"
        uri="http://10.3.77.15:8080/wsrf/services/FactRCService"
        fal = "org.globus.examples.stubs.FactRC.service.
            FactRCServiceAddressingLocator"
        sal = "org.globus.examples.stubs.RCService_instance.service.
            RCServiceAddressingLocator"
```

```
31          type="Factory"/>
32          <gpo:gsh name="Inst_Dionisio-mf2"
33          uri="http://10.3.77.15:8080/wsrf/services/FactRCService"
34          fal = "org.globus.examples.stubs.FactRC.service.
35              FactRCServiceAddressingLocator"
36          sal = "org.globus.examples.stubs.RCService_instance.service.
37              RCServiceAddressingLocator"
38          type="Factory"/>
39
40          <!-- Duas instancias criadas em Cronos: permitir paralelismo no envio-->
41          <gpo:gsh name="Inst_Cronos-mf1"
42          uri="http://10.3.77.16:8080/wsrf/services/FactRCService"
43          fal = "org.globus.examples.stubs.FactRC.service.
44              FactRCServiceAddressingLocator"
45          sal = "org.globus.examples.stubs.RCService_instance.service.
46              RCServiceAddressingLocator"
47          type="Factory"/>
48          <gpo:gsh name="Inst_Cronos-mf2"
49          uri="http://10.3.77.16:8080/wsrf/services/FactRCService"
50          fal = "org.globus.examples.stubs.FactRC.service.
51              FactRCServiceAddressingLocator"
52          sal = "org.globus.examples.stubs.RCService_instance.service.
53              RCServiceAddressingLocator"
54          type="Factory"/>
55
56          <!-- Instancia criada em Zeus -->
57          <gpo:gsh name="Inst_Zeus-mf1"
58          uri="http://10.3.77.5:8080/wsrf/services/FactRCService"
59          fal = "org.globus.examples.stubs.FactRC.service.
60              FactRCServiceAddressingLocator"
61          sal = "org.globus.examples.stubs.RCService_instance.service.
62              RCServiceAddressingLocator"
63          type="Factory"/>
64
65          <!-- Instancia criada em Apolo -->
66          <gpo:gsh name="Inst_Apolo-mf1"
67          uri="http://10.3.77.19:8080/wsrf/services/FactRCService"
68          fal = "org.globus.examples.stubs.FactRC.service.
69              FactRCServiceAddressingLocator"
70          sal = "org.globus.examples.stubs.RCService_instance.service.
71              RCServiceAddressingLocator"
72          type="Factory"/>
73
74          <!-- Instancia criada em Nix -->
75          <gpo:gsh name="Inst_Nix-mf1"
76          uri="http://10.3.77.21:8080/wsrf/services/FactRCService"
77          fal = "org.globus.examples.stubs.FactRC.service.
78              FactRCServiceAddressingLocator"
79          sal = "org.globus.examples.stubs.RCService_instance.service.
80              RCServiceAddressingLocator"
81          type="Factory"/>
82      </gpo:gservices>
83    </gpo:definitions>
84
85  <gpo:process name="CadSinc">
86
87      <!--Distribui o arquivo originel em 2 partes    -->
88      <gpo:invoke name="Inst_Dionisio-mf1" method="dataDistribution" tagname="PDG">
```

```
89          <gpo:argument value="2 /tmp/Docs/documento.txt" type="string"/>
90          <gpo:return variable="sRetCode" type="string"/>
91      </gpo:invoke>
92      <gpo:flow name="EnvioPrefetura_JUCE">
93          <!-- Envia slice files para Cronos e Zeus-->
94          <gpo:invoke name="Inst_Dionisio-mf1" method="runCommand">
95              <gpo:argument value="scp -P3015 -p /tmp/Docs/documento_out001.txt
                    geraldoms@cronos:/tmp/Docs/documento_i001.txt" type="string"/>
96              <gpo:return variable="retCode" type="int"/>
97          </gpo:invoke>
98          <gpo:invoke name="Inst_Dionisio-mf2" method="runCommand">
99              <gpo:argument value="scp -P3015 -p /tmp/Docs/documento_out002.txt
                    geraldoms@zeus:/tmp/Docs/documento.txt" type="string"/>
100             <gpo:return variable="retCode" type="int"/>
101         </gpo:invoke>
102     </gpo:flow>
103     <!-- Simula o processamento para a liberacao da prefeitura-->
104     <gpo:invoke name="Inst_Cronos-mf1" method="mProcess"
105             tagname="liberacaoPrefeitura">
106         <gpo:argument variable="libPrefeitura" type="string"/>
107         <gpo:return variable="retCode" type="int"/>
108     </gpo:invoke>
109
110     <!-- Verifica a saida do servico de verificacao da prefeitura-->
111     <gpo:if variable="retCode" operator="==" sExpression="1">
112         <!-- Envia o arquivo para Zeus-->
113         <gpo:invoke name="Inst_Cronos-mf1" method="runCommand">
114             <gpo:argument value="scp -P3015 -p /tmp/Docs/documento_out001.txt
                    geraldoms@zeus:/tmp/Docs/documento_i001.txt" type="string"/>
115             <gpo:return variable="retCode" type="int"/>
116         </gpo:invoke>
117     </gpo:if>
118
119     <gpo:if variable="retCode" operator="==" sExpression="0">
120         <!-- Realiza a duplicacao do arquivo -->
121         <gpo:invoke name="Inst_Cronos-mf1" method="dataDistribution"
122                 tagname="liberacaoPrefeitura">
123             <gpo:argument variable="2 /tmp/Docs/documento.txt" type="string"/>
124             <gpo:return variable="retCode" type="int"/>
125         </gpo:invoke>
126
127         <!-- Envia os arquivos para Apolo e Nix-->
128         <gpo:flow name="libCadastro">
129             <gpo:invoke name="Inst_Cronos-mf1" method="runCommand">
130                 <gpo:argument value="scp -P3015 -p /tmp/Docs/documento_out001.txt
                        geraldoms@apolo:/tmp/Docs/documento_i001.txt" type="string"/>
131                 <gpo:return variable="retCode" type="int"/>
132             </gpo:invoke>
133             <gpo:invoke name="Inst_Cronos-mf2" method="runCommand">
134                 <gpo:argument value="scp -P3015 -p /tmp/Docs/documento_out002.txt
                        geraldoms@nix:/tmp/Docs/documento_i001.txt" type="string"/>
135                 <gpo:return variable="retCode" type="int"/>
136             </gpo:invoke>
137         </gpo:flow>
138
139         <!-- Executa o cadastro Estadual e Federal simultaneamente-->
140         <gpo:flow name="CadEstadual_CadFederal">
141             <!-- Simula o processamento de dados para o Cadastro Federal-->
```

```
142             <gpo:invoke name="Inst_Apolo-mf1" method="mProcess" tagname="
                    CadastroFederal">
143                 <gpo:argument variable="cadFederal" type="string"/>
144                 <gpo:return variable="retCode" type="int"/>
145             </gpo:invoke>
146             <!-- Simula o processamento de dados para o Cadastro Estadual-->
147             <gpo:invoke name="Inst_Nix-mf1" method="mProcess" tagname="
                    CadastroEstadual">
148                 <gpo:argument variable="cadEstadual" type="string"/>
149                 <gpo:return variable="retCode" type="int"/>
150             </gpo:invoke>
151
152             <!-- Envio dos arquivos para Dionisio-->
153             <gpo:invoke name="Inst_Apolo-mf1" method="runCommand">
154                 <gpo:argument value="scp -P3015 -p /tmp/Docs/documento.txt
                        geraldoms@dionisio:/tmp/Docs/documento_i001.txt" type="string"/>
155                 <gpo:return variable="retCode" type="int"/>
156             </gpo:invoke>
157             <gpo:invoke name="Inst_Nix-mf1" method="runCommand">
158                 <gpo:argument value="scp -P3015 -p /tmp/Docs/documento.txt
                        geraldoms@dionisio:/tmp/Docs/documento_i002.txt" type="string"/>
159                 <gpo:return variable="retCode" type="int"/>
160             </gpo:invoke>
161         </gpo:flow>
162
163         <!-- Agrega as 2 partes recebidas -->
164         <gpo:invoke name="Inst_Dionisio-mf1" method="dataAggregation" tagname="
                JuntaArquivos_RecFederal">
165             <gpo:argument value="2 /tmp/Docs/documento_in.txt
166                 /tmp/Docs/documento.txt" type="string"/>
167             <gpo:return variable="sRetCode" type="string"/>
168         </gpo:invoke>
169         <!-- Simula o processamento na Receita Federal-->
170         <gpo:invoke name="Inst_Dionisio-mf1" method="mProcess" tagname="RecFederal">
171             <gpo:argument value="recFederal" type="string"/>
172             <gpo:return variable="RetCode" type="int"/>
173         </gpo:invoke>
174
175         <!-- Envia a JUCE -->
176         <gpo:invoke name="Inst_Dionisio-mf1" method="runCommand">
177             <gpo:argument value="scp -P3015 -p /tmp/Docs/documento.txt
                    geraldoms@zeus:/tmp/Docs/documento_i001.txt" type="string"/>
178             <gpo:return variable="retCode" type="int"/>
179         </gpo:invoke>
180     </gpo:if>
181     <!-- Simula o processamento na JUCE-->
182     <gpo:invoke name="Inst_Zeus-mf1" method="mProcess">
183         <gpo:argument variable="juce" type="string"/>
184         <gpo:return variable="retCode" type="int"/>
185     </gpo:invoke>
186
187     <!-- Envia para a maquina central (middleware)-->
188     <gpo:invoke name="Inst_Zeus-mf1" method="runCommand">
189         <gpo:argument value="scp -P3015 -p /tmp/Docs/documento.txt
                geraldoms@dionisio:/tmp/Docs/documento_final.txt" type="string"/>
190         <gpo:return variable="retCode" type="int"/>
191     </gpo:invoke>
192
```

```
193     <!-- Return value to Middleware -->
194     <gpo:return variable="retCode" type="int"/>
195  </gpo:process>
196 </gpo:job>
```

Listing A.1: Representação do *Workflow* CadSinc em GPOL

Apêndice B

Perfil Semântico para o Cadastro Síncrono

Exemplo de perfil semântico em OWL-S para o Cadastro Síncrono.

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!--=========================================================================
    cadSinc - Perfil Semantico simplificado
    @author: Geraldo Magela
    Atualizado em: 22/03/2013

========================================================================-- >

<!DOCTYPE uridef[
  <!ENTITY rdf          "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs         "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl          "http://www.w3.org/2002/07/owl">
  <!ENTITY service      "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY process      "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY profile      "http://www.daml.org/services/owl-s/1.1/Profile.owl">
  <!ENTITY actor         "http://www.daml.org/services/owl-s/1.1/ActorDefault.owl">
  <!ENTITY addParam      "http://www.daml.org/services/owl-s/1.1/
      ProfileAdditionalParameters.owl">
  <!ENTITY DEFAULT      "http://localhost:8080/midgov/services/CadSincService.owl">
]>

<rdf:RDF
  xmlns:rdf=          "&rdf;#"
  xmlns:rdfs=         "&rdfs;#"
  xmlns:owl=          "&owl;#"
  xmlns:service=      "&service;#"
  xmlns:process=      "&process;#"
  xmlns:profile=      "&profile;#"
  xmlns:actor= "&actor;#"
  xmlns:addParam= "&addParam;#"
  xmlns=              "&DEFAULT;#">

```

```
33   <owl:Ontology about="">
34     <owl:imports rdf:resource="&service;" />
35     <owl:imports rdf:resource="&profile;" />
36     <owl:imports rdf:resource="&process;" />
37     <owl:imports rdf:resource="&actor;" />
38     <owl:imports rdf:resource="&addParam;" />
39
40     <owl:imports rdf:resource="http://localhost:8080/midgov/ontologies/
           cadSincData.owl"/>
41   </owl:Ontology>
42
43   <profile:Profile rdf:ID="CadSincService">
44
45     <profile:serviceName>CadSincService</profile:serviceName>
46     <profile:textDescription>
47 The National Synchronized Register (CadSinc)
48 allows the integration of the CNPJ registry
49 procedures in the three levels of government.
50     </profile:textDescription>
51     <!-- Descriptions of the parameters that will be used by IOPEs -->
52       <profile:hasInput>
53         <process:Input rdf:ID="CompanyName">
54           <process:parameterType>http://localhost:8080/midgov/ontologies/
                 cadSincData.owl#CompanyName</process:parameterType>
55     </process:Input>
56        </profile:hasInput>
57
58       <profile:hasInput>
59         <process:Input rdf:ID="CompanyAddress">
60           <process:parameterType>http://localhost:8080/midgov/ontologies/
                 cadSincData.owl#CompanyAddress</process:parameterType>
61     </process:Input>
62        </profile:hasInput>
63
64       <profile:hasOutput>
65         <process:UnConditionalOutput rdf:ID="BusinessLicense">
66           <process:parameterType>http://localhost:8080/midgov/ontologies/
                 cadSincData.owl#BusinessLicense</process:parameterType>
67         </process:UnConditionalOutput>
68       </profile:hasOutput>
69
70   </profile:Profile>
71 </rdf:RDF>
```

Listing B.1: Perfil Semântico para o CadSinc