CoGPlat: Um Middleware para Serviços de Governo Eletrônico Centrados no Cidadão

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Ivo José Garcia dos Santos e aprovada pela Banca Examinadora.

Campinas, 16 de dezembro de 2008.

Prof. Dr. Edmundo Roberto Mauro Madeira
(Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Júlia Milani Rodrigues CRB8a / 2116

Santos, Ivo José Garcia dos
Sa59c CoGPlat: um middleware para serviços de governo eletrônico centrados no cidadão / Ivo José Garcia dos Santos -- Campinas, [S.P. :s.n.], 2008.

Orientador : Edmundo Roberto Mauro Madeira
Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Governo eletrônico. 2. Middleware. 3. Serviços na Web - Semântica. I. Madeira, Edmundo Roberto Mauro. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: CoGPlat: a middleware for citizen-centric e-government services.

Palavras-chave em inglês (Keywords): 1. E-government. 2. Middleware. 3. Semantic Web services.

Área de concentração: Ciência da Computação

Titulação: Doutor em Ciência da Computação

Banca examinadora:

Prof. Dr. Edmundo Roberto Mauro Madeira (IC-UNICAMP)
Prof. Dr. Fabio Kon (IME-USP)
Profa. Dra. Thaís Vasconcelos Batista (UFRN)
Profa. Dra. Cláudia Maria Bauzer Medeiros (IC-UNICAMP)
Profa. Dra. Islene Calciolari Garcia (IC-UNICAMP)
Prof. Dr. Jó Ueyama (EACH-USP)
Prof. Dr. Rogério Drummond (IC-UNICAMP)
Profa. Dra. Maria Beatriz Felgar de Toledo (IC-UNICAMP)

Data da defesa: 16/12/08

Programa de pós-graduação: Doutorado em Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 16 de dezembro de 2008, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Fábio Kon
IME - USP.

Profª. Drª. Thaís Vasconcelos Batista
DIMA – UFRN.

Profª. Drª. Cláudia Maria Bauzer Medeiros
IC – UNICAMP.

Profª. Drª. Islene Calciolari Garcia
IC – UNICAMP.

Prof. Dr. Edmundo Roberto Mauro Madeira
IC – UNICAMP.

Instituto de Computação
Universidade Estadual de Campinas

CoGPlat: Um Middleware para Serviços de Governo Eletrônico Centrados no Cidadão

Ivo José Garcia dos Santos[1]

Outubro de 2008

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (Orientador)
- Prof. Dr. Fabio Kon
  Instituto de Matemática e Estatística - USP
- Profa. Dra. Thaís Vasconcelos Batista
  Departamento de Informática e Matemática Aplicada - UFRN
- Profa. Dra. Cláudia Maria Bauzer Medeiros
  Instituto de Computação - UNICAMP
- Profa. Dra. Islene Calciolari Garcia
  Instituto de Computação - UNICAMP
- Prof. Dr. Jó Ueyama (Suplente)
  Escola de Artes, Ciências e Humanidades - USP
- Prof. Dr. Rogério Drummond (Suplente)
  Instituto de Computação - UNICAMP
- Profa. Dra. Maria Beatriz Felgar de Toledo (Suplente)
  Instituto de Computação - UNICAMP

[1]Suporte financeiro: Bolsa CNPq (2004–2005), Bolsa de Doutorado-Sanduíche CAPES/DAAD (2005–2006), Bolsa BIPED/PED-A Unicamp (2006–2008) e Bolsa CAPES (2008)

*"**Information is the currency of democracy.**"*

Thomas Jefferson

*"**Es ist nicht genug, zu wissen, man muß auch anwenden; es ist nicht genug, zu wollen, man muß auch tun.**"*
("Saber não basta, devemos aplicar. Desejar não basta, devemos fazer.")

Johann von Goethe

*"**The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency.**"*

Bill Gates

*"**Feliz o homem que acha sabedoria, e o homem que adquire conhecimento porque melhor é o lucro que ela dá do que o da prata, e melhor a sua renda do que o ouro mais fino. Mais preciosa é do que pérolas, e tudo o que podes desejar não é comparável a ela.**"*

Provérbios 3:13-15

Resumo

As Tecnologias de Informação e Comunicação (TICs) têm sido aplicadas de maneira cada vez mais intensa por entidades governamentais em todo o mundo, com o objetivo de melhor servir à sociedade, fenômeno este comumente chamado de *Governo Eletrônico* (*e-Government*). Inicialmente o uso das TICs tinha como objetivo principal agilizar os processos burocráticos internos dos órgãos públicos (por exemplo processando enormes quantidades de dados), sendo a melhora no atendimento ao cidadão fator considerado de menor importância. Nos anos recentes, isto tem caminhado rumo a uma grande transformação com aumento considerável da demanda por uma participação mais ativa dos cidadãos nos processos e decisões governamentais, pelo aumento da inclusão social, pela prestação eletrônica de serviços públicos cada vez mais complexos e transparentes e também pela redução da burocracia. A inerente heterogeneidade dos sistemas e processos, aspectos legais e também requisitos como privacidade, autonomia e rastreabilidade criam uma série de desafios tecnológicos que precisam ser enfrentados.

Neste contexto, esta tese, organizada como uma coletânea de artigos, apresenta um conjunto de mecanismos para a construção de novas aplicações de Governo Eletrônico Orientadas a Serviços e Centradas no Cidadão. A contribuição principal refere-se à infraestrutura da plataforma *CoGPlat*, concebida como um *middleware* que oferece suporte a aplicações e serviços de Governo Eletrônico inseridos no contexto da Web Semântica. A tese contribui ainda ao apresentar uma estratégia para realizar a composição de serviços de maneira transparente com o auxílio de anotações semânticas e a mediação através de políticas de interação. A implementação do protótipo do *middleware* e alguns cenários de aplicação também são discutidos.

Abstract

The Information and Communication Technologies (ICTs) are being applied vigorously by governmental entities around the world to better serve the society, a phenomenon often referred to as Electronic Government (*e-Government*). Initially the use of ICTs had a unique goal: to speed up internal bureaucratic processes (e.g. by processing huge amounts of data). Improvements in the delivery of citizen services were not considered a priority. In the recent years an enormous transformation is happening where one can observe increasing demands to improve citizen participation in public processes, to promote social e-Inclusion, to deliver new services which are more complex and more transparent and also to reduce bureaucracy. The inherent heterogeneity of the systems and processes, legal aspects and requirements such as privacy, autonomy and traceability create technology challenges that must be faced.

In this context, this thesis, organized as a collection of papers, presents mechanisms to facilitate the development of new service-oriented and citizen-centric applications. The main contribution is the proposal of the *CoGPlat* infrastructure, idealized as a middleware to support e-Government services and applications on the Semantic Web. The thesis also contributes by proposing a strategy to enable transparent compositions with support of semantic annotations and the mediation of interaction policies. Prototype implementation issues and some application scenarios are also discussed.

Agradecimentos

Primeiramente a **Deus**, Criador do Universo e Pai de toda a sabedoria e conhecimento. A possibilidade de realizar este trabalho foi mais uma das infinitas bênçãos derramadas até hoje em toda a minha vida.

Ao **Professor Dr. Edmundo Madeira** por sua impecável orientação, seu equilíbrio, sua compreensão e amizade. Este trabalho seria impossível sem a sua participação.

Aos meus **pais**, **tios** e **demais familiares** por todo amor e carinho.

À minha querida **esposa** pela compreensão, amor e companheirismo.

Aos meus queridos **amigos**, **colegas** e **professores** de **Santos**, de **Campinas**, de **Berlim** e de **Redmond**.

Ao **Instituto de Computação** e à **UNICAMP**, todo seu corpo docente, discente e de funcionários. Aos colegas do **LRC**, do **LSD** e do **LIS**. Foram 11 proveitosos anos nesta universidade. Sem dúvida alguma eu a considero meu segundo lar.

Ao **CNPq**, à **CAPES**, ao **DAAD** e à FAPESP pelo apoio financeiro em diferentes etapas do projeto.

Ao **Instituto Fraunhofer-FOKUS** (Alemanha) e ao seu Centro de Competência em Aplicações de Governo Eletrônico (ELAN). Agradecimentos especiais ao **Dr. Volker Tschammer**, a **Matthias Fluegge**, a **Gerd Schürmann** e ao **Prof. Dr. Dr. h.c. Radu Popescu-Zeletin** pelo acolhimento durante o período de doutoramento-sanduíche.

À **Microsoft Research** (EUA) e ao seu grupo de pesquisa em Banco de Dados. Agradecimentos especiais ao **Dr. Phil Bernstein**, ao **Dr. Sergey Melnik**, ao **Dr. Jonathan Goldstein** e ao **Dr. David Lomet**.

Lista de Acrônimos

AI Artificial Intelligence

BPMN Business Process Modeling Notation

CIM Computation Independent Model

CoGPlat Citizen-oriented e-Government Platform

CWM Common Warehouse Metamodel

DAG Directed Acyclic Graph

EDOC Enterprise Distributed Object Computing

EEC E-Governance and E-Democracy Center

EPAL Enterprise Privacy Authorization Language

GT4 Globus Toolkit 4.0

HTTP Hypertext Transfer Protocol

ICT Information and Communication Technology

IETF Internet Engineering Task Force

IOPE Input, Output, Preconditions and Effects

IRI Internationalized Resource Identifier

MDA Model Driven Architecture

MMC Metamodel Management Center

MOF Meta-Object Facility

OGSA Open Grid Services Architecture

OMG Object Management Group

ONG Organização Não Governamental

OWL Web Ontology Language

OWL-S Semantic Markup for Web Services

OWLS-MX Hybrid Semantic Web Service Matchmaker

P3P Platform for Privacy Preferences Project

PDDL Planning Domain Definition Language

PIM Platform Independent Model

PSM Platform Specific Model

QoS Quality of Service

RDF Resource Description Framework

RDFS Resource Description Framework Schema

RIF Rule Interchange Format

RM-ODP Reference Model for Open Distributed Processing

SAGA Standards and Architectures for e-Government Applications

SAWSDL Semantic Annotations for WSDL and XML Schema

SOA Service-Oriented Architecture

SOAP Simple Object Access Protocol

SPARQL Simple Protocol and RDF Query Language

SWS Semantic Web Service

TAC Traceability and Auditing Center

TIC Tecnologia de Informação e Comunicação

TSC Transparent Services Center

UDDI Universal Description, Discovery and Integration

UML Unified Modeling Language

UMM Unified Modeling Methodology

UN United Nations

UNSPSC United Nations Standard Products and Services Code

URI Uniform Resource Identifier

VO Virtual Organization

W3C World Wide Web Consortium

WF Windows Workflow Foundation

WS-BPEL Web Services Business Process Execution Language (ou BPEL)

WS-CDL Web Services Choreography Description Language (ou CDL)

WSDL Web Service Definition Language

WSDL-S Web Service Semantics

WSMF Web Service Modeling Framework

WSMO Web Service Modeling Ontology

XAML Extensible Application Markup Language

XML Extensible Markup Language

Sumário

Capítulo 1

Introdução

As Tecnologias de Informação e Comunicação (TICs) têm sido aplicadas de maneira cada vez mais intensa por entidades governamentais em todo o mundo com o objetivo de melhor servir à sociedade [Marchionini 03]. Este fenômeno, conhecido por *Governo Eletrônico* (ou *e-Government*), que inicialmente se limitava ao uso da informática para agilizar os processos burocráticos internos dos órgãos públicos, tem caminhado rumo a uma grande transformação. Os primeiros passos nesta direção foram dados com o início do oferecimento de serviços públicos tradicionais também por canais eletrônicos - no Brasil temos como exemplos de sucesso a *Declaração de Imposto de Renda via Internet* [Leite 98], a *Delegacia Eletrônica do Estado de São Paulo* [SSP-SP 08] e o projeto *e-Poupatempo* [Tokairim 03]. Mas isto representa somente o início da mudança: o preceito que estabelece que uma democracia requer cidadãos bem informados [Jefferson 89], de forma a aumentar a credibilidade dos administradores públicos e legitimar suas decisões, exige mecanismos que permitam a participação ativa e autêntica dos cidadãos nos mais diversos processos governamentais [Watson 01]. De forma a aumentar a eficiência, transparência e legitimidade destes processos, novos serviços e aplicações (em geral complexos e multi-organizacionais) precisam ser fornecidos. Questões como a heterogeneidade dos sistemas já existentes, a autonomia das diferentes entidades que participam nos processos, aspectos legais, e a privacidade dos dados sendo tratados, entre outras, criam uma série de novos desafios do ponto de vista computacional.

Considerando estes desafios, esta tese tem como principal objetivo propor uma plataforma de Governo Eletrônico, idealizada como um *middleware*, que facilite o desenvolvimento de novas aplicações centradas no cidadão, no contexto da Web Semântica, e que visem melhorar a qualidade, eficiência e abrangência dos serviços públicos.

Este capítulo introduz os principais conceitos, desafios e objetivos que permeiam o contexto desta tese e apresenta sua organização em formato de coletânea de artigos.

1.1 Interoperabilidade e Serviços

A **interoperabilidade**, definida como a *capacidade de dois ou mais sistemas compartilharem informação de maneira eficiente* [IEEE 90], é requisito fundamental no cenário atual de aplicações dinâmicas e distribuídas. Dentre as principais estratégias propostas para aumentar o grau de interoperabilidade de sistemas estão as baseadas na chamada **Arquitetura Orientada a Serviços** (*Service-Oriented Architecture - SOA*), abordagem que pode ser comparada a um modelo baseado em componentes que interrelaciona diferentes unidades funcionais (ou *serviços*) de uma aplicação através de interfaces bem definidas [Papazoglou 03]. Estas interfaces devem ser construídas de uma forma neutra, independente de plataforma (hardware e sistema operacional) e também da linguagem de programação usada para efetivamente implementar as funcionalidades do serviço. Isto permite que serviços construídos sobre uma gama heterogênea de sistemas possam interagir entre si de maneira uniforme [Papazoglou 03, IBM 05], formando novas aplicações. De maneira resumida, seguem algumas características encontradas nas soluções baseadas em SOA:

- **Abstração**: um serviço expõe publicamente somente a lógica descrita em seu contrato, escondendo os detalhes de implementação dos consumidores do serviço;
- **Autonomia**: um serviço controla somente a lógica que ele próprio encapsula;
- **Baixo acoplamento**: permite que a lógica encapsulada por um serviço seja modificada com praticamente nenhum impacto nos outros serviços que são parte da mesma arquitetura;
- **Contratos**: representam as descrições dos serviços e como estes podem ser acessados;
- **Reusabilidade**: é alcançada pela distribuição da lógica da aplicação entre diferentes serviços, de maneira que cada serviço possa ser potencialmente usado por mais de um serviço requisitante;
- **Capacidade de Composição**: representa a possibilidade de combinar os serviços em grupos com troca de dados coordenada;
- **Ausência de Estado**: serviços normalmente não mantêm seu estado com relação a uma atividade. Isso favorece o baixo acoplamento, a reusabilidade e a capacidade de composição. Quando é necessário manter o estado, isso normalmente é feito no nível das aplicações e/ou das composições;

- **Capacidade de Descoberta**: conjunto de mecanismos que permitem que as descrições de um serviço possam ser encontradas por seus consumidores em potencial.

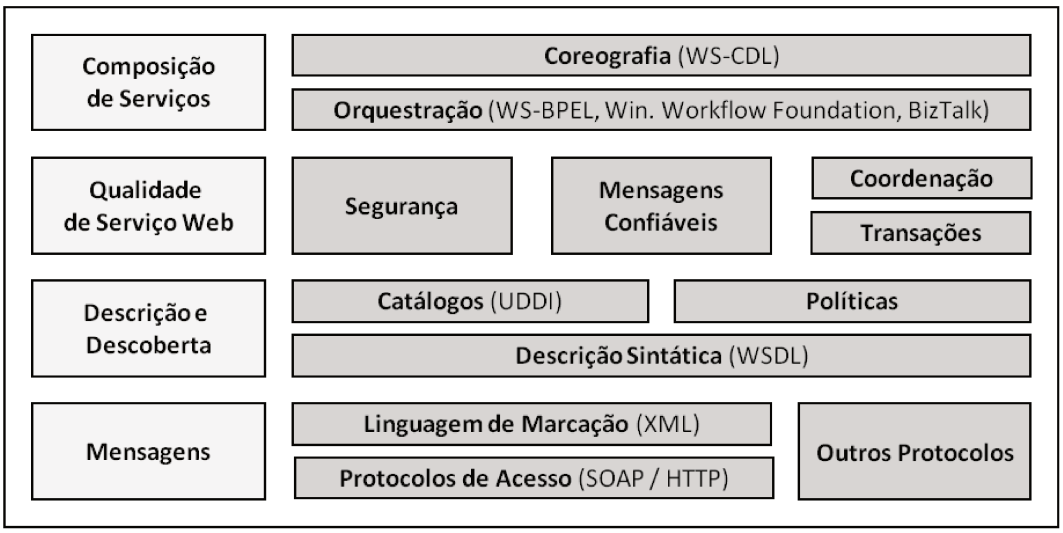

Figura 1.1: Pilha dos Serviços Web [Curbera 03]

No contexto da Internet, as tecnologias baseadas nos *Serviços Web* (Web Services) podem ser usadas para construir sistemas fundamentados em SOA. O *World Wide Web Consortium* (W3C) define um Serviço Web como "*um sistema de software projetado para suportar interações máquina-máquina em uma rede, que contém uma interface descrita em um formato processável pela máquina (WSDL - Web Services Description Language), e com o qual outros sistemas interagem orientando-se por sua descrição e usando mensagens SOAP (tipicamente sobre HTTP - HyperText Transfer Protocol) serializadas usando XML (eXtended Markup Language)*" [W3C 04d]. Diversas camadas formam a pilha SOA no mundo dos Serviços Web (Figura 1.1):

- *Transporte e Mensagens*: Trata do transporte e troca de mensagens entre *hosts*, usando normalmente os protocolos da Internet (HTTP, SOAP) e a linguagem XML como padrão de serialização dos dados;

- *Descrição e Descoberta*: preocupa-se com a descrição da interface de um Serviço Web (WSDL) e sua publicação e divulgação (via, por exemplo, um registro UDDI [Walsh 02]);

- *Qualidade de Serviço*: permite agregar a um sistema baseado em SOA requisitos não-funcionais como troca de mensagens confiáveis (WS-ReliableMessaging), aspectos de segurança (WS-Security) e outros relacionados com a Qualidade dos Serviços Web;

- *Composição de Serviços*: permite que novos serviços e aplicações sejam construídos a partir da combinação de serviços já existentes.

É importante salientar que para implementar um sistema baseado em SOA, não é necessário o uso de todas as camadas e especificações ilustradas na Figura 1.1. Na verdade, somente aquelas que contemplem funcionalidades desejadas podem ser adotadas para se tornarem parte do sistema. Este trabalho de doutorado dedica especial atenção às camadas de *Composição de Serviços* e de *Descrição e Descoberta*. Outros detalhes considerados relevantes sobre estas camadas serão apresentados nos demais capítulos da tese.

1.2 A Web Semântica

A Web está evoluindo para um estágio onde os dados passarão a ter *significado* para as máquinas, podendo ser *automaticamente compreendidos* e *processados*: a chamada *Web Semântica* [Berners-Lee 01, Shadbolt 06]. Ela fundamenta-se na publicação de metadados mais expressivos em um ambiente de conhecimentos compartilhados, trazendo as linguagens de representação de conhecimento e as ontologias para o contexto da Internet [Martin 07]. As anotações na Web Semântica expressam as relações entre fontes de informação na Web e as conectam a terminologias formais (as ontologias).

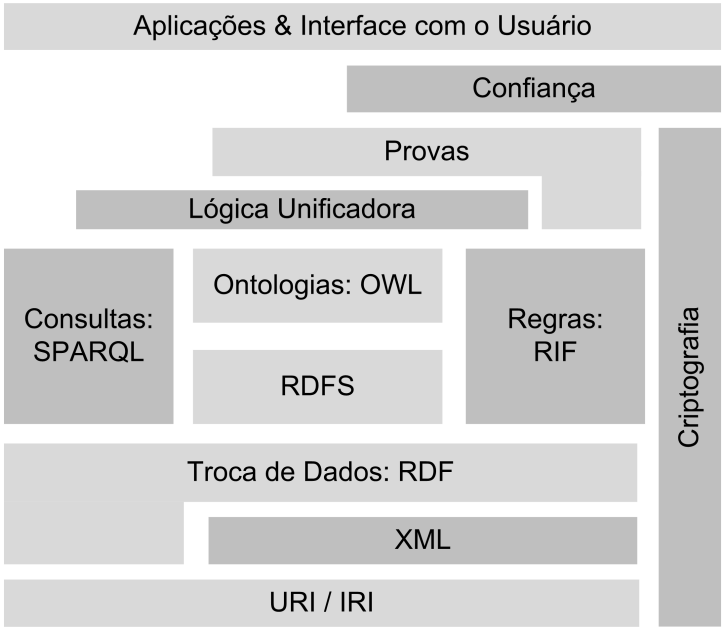

Figura 1.2: Camadas da Web Semântica [Berners-Lee 07]

Uma figura comumente usada para apresentar como a Web Semântica é estruturada é mostrada na Figura 1.2, o chamado "bolo em camadas" proposto por Tim Berners-Lee [Berners-Lee 07]. As camadas inferiores na figura (URI e XML, por exemplo) são formadas por padrões já existentes na Web e fornecem a base sintática para as camadas superiores. Iremos descrever ainda nesta seção algumas das camadas que formam o núcleo da Web Semântica e possuem maior relação com o trabalho apresentado nesta tese.

1.2.1 Ontologias

Inicialmente desenvolvidas na área de Inteligência Artificial, as *ontologias* tornaram-se tópico de estudo em diversas comunidades científicas. Uma *ontologia* pode ser definida como uma "especificação *formal* e *explícita* de uma *conceitualização compartilhada*" [Gruber 93], onde "*conceitualização*" se refere a uma abstração de um domínio que identifica os conceitos nele relevantes, e "*compartilhada*" significa que tais conceitos são consensuais, normalmente definidos de maneira cooperativa dentro de uma comunidade.

As ontologias formam a espinha dorsal da Web Semântica; o objetivo do seu uso é permitir que as máquinas compreendam as informações a partir dos relacionamentos entre as fontes de informação e os termos nas ontologias. Elas interligam a compreensão humana e a das máquinas sobre símbolos. Estes, também chamados de *termos* e *relacionamentos*, podem ser interpretados por ambos, homens e máquinas. O sentido para um humano é dado pelo próprio termo, normalmente uma palavra em linguagem natural, e pelos relacionamentos semânticas entre os termos (por exemplo uma relação superconceito-subconceito "*é um*", onde um conceito é mais geral que o outro). Como os significados dos relacionamentos são definidos formalmente, uma máquina pode analisá-los e obter as mesmas conclusões que um humano sobre eles [Fensel 06].

Formalmente, uma ontologia Ω contém um conjunto de *conceitos* (ou *classes*) $\{c-1, ..., c-n\}$, tendo cada classe um conjunto associado de *propriedades* $P-i = \{p-i1, ..., p-im\}$. As classes são relacionadas entre si de maneira similar ao mundo de orientação a objetos (sub-classes, super-classes, etc) [Medjahed 04]. A literatura apresenta normalmente três tipos diferentes de ontologia [Fensel 06] classificados de acordo com sua generalidade:

- **Genéricas** (ou de **nível superior**): capturam conhecimentos gerais (por exemplo tempo e espaço) que são independentes de domínio. São compartilhadas por um número grande de pessoas em diferentes domínios (exemplos: WordNet [Fellbaum 98] e CYC [Lenat 95]);
- **Domínio**: capturam o conhecimento específico em um domínio (exemplo: UNSPSC [UNSPSC 08], um esquema de classificação de produtos);

- **Aplicação**: capturam o conhecimento necessário para uma aplicação específica. Um exemplo poderia ser uma ontologia que representa a estrutura de um sítio Web particular. Podem não ser consideradas realmente ontologias, já que não são compartilhadas.

Outra classificação encontrada na literatura, ortogonal à generalidade, é a expressividade das ontologias. Os seguintes níveis são normalmente definidos [Fensel 06]:

- **Thesaurus**: relações entre termos (como sinônimos) são descritas;
- **Taxonomia informal**: existe uma hierarquia explícita (generalização e especialização) mas não existe herança estrita; uma instância de uma subclasse não é necessiariamente uma instância da superclasse;
- **Taxonomia formal**: existe herança estrita, ou seja, cada instância de uma subclasse também é instância da superclasse;
- **Frames**: um *frame* contém um número de propriedades e estas são herdadas pelas subclasses e instâncias (exemplo: ontologias expressas em RDFS);
- **Restrições de Valores**: os valores das propriedades são limitados (exemplo: ontologias escritas em OWL Lite);
- **Condições lógicas genéricas**: os valores das propriedades podem ser limitados por fórmulas lógicas ou matemáticas que usam valores de outras propriedades (exemplo: ontologias escritas em OWL DL);
- **Condições lógicas expressivas**: permitem o uso de condições em lógica de primeira ordem e relações mais ricas tais como disjunção de classes, relacionamentos inversos e de parte-todo.

As especificações RDF (*Resource Description Framework*) e RDF-*Schema* (RDFS) [W3C 04b] oferecem um framework para modelar meta-dados referentes a *recursos* na Web (onde um *recurso* é qualquer coisa que possa ser identificada por uma URI - *Uniform Resource Identifier*). O modelo de dados de RDF descreve triplas do tipo *sujeito-predicado-objeto*: por exemplo na frase "*Brasília é a capital do Brasil*", "*Brasília*"é o sujeito, "*é a capital do*"é o predicado e "*Brasil*"é o objeto. Um objeto de uma tripla pode também servir de sujeito de outra tripla, formando um grafo orientado (com rótulos) onde os recursos (sujeitos e objetos) correspondem aos nós e os predicados correspondem às arestas. RDFS representa uma extensão de RDF com um vocabulário para definir classes, hieraquia de classes e restrições a propriedades.

A linguagem de ontologia para a Web (**OWL** - *Web Ontology Language* [W3C 04a]) estende RDFS, aumentando seu vocabulário com a inclusão, por exemplo, de novos relacionamentos entre classes. OWL é definida em três variantes com níveis crescentes de expressividade:

- **OWL Lite**: a variante menos expressiva de OWL. Comparada com RDFS, ela adiciona restrições com faixas locais, restrições existenciais, restrições de cardinalidade simples, igualdade, e vários tipos de propriedades (inversão, transitividade e simetria);
- **OWL DL** (*Description Logic*): Comparado com OWL Lite, OWL DL adiciona suporte completo a negação, disjunção, restrições de cardinalidade, enumerações e restrições de valores. Garante completude e computabilidade. Note que apesar das restrições sintáticas, a expressividade de OWL Lite é bem próxima à de OWL DL;
- **OWL Full**: suporta máxima expressividade, mas não oferece garantias computacionais. Permite ainda modificações na própria linguagem. Devido a esta liberdade sintática, alguns problemas de inferência importantes são indecidíveis em OWL Full.

1.2.2 Serviços na Web Semântica

Como o próprio nome indica, os *Semantic Web Services* (SWS) situam-se na intersecção de dois importantes movimentos na evolução da Web descritos anteriormente neste capítulo: os Serviços Web e a Web Semântica. O tema central dos SWS é o uso de descrições mais ricas e declarativas dos elementos participantes em uma computação distribuída: serviços, processos, conversas baseadas em mensagens, transações, entre outros. Essas descrições permitem uma maior automatização e flexibilização na provisão e uso dos serviços [Martin 07]. Esta possibilidade motivou a proposta de diferentes especificações, descritas a seguir, para modelagem e anotação semântica de serviços. Apesar de apresentarem algumas diferenças na abordagem, todas possuem como objetivo principal facilitar o caminho para a descoberta, invocação, composição e inter-operação automáticas entre serviços.

A linguagem **OWL-S** (*Semantic Markup for Web Services*) [DAML 06, Coalition 04] é construída sobre OWL, agregando um conjunto de ontologias inter-relacionadas que oferecem um repertório de termos usados em aplicações baseadas em serviços [Ankolekar 06]. Já a ontologia para modelagem de serviços Web **WSMO** (*Web Service Modeling Ontology*) [W3C 05] é outra proposta importante que visa descrever todos os aspectos relacionados a serviços genéricos que podem ser acessados através de interfaces Web, tendo sua base conceitual no WSMF (*Web Services Modeling Framework*) [Fensel 02]. WSMO identifica quatro elementos de alto nível como seus principais conceitos: ontologias, serviços,

objetivos e mediadores. Enquanto OWL-S é especificada em OWL, WSMO usa um modelo MOF (Meta-Object Facility) abstrato. Existem, entretanto, algumas similaridades conceituais entre OWL-S e WSMO: um perfil de serviço OWL-S é próximo a um objetivo em WSMO, embora WSMO faça uma diferenciação conceitual entre a visão de um provedor e a visão de um requisitante, o que não ocorre em OWL-S.

Recentemente o W3C adotou como padrão para anotações semânticas o **SAWSDL** (*Semantic Annotations for WSDL and XML Schema*) [Kopecký 07], uma abordagem minimalista que propõe uma extensão direta ao WSDL (seu nome anterior era inclusive WSDL-S). Nele um conjunto de anotações semânticas é adicionado ao esquema XML do WSDL, permitindo a descrição semântica das entradas, saídas e operações de um serviço Web. O modelo semântico fica fora do escopo de SAWSDL, sendo este imparcial em termos da linguagem de representação de ontologias (a especificação menciona como exemplos WSML, OWL ou ainda UML). Além disso, atributos para referenciar mapeamentos entre tipos complexos em esquemas XML e seus conceitos correspondentes nas ontologias também são tratados por SAWSDL. Devido à sua abordagem minimalista, muitos autores tem considerado SAWSDL ortogonal tanto a WSMO quanto a OWL/OWL-S [Fensel 06]. Esta tese escolheu OWL-S como padrão para a descrição semântica dos serviços Web por sua maior proximidade com OWL (padrão do W3C) e também por sua maior adoção na comunidade acadêmica e na indústria.

1.3 Governo Eletrônico

O termo *Governo Eletrônico* (*e-Government*), apesar de recente, designa um campo de atividade que já existe há algumas décadas e que tem alcançado um alto grau de penetração em diversos países [Lenk 02]. Todos os processos relacionados à tomada de decisões ou à prestação de serviços na política, governo e administração pública e que fazem uso de tecnologias de informação e comunicação podem ser enquadrados em seu escopo [KBSt 06].

Nos últimos anos mudanças significativas têm sido observadas nos objetivos das aplicações de Governo Eletrônico: o foco passou a ser não somente informatizar processos internos do setor público ou ainda disponibilizar o acesso a serviços tradicionais por meios eletrônicos, mas também oferecer serviços novos, centrados no cidadão e criados dinamicamente a partir das suas necessidades. O avanço e a proliferação das TICs (Tecnologias de Informação e Comunicação) facilitam a criação destes novos serviços que exigem a colaboração entre diversos órgãos governamentais. Ganha momento também o interesse por uma participação maior dos cidadãos nos processos e decisões governamentais como forma de aumentar a transparência e legitimidade dos gestores públicos, fenômeno conhecido como *Participação Eletrônica* (*e-Participation*).

O *Governo Eletrônico Centrado no Cidadão* pode ser definido, de maneira simples, como aquele que trata os cidadãos como seus clientes, de maneira a privilegiar suas necessidades em detrimento da burocracia ou de outros imperativos dentro da máquina pública, atuando sempre dentro dos limites impostos pela legislação. Os seguintes princípios marcam uma postura centrada no cidadão (*citizen-centric*) [GOV3 06, Santos 07]:

1. Tratar os cidadãos e entidades como clientes do governo como um todo (e não somente de um órgão/departamento específico);
2. Utilizar uma arquitetura orientada a serviços que alcance o governo como um todo e apoie todo e qualquer tipo de interação;
3. Desenvolver um local único (*on-line* e/ou físico) para que os cidadãos possam ter acesso a todas as informações e serviços transacionais oferecidos pelo governo;
4. Estar preparado para a necessidade de evoluir sempre, de forma dinâmica, aprendendo com os erros, experiências e pronto a atender novas necessidades dos cidadãos.

Levando em conta os aspectos tecnológicos, os seguintes requisitos podem ser identificados como fundamentais para o sucesso de qualquer aplicação de governo eletrônico [KBSt 06, Santos 07]:

- **Interoperabilidade**: os processos administrativos devem ser co-orientados de maneira que diferentes aplicações de governo eletrônico possam interagir sem maiores dificuldades. A interoperabilidade pode ser classificada em três níveis:
  1. *Organizacional*: refere-se à determinação de quando e porque uma certa informação é trocada;
  2. *Técnica*: refere-se à possibilidade de trocar informação, incluindo a definição de protocolos e rotas de transmissão;
  3. *Semântica*: existe quando dois sistemas trocam informação de maneira que esta é interpretada da mesma maneira por ambos. Aplica-se não somente ao formato mas também ao conteúdo dos dados transmitidos.
- **Reusabilidade**: o reuso pode ocorrer em diferentes níveis de abstração, como por exemplo no intercâmbio de experiências entre agências, no uso de modelos compartilhados de dados e processos ou ainda de serviços comuns;
- **Abertura**: as aplicações devem oferecer interfaces bem definidas e bem documentadas. Padrões abertos devem ser adotados sempre que possível;

- **Escalabilidade**: deve-se garantir a usabilidade das aplicações à medida que o volume e freqüência de transações aumenta;

- **Segurança**: as aplicações e o *middleware* de suporte devem garantir que a informação somente possa ser acessada, modificada ou publicada respeitando-se uma política de segurança pré-definida, preservando assim a confidencialidade e integridade dos dados. Além disso, algumas aplicações de governo eletrônico exigem outros requisitos de segurança, como anonimato, proteção ao roubo de identidade e irretratabilidade.

1.4 Desafios

Inicialmente pode parecer que os desafios tecnológicos que um *middleware* de Governo Eletrônico precisa enfrentar não são diferentes daqueles encontrados nos projetos computacionais de larga escala do setor privado. Entretanto, os requisitos de governança eletrônica vão estritamente além dos encontrados no mundo dos negócios, principalmente nos aspectos relacionados com flexibilidade, transparência e interoperabilidade. Para compreender esse fato é importante considerar a origem destes requisitos: eles refletem as necessidades e ambições coletivas da nossa sociedade, expressas através da combinação da legislação com a opinião pública. A extensão da colaboração requerida é muito maior, e os resultados são muitas vezes diferentes dos objetivos imediatos de negócio das organizações participantes. De acordo com *Davies et. al.* [Davies 07], as principais características que diferenciam a governança eletrônica e a administração pública dos negócios eletrônicos e da administração privada podem ser organizadas nas seguintes categorias:

1. **Aspectos Regulatórios e de Políticas**: relacionados com a estruturação legal dos processos administrativos. Incluem:

   (a) **Proteção à Privacidade**: as agências do governo tipicamente precisam coletar informações dos cidadãos quando estes solicitam serviços eletrônicos. Algumas vezes, dentro do contexto de serviços multi-organizacionais, parte das informações precisa ser compartilhada com outras agências para que os serviços possam ser realizados. O problema é que em muitos casos a legislação proíbe que uma agência compartilhe informações de cidadãos com outras sem o expresso consentimento do mesmo e esse direito deve ser garantido.

   (b) **Critério da Elegibilidade**: a maioria dos serviços públicos estabelece um conjunto de critérios que precisam ser verificados para determinar se o solicitante (cidadão) tem direito ou não ao serviço.

(c) **Gerenciamento de Identidade**: a verificação positiva da identidade de um solicitante (e o correto gerenciamento e guarda dessa informação) é fundamental para o fornecimento da maioria dos serviços públicos.

(d) **Proteção ao Anonimato**: Ao mesmo tempo em que a identificação positiva é necessária em muitas das interações com o governo, existe um bom número de processos governamentais, como consultas públicas e votações, que exigem anonimato. Embora inicialmente um cidadão precise ter sua identificação confirmada para comprovar que tem direito de participar de um referendo, não deve ser possível identificar seu voto. Apesar do anonimato também ser uma característica importante em aplicações do setor privado, as conseqüências legais de violações de anonimato no setor público são geralmente mais severas.

(e) **Acessibilidade**: ao contrário de algumas aplicações do setor privado, onde o fato do público alvo ser restrito minimiza os requisitos de acessibilidade, as aplicações do setor público devem atender a todos os segmentos da sociedade, o que torna mecanismos de acessibilidade essenciais.

(f) **Adoção de Padrões**: considerando a heterogeneidade de processos e tecnologias encontrados nas agências do setor público, a adoção de padrões, tanto administrativos quanto de tecnologia, passa cada vez mais a ser fundamental para garantir um mínimo de interoperabilidade.

(g) **Dinamismo dos Mecanismos Regulatórios**: os sistemas de informação da administração pública precisam estar preparados para se adaptar a mudanças na legislação, principalmente relacionadas aos aspectos anteriormente citados (privacidade, anonimato, elegibilidade, identificação e acessibilidade).

2. **Aspectos Organizacionais**:

(a) **Colaboração**: as agências governamentais precisam se envolver em diferentes formas de colaboração dentro e fora do governo, tanto com organizações públicas quanto privadas. As colaborações entre agências, por exemplo, podem envolver o compartilhamento de informações, de opiniões técnicas que apóiem processos de tomada de decisão, a certificação e validação de informações. Um desafio grande nos processos governamentais colaborativos é garantir a não violação das regras de privacidade estabelecidas na legislação, considerando em especial a possibilidade de dados dos cidadãos estarem em custódia de organizações do setor privado.

(b) **Serviços Administrativos**: tipicamente um grande número de serviços públicos é oferecido pelos governos. Entretanto, muitos destes serviços são frequentemente similares do ponto de vista dos processos administrativos internos que

os implementam, fato que pode ser muito valioso no processo de modelagem e implementação de plataformas de suporte a Governo Eletrônico.

(c) **Conhecimento**: muitos dos serviços prestados pelos governos envolvem processos complexos de tomada de decisão e conhecimentos especializados, como por exemplo: conhecimentos legais, conhecimentos sobre os recursos que o governo tem à disposição, conhecimento sobre a eficácia das diferentes medidas possíveis e também uma memória dos processos que é construída gradualmente a partir de casos anteriores. Essas características tornam os requisitos do gerenciamento de conhecimento (*knowledge management*) no setor público diferentes dos requisitos existentes na maioria das aplicações encontradas no setor privado: o conhecimento é aplicado principalmente para ações administrativas no domínio público enquanto que no domínio privado é principalmente usado como ferramenta na criação de inovações.

A Tabela 1.1 mostra quais destes desafios serão atacados na tese e quais as soluções propostas, detalhadas nos capítulos seguintes.

Desafio	**Solução Proposta**
1.(a) / 1.(c) / 1.(d) / 2.(a)	Políticas que regulam as interações entre parceiros que colaboram nos processos (interoperabilidade organizacional)
1.(f)	Arquitetura orientada a serviços + padrões nos mecanismos de comunicação para facilitar a interoperabilidade técnica
1.(g) / 2.(b)	Mecanismo de composições abstratas e o uso de políticas que facilitam a adaptação dos sistemas a mudanças e também o reuso de composições em serviços administrativos similares
2.(c)	Estratégias de rastreamento e auditoria dos processos

Tabela 1.1: Desafios e Proposta de Soluções

1.5 Objetivos, Requisitos e Estratégias

Esta tese tem como principal meta propor uma plataforma de Governo Eletrônico centrada no cidadão que facilite o desenvolvimento de novas aplicações com o objetivo de:

- Melhorar a qualidade, eficiência e alcance dos serviços públicos;

- Aumentar o grau de transparência dos processos governamentais;
- Incentivar uma maior participação dos cidadãos, legitimando atos do gestor público e fortalecendo a democracia;
- Valorizar aplicações e processos centrados principalmente no cidadão e não somente na máquina administrativa e nos processos burocráticos.

Levando em conta estes objetivos, a Tabela 1.2 apresenta os requisitos mais relevantes introduzidos pelo novo cenário de serviços de Governo Eletrônico centrados no cidadão e as respectivas estratégias que adotamos no projeto apresentado nesta tese.

Requisito	**Estratégia**
Tratar a **heterogeneidade** de plataformas	Orientação a **Serviços** e **Padrões**
Oferecer **novos serviços**, **complexos** e **multi-organizacionais**	**Estratégias de Composição** baseadas em recursos da **Web Semântica**
Obedecer a requisitos impostos pela **legislação** como **autonomia** + **privacidade** + **segurança** + **rastreabilidade**	**Políticas** de Interação
Simplificar o processo de desenvolvimento de **novas aplicações**	Facilidades de **Democracia e Governança Eletrônica** + Estratégias anteriores

Tabela 1.2: Requisitos e Estratégias

O primeiro requisito tratado é possibilitar a colaboração entre as aplicações e sistemas já existentes (inerentemente heterogêneos) e a plataforma: a interoperabilidade é alcançada com a adoção de uma abordagem orientada a serviços e baseada nos padrões da Internet e dos Serviços Web.

O segundo requisito é oferecer serviços novos e adaptados às necessidades do cidadão, o que muitas vezes pode envolver múltiplas organizações e requerer características dinâmicas: a estratégia escolhida é a composição de serviços usando os recursos da Web Semântica.

O terceiro requisito está relacionado com exigências da legislação (ou mesmo requisitos não funcionais de determinadas aplicações) que impliquem em garantir que as composições, muitas vezes elaboradas de maneira dinâmica, respeitem a autonomia das entidades envolvidas, a privacidade e segurança dos dados ou ainda ofereçam recursos para acompanhamento e auditoria dos processos em andamento. Para satisfazer esses requisitos as composições são regidas por um conjunto de Políticas de Interação.

O quarto e último requisito que esta tese trata é a simplificação do processo de desenvolvimento de novas aplicações oferecendo, diretamente no *middleware*, um conjunto de funcionalidades básicas encontradas normalmente nos domínios de Democracia e Governança Eletrônica. Além disso, as estratégias propostas para atender aos outros requisitos da tabela também contribuem diretamente para este facilitar este processo.

1.6 Contribuições

A principal contribuição desta tese é a proposta de uma plataforma (***CoGPlat*** - ***C****itizen-****o****riented e-****G****overnment* ***Plat****form* [Santos 05]) que oferece suporte para serviços e aplicações de Governo Eletrônico (*e-Government*) centrados no cidadão e inseridos no contexto da Web Semântica. O objetivo primordial é facilitar a construção e a operação de aplicações que habilitem a interação e colaboração entre entidades governamentais, organizações e cidadãos em diferentes cenários da administração pública.

Uma segunda importante contribuição da tese está relacionada com as técnicas utilizadas para implementar Composições de Serviços no contexto da Web Semântica. O processo proposto inclui um mecanismo de seleção dinâmica de serviços considerando suas descrições semânticas e a mediação das interações entre os serviços de acordo com diversos tipos de políticas que tratam aspectos como privacidade, autonomia e anonimato.

1.7 Organização da Tese

Esta tese está organizada como uma coletânea de artigos que detalham as principais contribuições do doutorado, sendo que apenas os artigos mais recentes e atualizados foram selecionados (consulte o Apêndice A para uma relação completa das publicações). São eles:

Capítulo 2: "Challenges and Techniques on the Road to Dynamically Compose Web Services"

> Matthias Fluegge, Ivo J. G. Santos, Neil Paiva Tizzo e Edmundo R. M. Madeira. Em *ICWE '06: Proceedings of the 6th international conference on Web engineering*, páginas 40-47, Palo Alto, Califórnia, EUA, 2006. ACM Press.

O principal objetivo deste artigo é discutir de maneira crítica os diversos desafios e alternativas no caminho da composição dinâmica de serviços Web. Ao apresentar uma revisão conceitual e bibliográfica da área, o artigo identifica que, apesar da relevância do tema, existe ainda uma falta de consenso no tocante a uma definição clara do conceito

de *composição dinâmica de serviços*. A partir desta constatação, a primeira contribuição do artigo é a proposta de critérios claros de classificação de diferentes estratégias de composição em diferentes níveis de dinamismo e automatização.

Além disto, outra importante contribuição do artigo é a discussão de uma abordagem genérica orientada a modelos (MDA) para realizar a composição de serviços. Esta abordagem é aprofundada para ilustrar diferentes técnicas que podem ser aplicadas para possibilitar maior grau de dinamismo nas composições. Tópicos como a seleção dinâmica de serviços e o uso de técnicas de inteligência artificial na elaboração dos planos de execução também são discutidos. Finalmente, um exemplo de serviço no contexto do Governo Eletrônico é apresentado para ilustrar o uso de algumas das técnicas discutidas.

No contexto da tese, apesar de não tratar diretamente do projeto *CoGPlat*, este artigo foi selecionado para compor a coletânea por introduzir uma revisão bibliográfica e conceitual da área e por discutir criticamente abordagens que estão diretamente relacionadas aos fundamentos tecnológicos aplicados em *CoGPlat*.

Capítulo 3: "A Semantic-enabled Middleware for Citizen-centric E-Government Services"

> Ivo J. G. Santos e Edmundo R. M. Madeira. *ACM Transactions on the Web*, 2008. Artigo em processo de revisão (ainda não publicado).

Considerado o principal desta coletânea, o artigo apresenta em detalhes a infraestrutura da plataforma *CoGPlat*, concebida como um *middleware* que oferece suporte a aplicações e serviços de Governo Eletrônico inseridos no contexto da Web Semântica.

Após uma ampla revisão dos trabalhos e conceitos correlatos, o artigo detalha as facilidades do núcleo da plataforma: o *Centro de Serviços Transparentes*, o *Centro de Gerência de Metamodelos*, o *Centro de Governança e Democracia Eletrônica* e o *Centro de Auditoria e Rastreabilidade*. Uma estratégia para facilitar a composição de serviços de Governo Eletrônico baseada em descrições semânticas também é proposta e representa uma das principais contribuições do artigo.

Requisitos como autonomia, privacidade, rastreabilidade e anonimato são tratados com o uso de políticas de interação entre os serviços. Uma discussão sobre as opções tecnológicas adotadas para viabilizar a implementação do protótipo do *middleware* também é apresentada. Finalmente, um cenário de aplicação inspirado em processos burocráticos do setor da construção civil é introduzido e sua implementação sobre *CoGPlat* é ilustrada. O artigo apresenta ainda uma avaliação qualitativa da plataforma, com destaque para aspectos como flexibilidade, dinamismo, extensibilidade e interoperabilidade.

Capítulo 4: "Transparency in Citizen-Centric Services: A Traceability-based Approach on the Semantic Web"

Ivo J. G. Santos e Edmundo R. M. Madeira. Em: *Proceedings of the Tenth International Conference on Enterprise Information Systems (ICEIS)*, Volume SAIC, páginas 184-189, Barcelona, Espanha, Junho 2008.

O artigo fundamenta-se na constatação de que a busca por estratégias eficientes para aumentar os níveis de transparência nos processos públicos tem se tornado aspecto fundamental para o sucesso das novas aplicações de Governo Eletrônico.

Dentro do contexto do projeto *CoGPlat*, o artigo apresenta uma abordagem para monitorar e auditar serviços compostos oferecidos sobre o *middleware*. A solução é baseada em um conjunto de *políticas de rastreabilidade* (*traceability*) que regulam como as composições devem ser monitoradas pela plataforma.

As contribuições deste artigo relacionam-se principalmente com o *Centro de Rastreabilidade e Auditoria* de *CoGPlat*. O artigo apresenta ainda um exemplo de composição com o objetivo de ilustrar as estratégias adotadas na implementação da abordagem.

Capítulo 5: "E-Government and Grid Computing: Potentials and Challenges towards Citizen-Centric Services"

Ivo J. G. Santos e Edmundo R. M. Madeira. Em: *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS)*, páginas 144-148, Portugal, 2007.

Idealizado como um *position paper*, o artigo discute a potencial aplicabilidade das Grades Computacionais nas aplicações de Governo Eletrônico, principalmente devido ao seu alto poder computacional, de armazenamento e a sua recente convergência com o mundo da orientação a serviços.

Partindo de uma investigação do estado da arte, desafios como interoperabilidade, escalabilidade e segurança são discutidos, bem como possíveis estratégias e questões ainda consideradas em aberto pela comunidade científica, dentre elas: o suporte avançado a processos baseados em *workflows* complexos, mecanismos robustos de tolerância a faltas, requisitos específicos de segurança e suporte a descrições semânticas.

No contexto desta coletânea e também do projeto *CoGPlat*, este artigo foi selecionado principalmente por apontar diferentes direções de trabalhos futuros. Uma destas seria a implementação de uma arquitetura inspirada em *CoGPlat* sobre um ambiente de Grades de Serviços Computacionais.

Apêndices

A tese inclui ainda 4 apêndices, assim organizados:

- **Apêndice A**: apresenta a lista completa de publicações realizadas a partir deste trabalho de doutorado;
- **Apêndice B**: apresenta o código fonte da ontologia dos serviços e políticas de *CoGPlat*;
- **Apêndice C**: apresenta os diagramas UML e o código fonte em C# das interfaces do barramento de serviços de *CoGPlat*;
- **Apêndice D**: apresenta um exemplo de perfil OWL-S do serviço de autorização para construção de casas.

Capítulo 2

Challenges and Techniques on the Road to Dynamically Compose Web Services

2.1 Introduction

The term *interoperability* can be defined as *'the ability of two or more systems or components to exchange information and to use the information that has been exchanged'* [IEEE 90]. In fact, interoperability plays a key role in the new world of networked applications, specially in the e-Business and e-Government domains. In this context, much has been discussed in the literature regarding SOA (*Service Oriented Architectures*) and the so-called *dynamic* (or *automatic*, *adaptive* and even *autonomic*) *Service Composition*, but the fact is that there is up to the moment no consensus regarding the exact definition and broadness of these terms.

The first contribution of this paper is the proposal of a set of parameters to classify the compositions into different levels of dynamism and automation. Based on these parameters, we then identify that most composition approaches applied in real-world contexts using traditional Web service technologies [Milanovic 04] (such as WSDL and BPEL) can be classified as *static* since the process model is created manually and the services are bound at design time. There are some attempts to apply late binding of services based on fixed interfaces and message formats, being these considered to be semi-dynamic service compositions. Approaches exposing a higher degree of dynamics can hardly be found in a real world context. Apart from obstacles such as performance and trust, the reason is clearly related to the fact that *traditional* Web services just partially kept their promise of being self-contained and self-describing software components.

Taking all this into account, we propose a path to increase the levels of dynamism and automatization in the service composition process, showing where resources such as ontologies and Artificial Intelligence (AI) techniques could be applied using the Model Driven Architecture (MDA) approach. The strategy is then exemplified through a composition in the e-Government context. Throughout the paper, our main goal is to discuss more general strategies and techniques, which can be further applied in different scenarios, instead of focusing on some specific technology or implementation. This article is organized as follows: Section 2 presents an overview of the steps of a service composition process; Section 3 compares different composition strategies and draws one path towards a fully dynamic composition process; Section 4 presents an example in the e-Government context; and finally in Section 5 conclusions and final remarks are stated.

2.2 The Service Composition

A composite service can be regarded as a combination of activities (which may be either atomic or composite services), invoked in a predefined order and executed as a whole. In this way, a complex service has the behavior of a typical business process. In order to build a service composition, some steps must be taken (not necessarily in this order): (1) A *process model* specifying control and data flow among the activities has to be created; (2) Concrete services to be bound to the process activities need to be discovered. The service composer usually interacts with a broker, e.g. a service registry, in order to look up services which match with certain criteria; (3) The composite service must be made available to potential clients. Again the broker is used to publish a description and the physical access point of the service; (4) During invocation of a composite service a coordinating entity (e.g. a process execution engine) may manage the control flow and the data flow according to the specified process model (see Section 2.2.3). Next we further analyze some characteristics of these steps which we will use later as criteria to classify the compositions.

2.2.1 Discovery, Selection and Binding

The selection of the activities which will participate in a service composition may be done either at design time or at run-time. In the former, the bindings are static, i.e. each instantiation of the composite service will be made up of the same constituent services. In the latter, the constituent services are selected at run-time, based on automatically analyzable criteria, such as service functionality, signature and QoS parameters. Late binding implies the dynamic invocation of the constituent services, i.e. a sufficient level of interoperability has to be established, either through fixed interfaces or by applying

more sophisticated matchmaking and mapping mechanisms. For a service provider, the applied binding mechanism has several business implications. In a growing service market, third party service providers may offer the same functionality at different conditions, e.g. regarding QoS parameters like price. Applying late binding, the discovery and invocation may become scalable as the number of services increases. Thus the costs of a composite service offered by a provider may decrease along with the growing competition in the associated marketplace. The cost advantage can be either handed over to the consumer or it will increase profitability at the provider's side. Furthermore, late binding may enhance fault-tolerance and thus reliability. Since the actions in a process are not hardwired to concrete services, the unavailability of a service may be compensated through the invocation of a functionally equivalent one. In addition, there are some scenarios where important service characteristics (like price) change constantly, what makes the use of run-time service discovery almost essential for the success of the composition. On the other hand, in some specific application domains, the lack of determinism, i.e. the fact that it is not possible to previously know which service is going to be selected, is not acceptable.

2.2.2 Creation of the Process Model

Another significant characteristic of a service composition strategy is the degree of automation in the creation of the process model. Traditional service composition methods require the user to define the data flow and the control flow of a composite service manually, either directly or by means of designer tools, e.g. in a drag-and-drop fashion. Subsequently the process description is deployed in a process execution engine. Depending on the abstraction level provided by the tools and also on the applied binding mechanism, the user either creates the process model based on concrete service descriptions or based on abstract service templates which are representatives for sets of services, i.e. for service classes. With respect to the multitude of available services and service templates, it may be a time-consuming task to manually select reasonable building blocks for the composite service. Furthermore, the creation of the data flow, i.e. the parameter assignments between the activities, can be complex and might require the user to have extensive knowledge about the underlying type representations. More advanced composition strategies actively support the user with the creation of the process model, which is often referred to as *semi-automated service composition*. Corresponding modeling tools may interact with a broker in order to automatically look up services which match (regarding IOPEs - Inputs, Outputs, Preconditions, Effects) with the already available control and data flow, thus facilitating and accelerating the creation of the process model. The same applies for the creation of models that are based on abstract functional building blocks (which will be

bound to concrete services at run-time). Parameter assignments between these building blocks may be automatically recommended based on an analysis of the underlying types and concepts.

Fully-automated composition approaches intend to generate a service composition plan without human interaction. Mostly AI inspired methods based on formal logic are used for that matter, such as automated reasoning through theorem proving. By means of a planning algorithm a workflow graph containing available activities or concrete services is generated to satisfy the requirements established by the requestor. If there are multiple solutions for the problem, i.e. several plans satisfy the given set of requirements, a selection is made based on QoS parameters. This selection can either be made by the process designer or automatically through predefined weighting and ranking functions. Combining the latter with late service binding implies that the complete service composition (i.e. process model generation and service selection) can be performed at run-time. The question to which extent the composition procedure can be automated is subject to research. Fully automated service composition may work in narrow and formally well defined application domains. The more complex the context however the more difficult it will be to apply the automated service composition approach in real-world applications. Again the applied degree of automation for generating the process model has significant business implications for a provider who composes services and delivers them to consumers. As mentioned above, modeling the control flow and the data flow of a composite service may be time-consuming tasks. (Semi-) automated composition techniques promise to speed up this procedure, thus bringing down the costs for developing new services. Furthermore time-to-market is accelerated since the provider may react faster and more flexible to the customer requirements. In addition the designed composite services improve in quality as the application of "intelligent"tools helps to create more efficient processes, e.g. by proposing parallel execution of functionally independent activities. One interesting (and feasible) approach for semi-automated compositions was proposed in the SATINE project [Dogac 04]. It is based on *self-contained activity components* [Fluegge 04], which are created semi-automatically based on OWL-S [Coalition 04] service ontologies.

2.2.3 Execution

When composing Web Services, two different execution models are usually applied: *Orchestration* and *Choreography*. There is not a common sense regarding these two definitions, but we can consider that in an *Orchestration* all interactions that are part of a business process (including the sequence of activities, conditional events, among others) must be described, like on a traditional workflow system. This description is then executed by an orchestration engine, which has control of the overall composition. On the

other hand, a *Choreography* is more collaborative and less centralized in nature. Only the public message exchanges are considered relevant and more, each service only knows about its own interactions and behavior. Differently from Orchestration, there is not an entity that has a global view/control of the composition [Peltz 03, Ross-Talbot 05]. If we refer to the origin of the words, a good comparison can be made. The first, orchestration, can be compared to a set of musicians (*services*) commanded by a conductor (*engine*). The second, choreography, can be compared to a group of dancers (*services*) that already know how to perform and that don't obey to a central coordination. Usually real scenarios involving complex systems with multi-part interactions demand both approaches. In [Santos 06] the authors propose a set of policies to regulate service compositions and establish a relationship among these policies and the execution models.

2.3 Towards a Dynamic Service Composition Process

In this section we first present a clear classification of different composition strategies in terms of dynamism and automation. Then we draw a possible path towards a fully dynamic composition process based on a model-driven approach.

2.3.1 Service Composition Strategies

Besides the execution model (orchestration or choreography), two service composition characteristics have been examined, namely the type of service discovery/selection/binding and the degree of automation applied for the creation of a process model: service composition approaches may use early binding or late binding; the process model can be created manually, semi-automatically or automatically. As illustrated in Figure 2.1, these characteristic values can be used for a classification of existing service composition strategies in six main categories. The fact that the borders between these categories are not strict but fluent is made clear through the smooth transitions between the squares. Some categories may overlap, i.e. there are composition approaches that may be assigned to two or more categories. To give an example: besides early and late binding there may be several variations in between, such as the specification of a restricted set of service candidates at design time from which one service is chosen and invoked at run-time.

There are numerous examples for each of the identified categories. In EFlow [Casati 00] and similar systems the data flow and the control flow of a composite service need to be defined manually. The services however can be bound early at design time or at run-time by means of appropriate proxies. An example for semi-automated creation of the process model based on semantic service descriptions has been presented by *Sirin et. al.* [Sirin 03]. All possible services that match with the current activity are presented to the

user. The semi-automated composition approach described by *Fluegge and Tourtchaninova* [Fluegge 04] uses abstract functional building blocks which are bound to concrete services at run-time. The user is supported with the creation of the process model by automatically analyzing the input and output concepts of the building blocks. An example for fully automated service composition based on AI-inspired planning algorithms is given by *Ponnekanti and Fox* [Ponnekanti 02]. A strategy for automatic composition based on a goal description language (GDL4WSAC) is proposed by *Lin et. al.* [Lin 05].

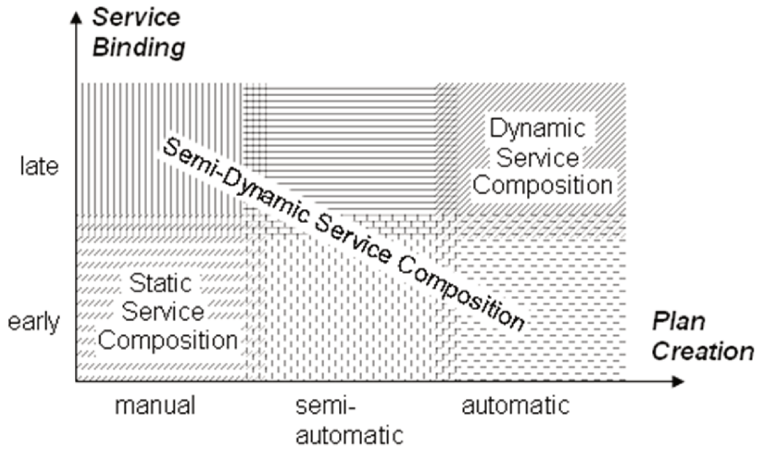

Figura 2.1: Classification of Service Composition Strategies

In the previous discussion regarding the implications for an actor who creates and provides composite services, it was argued that composition approaches applying late binding mechanisms are more adaptable to a changing environment, where third party providers are frequently leaving and joining. Furthermore it was argued that a high degree of automation during creation of a process model cuts down development costs and accelerates time-to-market, thus resulting in a higher flexibility of a composite service provider. In addition, quality aspects, such as reliability, have been considered. When combining the terms adaptiveness and flexibility to the more generic term dynamics, a coarse-grained and more business-oriented classification in static, semi-dynamic and dynamic service composition strategies can be made (see Figure 2.1). Taking into account the above mentioned attributes cost efficiency, time-to-market and reliability, it can be argued that a high degree of dynamics for service composition has positive effects on the providers' profitability.

On the other hand this does not inherently mean the more automation the better. The degree of dynamics applicable in a real world context is limited by many more factors, being trust (see Section 2.3.2) one of the most relevant. In addition, performance issues can also represent a problem, since interacting with a broker for service discovery

and matchmaking as well as applying sophisticated AI algorithms for automated plan generation may be time-consuming tasks.

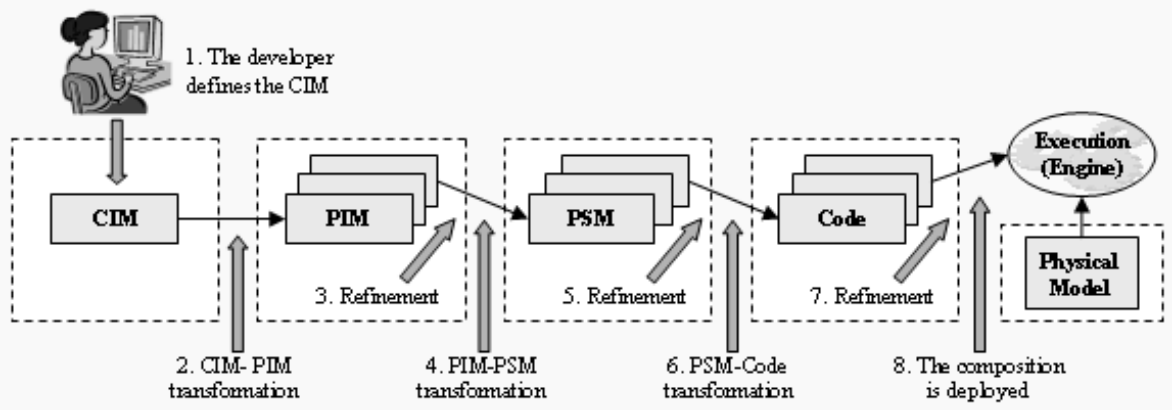

Figura 2.2: The Composition process following an MDA approach

2.3.2 Trust

Dini [Dini 05] states that "*functional optimization addresses very well the technological part of a problem, but does not pay enough attention to the intentions of the users and the structure and dynamics of social groups and user communities*". Applying dynamic service composition usually results in limited control over the way *how* a problem will be solved, respectively how a service will be composed. As a consequence it might become opaque with which partners the provider of a composite service and its customers collaborate and share data in the scope of a process. Trust is the keyword in this respect and the on-demand establishment of trustable relationships is still an unsolved problem.

2.3.3 Model Driven Approach

The Model Driven Architecture (MDA)[OMG 03] is a new approach proposed by the Object Management Group (OMG) to develop applications and write software specifications. It is based on three standards: the Unified Modeling Language (UML), the MetaObject Facility (MOF) and the Common Warehouse Metamodel (CWM). These standards should facilitate the design, description, storage and exchange of models.

The MDA approach separates the specification of the operation of a system from the details of the way that system uses the capabilities of its platform. It uses abstract models to specify all the logic of the application where concepts on languages or platform are irrelevant. Later, these models are used to create new models which express the requirements of the system in a specific platform. Note that *Platform independent* and *platform specific*

are not absolute concepts: what is specific to one system can be independent to another. The MDA specifies that the following models should be created during a development process [OMG 03]:

- *Computation Independent Model* (CIM): also called a domain model, focuses on the environment and requirements of the system. The details of the structure and processing are hidden;
- *Platform Independent Model* (PIM): provides a description of the system from a platform independent viewpoint, focusing only on the system functionalities;
- *Platform Specific Model* (PSM): describes the system combining the specifications in the PIM with the details regarding the platform where the system will run.

As the use of MDA implies an increase in the abstraction level during system development, the developer, starting from a more abstract level, can successively produce lower abstract models until in the end an executable model is reached. All models must be written in a formal model language, in order to enable the transformation between models to be computer aided. But note that it is not mandatory to use the same language in all models, i.e., it is also possible that a language transformation between different models of the same system becomes necessary. The great advantage in using MDA is the ability to transform a platform independent model (PIM) into a model capable of running into a great variety of technologies. MDA assumes that technology is very volatile, thus an automatic PIM-PSM transformation can save time and money by improving the efficiency of steps like implementation, integration, maintenance, testing and simulation. In an ideal world, the developer would simply submit the PIM to generators which would produce in the end executable code. But the reality is different and we are far away from this - in practice a lot of manual work must still be done.

In a recent work, ISO/IEC [ISO 05] defined the use of the UML for expressing system specifications in terms of the five viewpoints defined by the *Reference Model for Open Distributed Processing* (RM-ODP): enterprise, information, computational, engineering and technology. A relationship with RM-ODP viewpoints and MDA can be made as following: the CIM is provided by an enterprise viewpoint with relevant parts of an information viewpoint; the PIM is composed of the information and computational viewpoints; and finally, the PSM is related to the engineering viewpoint. The technology specification does not have an equivalent in the MDA. For each viewpoint there is an associated viewpoint language which can be used to express a specification of the system from that viewpoint, being the model/language transformations provided by the MOF.

In Figure 2.2 we see a composition process following the MDA approach. The process starts at the definition of CIM, usually a manual task (step 1). This definition must

include, among other things, the identification, specification and modeling of the composition. The UMM (UN/CEFACT Modeling Methodology) [CEFACT 03] is an example of methodology that could be used at this phase of the project. The first transformation takes place into CIM-PIM (step 2). The transformation between models can be complex and, in almost every case, parameters need to be set in the source model in order to drive the transformation. After the transformation, refinements should be performed in the PIM in order to go in the direction of the executable code (step 3). In order to stay aligned with the RM-ODP, the information and computational viewpoint languages must be used. Otherwise, BPMN (Business Process Modeling Notation), UML activity diagram or EDOC profiles are also typical languages used to design the PIM in a service composition context.

Next, a transformation from a PIM into a PSM takes place (step 4). This transformation plays a special role in MDA and a mapping language can be defined to automate it. In an ideal world, the same PIM could be transformed in several PSMs, for instance, a PSM-J2EE, PSM-CORBA, PSM-.NET, PSM-WS or any other. The PSM can be specified using the engineering viewpoint language defined by ISO/IEC or UML activity diagrams with extensions for a specific platform. As the previous transformation, some parameters can be set in the PIM to drive the transformation and, after it, the refinement of the PSM should be necessary (step 5). *Bézivin et al.* [Bézivin 04] describes a PIM (UML and EDOC) transformation to PSM (Java and Web Services), focusing on the static aspects of the mappings. *Patrascoiu* [Patrascoiu 04] presents the mapping from EDOC profiles to Web Services using a transformation language called YATL, also considering the dynamic aspects of a service composition. A framework proposal for service composition using the MDA approach applied to the e-Government area was presented by *Tizzo et. al.*[Tizzo 04].

Finally, the last transformation: PSM-code (step 6). The code is the composition described in some executable language (BPEL for example). A very detailed PSM can describe the whole service composition logic. So, the gap between PSM and code can be very small. But, as the previous transformations, fine adjustments can be necessary before actually running it (step 7). The code, added to the deployment description, completes the models that are necessary to run the composition (step 8). Analyzing the MDA approach in a reverse way, the automatic code generation would start in the PSM-code transformation. When it's possible to produce a full executable code from this transformation, the code would be hidden and in fact the PSM would be executed by a virtual machine. This process is analogous to the one that happens with a traditional compiler or interpreter. Going further, if we apply this automatization within the PIM-PSM transformation, a virtual machine could execute the PIM. The next section presents the challenges and guidelines in order to achieve this goal.

2.3.4 Using Semantics and MDA to Increase the Dynamism in Service Composition

According to the MDA (see previous section), the automation may take place in two different directions: from one model to another (model transformations) and inside a model (model refinement). During these steps, the abstraction level is gradually reduced (see Figure 2.3).

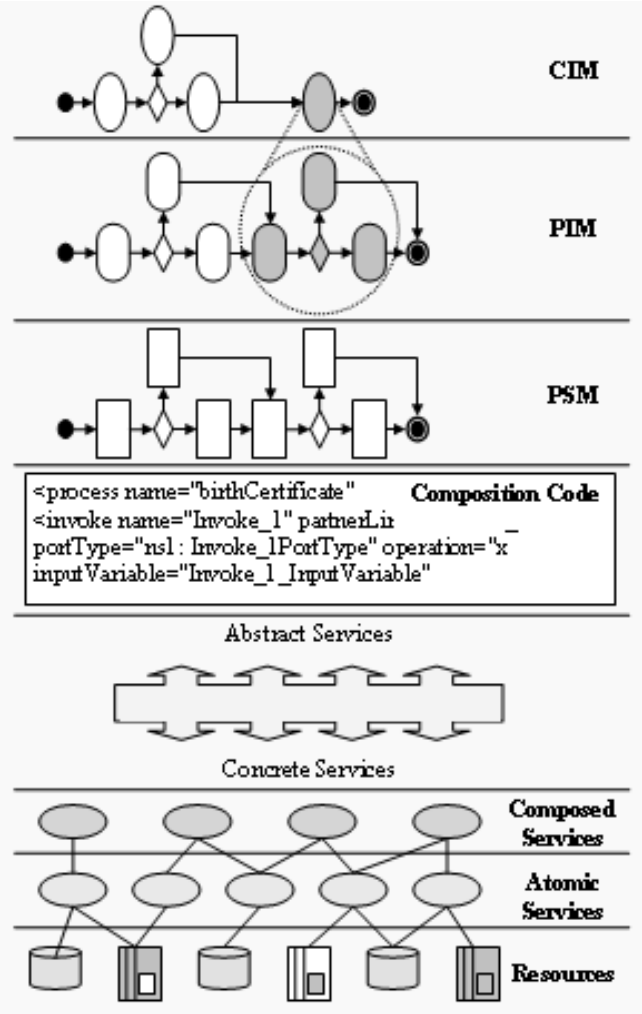

Figura 2.3: Viewpoints and Abstraction Levels

Note that the service compositions, which are described using abstract services, need to be bound to concrete services at a certain moment in time. The use of semantic descriptions and ontologies plays a special role in this stage. The composition described in a CIM is a description of a sequence of tasks and its data and control flow. At

this point, a task is an abstract service, i.e., it is only a service description and it may not have an implementation. The information used to describe a task is composed of four fundamental elements: signature, preconditions, post conditions and information invariants [Frankel 05]. Non-functional aspects can also be described. The activities in a CIM must then be detailed, process done during the CIM-PIM transformation and/or PIM refinement. The automatic (or semi-automatic) transformation and refinement must rely on semantic descriptions to provide machine-readable information about each task. Algorithms based on AI techniques [Rao 04] (such as Situation calculus, PDDL, Rule-based planning or Linear Logic) can use these descriptions to decompose CIM tasks or refine the PIM. As already mentioned in Section 2.2.1, one of the characteristics that can be found in dynamic compositions is the possibility of selecting the participant services. Theoretically, this can be done at the PIM, PSM, code or run-time (late-binding): when it will happen can be determined by a technology restriction or by a project decision: (1) the BPEL language does not allow service selection at run-time, forcing the developer to do it before; (2) on the other hand, when the process environment changes over time, the developer can decide to do the selection as later as possible.

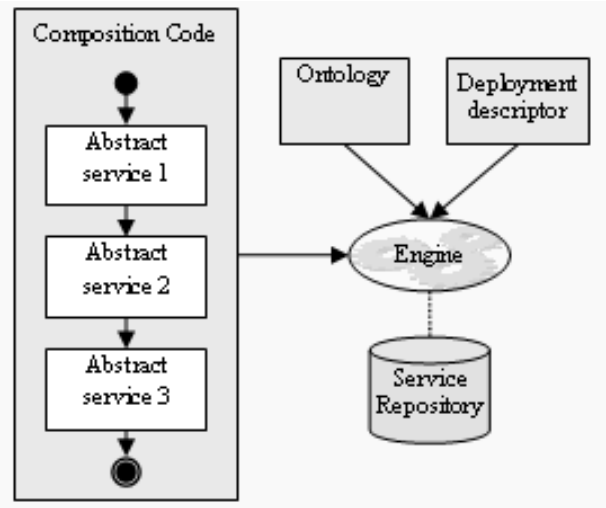

Figura 2.4: Composition with late (semantic-based) binding

In order to perform the selection of services and considering the late-binding scenario, the result of step 8 (Figure 2.2) would be a composition of Abstract Services, to be bound to Concrete Services at run-time. In Figure 2.4, this process is illustrated. An engine receives the composition description (i.e. the code), starts to run it and for each abstract service (described with semantic information), a service repository (or a service broker) is contacted in order to discover which service will be the responsible for executing that activity, always considering the associated ontology.

2.4 An Example

In order to apply the techniques presented in Section 2.3, we consider now an example in the e-Government context. The goal of the composition is to provide *birth certificates* to citizens through an e-Government portal. First of all, the portal checks the municipality where the citizen was born (*city of birth*). This is an essential step in order to correctly determine which services will participate in the composition. For different municipalities, even though the CIM is the same, different services should be selected and a different sequence of activities might be performed. That is also an important fact to justify the use of a dynamic strategy in this scenario. Furthermore, each one of the activities involved in the process can be performed by services located in different organizations. This would also create the necessity of considering aspects such as privacy, autonomy and security, omitted in the example for not being in the scope of this paper. An interesting strategy for handling these cross-border issues is the use of *Interaction Policies* [Santos 06].

Referring back to the strategies presented in Section 2.3, and considering a unique CIM as input, the information "*city of birth*"may be used in one of the following manners:

1. Considering that each activity of the CIM could be implemented in a different way for different cities, the "*city of birth*"is used to determine the result of the CIM-PIM transformation and refinement. That is, each city might have a different associated PIM;

2. If besides the same CIM, the PIM is also identical for all municipalities, the differences might occur in the PSM. It means that the activities are exactly the same for all cities, but each city may rely on a different technological platform;

3. When the CIM, PIM and PSM are identical, only one composition is built, deployed, and the "*city of birth*"is used as parameter for selecting the correct concrete services to perform each of the activities in a given execution instance.

Note how one single variable, depending on the strategy adopted and on the characteristics of the problem, may have a great impact on the result composite service.

Remember that in our example, the citizen requests the emission of a birth certificate at an e-Government portal. The steps of this process described in the CIM (Figure 2.5) are:

1. The citizen has to pay for the emission of the certificate;

2. If the payment succeeds, the process continues and the request is forwarded to a public servant (step 3). If not, the citizen must be notified about the problem;

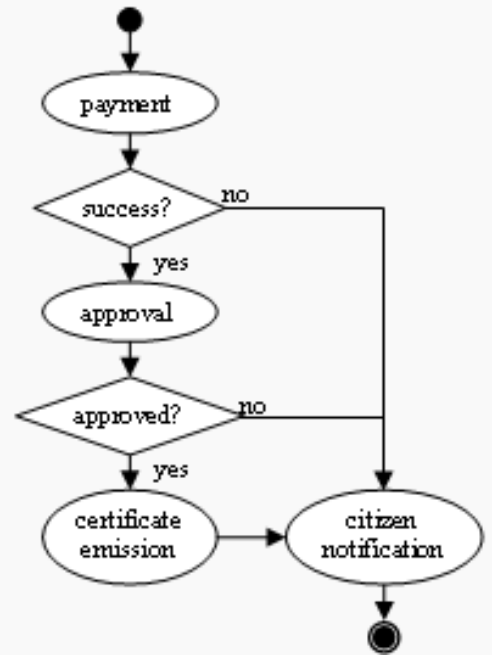

Figura 2.5: CIM (UML Activity Diagram)

3. A municipality servant must check the pending certificate emission request and validate them. In case of his approval, the request proceeds to step 4. If the servant, for any reason, denies the request, the citizen must be notified;

4. Finally, the certificate must be emitted and the citizen notified about the status of his request.

Other requirements of the process could also be modeled in the CIM (in fact they are necessary if we desire to automatically proceed to next step in the MDA process), but it is not in the scope of this article to further detail them.

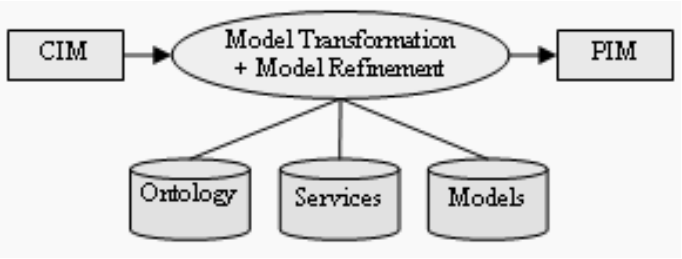

Figura 2.6: CIM-PIM transformation and refinement

As shown in Figure 2.2, the next steps in our strategy are then the transformation of the CIM into a PIM and the PIM refinement (Figure 2.6). We consider in our example

that each municipality may have its own PIM, and that the information "city of birth" will be important in all transformations and refinements and also in the execution phase (to correctly select and bind the abstract services to concrete implementations). Therefore, the PIM-PSM and PSM-Code transformation would be analogous to the CIM-PIM transformation illustrated in Figure 2.6.

In order to implement these transformations we consider the associated ontology, the services registry and also a database with previous existing activity models. This model database is particularly important because it contains, at each level, a description of the possible activity mappings to the immediate lower abstraction level in the composition (see Figure 2.3). Note that if we do not rely on this strategy, we must them adopt some AI technique (Section 2.2.2) to perform the abstraction level change.

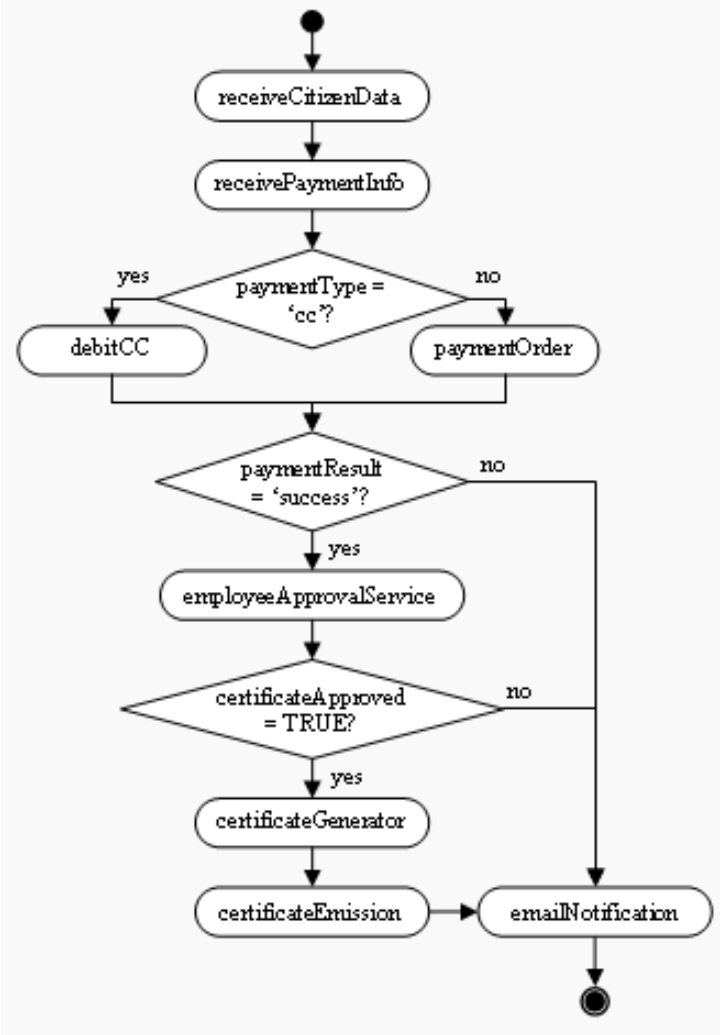

Figura 2.7: PIM (UML Activity Diagram)

In Figure 2.7 we see the UML Activity Diagram which is part of the PIM for a given municipality. Differently from the CIM, the PIM describes in more details the sequence of activities that should be performed to successfully complete the process:

1. An activity called "*receiveCitizenData*"is responsible for getting the citizen request and information;

2. The payment information is sent to the activity "*receivePaymentInfo*", that checks whether the citizen will pay by Credit Card or by Payment Order, and then calls the associated activity to actually perform the payment ("*debitCC*"or "*paymentOrder*");

3. If the payment was successful, the process proceeds to step 4. If not, an e-mail notification activity must inform the citizen the reason of the failure;

4. A public servant participates in the activity "*employeeApprovalService*"and checks if everything is fine with the request, validating it. In case of approval, the process proceeds to step 5. In case of failure, the citizen must be informed by the activity "*emailNotification*";

5. The certificate is generated electronically ("*certificateGenerator*") and then printed and stamped ("*certificateEmission*"), being finally the citizen notified of the success of his request and informed about the procedures to pick up the document.

Next follows a (possible automatic) transformation from the PIM into a PSM which can be, for instance, a BPEL specific one. Finally a PSM-Code transformation takes place, and the resulting composition code (made of abstract services) is deployed. At run-time, the binding to the concrete services is performed like Figure 2.4 illustrated.

2.5 Conclusions and Final Remarks

The proposal of effective solutions to enable interoperability among heterogeneous and inter-organizational systems remains a key issue in the development of new Web-based applications, especially in the e-Business and e-Government contexts. The so called *Service-Oriented Architecture*, and more specifically its *Composition* layer, appears as a promising solution to deal with these demands. Even though most part of the academia and industry considers that *Dynamic* (and/or Automatic) Service Composition is the next stage to be reached, there is still a lot of misuse and misunderstanding regarding these concepts. In this paper, our first contribution is the proposal of a clear classification of different composition strategies with regards to their levels of dynamism and automatization.

Besides this classification, another important contribution of our paper is the introduction of a generic model driven approach for composing services, which is further specialized to illustrate the techniques that can be applied to enable fully dynamic service compositions. Aspects like dynamic selection of services and the use of AI techniques to automatically generate the execution plans are also discussed. An example service in the e-Government context is presented to illustrate the use of the proposed techniques. As already mentioned, the goal of this paper was not to focus on any specific technology or implementation solution, but rather to critically analyze the challenges and possible alternatives regarding service compositions in general. An interesting extension to this work would be to apply the proposed strategies in some technology specific scenario and evaluate its potentials and limitations considering different domains and platform characteristics.

We believe that much research is still necessary and also that a fully dynamic composition may be still far from becoming a reality, except for specific and well-defined application domains. Nevertheless, the research community, having recognized the potential of the evolving *Semantic Web*, has spawned several activities in the direction of *Semantic Web Services*. With languages like the *Web Ontology Language* (OWL), machine-understandable Web service descriptions can be created and shared. Generic service ontologies, such as *OWL-S* [Coalition 04] and *WSMO* [Feier 05], in combination with appropriate rule languages and new AI techniques lay the foundations for semantically describing the functionality and the behavior of services, and keep the road open for a future of dynamic service compositions.

2.6 Acknowledgments

The authors would like to thank the Brazilian Agencies CAPES and CNPq for financial support. Furthermore this work is also supported by the European Commission through the IST-1-002104-STP SATINE (Semantic-based Interoperability Infrastructure for Integrating Web Service Platforms to Peer-to-Peer-Networks) project.

Capítulo 3

A Semantic-enabled Middleware for Citizen-centric E-Government Services

3.1 Introduction

The demands for the creation of mechanisms to increase the transparency of the public administration processes have dramatically increased over the recent years [Watson 01]. It represents a requirement for every government identified since the origins of the modern democracies: Thomas Jefferson once wrote that "*whenever the people are well-informed, they can be trusted with their own government*"[Jefferson 89]. According to a UN (United Nations) recent report the "*strategic and meaningful application of Information and Communication Technologies for the purpose of improving the efficiency, transparency, accountability and accessibility of government is possible if the ultimate objective of e-government is to promote social inclusion*" [Ahmed 06].

In order to transform these demands into reality some important technological issues must be handled. The first one is the implicit heterogeneity of the information systems spread throughout the government entities and their partners and therefore a challenge is to find an efficient way to enable full interoperability with the existing systems. Concerns like privacy, trust, autonomy and identity management are also of equal (or sometimes greater) importance. Many current solutions are moving towards adopting SOA-based (Service-Oriented Architecture) approaches, what itself introduces new and probably bigger challenges: questions like how to successfully describe the services, how to compose them as dynamically as possible and also how to mediate (or not) their interactions were not fully answered by traditional SOA-based specifications/implementations and their most famous counterparts, the *Web Services* related technologies.

The first important contribution of this paper is the proposal of a semantically enriched middleware for e-Government services, the ***CoGPlat*** (**C**itizen-**o**riented e-**G**overnment **Plat**form), which provides a set of functionalities that simplify the development and operation of citizen-centric applications. This paper also contributes presenting strategies to dynamically compose Semantic Web Services and techniques to increase the public administration processes' transparency. The introduction of a set of policies to govern the service compositions (providing different levels of autonomy, privacy, traceability and identity management) is another important contribution. The remainder of this paper is organized as follows: Section 3.2 presents related concepts, technologies and a literature review; Section 3.3 introduces our middleware architecture and discusses the dynamic composition strategies; in Section 3.4 implementation aspects are presented and the use of the platform is illustrated through example scenarios; and Section 3.5 presents the conclusions and final remarks.

3.2 Concepts, Technologies and Literature Review

Our proposal relies on concepts, technologies and strategies oriented towards providing interoperability and dynamism for web-based services and applications in the e-Government domain. Next we discuss these underlying concepts and technologies and present also a literature review on related work.

3.2.1 Interoperability and Services

The **Interoperability**, defined as "*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*" [IEEE 90], is nowadays a fundamental requirement in the context of distributed and dynamic applications. In order to achieve higher levels of interoperability, many systems are being implemented following the Service-Oriented Architecture (SOA) approach and its most famous manifestation, the Web Services technologies. The SOA may be described as a component-based model which interrelates different functional units (or *services*) through well-defined interfaces that should be neutral, platform- and language-independent. Services running over heterogeneous systems may then interact and be used as building blocks for new applications [Papazoglou 03, IBM 05]. A composite service, the basis for the construction of applications in the SOA world, can be regarded as a combination of activities (which may be either atomic or composite services), invoked in a predefined order and executed as a whole [Fluegge 06]. Two different execution models for composite services are commonly defined: *orchestration* and *choreography*. In an *orchestration* all interactions that are part of the process are described and then executed by an orchestra-

tion engine, which has control of the overall composition. In contrast, a *choreography* is more collaborative and less centralized, with only the public message exchanges considered relevant [Peltz 03, Ross-Talbot 05].

The specification, enactment and management of composite e-services are issues studied in the *eFlow* project [Casati 01]. Both adaptative characteristics and also strategies on how to implement dynamic service process modifications are discussed and classified into two categories: *ad hoc changes*, when modifications are applied to a single running process instance; and *bulk changes*, when modifications are collectively applied to a subset (or to all) the running instances of a service process. A broker-based architecture (*QBroker*) is proposed in [Yu 07] to facilitate the selection of services based on Quality-of-Service (QoS) criteria. The approach uses both a combinatorial and a graph model and defines multiple QoS criteria, taking global constraints into account. The *VieDAME* system [Moser 08] proposes an approach to monitor composite services (BPEL processes) according to QoS attributes. It also introduces partner service replacement strategies that can be applied automatically at runtime without interrupting the system. In [Schäfer 08] techniques to perform advanced compensations of web services transactions are introduced. The strategy is based on forward recovery principles and on contracts. It defines, like our work (see later), the concept of *abstract service*, but with a different perspective: they work as management units that invoke concrete services, replace them with compatible ones in case of failures and also process externally triggered compensations. The dynamic evolution of business protocols is the subject of [Ryu 08], where a method to automatically classify conversations based on the impact of the protocol change is introduced. In [Fluegge 06] a discussion about the criteria that can be used to identify the levels of dynamism and automatization in service compositions is held. Considering a Model Driven Architecture, the authors propose a strategy using different techniques to compose services more dynamically.

3.2.2 The Semantic Web

It has been widely identified that both the Web and the traditional Web Services lack support to describe the semantics of data. To overcome this weakness, the **Semantic Web** [Berners-Lee 01] proposes a scenario where data becomes meaningful to machines and can be automatically processed and understood. In it, ontologies play a fundamental role in the definition of the concepts that are used to annotate data [Medjahed 04]. An ontology can be defined as a *formal*, *explicit* specification of a *shared conceptualization* [Gruber 93, Borst 97], where *conceptualization* refers to an abstraction of a domain that identifies its relevant concepts, and *shared* implies that these concepts are equally defined and understood (consensus) by the participants of that domain.

The RDF (*Resource Description Framework*) and RDFS (*RDF-Schema*) specifications represent the first W3C (World Wide Web Consortium) initiatives to model meta-data related to Web resources [W3C 04b, W3C 04c]. A successful example of an RDFS compatible ontology is the YAGO project [Suchanek 07], that contains 1 million entities and 5 million facts which were automatically extracted from *Wikipedia* and *WordNet* [Miller 95] using a combination of rule-based and heuristic methods. The current W3C standard for specifying ontologies on the Web is the OWL (*Web Ontology Language*), which extended RDF, augmenting its vocabulary with the inclusion, for instance, of new class relationships [W3C 04a].

The possibility of applying a similar strategy to describe services semantically, opening the road towards their automatic discovery, invocation and composition, motivated the proposal of different approaches for the definition of what was baptized as the **Semantic Web Services** (SWS). OWL-S (*Semantic Markup for Web Services*), the proposal adopted in our work, combines a set of inter-related OWL ontologies that define terms used in service-oriented applications [Coalition 04]. An OWL-S semantic description is organized into four parts:

1. *Process Model*: describes how a service performs its tasks. Includes information about inputs, outputs, pre- and post-conditions. The processes may be composed, atomic or simple (abstract);

2. *Profile*: provides a more generic description of the Web Service to be published and shared. The profiles may include both functional (inputs, outputs, pre- and post-conditions) and non-functional (name, textual description, category and additional parameters) properties. The functional properties are derived from the process model, but it is not mandatory to include all functional properties from a process model into a profile;

3. *Grounding*: describes how a service is invoked, detailing how the atomic process from the process model is mapped into concrete message exchange protocols;

4. *Service*: encapsulates the previous three parts into a unit that can be published and invoked.

In addition to OWL-S, two other proposals are relevant in the Semantic Web Services domain: the WSMO (*Web Services Modeling Ontology*) [W3C 05]) and the SAWSDL (*Semantic Annotations for WSDL and XML Schema*) [W3C 07], recently adopted as a W3C recommendation.

Despite being a relatively new area of study, it is already possible to find a considerable amount of research around the Semantic Web Services. The markup and automated

reasoning technology to describe, simulate, test, and verify compositions of Web services are discussed in [Narayanan 02]. The authors define the semantics for a relevant subset of OWL-S in terms of a first-order logical language (*Situation Calculus*). With the semantics in hand, service descriptions are encoded in a Petri Net formalism. The implemented system is able to read in OWL-S service descriptions and perform simulation, enactment and analysis. The use of advanced workflow and activity concepts in the composition of Web services is the proposal of [Fileto 03]. The approach is called POESIA (*Processes for Open-Ended Systems for Information Analysis*), an open environment for developing Web applications using metadata and ontologies to describe data processing patterns developed by domain experts. It supports Web service composition using domain ontologies with multiple dimensions (e.g., space, time, and object description). In [Küster 07], a strategy to integrate the tasks of automated service discovery, matchmaking and composition is presented, focusing on services with multiple connected effects. A derivation mechanism that allows the (semi-)automatic annotation of operation parameters starting from already annotated ones within service compositions is discussed in [Belhajjame 08]. The mediation of SWS compositions involving heterogeneous data is handled using context information in the strategy proposed in [Mrissa 07]. An end-to-end model-driven approach to design and develop WSMO-based semantic Web services is investigated in [Brambilla 07].

3.2.3 E-Government

The **electronic Government** (e-Government) domain includes the "*set of all processes which serve decision-making and services in politics, government and administration and which use information and communication technologies*"[KBSt 06]. Recently a new approach to e-Government is gaining momentum: the citizen-centric government, where citizens and businesses are considered customers of the public administration, so that their needs come first, rather than bureaucracy or other imperatives inside the government machine [GOV3 06, Lee 05, Marchionini 03]. In this context, usually a government-wide service-oriented architecture is applied to develop a single place that offers access to all government informational and transactional services.

Several research efforts in the e-Government domain can be found in the literature. For instance, a system which automatically generates Web services customized to citizens' needs and also to government laws and regulations is presented in [Medjahed 05]. It proposes three levels of service customization: the *Citizen* level, the *Service* level and the *User interface* level. A *metadata ontology*, used to describe e-Government services and operations, is also introduced. An approach for the semi-automated design of data flows between Web Services that are semantically described using different ontologies and data representations is introduced in [Barnickel 06], including a rule-based mechanism for user-

transparent mediation between ontologies spanning multiple application domains. The potential benefits of adopting Semantic Web Services in the e-Government domain are discussed in [Gugliotta 08] by analyzing motivations, requirements and expected results and illustrated through a reusable framework based on the IRS-III broker [Cabral 06].

3.2.4 Privacy

Being also of special interest to the e-Government domain, several efforts in the industry are seeking to better protect sensitive information available online in the form of privacy policies. These policies specify who can use a service and under which conditions, how information should be provided to the service and how the provided information will be used [Kagal 04]. They can be seen as a way to dynamically constrain and regulate a system's behavior without changing code or requiring the cooperation of the components being governed [Uszok 04]. Multiple approaches for policy specification which range from formal to rule-based languages have been proposed by researches. Two of those languages that have emerged are P3P (Platform for Privacy Preferences) and EPAL (Enterprise Privacy Authorization Language). The major problem with both is that they lack unambiguous semantics and significant expressive power [Antón 07]. Therefore the proposal of an efficient language for specifying privacy policies remains an open research issue.

Besides providing effective ways to specify the policies, it is fundamental to offer mechanisms to enforce privacy during the complete life cycle of the applications and also to allow post-execution auditing. In [Antón 07], for instance, a framework to support the privacy policy life cycle is discussed. Authorization and privacy requirements of Semantic Web Services are the subject of [Kagal 04], where an extension of OWL-S that supports policy annotations is presented. The KAoS project [Uszok 04] provides also a policy representation language based on OWL. It defines policy constructs that distinguish between positive and negative authorizations and obligations. A solution to enable privacy-preserving secure semantic access control and sharing of data among heterogeneous databases without having to share metadata is presented in [Mitra 06]. It uses encrypted ontologies, mapping tables, conversion functions, role hierarchies and queries. The encrypted results of queries are sent directly from the responding system to the requesting system, bypassing the mediator to further improve the security of the system. According to the authors, one of its distinguishing features is that it requires very little changes to underlying databases.

3.3 CoGPlat: The Citizen-oriented e-Gov Platform

The requirements imposed to a middleware that aims to support citizen-centric applications and electronic governance go beyond those found in the e-Business world, especially with respect to flexibility, transparency and interoperability demands. According to [Davies 07] these requirements "*reflect the collective needs and ambitions of our society, expressed through a combination of legislation and public opinion*".

The contributions introduced in this paper are in the context of the ***CoGPlat*** (***C****itizen-**o**riented e-**G**overnment **Plat**form*) project [Santos 05]. Its main goal is to provide a middleware that supports the interaction and collaboration of *entities* (governmental offices, private companies, non-governmental organizations and service providers) and *citizens* in different public administration scenarios, ranging from the electronic delivery of integrated services to the support for citizen participation in government decisions. *CoGPlat* is a service-oriented middleware, not an end-user application, and provides a set of generic services and facilities that can be used by different e-Gov applications and tools.

3.3.1 Middleware Infrastructure

The *CoGPlat* middleware infrastructure is composed of the following elements (Figure 3.1):

- The **Service Bus** is an interface between the middleware and the applications. All services provided by the platform core facilities are exposed to the applications through this interface;
- Four **core** facilities:
    1. The **Transparent Services Center**, responsible for dynamically building the service compositions according to the application requests;
    2. The **Metamodel Management Center**, which offers services and tools to manage the models, metamodels and ontologies used in the description of services, compositions, processes and entities;
    3. The **E-Governance and E-Democracy Center**, which delivers generic services that aim to increase citizen participation in the public administration and to facilitate the decision-making processes;
    4. The **Traceability and Auditing Center**, which offers services and tools to monitor running processes and also to audit processes that have already been concluded.

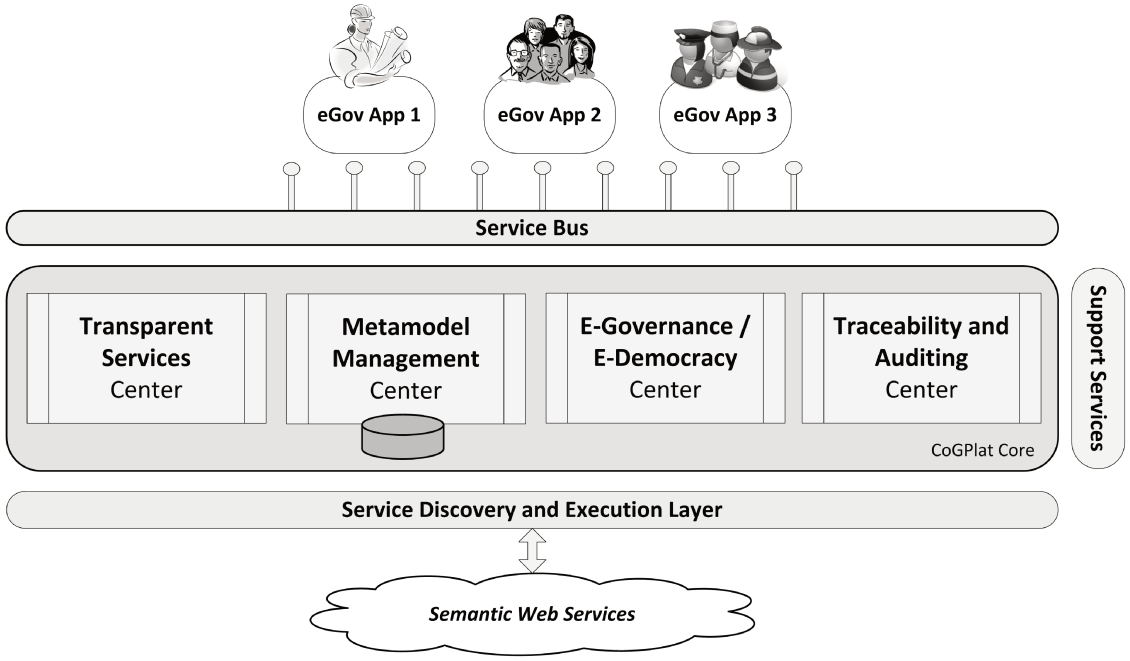

Figura 3.1: CoGPlat - General Infrastructure

- The **Service Discovery and Execution Layer**, responsible for selecting the services that will participate in the compositions and also for interacting with these services during execution time;
- A set of **Support Services** which provides security, persistence, reliable messaging and transaction support to the processes running over the platform.

The remainder of this section will further detail the middleware infrastructure, with special attention given to the four core facilities.

3.3.2 Collaborations Regulated through Interaction Policies

When composing e-Government services, the autonomy of the entities, the privacy and security of the data, the traceability of the processes and efficient identity management strategies are always important topics to be considered. In order to enable compositions that consider these demands, *CoGPlat* implements a set of *Interaction Policies*, first proposed and successfully applied in an e-Business scenario [Santos 06] and now extended to match these new e-Government application requirements.

The interaction policies are applied to regulate the *entity collaborations* that take place over the middleware infrastructure. In *CoGPlat*, an *entity* is any governmental office,

private company, non-governmental organization or independent service provider which acts either as a service provider or consumer. We consider that two entities *collaborate* when both participate in the same composite service and there is at least one message exchanged between them through the middleware facilities. These *collaborations* can be classified into three (well-known) groups:

1. **G2G** (Government-to-Government) collaborations: processes involving two governmental entities;
2. **G2B** (Government-to-Business) collaborations: processes involving private and/or non-profit organizations and governmental entities;
3. **C2G** (Citizen-to-Government) interactions: processes involving citizens and governmental entities. Though citizens are not explicitly considered entities, the interactions where they participate are of equal (or even greater) importance and must also be regulated by the policies.

The active policies are determinant in the dynamic construction of the composition flow and also during its execution. They are organized into the following categories:

- **Entity Autonomy Policies** (*CoGPlat x Entity*): determine the level of control the platform and the applications running over it may have over the internal operations of a composite service;
- **Data Privacy Policies** (*Entity x Entity*): determine the collaboration levels on the interactions between two entities that participate on the same composite service instance and also establishes the privacy of information exchanged between these two partners;
- **Service Traceability Policies** (*CoGPlat x Service*): determine to what extent the internal operations of a service may/must be monitored/audited by the middleware;
- **Identity Management Policies** (*Citizen/Entity x CoGPlat*): establish where to provide mechanisms like anonymousness, identity theft protection and non-repudiation for citizens and entities (when acting as service consumers).

Figure 3.2 presents an excerpt of *CoGPlat*'s ontology metamodel describing the relationships among policies, compositions and services. Based on the metamodel, it is possible to infer that:

1. **Identity Management Policies** and **Traceability Policies** are applied to **Compositions**;

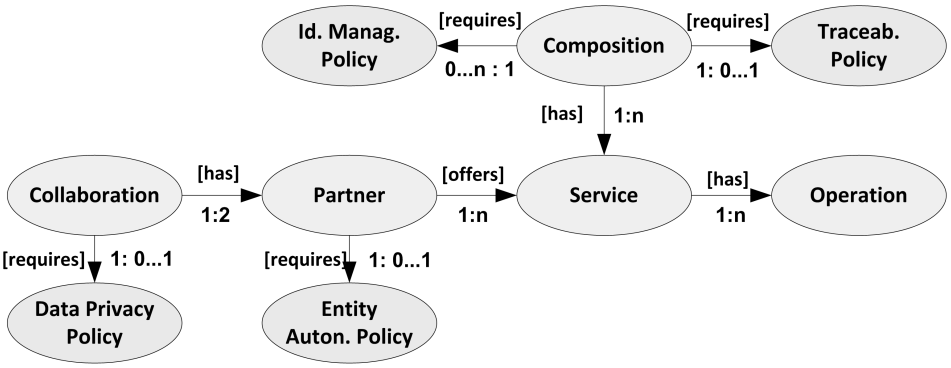

Figura 3.2: Compositions, Services and Policies in the Ontology

2. **Privacy Policies** are applied to **Collaborations**, each of these involving 2 **Partners**;

3. **Autonomy Policies** are applied to **Partners**. A *partner* represents an entity that participates in a composite service.

Note that a successful composition requires not only a combination of the service functionalities, but also of all active policies.

The *Entity Autonomy Policies* influence the composition strategy adopted. When for instance one participant entity has control/supremacy over others, an orchestration approach seems to be more appropriate. On the other hand, when there is only collaboration among the entities (no administrative links or hierarchy and fully decentralized control), choreography might be the most appropriate choice.

It is also possible to mediate how data will be exchanged when two entities collaborate through the use of *CoGPlat*'s *Data Privacy Policies* (*Total Trust*, *Moderate Trust* and *No Trust*)). They determine the level of trust between the two partners, and therefore whether the internal process information should be or not shared between them. Security requirements (e.g. encryption, certification, specific protocols etc) are handled by *CoGPlat*'s support facilities (see Section 3.3.1).

Another important requirement of many e-Government applications is the ability to monitor and audit the execution of public processes. A straightforward example is a situation where a citizen wants to know details about the status of a given service he requested. This, in general, represents a significant increase in the transparency of the public processes. On the other hand, there are some scenarios where a limitation on this ability is necessary and/or mandatory. The *Service Traceability Policies* intend to regulate (and therefore guarantee) the desired traceability levels for a given service (see more in Section 3.3.6).

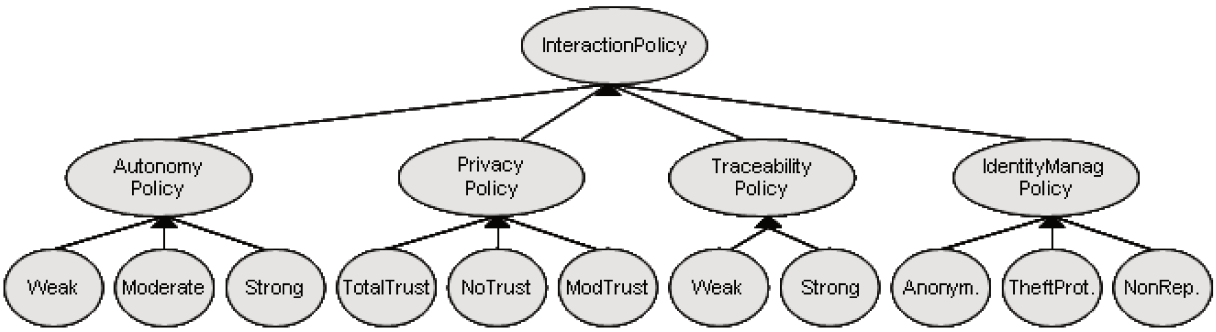

Figura 3.3: Interaction Policies: class hierarchy in the Ontology

Last but not least, e-Government applications should also consider the correct management of the identities of actors (especially citizens). There are scenarios (e.g. e-Democracy applications) where anonymity is desired. In others, an actor should not be able to negate an action he performed. There are also situations where a service provider must guarantee, usually according to legal requirements, that the actor identity and private data is protected and cannot be taken by another actor or external agent. The *Identity Management Policies* are used to enforce these requirements.

The Interaction Policies are described as classes in *CoGPlat*'s metadata ontology. Figure 3.3 illustrates this definition. Note that all policies derive from the *InteractionPolicy* Class, being the four categories (*AutonomyPolicy*, *PrivacyPolicy*, *TraceabilityPolicy* and *IdentityManagementPolicy*) we introduced previously just below it in the hierarchy.

3.3.3 The Transparent Services Center

The *Transparent Services Center* (TSC) is the facility in *CoGPlat* responsible for composing Web services as transparently and dynamically as possible. The composite **transparent services** have the following characteristics:

1. The service's client application needs no knowledge about the internals of a composite service - it knows only its semantic description. This introduces dynamism and a higher abstraction level when compared with what happens with the traditional Web services composition approach where the creation of composite services must still be handled by the application;

2. Even though the application is not aware of the internal details of the composition, it should be possible for it to monitor the processes execution status also transparently.

Inside the middleware, the processes are created based on abstract compositions that were previously defined by a designer with the help of a domain expert. A composition is

considered *abstract* when its activities are not bound to concrete services during design time. These activities represent service classes that are described using criteria such as **I**nputs, **O**utputs, **P**re-conditions and **E**ffects (IOPE's). The binding with concrete services is done at execution time, as we will show in Section 3.3.7.

The *Transparent Services Center* is also responsible for managing the process executions, governed by the active interaction policies. Its internal infrastructure contains the following elements (see Figure 3.4 for the corresponding class diagram):

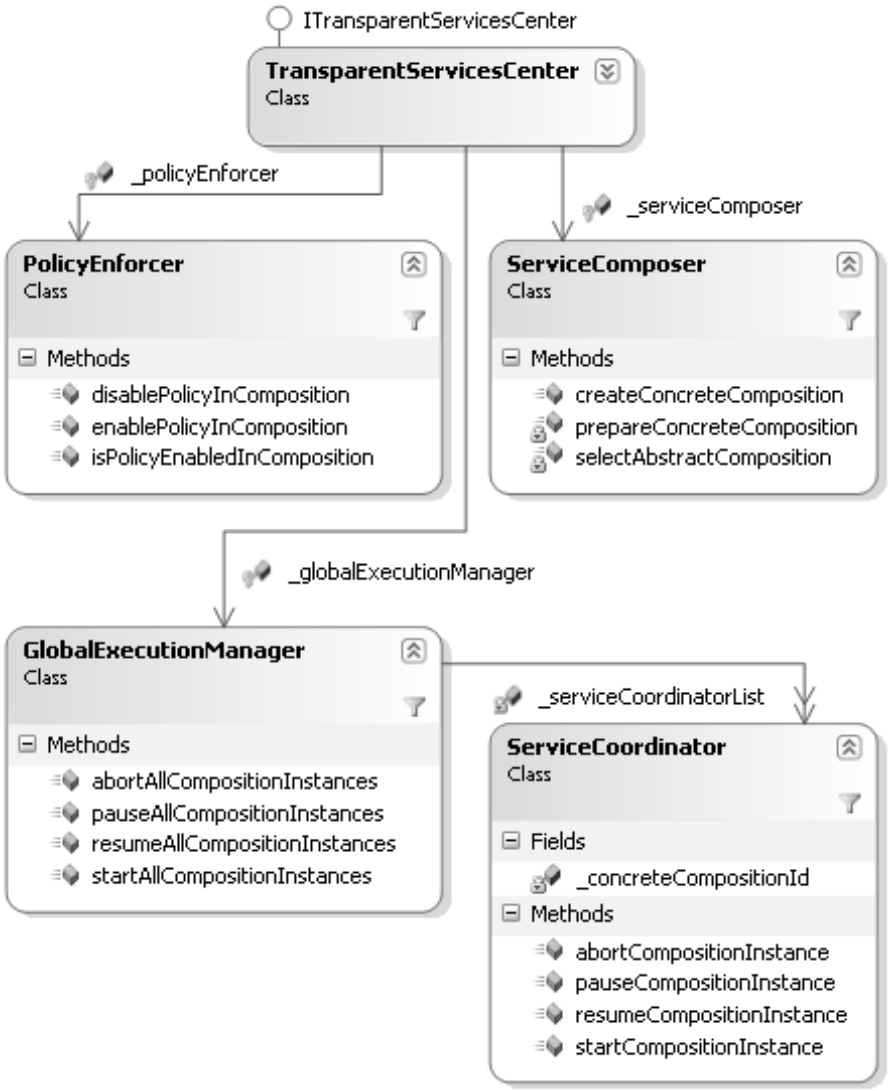

Figura 3.4: Class Diagram: The Transparent Services Center Infrastructure

- **Service Composer**: provides services to transform abstract compositions into concrete composite service instances;

- **Global Execution Manager**: interacts with all service coordinators and has a global view and control over all running processes;
- **Service Coordinator**: controls the execution of one composite service instance;
- **Policy Enforcer**: verifies the correct application of the active interaction policies among the running composition instances and also offers a mechanism to enable or disable policies.

3.3.4 The Metamodel Management Center

The *Metamodel Management Center* (MMC) offers services and tools to manage the models, metamodels and ontologies used in the description of services, compositions, processes and entities. It is internally organized as follows (see Figure 3.5):

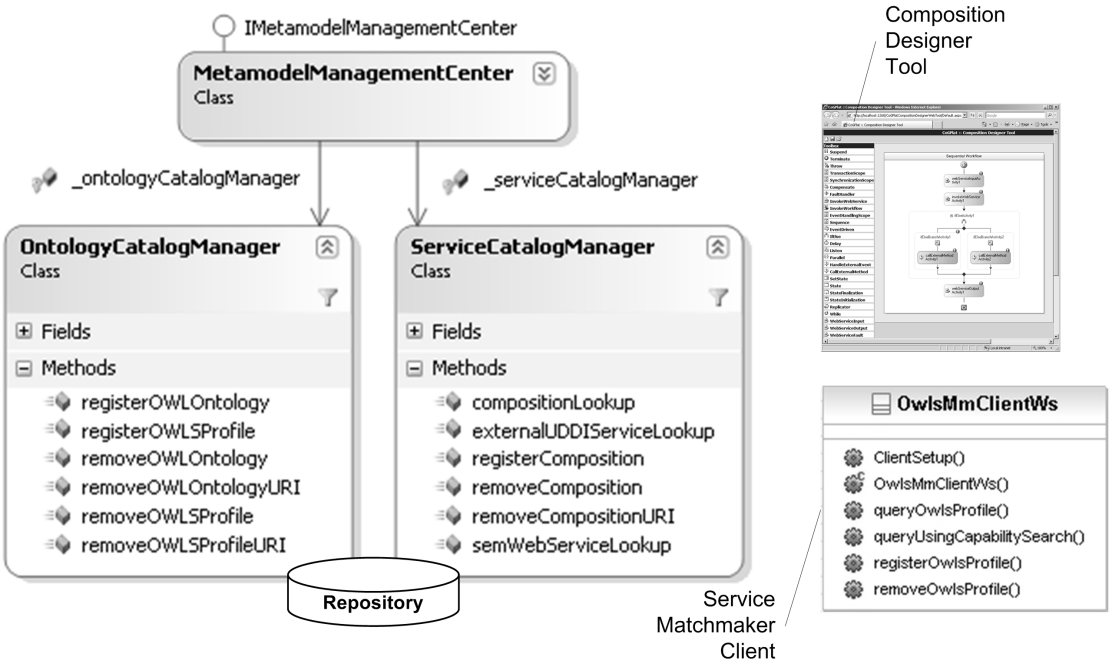

Figura 3.5: The Metamodel Management Center

- **Ontology Catalog Manager**: provides services to register and unregister ontologies and service profiles from the middleware catalog (both in the service matchmaker knowledge base and in the repository);
- **Service Catalog Manager**: provides services to search, register and unregister compositions and services from the middleware catalog;

- **Composition Designer Tool**: web-based tool to create and edit abstract and concrete compositions that are stored in the middleware catalog and then delivered to the applications;

- **Service Matchmaker**: provides a semantic match between application service requests and the compositions and services registered in the middleware;

- **Repository**: provides persistence to the catalog (standard relational database).

3.3.5 The E-Governance and E-Democracy Center

The *E-Governance and E-Democracy Center* (EEC) provides generic services that can be used as part of the application compositions with the aim to increase citizen participation in the public administration and to facilitate and legitimate the decision-making processes. It provides the following service categories (see Figure 3.6 for the corresponding class diagram):

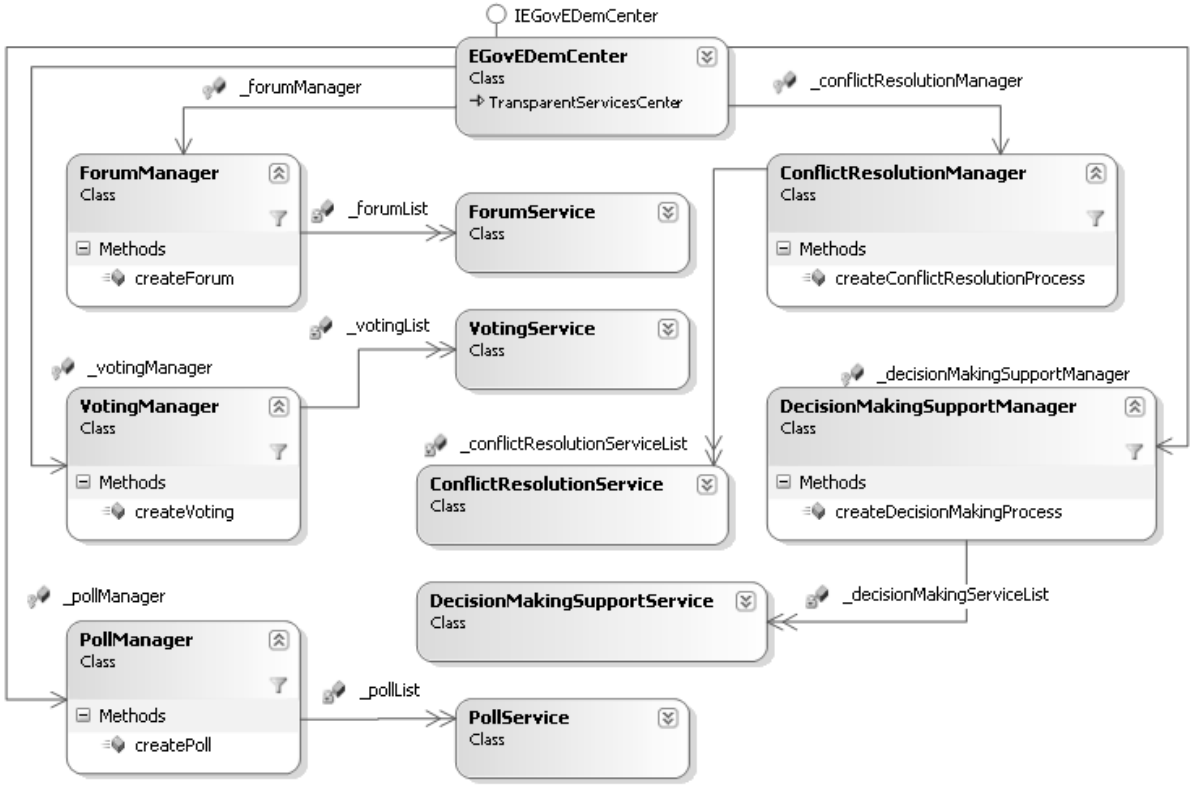

Figura 3.6: Class Diagram: E-Governance and E-Democracy Center

- **Forum**: service that allows the citizen participation in a (mediated or free) discussion about some pre-defined topic;
- **Poll**: service to gather public opinion statistics;
- **Voting**: service that provides an abstract definition of an e-voting tool (must be further specialized according to the local legal requirements);
- **Decision Making Support**: service that, using some pre-defined criteria (e.g. poll results, legislation etc), helps the public administrator in determining the political legitimacy of a decision;
- **Conflict Resolution**: service that tries to promote a friendly interaction between entity administrators when a conflict exists and needs to be solved before continuing the execution of a given process (e.g. interaction policy conflicts, legislation incompatibilities, conflicting poll results).

Each of these services is defined as an abstract composition, using the same approach specified for the *Transparent Services* - see in the class diagram that the *EGovEDemCenter* class inherits from the *TransparentServicesCenter* class. When a request for a new service arrives, the concrete service is produced using the corresponding abstract definition following the same strategy discussed in Section 3.3.7. In order to support multiple service instances and to facilitate the coordination of these instances, each service category has an associated manager (*ForumManager*, *VotingManager*, *PollManager*, *ConflictResolutionManager* and *DecisionMakingSupportManager*).

3.3.6 The Traceability and Auditing Center

The *Traceability and Auditing Center* (TAC) implements the necessary mechanisms to guarantee that the active traceability policies are respected during the execution of the composite services. Its internal infrastructure is composed of the following elements (see Figure 3.7 for the corresponding class diagram):

- The **Standard Tracking Service** collects all data from the generated tracking events, according to the active policy, and saves it into a repository;
- The **Real-time Tracking Service** collects only representative data from the tracking events, also according to the active policy, but provides it in real-time to the platform client applications instead of saving it to a repository;
- The **Auditing Service** offers mechanisms to access tracking data of finished processes;

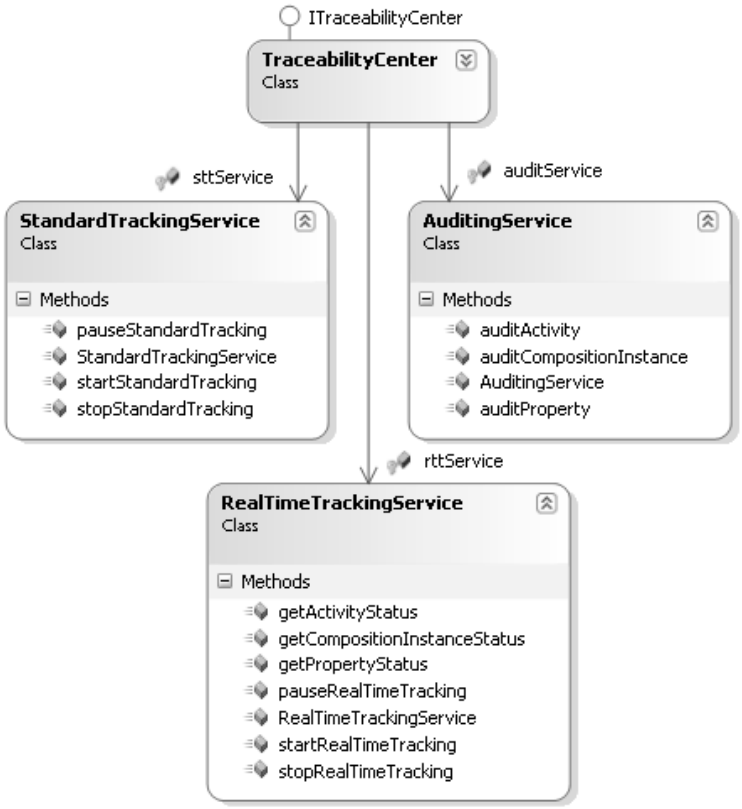

Figura 3.7: Class Diagram: Traceability and Auditing Center

- The **Tracking Data Repository** stores the information generated by the tracking service events.

An abstract composition and individual abstract activities can be semantically annotated with the following **Service Traceability** policies:

- **Strong-traceability**: all events both at composition and at activity and service level are monitored. Customized traceability requirements can also be specified, for instance to monitor the behavior of an specific property or value throughout the instance execution;
- **Weak-traceability**: only composition level events are monitored. No customizations are allowed;
- **Zero-traceability**: There is no traceability at all. Indeed, if this policy is specified, there must be no trace (neither during nor after) of a given composition instance.

For more details on the *Traceability and Auditing Center* please refer to [Santos 08] (Chapter 4).

3.3.7 Service Compositions in CoGPlat

Given the quantity and diversity of public administration agencies, integration of services delivered by these agencies becomes a challenging task. The service composition process described next tries to shift to the middleware level the solution for the associated interoperability and integration problems. It includes two main processes: the *definition of abstract compositions* and the concrete creation of the *transparent services*. The abstract compositions describe the generic public processes that may run over the platform combining services from different agencies. No concrete binding is necessary at design time - actually only service classes, semantically annotated with IOPE's and Policies are included in the abstract composition flow.

The abstract compositions repository may be maintained/updated by a process designer / domain expert (or also by an application on their behalf) through special *CoGPlat* administrative services offered by the *Metamodel Management Center*. This process has the following steps (see Figure 3.8):

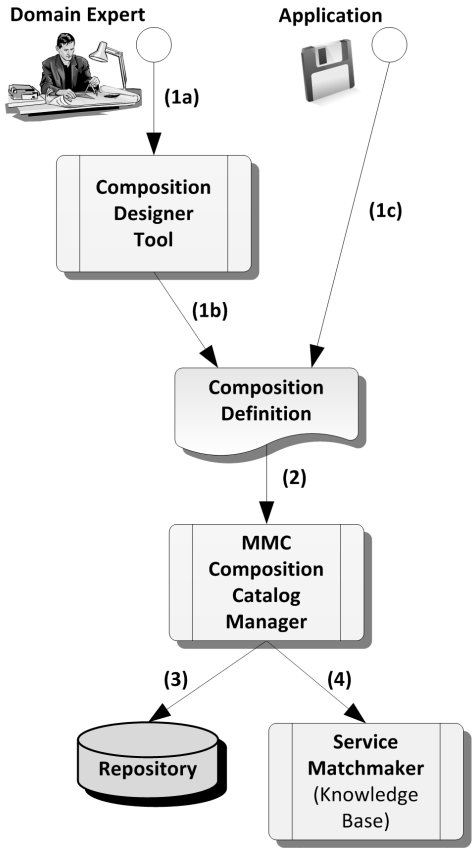

Figura 3.8: Defining a new abstract Composition

(1): A domain expert uses the web-based composition designer tool to define the workflow associated with the abstract composition (1a). This tool exports the workflow definition as an XML-based document (using for instance the Windows Workflow Foundation XAML format or WS-BPEL) (1b). Alternatively an application may provide the composition XML-based definition directly (1c);

(2): The abstract composition definition is received by the metamodel management center;

(3): The metadata associated with the composition and the XAML abstract workflow description are stored in a repository (e.g. standard relational database);

(4): The semantic service profile description (OWL-S), which includes the IOPE's of the service, is registered in the service matchmaker knowledge base.

The process to create concrete composite services based on the application requests has the following stages (see Figures 3.9 and 3.10):

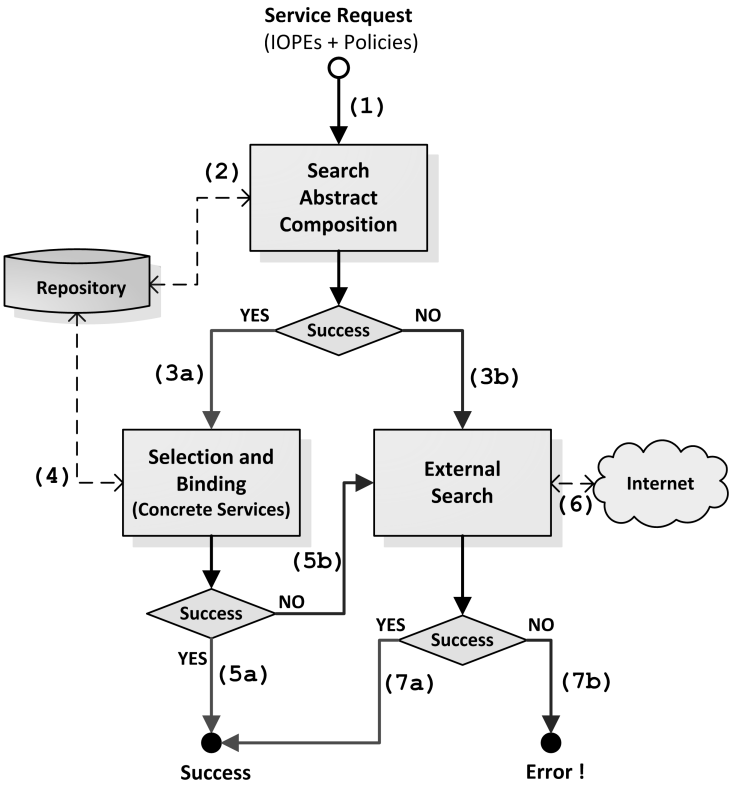

Figura 3.9: Composite Service Creation

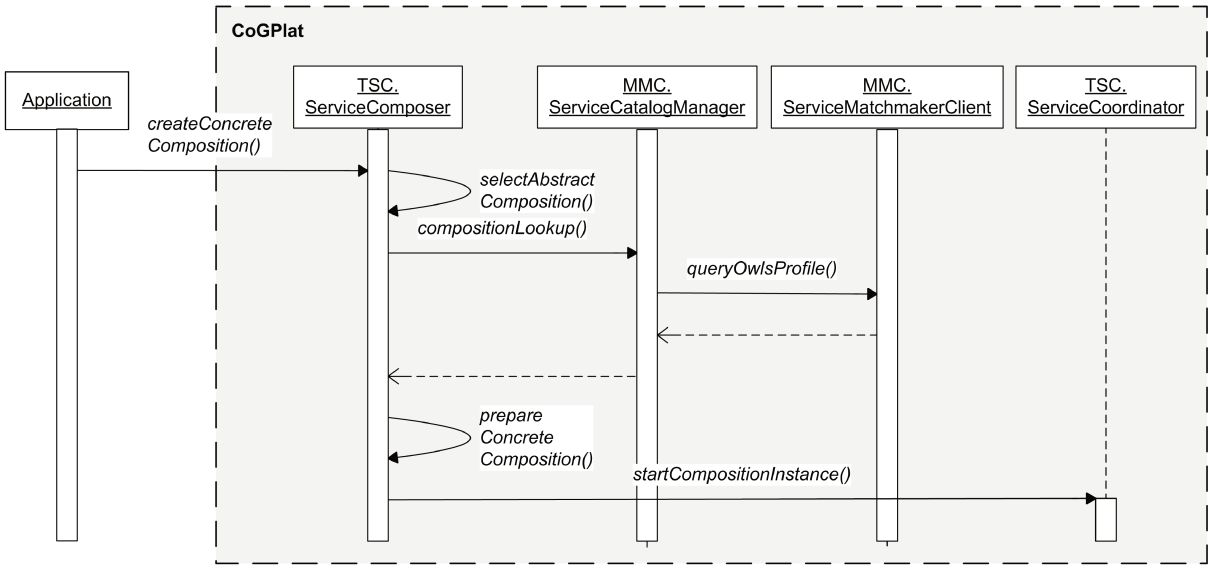

Figura 3.10: Composite Service Creation (UML Sequence diagram)

(1): An application sends a request (an OWL-S profile) to the *Transparent Services Center* for a composite service (*TSC.ServiceComposer.createConcreteComposition()*). The request includes the IOPE's and the demanded interaction policies;

(2): The *TSC Service Composer* uses the *Metamodel Management Center* to perform a search in the repository looking for an abstract composition that satisfies the application requirements (*MMC.ServiceCatalogManager.compositionLookup()*). The OWL-S matchmaker is queried to fulfill this task (*MMC.ServiceMatchmakerClient.queryOwlsProfile()*);

(3a, 4): If an abstract composition is found, a Service Coordinator is created within the TSC (*prepareConcreteComposition()*) and the associated workflow starts to run (*startCompositionInstance()* call). A search for concrete services that will execute each of its abstract activities is done (see in Section 3.4 the *InvokeTransparentService* activity). The interaction policies are, together with the IOPE's, used as selection criteria. Else **(3b, 6, 7a, 7b)**: a search for external services (not pre-registered in the platform) is tried;

(5a): If a concrete composition was successfully built and executed, the results are returned to the calling application. Else **(5b, 6, 7a, 7b)**: a search for an external service is tried within some trusted third party UDDI or semantic-enabled catalog (as in **(3b)**).

The chances of success on the initial stages of the transparent composite service creation process (steps **1** and **2** in Figure 3.9) are directly related to the amount and diversity of abstract compositions registered in the platform.

3.4 Implementation Issues

The *CoGPlat* infrastructure was modeled to be platform-independent and an implementation of it over different technologies would be perfectly possible. Next we discuss our implementation choices.

3.4.1 Middleware Prototype

The *CoGPlat* middleware infrastructure is implemented with the support of the following technologies:

1. the **Service Bus** exports traditional Web Services (WSDL + SOAP/HTTP);
2. the middleware **core** runs over the Microsoft .NET framework and is written in C#;
3. the **matchmaker** included in the Metamodel Management Center is based on the OWLS-MX [Klusch 06], a JAVA/Jena based matchmaker that performs both semantic (OWL-S Profiles) and syntactic matching between requisitions and services;
4. the *CoGPlat* **Composition Designer Tool** is an extension of the *Atlas Workflow Designer* component [Flanders 06];
5. the **composition executions** are performed by the .NET 3.0 *Windows Workflow Foundation* (*WF*) runtime engine [Bukovics 07]. The composite service definitions are based on the XML-based XAML format defined in the WF (WS-BPEL compositions are supported through the *BPEL for Windows Workflow Foundation* Microsoft add-on).

The WF offers native persistence mechanisms, facilitating the fault recovery process, and provides basic tracking services, used as a base to implement the Traceability and Auditing Center functionalities.

It also natively offers activities to perform interactions with traditional Web Services, which only partially fulfill *CoGPlat*'s composition process demands. To handle this WF limitation and considering that it supports dynamic updating (in execution time) of the execution plans through reflection mechanisms, new WF custom activities were implemented. They follow the service lookup strategy discussed in Section 3.3.7, with support for

semantic annotations (IOPE's and Policies) and dynamic web service binding and invocation. The *InvokeTransparentService* activity presented in Figure 3.11 represents the basic building block in *CoGPlat*'s Compositions. It corresponds, for each individual concrete service, to the step (4) of the process described previously in Figure 3.9. Initially it receives as input annotations containing IOPE's and policy requirements and tries to find a suitable concrete service using the MMC semantic match services (*SemWServLookup* code activity) or a traditional UDDI lookup if not successful (*TraditionalWSUDDILookup* activity). Next, the compliance with the active interaction policies is checked (*VerifyPolicies* / *VerifyPolicies2*) and then the dynamic binding and invocation of the selected concrete service is done through instances of the custom WF activities *DynamicWebServiceBind* and *DynamicWebServiceInvoke*.

3.4.2 CoGPlat Web Administrative Center

The *CoGPlat Web Administrative Center* is a web-based application that allows a domain administrator to completely interact with the middleware facilities and services. It basically offers the same services delivered to the middleware client applications, including the possibility of managing the repositories, requesting transparent services, monitoring and auditing processes and also designing new abstract compositions. Figure 3.12 presents a screenshot of it.

3.4.3 Application Scenarios

An application in the **Civil Construction** domain was prototyped over *CoGPlat* to evaluate its infrastructure. It implements the complete bureaucratic process required to obtain an authorization to build houses, inspired in the Brazilian legislation. This scenario is considered dynamic because the selection of the concrete services that will be responsible for the actual activities is done at execution time according to IOPE's specified in the application request. They may change, for instance, depending on the location of the building and also on the selected interaction policies. Two composite services are included in this example and accessed by the client application: the first is responsible for the process to obtain an authorization to build the house and the second is responsible for the final authorization to move after the construction is concluded. The main steps of these interactions are (see Figure 3.13):

1. : The application requests through the *CoGPlat Service Bus* the creation of a concrete composition that will handle the *build authorization* process (using the *createConcreteComposition()* operation);

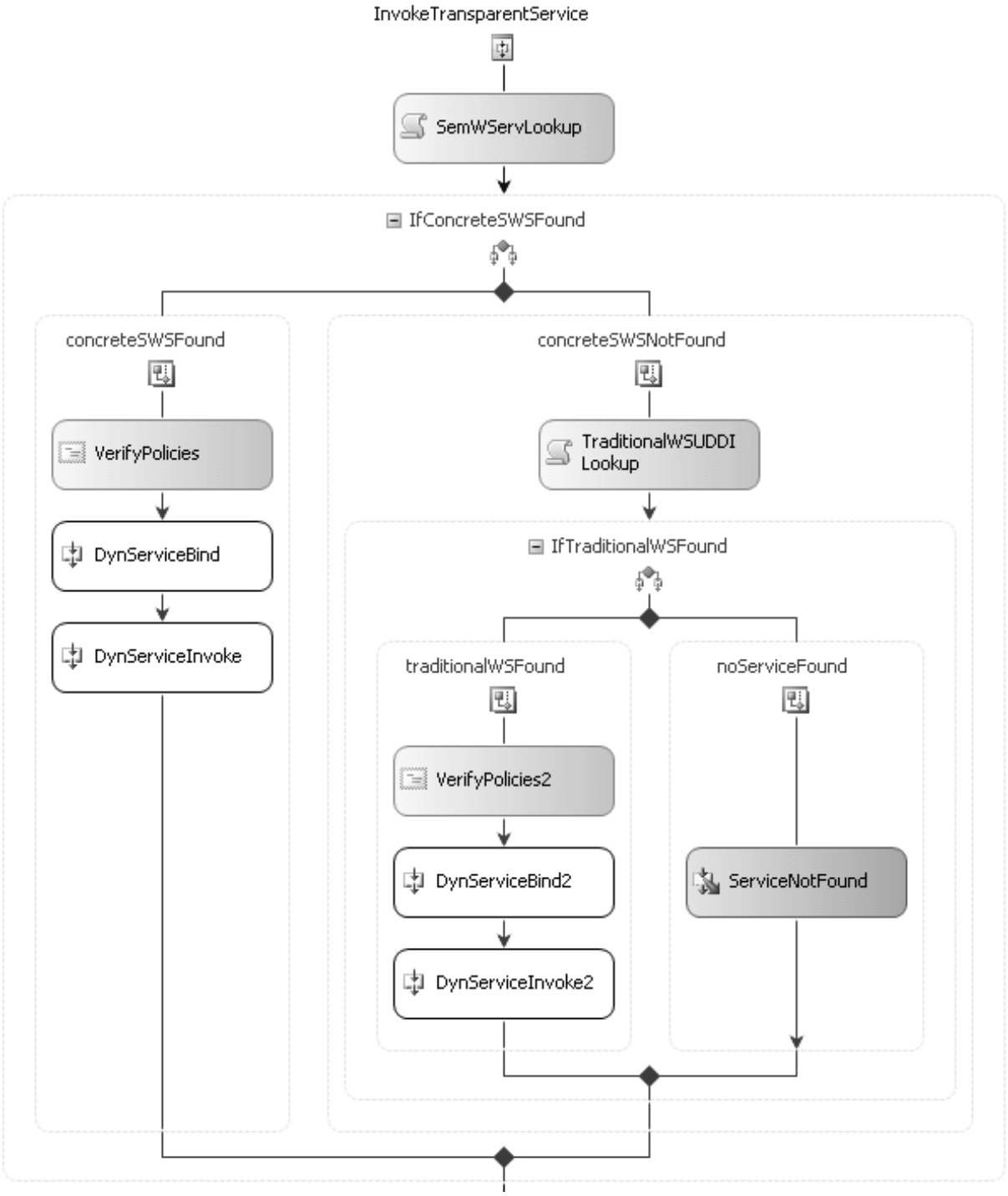

Figura 3.11: CoGPlat Custom WF Activity: *InvokeTransparentService*

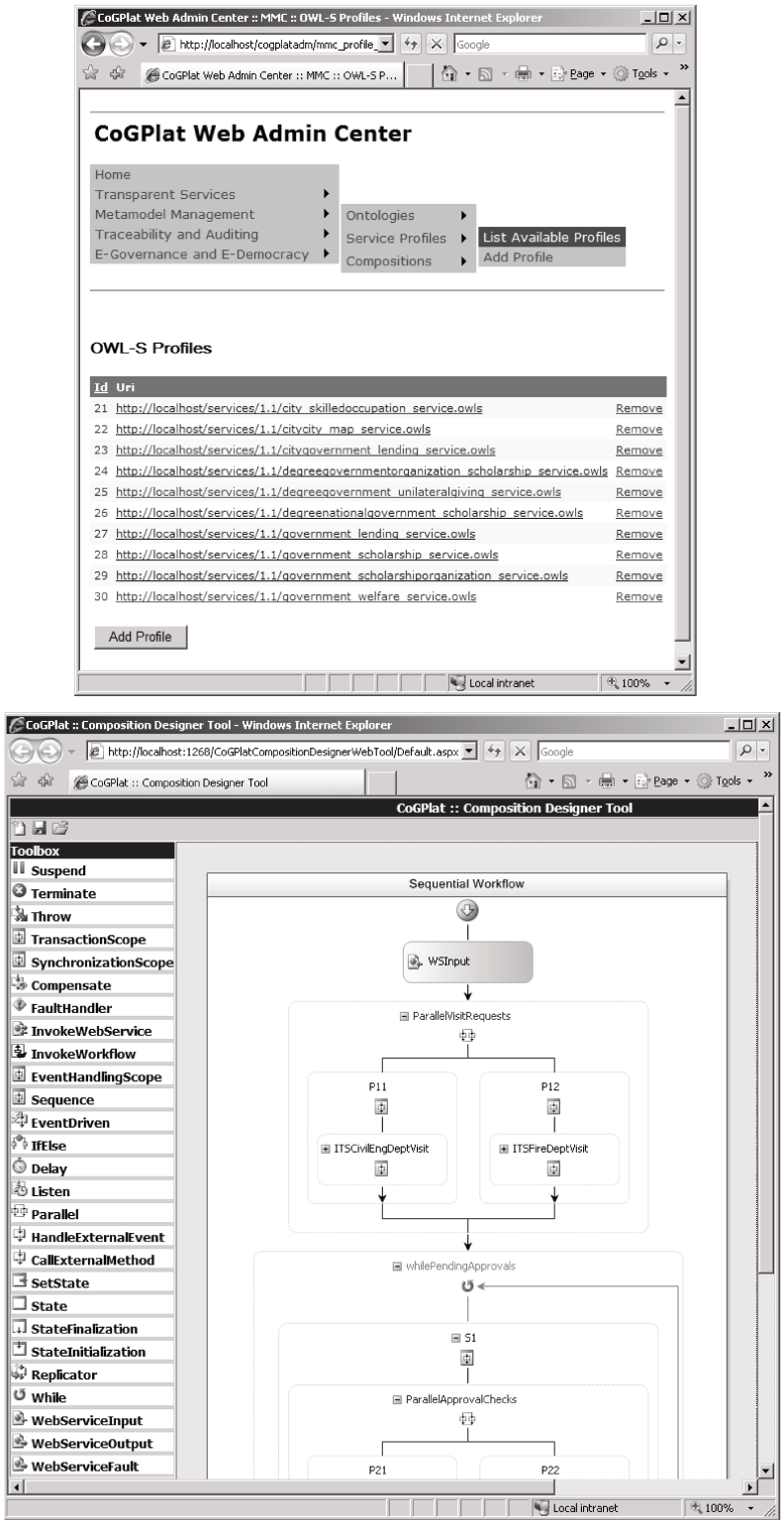

Figura 3.12: CoGPlat Web Administrative Center Screenshot

2. : Internally (in a transparent manner for the application), *CoGPlat*'s facilities prepare the composite service following the strategy discussed previously in Section 3.3.7. As a result the *BuildAuthorizationService* workflow is selected and an instance of it is created (detailed later in Figure 3.14);

3. : The application, after receiving a notification that the composition is ready to run (through the return value of the operation called in step 1), calls the *startCompositionInstance()* operation to request its execution;

4. : The service execution is initiated through a call to the *reqBuildAuth()* operation exposed by the workflow;

(5, 6) : The results are returned to the associated *ServiceCoordinator* and then back to the calling application;

(7, 8) : The application now requests the creation of a concrete composition to handle the *move authorization* process. Note that this step will most likely happen a long time (months or even years) after the previous one. The result is an instance of the *MoveAuthorizationService* workflow (detailed later in Figure 3.15);

(9-12) : As in the steps 3-6, the instance execution is initiated and the results are returned to the associated *ServiceCoordinator* and then to the calling application.

It is important to remember that the two selected (abstract) workflows were previously defined by a domain expert using *CoGPlat*'s *Composition Designer Tool*. The first, which defines the *BuildAuthorizationService*, is shown in Figure 3.14 (which uses the WF notation). It performs the following activities:

- *WSInput*: input activity which defines that this workflow is accessed as a web service. Its input arguments are in conformance with the semantic descriptions defined in the associated OWL-S profile and registered in the MMC repository;

- *ITSProjectApprDept*: this is an instance of the *InvokeTransparentService* activity presented in Figure 3.11. It defines a call to a service provided by some project approval department that should be selected, according to the activity definition, at execution time based on the IOPE's;

- *checkRegionCategory* (if-else): verifies whether the building address is part of a region regulated by some military organization (e.g., in Brazil buildings close to the beach are under navy jurisdiction).

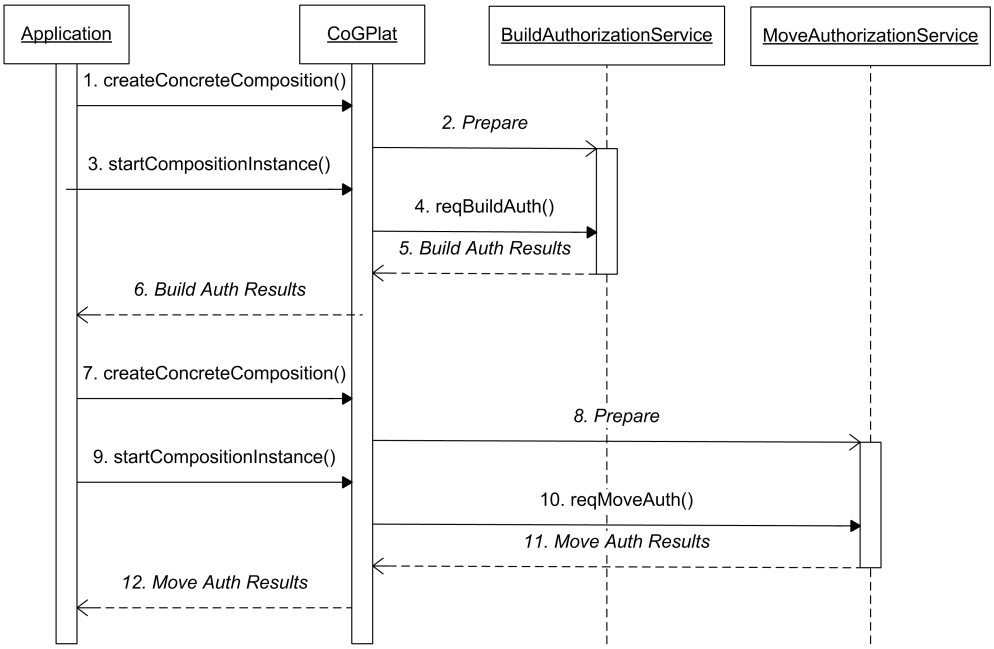

Figura 3.13: UML Sequence Diagram: Interactions between the application, *CoGPlat* and the new composite services in the Civil Construction example

- *NonMilitaryRegion* :: *ITSNeighbApprovalService*: if the region is not under military jurisdiction, a transparent service which manages a neighborhood approval process is invoked (similar to the U.S. *Public notice of action*), where comments or protests from the community concerning the proposed building plan may be collected and evaluated by a designated authority. One interesting alternative here is to select inside the transparent service activity another customized workflow (pre-defined by a domain expert) that relies on some of the services provided by *CoGPlat*'s *E-Governance and E-Democracy Center*;

- *MilitaryRegion* :: *checkMilitaryRegionCategory* (if-else): if the region is under military jurisdiction another check is performed to verify if it is under the army (e.g. close to an international border) or navy (e.g. close to the ocean) jurisdiction. Constructions in these areas are possible with a special authorization from the corresponding entity and are subject to additional federal taxes;

- *BldgCloseToSecurityZone* :: *ITSLocalArmyAuthority* / *BldgCloseToOcean* :: *ITSLocalNavyAuthority*: the corresponding local army or navy authority approval service is accessed (through an *InvokeTransparentService* activity);

- *WSOutput*: activity associated with *WSInput* that indicates the returning values.

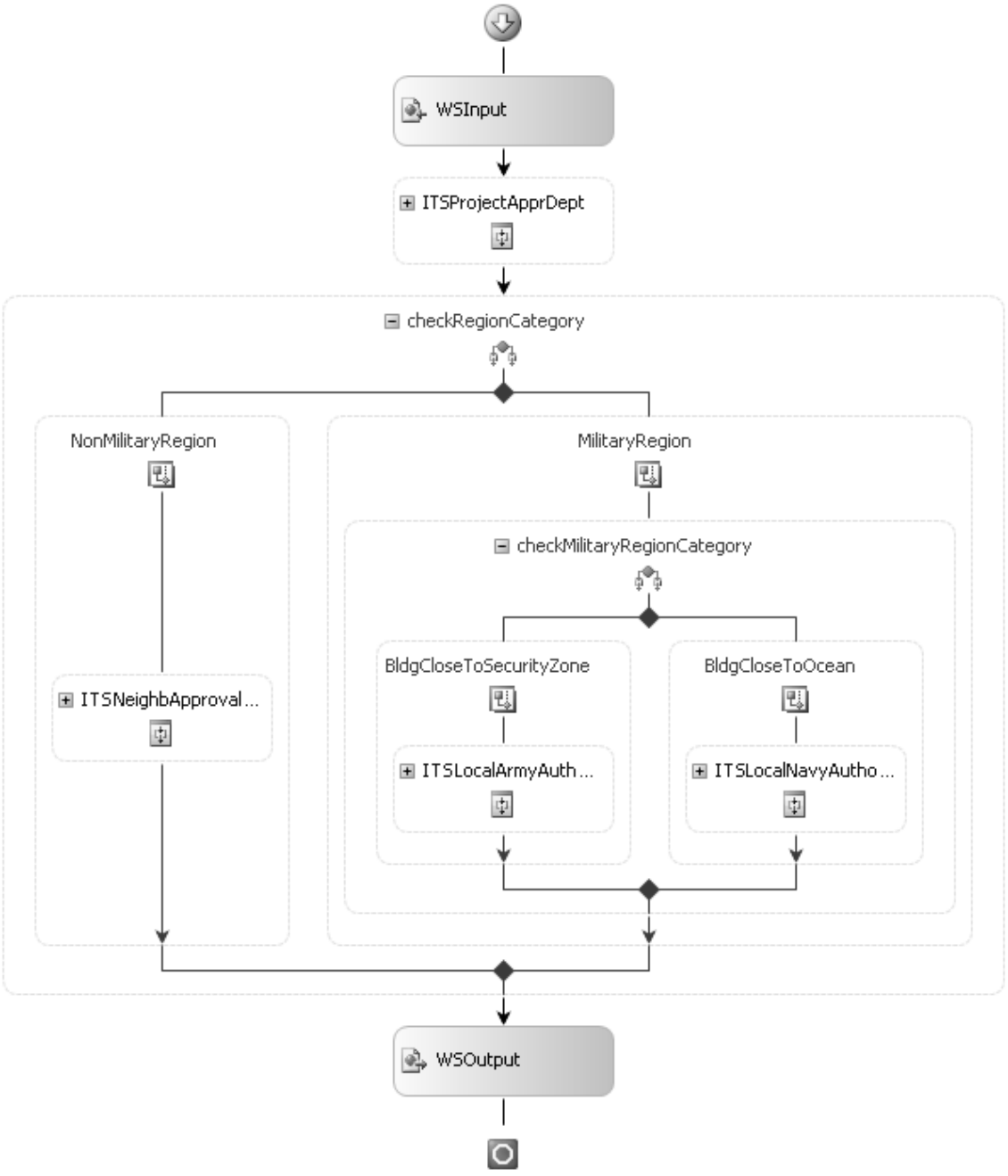

Figura 3.14: *BuildAuthorizationService* workflow (WF notation)

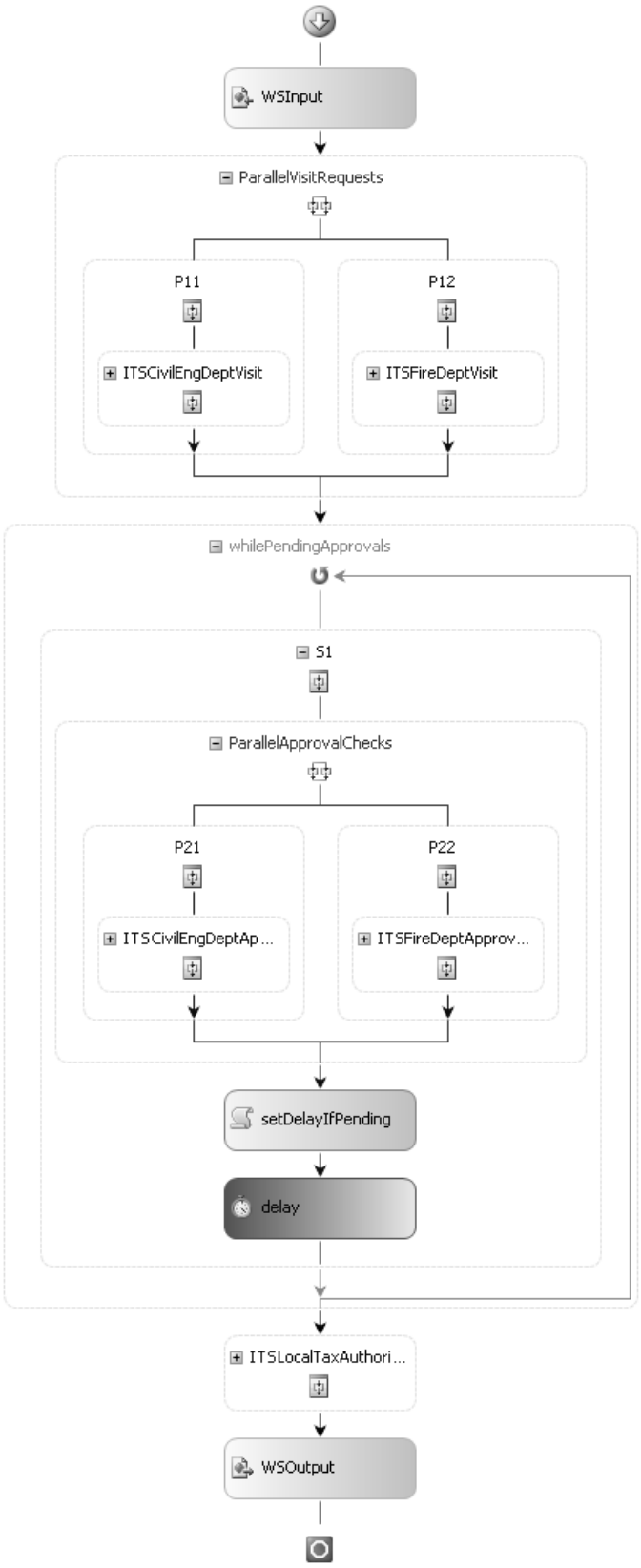

Figura 3.15: *MoveAuthorizationService* workflow (WF notation)

The second workflow of this example defines the *MoveAuthorizationService* and is presented in Figure 3.15 using the WF notation. It consists of the following activities:

- *WSInput*: input activity which defines that this workflow is accessed as a web service. Its input arguments are in conformance with the semantic descriptions defined in the associated OWL-S profile and registered in the MMC repository;
- *ParallelVisitRequests*: the activities inside this one are executed in parallel (P11 and P12);
- *P11 / P12*: the *InvokeTransparentService* activity is used to request inspection visits from the local civil engineering department (*ITSCivilEngDeptVisit*) and from the local fire department (*ITSFireDeptVisit*);
- *whilePendingApprovals*: traditional *while* loop that repeats the execution of its child activities while some control condition is true (i.e. approvals are still pending);
- *S1* :: *ParallelApprovalChecks*: the activities inside this one are executed in parallel (P21 and P22);
- *P21 / P22*: invoke transparent services that check if the approvals from the local civil engineering department (*ITSCivilEngDeptApprovals*) and from the local fire department (*ITSFireDeptApprovals*) have already been issued. Note that the visit requests made inside P11 and P12 do not output an immediate approval response (they are long-running operations), therefore it is necessary to periodically check their results;
- *S1* :: *setDelayIfPending*: if any of the wanted approvals is still pending a delay is set (e.g. 12 hours);
- *S1* :: *delay*: pauses the workflow for a specific amount of time before resuming the while loop execution;
- *ITSLocalTaxAuthority*: the local tax authority service is informed of the results of the inspections (approvals and/or rejections) and issue the corresponding fees and taxes to the requesting citizen;
- *WSOutput*: activity associated with *WSInput* that indicates the returning values of the workflow.

Two additional application scenarios, suggested as future work, are:

1. **e-Ombudsman**: an interactive on-line channel to collect and investigate complaints (and also suggestions) by private citizens against government services and agencies. The application could handle the complete life-cycle of the complaints, until the final feedback from the administrator is informed and accepted by the citizen. It should be fully integrated with all public administration agencies;

2. **Emergency Management Systems**: due to *CoGPlat*'s inherent interoperability capabilities, the middleware infrastructure could facilitate the implementation of an application that collects and manages essential information used by emergency response teams, serving as a powerful tool to organize their efforts.

3.4.4 Qualitative Evaluation

Based on the process to implement the civil construction scenario it was possible to observe that the application development efforts were mostly related with the user interaction mechanisms. The **complexity** usually associated with the definition and execution of the composite public processes was transferred mostly to the middleware components.

Nevertheless, it is of utmost importance to remember that the success (or failure) of the dynamic composition processes is directly affected by the variety and coverage of the abstract compositions previously registered in the platform. The middleware administrators and the domain experts must guarantee that all public processes that should be provided on-line through client applications have a abstract definition. This may at first sound as a problem or deficiency of the platform, but it is fundamental to remember that the peculiarities of the e-Government domain require exactly that, i.e., only services defined in the legislation may be delivered.

In addition, if well designed, the same application can be applied to different contexts due to the **flexibility** inherently provided by the **dynamic** service composition process. In the civil construction example, different categories of authorizations to build in different regions can be handled by the same application as the corresponding services are selected at execution time based on the IOPE's. This property, when well explored by the domain experts and application designers, facilitates the development of one-stop government portals. The adoption of *Interaction Policies* also enhances the flexibility and dynamism of the compositions as they allow a **finer-grained specification** of **autonomy**, **privacy** and **traceability** requirements.

The middleware functionalities are fully service-oriented and exposed as traditional Web Services to facilitate the **interoperation** with the applications and also with other e-Government platforms when required. *CoGPlat*'s core facilities are internally modeled

and implemented in an object-oriented fashion in order to make the efforts to **extend** it and **adapt** it easier.

3.5 Concluding Remarks

The proposal of effective solutions to enable interoperability among heterogeneous and inter-organizational systems remains a key issue in the development of new citizen-centric e-Government services. Considering the challenges introduced by these new applications, we contribute in this paper proposing a semantically-enriched and service-oriented middleware (***CoGPlat***). Its main goal is to enable the collaboration among governmental entities, organizations and citizens in various public administration scenarios. We also contribute proposing a mechanism based on interaction policies and on semantic descriptions to build composite services.

The experience to build the prototype, which involved the integration of a variety of technologies, was also discussed in the paper. A complete application example was shown to demonstrate important platform characteristics. The use of semantics and ontologies proved itself to be very useful in the service composition process as it took advantage of the fact that the search domain was limited by the abstract compositions previously specified in the middleware repository. The approach proposed and tested in the *CoGPlat* prototype represents an end-to-end alternative that enables access to composite e-Government services, having the solutions for issues such as interoperability, privacy and traceability transferred to the middleware level.

Future works include the development of an AI-based facility to automatically generate execution plans and also an alternate implementation of *CoGPlat* running over a Grid Services infrastructure. There are still many challenges to be faced before a fully citizen-centric e-Government becomes a reality. However, the visioned changes create the expectations of a future where public administration processes become really transparent and efficient, motivating further research in this field.

Acknowledgments

We would like to thank CAPES, CNPq and DAAD for financially supporting the project. Additional thanks to the Fraunhofer FOKUS Institute (Berlin, Germany).

Capítulo 4

Transparency in Citizen-Centric Services: A Traceability-based Approach on the Semantic Web

4.1 Introduction

The demands for the creation of mechanisms to increase the transparency of the public administration processes have dramatically increased over the recent years. It represents a requirement for every democracy identified since the origins of the modern republic: Thomas Jefferson once wrote that "*whenever the people are well-informed, they can be trusted with their own government*" [Jefferson 89]. In parallel, new citizen-centric services and applications are becoming more and more investigated, mainly due to the proliferation of the new ICTs (*Information and Communication Technologies*). The use of Service-oriented architectures and, more recently, semantics and ontologies, is also gaining momentum to enable interoperability and to increase the dynamism of e-Government applications. Considering this scenario, this paper contributes by presenting an strategy to achieve transparency in composite e-Government Services based on a set of Traceability Policies modeled and implemented over a semantically-enriched and service-oriented middleware (***CoGPlat***). The remainder of this paper is organized as follows: Section 4.2 introduces the concepts and technologies related to the contributions of the paper and presents relevant related work; Section 4.3 introduces the proposed approach to increase transparency in composite e-Government services; and finally, Section 4.4 presents the concluding remarks.

4.2 Literature Review

The **Interoperability**, defined as "*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*" [IEEE 90], is a fundamental requirement in the context of distributed and dynamic applications. In order to achieve higher levels of interoperability, many systems are being implemented following the Service-Oriented Architecture (SOA) approach [Papazoglou 03] and its most famous manifestation, the Web Services technologies. One of the current Web biggest limitations is the lack of support to describe the semantics of data. To overcome this weakness, the so-called **Semantic Web** [Berners-Lee 01] proposes a scenario where data becomes meaningful to machines and can be automatically processed and understood. In it, ontologies play a fundamental role in the definition of the concepts that are used to annotate data [Medjahed 04]. Specifications such as RDF (*Resource Description Framework*) and RDF-Schema were among the first W3C (World Wide Web Consortium) initiatives to model meta-data related to Web resources. Later, the OWL (Ontology Web Language) extended RDF, augmenting its vocabulary with the inclusion, for instance, of new class relationships. It is the current W3C standard for specifying ontologies on the Web. The possibility of applying a similar strategy to semantically describe services, opening the road towards their automatic discovery, invocation and composition, motivated the proposal of different approaches for the definition of what was baptized as the **Semantic Web Services**. OWL-S (*Semantic Markup for Web Services*), for instance, combines a set of inter-related OWL ontologies that define terms used in service-oriented applications. Besides OWL-S (adopted in the work described in this paper) two other proposals already play an important role in the Semantic Web Services scenario: the WSMO (Web Services Modeling Ontology) [W3C 05]) and the SAWSDL (Semantic Annotations for WSDL and XML Schema) [W3C 07], recently adopted as a W3C recommendation.

Techniques which enable markup and automated reasoning technology to describe, simulate, test, and verify compositions of Web services are discussed in [Narayanan 02]. The authors define the semantics for a relevant subset of OWL-S in terms of a first-order logical language (*Situation Calculus*). With the semantics in hand, service descriptions are encoded in a Petri Net formalism. The implemented system is able to read in OWL-S service descriptions and perform simulation, enactment and analysis. The use of advanced workflow and activity concepts in the composition of Web services is the proposal of [Fileto 03]. The approach is called POESIA (*Processes for Open-Ended Systems for Information Analysis*), an open environment for developing Web applications using metadata and ontologies to describe data processing patterns developed by domain experts. It supports Web service composition using domain ontologies with multiple dimensions (e.g., space, time, and object description).

The **electronic Government** (e-Government) domain includes the "*set of all processes which serve decision-making and services in politics, government and administration and which use information and communication technologies*"[KBSt 06]. Recently a new approach to e-Government is gaining momentum: the citizen-centric government, where citizens and businesses are considered customers of the public administration, so that their needs come first, rather than bureaucracy or other imperatives inside the government machine [GOV3 06]. In this context, usually a government-wide service-oriented architecture is applied to develop a single place that offers access to all government informational and transactional services. Several research efforts in the e-Government domain can be found in the literature. For instance, a system which automatically generates Web services customized to citizens' needs and also to government laws and regulations is presented in [Medjahed 05]. It proposes three levels of service customization: the *Citizen* level, the *Service* level and the *User interface* level. A *metadata ontology*, used to describe e-Government services and operations, is also introduced. An approach for the semi-automated design of data flows between Web Services that are semantically described using different ontologies and data representations is introduced in [Barnickel 06], including a rule-based mechanism for user-transparent mediation between ontologies spanning multiple application domains.

4.3 Enabling Transparency

As already mentioned, transparency is becoming a fundamental requirement in citizen-oriented applications. One of the possible strategies to improve transparency is to guarantee **traceability** at the service level. Traceability is defined by the International Organization for Standardization [ISO 94] as the "*ability to trace the history, application or location of an entity by means of recorded identification*". We introduce a strategy based on traceability policies to regulate the execution monitoring of composite e-Government services and detail it next.

4.3.1 The Middleware

The strategy presented in this paper is modeled and implemented over an e-Government service middleware called ***CoGPlat*** (***C****itizen-****o****riented e-****G****overnment* ***Plat****form*)
[Santos 05]. Its main goal is to support applications that enable the interaction and collaboration among governmental entities, organizations and citizens and its infrastructure includes (see Figure 4.1): a **Service Bus**, an interface between the middleware services and the applications; four **core** facilities (described next); a **Service Discovery and Execution Layer**, responsible for selecting the Semantic Web Services to participate in

the compositions and also for interacting with those services at process run-time; and a set of **Support Services** which provides security, persistence, reliable messaging and transaction support to the processes running over the platform.

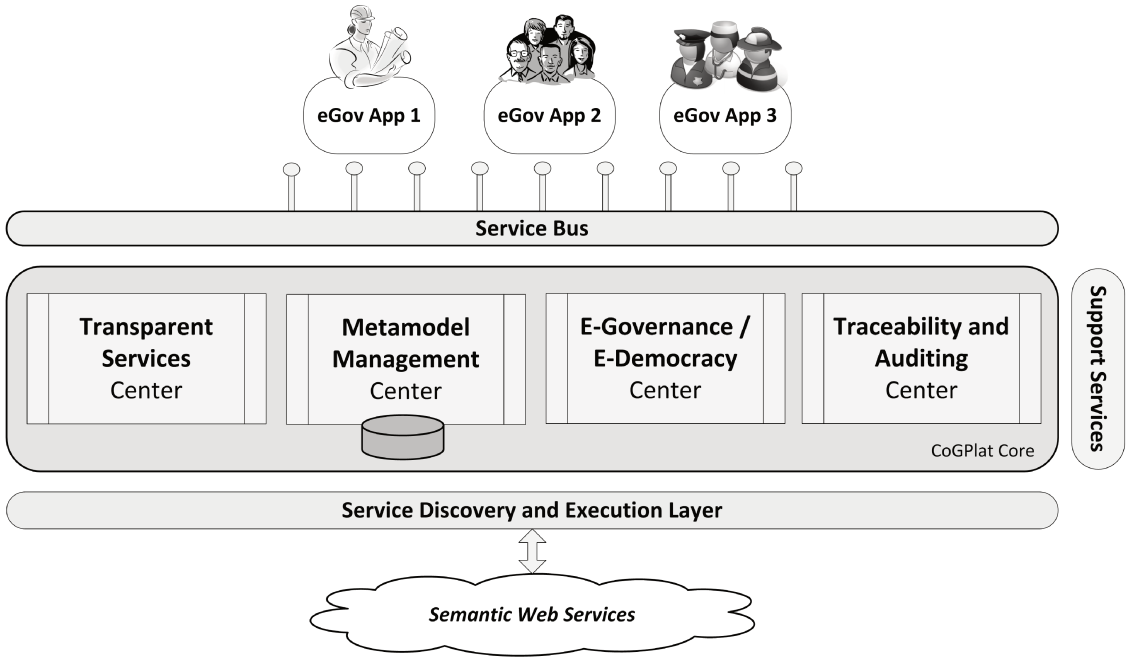

Figura 4.1: CoGPlat Architecture Overview

The four middleware core facilities are (see also Figure 4.1): the **Transparent Services Center**, responsible for dynamically building the service compositions according to the application requests; the **Metamodel Management Center**, which offers services and tools to manage the models, metamodels and ontologies used in the description of services, compositions, processes and entities; the **E-Governance and E-Democracy Center**, which delivers generic services which aim to facilitate the decision-making processes and to increase citizen participation in the public administration; and the **Traceability and Auditing Center**, which offers services and tools to monitor running processes and also to audit processes that have already been concluded. The contributions of this paper are concentrated in this last facility.

4.3.2 Policies

To effectively compose e-Government services, issues like the autonomy of the entities, data privacy, process traceability and an efficient identity management usually need to be

considered. In order to enable compositions which handle these demands, *CoGPlat* implements a set of *Interaction Policies* which were first proposed and successfully applied in an e-Business scenario [Santos 06] and then extended to match the new requirements demanded by e-Government applications. These policies are applied to regulate the *collaborations* that take place over the middleware infrastructure - we consider that two entities *collaborate* when both participate in the same composite service (as providers and/or consumers) and there is at least one information or message exchange between them through the middleware facilities. The *CoGPlat Interaction Policies* are classified into the following categories (all policy terms and relations are specified in an OWL ontology):

1. **Entity Autonomy Policies**: determine the level of control the platform (and therefore the applications running over it) may have over the internal operations of a composite service;

2. **Data Privacy Policies**: determine the collaboration levels on the interactions between two entities that participate on the same composite service, defining how the privacy of the data exchanged between them is handled;

3. **Identity Management Policies**: establish where to provide mechanisms like anonymousness, identity theft protection and non-repudiation for citizens and entities;

4. **Service Traceability Policies**: determine to what extent the operations of a composite service should be tracked by the middleware. This is the category closely related to the contributions presented in this paper and will be further detailed next.

An abstract composition and individual abstract activities can be semantically annotated with the following **Service Traceability** policies:

- **Strong-traceability**: all possible events both at composition and at activity/service level are monitored. Customized traceability requirements can also be specified (e.g.: monitor the behavior of an specific property/value throughout the instance execution);

- **Weak-traceability**: only composition level events are monitored. No customizations are allowed;

- **Zero-traceability**: There is no traceability at all. Indeed, if this policy is specified, there must be no trace (neither during nor after) of a given composition instance.

4.3.3 The Composition Process

One of the main goals of *CoGPlat* is to facilitate the development and operation of new e-Government applications that present higher levels of dynamism and citizen-orientation. Given the quantity and diverseness of public administration agencies, to integrate and interoperate the services delivered by those agencies become a challenging task. The service composition process described next tries to shift to the middleware level the solution for those integration problems. It is important to understand how *CoGPlat* builds composite services before a deeper discussion on the Traceability mechanisms is held. The composition process has the following stages (see Figure 4.2):

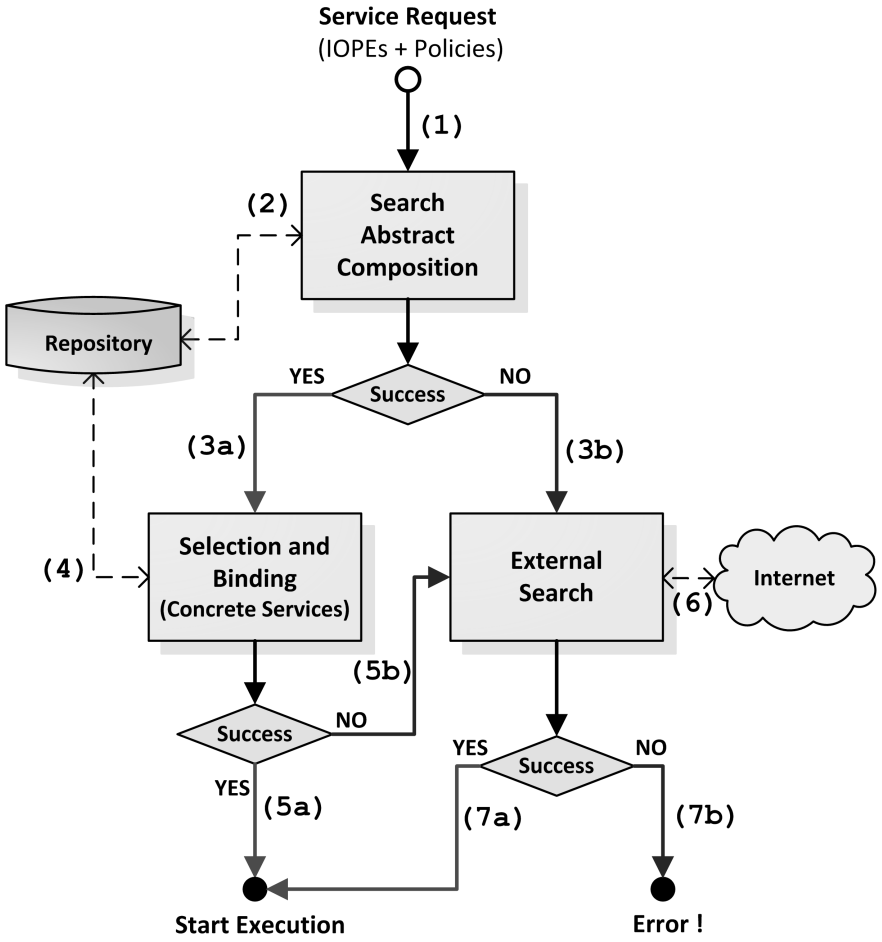

Figura 4.2: Composition Process in CoGPlat

(1): An application sends a request (an OWL-S profile) to the *Transparent Services Center* for a composite service. The request includes the IOPEs (Input, Output, Precondition, Effects) and the demanded interaction policies;

(2): The *Metamodel Management Center* performs a search in the repository for an abstract composition that satisfies the application requiremens. An OWL-S matchmaker is used in this stage;

(3a, 4): If an abstract composition is found, a search for concrete services that will execute the abstract composition activities is started. The interaction policies are, together with the IOPEs, used as selection criteria. **(3b, 6, 7a, 7b)**: Else, a search for external services (not pre-registered in the platform) is tried;

(5a): If a concrete composition has been successfully built, its specification is prepared for execution. **(5b, 6, 7a, 7b)**: Else, an external search is tried (as in **(3b)**).

The chances of success on the initial stages of the composition process (**1** and **2** in Figure 4.2) are directly related to the amount and diversity of abstract compositions registered in the platform. These abstract compositions describe the generic public processes that may run over the platform combining services from different agencies. No concrete binding is necessary in design time - actually only service classes, semantically annotated with IOPEs and Policies are included in the abstract composition flow.

4.3.4 Traceability and Auditing Center

The **Traceability and Auditing Center** implements the mechanisms necessary to guarantee that the active traceability policies are respected during the execution of the composite services. Its internal infrastructure is composed of the following elements (see also Figure 4.3): the **Standard Tracking Service** which collects all data from the generated tracking events, according to the active policy, and saves it into a repository; the **Real-time Tracking Service** which collects only representative data from the tracking events, also according to the active policy, but provides it in real-time to the platform client applications instead of saving it to a repository; the **Auditing Service** which offers mechanisms to access tracking data of already concluded processes; and a **Tracking Data Repository** that stores the information generated by the tracking service events.

4.3.5 Composition Example

In order to illustrate how the Traceability Policies are evaluated and executed in *CoGPlat*, let's follow the example of a generic process of requesting a document. The process is composed of the following steps: **(1)** Validation of the requester personal information; **(2)** Payment of administrative fees; **(3)** Forward of the request to the responsible government office (determined by the document type).

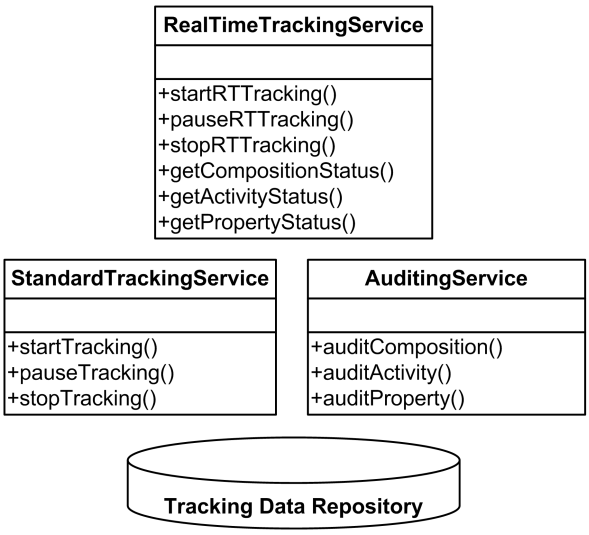

Figura 4.3: Traceability and Auditing Center Infrastructure

The composition process follows the steps previously presented in Figure 4.2. First, the application sends a request containing IOPEs and demanded policies. A search is then performed in the abstract compositions repository. In Figure 4.4 the selected abstract composition is shown. Note that the composition and the individual activities are annotated with IOPEs and with the supported traceability policies.

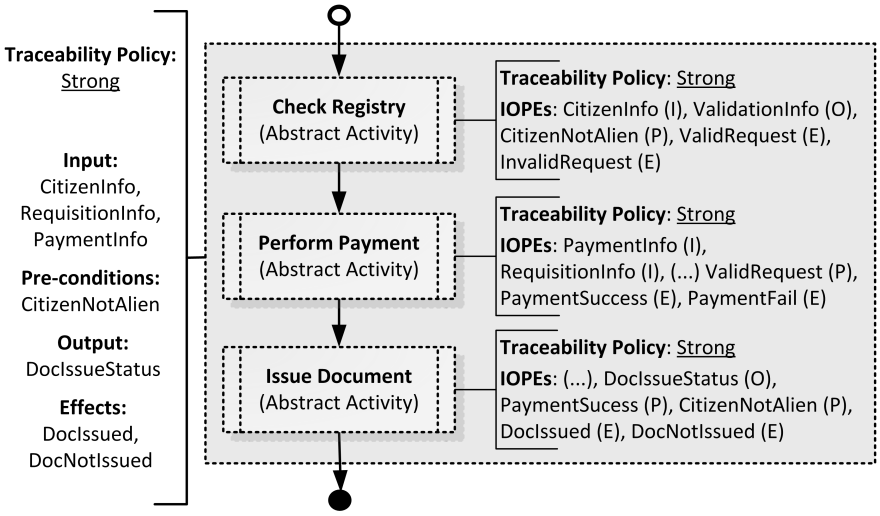

Figura 4.4: Abstract Composition with Policies and IOPEs

Next, according to the process in Figure 4.2, the semantic annotations are used to select concrete services that will implement each of the activities. When the composition starts to execute, the **Standard Tracking Service** of the **Traceability and Auditing**

Center is activated. In the example, the composition and all activities require a strong traceability policy, so all composition and service level events are tracked and stored in the **Tracking Data Repository** (a database). While the composition is running, it is possible monitor its status in real-time. This is achieved in the platform through the following steps (see Figure 4.5):

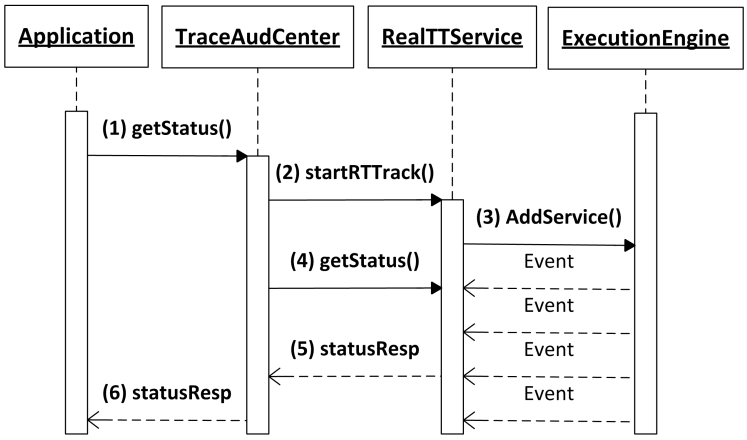

Figura 4.5: Finding the current status of a running process

(1) The *Application* calls the *getStatus()* operation from the **Traceability and Auditing Center** (*TraceAudCenter*) service;

(2) The *TraceAudCenter* informs the **Real Time Tracking Service** (*RealTTService*) that a given composition instance should be tracked following a given policy;

(3) The *RealTTService* subscribes to the tracking events generated by the WF Runtime Engine (*ExecutionEngine*) for the given instance. It uses a custom profile that implements the active traceability policy (see also Section 4.3.6). From this moment, the *events* specified in the profile start to be delivered to the *RealTTService*;

(4) The *TraceAudCenter* asks the *RealTTService* for the status of a given composition (or also of an activity or property);

(5) The status response (*statusResp*) is sent back to the *TraceAudCenter* and then to the *Application* **(6)**.

The next time the application requests real-time status information of the same composition instance, the steps (2) and (3) do not need to be repeated (the flow starts then from step (4)) as the instance is already being monitored. It is also possible to audit the process

execution anytime after its conclusion using the **Auditing Service**, which offers to the applications a convenient way to access the tracking information stored in the repository. Note, though, that only information in accordance with the selected traceability policy will be present there.

4.3.6 Implementation Issues

The *CoGPlat* middleware infrastructure is implemented with the support of the following technologies: the **Service Bus** export traditional Web Services (WSDL + SOAP/HTTP); the middleware **core** runs over the Microsoft .NET framework and is written in C#; the **service discovery** layer uses the OWLS-MX matchmaker [Klusch 06]; the **composition executions** are performed by the *Windows Workflow Foundation* (WF) runtime engine [Bukovics 07]. The **Traceability and Auditing Center** implementation relays on the WF *Tracking Service* functionalities. It provides three event categories: **Workflow Events** which correspond to changes in the composition instance status; **Activity Events** which correspond to changes in the execution status of individual activities (services); and **User Events** that can be generated at any point in the life cycle of the composition instance (those events are customized and defined during the construction of the abstract compositions).

The default WF Tracking Service behavior generates events for all workflow and activity types. Therefore, to implement a behavior that corresponds to the chosen traceability policy, *CoGPlat* defines three custom WF tracking profiles, one for each of the possible traceability policies (**strong-**, **weak-** and **zero-** traceability). Note that as the first policy (strong) allows customization, additional tracking behavior might be specified during the construction of the abstract compositions. Besides what is (and what is not) being monitored, another important question is what to do with the information produced by the tracking service. The Traceability and Auditing Center uses the default WF *SQLTrackingService*, which provides ready-to-use functionalities to record data to a SQL Server database. This recorded data can be later used for auditing purposes. In addition, *CoGPlat* implements a custom tracking service which allows the on-line monitoring of an executing composition.

4.4 Concluding Remarks

The proposal of effective solutions to increase the public administration transparency is becoming a key issue for the success of the new citizen-centric services and applications. Considering the challenges introduced by this new scenario, this paper contributes proposing a strategy to monitor and audit composite e-Government services. It includes

the proposal of a set of Traceability Policies, used to semantically annotate the services, specifying the desired traceability behavior. The paper also contributes by presenting an application scenario and by discussing its implementation, based on the *Windows Workflow Foundation Tracking Service*. The work presented in the paper is in the context of the *Traceability and Auditing Center*, one of the core facilities of the *Citizen-oriented e-Government Platform* (***CoGPlat***). Future works may include the extension of the traceability policies, the implementation of new applications and a study on the effects of the tracking mechanisms on the performance of time-critical compositions, not so frequent in the e-Government domain.

There are still many challenges to be faced before a fully citizen-centric e-Government becomes a reality. However, the visioned changes create the expectations of a future where public administration processes may become really transparent and efficient, stimulating the continuous research efforts in this field.

Acknowledgements

The authors would like to thank the Brazilian agencies CAPES and CNPq for the financial support. Additional thanks to the Fraunhofer-FOKUS Institute (Berlin, Germany).

Capítulo 5

E-Government and Grid Computing: Potentials and Challenges towards Citizen-Centric Services

5.1 Introduction

An increasing demand for the delivery of integrated and efficient public services has been observed worldwide over the recent years and applications in the area of e-Government, e-Democracy and e-Participation are gaining momentum. A recent UN (United Nations) report stated that the "*strategic and meaningful application of Information and Communication Technologies for the purpose of improving the efficiency, transparency, accountability and accessibility of government is possible if the ultimate objective of e-government is to promote social inclusion*" [Ahmed 06].

Grid computing intends to provide a vehicle for high computation and massive storage, and no single application domain can be excluded from its potential benefits [Maad 05], scalability being the most cited one. In addition, the convergence between Grids and Service-oriented Architectures (SOA) is transforming the Grid into a powerful solution for the integration of heterogeneous cross-domain applications.

This position paper intends to investigate the state of the art concerning the use of Grid technologies in e-Government scenarios, the strategies, challenges and what still demands further research. It is organized as follows: Section 2 presents a discussion on e-Government, its applications and requirements; Section 3 introduces Grid computing and shows how it could serve as a supporting platform for e-Government applications; Section 4 critically presents the current challenges and open issues towards a full Grid support for e-Government; and finally Section 5 presents our final remarks.

5.2 e-Government

The term Electronic Government (*e-Government*), as an expression, was coined after the example of Electronic Commerce. It designates a field of activity that has been with us for several decades and which has attained a high level of penetration in many countries [Lenk 02]. E-Government can be defined as being all processes which serve decision-making and services in politics, government and administration and which use information and communication technologies [KBSt 06]. What has been observed over the recent years is a shift on the broadness of the e-Government concept: the demand now is not anymore to deliver traditional services on-line, but to deliver new and dynamic services, which are citizen-centric. In addition, the need for citizen participation in governmental processes and decisions is gaining momentum as a way to enforce governments' transparency and legitimacy, a phenomenon called *e-Participation*.

A citizen-centric government can be simply described as one that treats citizens and businesses like customers, so that their needs come first, rather than bureaucracy or other imperatives inside the government machine. It considers the following principles [GOV3 06]:

1. Treat citizens and businesses as customers of the government as a whole (and not only of a specific agency);

2. Use a government-wide service-oriented architecture to support all interactions;

3. Develop a single place for citizens to get all government information and transactional services;

4. Do not expect to get it right first time, but aim to move quickly and learn from experience.

5.2.1 Applications

The e-Government applications can be classified into the following categories:

- **e-Services**: On-line delivery of traditional services for citizens (tax declarations, document requests etc) and initiatives like One-Stop Government Portals;

- **e-Democracy / e-Participation**: Applications to improve citizen participation in the government decisions, including e-Voting, polls and discussion forums. Also, tools that increase the public administration transparency, like budget reports, are important in this category.

- **e-Business**: All on-line interactions between the public and the private sector, ranging from e-Procurement initiatives to legal and fiscal transactions;
- **e-Archiving**: Services for storage and retrieval of public documents, reports, and also public libraries.

These applications can be also classified according to three interaction levels [KBSt 06]:

1. *Information*: covers the provision of information to people, businesses and other elements of society. Users on this level merely act as recipients of information;
2. *Communication*: interactive and participation services which enable the exchange of news, messages and information. These services range from simpler solutions, such as e-mail or web-based discussion forums, right through to more complex applications, such as video conference systems;
3. *Transaction*: represent the highest interaction level and include, for instance, the electronic receipt and processing of applications or orders as well as the provision of forms which can be filled in on the computer and directly sent to the correct recipient. Electronic payment or tendering systems also belong to this category.

5.2.2 Requirements

Considering the technological aspects, the SAGA (*Standards and Architectures for e-Government Applications*) specification cites the following as key requirements for the success of any e-Government effort [KBSt 06]:

- **Interoperability**: the administration processes must be co-orientated so that the e-Government applications implemented can interact with each other. The interoperability can be further classified into three levels:
    1. *Organizational* interoperability primarily determines when and why certain data are exchanged;
    2. *Technical* interoperability refers to the mere possibility to exchange information, including the definition of transmission routes and protocols;
    3. *Semantic* interoperability exists when two systems exchange data in such a manner that the data is interpreted in the same way by both communication partners and misunderstandings are ruled out. This applies not just to the form but also to the content of the data transmitted.

- **Reusability**: reuse can take place on several different levels of abstraction, e.g. exchange of experience between agencies and the use of joint data and process models, architecture samples and central services.

- **Openness**: applications should feature well-defined and documented interfaces or be encapsulated in such a manner that they can be integrated via portals, and standards should be adopted whenever possible;

- **Scalability**: ensure the usability of applications as requirements change in terms of volume and transaction frequency;

- **Security**: the applications and the supporting middleware should guarantee that information can only be accessed, modified or published in compliance with a predefined (and possibly fine-tuned) security policy, preserving its confidentiality and integrity. In addition, some e-Government applications demand further security requirements like user anonymity, identity-theft protection and non-repudiation.

5.2.3 Approaches

Different approaches are found in the literature to implement e-Government applications and platforms. An approach for the semi-automated design of data flows between Web Services that are semantically described using different ontologies and data representations is introduced in [Barnickel 06]. The approach includes a rule-based mechanism for user transparent mediation between ontologies and is intended to be used in e-Government scenarios spanning multiple application domains. In [Medjahed 05] the authors present a system which automatically generates Web services customized to citizens' needs and also to government laws and regulations. The project, called WebSenior, proposes three levels of service customization: the Citizen level, the Service level and the User interface level. A metadata ontology, used to describe e-Government services and operations, is also introduced.

In [Senger 06] the authors propose the adoption of a grid computing platform as an enabling infrastructure for the development of large, distributed, e-democracy applications, illustrated with an example in the field of policy formulation. A semantically-enriched and service-oriented middleware for the support of e-Government applications is the proposal of [Santos 05]. In this project, called *CoGPlat*, new services are dynamically composed with the help of semantic descriptions and their execution is mediated through a set of interaction policies.

5.3 Grid Computing for e-Government

We consider a Grid to be a "*sharing environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities*"[Foster 03]. These resources (for instance computers, applications, data) are owned by various administrative organizations and shared under locally defined policies (this administrative domain is usually called a Virtual Organization, or VO).

The Open Grid Services Architecture (OGSA) represents an evolution towards a Grid system architecture based on Web services concepts and technologies. The Globus Toolkit [Foster 06] is a community-based, open-architecture, open-source set of services and software libraries that support Grids and Grid applications. It addresses issues of security, information discovery, resource management, data management, communication, fault detection, and portability. Its latest release, GT4, is fully based on Web Services and represents a definite step towards the convergence of Grid technologies and Service-oriented architectures.

This convergence to SOA, added to the inherent support for large scale processing and storage, makes the Grid an interesting middleware approach for the support of e-Government applications. In Table 5.1 we present a relationship between the e-Gov application requirements presented in Section 5.2.2 and how the Grid infrastructures support (or not) them.

Tabela 5.1: e-Government x Grid

Requirement	**Support**	**No Support**
Interoperability	Technical	Semantic
Reusability	Yes	-
Openness	If standards are used	-
Scalability	Processing and Storage	Reliable Descentralized Control
Security	Authentication, Message Protection	Specific e-Gov Application Security Requirements

The *interoperability* requirements are satisfied by the Grid mostly in the technical level, due to the use of approaches like SOA and technologies like Web Services. The lack of semantic support is further discussed in the next Section. The *reusability* requirements are much more related to the application/service model itself than to the middleware infrastructure, but we believe the Grid, due to its cooperative and relative open environment facilitates the reuse of services. Concerning the third requirement, *openness*, the use of standards and well defined interfaces are supported by the latest Grid solutions, like the GT4.

The fourth requirement, *scalability*, is definitely considered the biggest Grid advantage over other middleware approaches, but it is important to note that this scalability is only an advantage in terms of processing power and storage space. The e-Government process security requirements impose a limit to this scalability in terms of process control, i.e., a fully decentralized control is not acceptable for most e-Gov scenarios and therefore this represents a bottleneck for the growth of the Grid (what is not true in other application domains). The last requirement, *security*, has very good support in the Grid environments when we consider user authentication and message protection/cyphering. Next section discusses why additional e-Government security requirements are still not fully supported by the Grid.

5.4 Challenges

In the previous section we discussed the relationships between the Grid infrastructure and the middleware requirements imposed by the applications in the e-Government domain, showing what the Grid supports and what it does not support. In this section we detail our analysis focusing on the main challenges for the requirements interoperability, security and scalability (see Table 5.2 for a summary of the open issues presented in this Section).

Tabela 5.2: Challenges towards a full Grid Support for e-Gov

Interoperability	- Semantic descriptions + Ontology Support - Workflow (DAG) Support + efficient fault Tolerance mechanisms - Accessibility
Security	- Fine-tuned information access policies - Anonymity and Identity-theft protection mechanisms
Scalability	- Correct VO model choice - Enhanced control mechanisms - Traceability + Auditing support

5.4.1 Interoperability

The so-called Semantic Web Services have gained momentum over the last years and are being considered as powerful (or even essential) mechanisms to enable fully interoperable services (see Section 5.2.2). The actual Grid implementations still lack support for semantic descriptions and ontologies and this is certain an issue that is demanding efforts from the research community.

In addition, most Grid platforms offer full support only for "Bag-Of-Task"applications, i.e., applications composed of independent tasks that do not need to communicate within each other in order to complete. That is for sure a big challenge to be faced, as e-Government processes are usually described as workflows or DAGs (Direct Acyclic Graphs) and many of its tasks are interdependent. There are already some alternatives, like the use of a specific scheduler (e.g. CONDOR over Globus), but they are still poor in handling aspects like fault tolerance and recovery. A scalable and autonomic management of the Grid (including efficient fault handling mechanisms) remains a goal, and experiences are being learned from solutions implemented in approaches like the Peer-to-Peer networks [Foster 03].

Last, but not least, middleware-level support for mobile devices required by some e-Government applications, added to accessibility mechanisms for disabled and senior citizens is still subject of research.

5.4.2 Security and Scalability

Due to the nature of the data and processes involved, many e-Government applications have specific security requirements, some of them determined by local specific legislation. Though some Grid implementations like the GT4 have very good security mechanisms, usually focused on authentication and message privacy, the support for demands like anonymity and identity-theft protection still need enhancements. Also, the support for fine-tuned information access policies fully integrated to the service environments needs to be improved.

If we consider Grid environments based on Peer-to-Peer infrastructures and also opportunist Grid approaches, further security issues must be considered. These issues are mainly related to the lack of control over the tasks imposed by the inherent decentralization of these approaches.

Besides that, a critical issue for a successful e-Government support over the Grid seems to be the correct modelling of the Virtual Organizations. This is strongly dependent on the application category (see Section 5.2.1), on the interaction model and also on the participating entities. For example, if we consider a global e-Procurement application, that can be classified in the category e-Business/Transaction, a VO could be defined having as members all Governmental agencies together with all credited private business partners/suppliers. On the other hand, if the application fits in the category e-Democracy/Transaction, the VO would be formed only by the Government departments involved in decision making processes.

Finally, the ability to follow on-line the status of the different running processes and the possibility of performing late auditing procedures is also required by many e-Government

applications as a way to increase the public administration transparency. On one hand, this requires both extra storage and processing power, which might be successfully provided by the Grid. On the other hand, this represents an extra control issue that should be better solved in the middleware (Grid) level and not left to the applications.

5.5 Final Remarks

We believe that the proposal of effective solutions to enable full interoperability among heterogeneous and inter-organizational systems remains a key issue in the development of new citizen-centric e-Government services. Grid computing's promise to provide a vehicle for high computation and massive storage added to its recent convergence towards service-orientation has transformed it into an interesting middleware solution for supporting these new applications.

On one hand many inherent characteristics of the Grid middlewares are of great importance for e-Government applications, namely processing and storage scalability and also the support for technological interoperability. On the other, there are some important issues that still need to be handled: support for complex workflow-based processes, enhanced fault tolerance mechanisms, specific security requirements and support for semantic and ontologies for instance.

Nevertheless, we believe the potentials for the Grid are very high and as long as these challenges are faced, it may become a very solid and powerfull middleware solution for e-Government applications.

Acknowledgements

The authors would like to thank the Brazilian agencies FAPESP, CAPES and CNPq, and the project GigaBOT for the support.

Capítulo 6

Conclusão

São muitas as etapas a serem completadas até que as novas idéias propostas no contexto do Governo Eletrônico Centrado no Cidadão se tornem realidade. As transformações por elas prometidas criam a expectativa de um futuro com processos públicos mais transparentes e eficientes e servem de grande incentivo para a continuidade da pesquisa nesta área. Levando em conta alguns dos desafios a serem vencidos, esta tese, organizada como uma coletânea de artigos, discutiu um conjunto de mecanismos que visam facilitar a construção de novos serviços e aplicações de Governo Eletrônico.

6.1 Contribuições

O Capítulo 2 apresentou uma ampla revisão bibliográfica e conceitual de temas relacionados à composição dinâmica de serviços. Este capítulo contribuiu propondo uma classificação clara das diferentes estratégias de composição de serviços a partir da análise de seus níveis de dinamismo e automatização. Outra importante contribuição é a apresentação de uma abordagem genérica orientada a modelos e que serviu também de ilustração para diferentes técnicas que objetivam aumentar o dinamismo das composições.

Já o Capítulo 3 introduziu as principais contribuições desta tese, relacionadas com o *middleware* ***CoGPlat*** (**C**itizen-**o**riented e-**G**overnment **Plat**form), uma plataforma orientada a serviços que tem como principal objetivo facilitar o desenvolvimento de aplicações que permitam a interação e a colaboração entre entidades governamentais, organizações (públicas, privadas e ONGs) e cidadãos. Recursos da Web Semântica são utilizados para descrever as funcionalidades dos serviços e composições, aumentando o grau de dinamismo e permitindo uma maior flexibilidade e adaptabilidade das aplicações. Requisitos legais e não-funcionais das aplicações de Governo Eletrônico, como autonomia, privacidade e rastreabilidade, são atendidos através de um conjunto de políticas de interação.

Foram discutidos também aspectos relacionados com a implementação do protótipo do *middleware* e também de cenários de aplicação. A Seção 3.3.7 apresenta os passos necessários para que uma aplicação possa utilizar os recursos de composição de serviços oferecidos por *CoGPlat*. O artigo apresenta uma avaliação qualitativa da plataforma com destaque para os seguintes resultados:

1. A **complexidade** normalmente associada ao processo de definição e execução das composições foi transferida das aplicações para o *middleware*;

2. Uma mesma aplicação, quando bem projetada, pode ser utilizada em diferentes contextos devido à **flexibilidade** inerente ao processo de composição dinâmica de serviços de *CoGPlat*. Essa flexibilidade é alcançada principalmente pela adoção de descrições semânticas para os serviços;

3. A adoção das políticas de interação também aumenta a **flexibilidade** e o **dinamismo** das composições;

4. A orientação a serviços facilita a interoperabilidade do *middleware* com outras plataformas e aplicações.

O Capítulo 4 apresentou uma estratégia, ainda no contexto de *CoGPlat*, para aumentar a transparência dos processos públicos através de políticas de rastreabilidade e auditoria. Aspectos da implementação destes mecanismos também foram discutidos.

Finalmente, o artigo apresentado no Capítulo 5 investigou o estado da arte no tocante ao uso de Grades Computacionais no contexto de aplicações de Governo Eletrônico. Estratégias, desafios e questões em aberto foram discutidos, com destaque para interoperabilidade, segurança e escalabilidade.

6.2 Avaliação Geral das Estratégias Adotadas

A Seção 1.5 e a Tabela 1.2 apresentaram um conjunto de requisitos introduzidos pelos serviços de Governo Eletrônico Centrados no Cidadão e também apontaram as estratégias escolhidas neste trabalho de doutorado para atender a estes requisitos. A seguir é apresentada uma avaliação geral da experiência adquirida com a adoção destas estratégias, incluindo os sucessos e as dificuldades observadas.

Como as técnicas de composição adotadas em *CoGPlat* estão diretamente relacionadas com as principais contribuições da tese, um detalhamento maior é dedicado a elas, sendo então apresentada análise comparativa destas com trabalhos correlatos.

Orientação a Serviços e Padrões

A heterogeneidade inerente das plataformas e serviços de governo eletrônico já existentes foi tratada com a adoção de uma arquitetura orientada a serviços e padrões da Web. A estratégia mostrou-se bem sucedida principalmente por facilitar a interoperabilidade com outros sistemas e por utilizar como infra-estrutura de comunicação a Internet.

As principais dificuldades observadas com o uso desta abordagem foram as mesmas de qualquer outra solução baseada em serviços Web: imprevisibilidade no desempenho dos serviços externos que fazem parte das composições e maior chance de falhas. Isto precisa ser previsto e tratado de alguma forma, seja pelo projetista ao especificar as composições, utilizando por exemplo mecanismos de compensação, ou ainda pelas aplicações clientes que conhecem as peculiaridades do processo sendo acessado.

Estratégias de Composição de Serviços com Recursos da Web Semântica

O oferecimento de novos serviços compostos, com maior grau de transparência e envolvendo múltiplas organizações, foi tratado com a aplicação de estratégias de composição de serviços baseadas em recursos da Web Semântica. Os resultados foram positivos pois observou-se, como esperado inicialmente, um maior grau de dinamismo nas composições e também o reaproveitamento das composições abstratas em diferentes contextos. Além disso, o próprio processo de especificação das composições, normalmente efetuado por um *expert* no domínio, foi facilitado pelas ferramentas e estratégias oferecidas por *CoGPlat*.

As principais dificuldades encontradas surgiram do fato da Web Semântica ainda ser uma área nova, onde apesar de existir muita atividade de pesquisa, poucos são os padrões estabelecidos, quer seja pelo uso da maioria da comunidade (padrões *de facto*) ou por órgãos como W3C, IETF ou OMG. Como conseqüência, não há ainda ferramentas comerciais de desenvolvimento ou ainda plataformas que suportem em sua totalidade as especificações da Web Semântica. Outra dificuldade observada, relacionada diretamente com o processo de composição, foi o não-determinismo na escolha dos serviços concretos candidatos a implementar as atividades que fazem parte das composições abstratas. Devido a este fenômeno, foi de fundamental importância o uso das políticas de interação para garantir que todas os requisitos e normas determinadas pela composição e pela aplicação fossem observados pelos serviços participantes.

Diferentes trabalhos que aplicam técnicas da Web Semântica para facilitar a composição de serviços em diversos contextos, inclusive de Governo Eletrônico, podem ser encontrados na literatura (alguns destes já foram inclusive citados nos diferentes artigos que compõem esta coletânea) e têm a seguir suas principais contribuições comparadas com os mecanismos propostos na plataforma *CoGPlat*.

Um arcabouço baseado em ontologias para realizar a composição de maneira automática a partir de descrições em alto nível e usando **regras de compatibilidade** (sintática e semântica) entre os serviços é proposto por [Medjahed 03] e aplicado no projeto *WebSenior* [Medjahed 05]. Este gera automaticamente Serviços Web adaptados às necessidades dos cidadãos no contexto da seguridade social e introduz três níveis de **adaptação de serviços**: nível do cidadão, nível de serviço e nível de interface com o usuário. Na plataforma *CoGPlat*, além dos mecanismos de inferência lógica, o *matchmaker* (*OWLS-MX*) também avalia similaridades sintáticas dos termos e a adaptação é facilitada através da definição das composições a partir de atividades abstratas.

Em [Agarwal 05] uma **metodologia fim-a-fim** para composição de serviços é apresentada, onde dada uma **especificação formal** de um requisito para um novo serviço, componentes semanticamente anotados são agregados em um fluxo WS-BPEL. A abordagem possui três etapas: (1). Casamento de ontologias e tipos (composição lógica); (2). Seleção de Instâncias (composição física); (3). Implantação da composição em uma infra-estrutura descentralizada de orquestração. Já em *CoGPlat*, formalismos não são usados nas requisições de serviços, mas o suporte a solicitações em *Pi-Calculus*, com sua transformação em *WS-CDL / WS-BPEL / WF*, é considerado como uma possível futura extensão. O uso de formalismos pode ser encontrado também em [Narayanan 02], onde a semânti-ca de um subconjunto de OWL-S é escrita em lógica de primeira ordem e as descrições dos serviços são codificadas em um formalismo baseado em *Redes de Petri* (*Situation Calculus*), oferecendo procedimentos de decisão para a simulação, verificação e composição. Em *CoGPlat*, a notação adotada para descrever visualmente os processos é a proposta pelo *Windows Workflow Foundation*, com elementos funcionais similares aos de linguagens como WS-BPEL.

A abordagem proposta no modelo *CoSMoS* (*Component Service Model with Semantics*) [Fujii 04] integra informações semânticas e funcionais de um componente em um **grafo** e oferece um mecanismo denominado *SeGSeC* (*Semantic Graph based Service Composition*) que permite a um usuário solicitar um serviço a partir de uma **sentença em linguagem natural** (exemplo: "*imprima as instruções para ir de casa ao restaurante*"). Na estratégia adotada na plataforma *CoGPlat* o ponto de partida para as composições são os IOPEs (escritos como um perfil OWL-S) enviados pela aplicação. Já [Solanki 04] trata do enriquecimento das descrições semânticas dos serviços e propõe a definição de duas novas características relacionadas à composição: *assumption* e *commitment*. Estas são especificadas usando predicados em lógica de primeira ordem com operadores temporais que tratam somente do comportamento observável dos serviços e não dependem de qualquer conhecimento adicional sobre como estes são executados. As *composições abstratas* propostas em *CoGPlat* também pressupõem apenas conhecimento sobre o comportamento externo dos serviços.

Uma abordagem para o **projeto semi-automático** de fluxos de dados entre Serviços Web que são descritos semanticamente usando **ontologias distintas** é introduzida em [Barnickel 06]. Um mecanismo baseado em **regras** é proposto para a mediação entre as ontologias. Em *CoGPlat* o mapeamento entre ontologias não é tratado ainda.

Políticas de Interação

Aspectos relacionados com exigências da legislação e/ou com requisitos não funcionais de determinadas aplicações, como autonomia das entidades participantes em uma composição, a privacidade e segurança dos dados ou ainda o oferecimento de recursos para acompanhamento e auditoria dos serviços compostos foram tratados com a adoção um conjunto de Políticas de Interação. Além do impacto positivo no dinamismo e na flexibilidade das composições, as políticas também contribuíram muito ao aumentar a granularidade da especificação dos requisitos das aplicações.

Facilidades de Democracia e Governança Eletrônica

O *Centro de Democracia e Governança Eletrônica* teve, desde sua concepção, como principal meta simplificar o processo de desenvolvimento de novas aplicações ao oferecer, diretamente no *middleware*, um conjunto de funcionalidades básicas encontradas normalmente nos domínios da Democracia e Governança Eletrônica. Levando-se em conta os inerentes desafios de tal proposta, pode-se considerar os resultados obtidos com esta estratégia como sendo positivos, apesar de em parte ainda preliminares. As maiores dificuldades encontradas ocorreram principalmente na escolha e especificação de tais funcionalidades. Parte da razão está relacionada com a falta de uma legislação clara ou ainda de consenso sobre o assunto tanto dentre os pesquisadores da área quanto dentre os governos considerados pioneiros em sua aplicação. Apesar disso, o surgimento recente de fenômenos como a *Web 2.0* prometem revolucionar a forma como os cidadãos interagem com os órgãos governamentais e merecem ser estudados mais profundamente [Zappen 08, Lewis 06].

6.3 Considerações adicionais sobre a Implementação

Justificativa das escolhas Tecnológicas

O modelo da plataforma e as estratégias de composição de serviços apresentadas nesta tese podem, de maneira geral, ser implementadas sobre diferentes tecnologias. Levando isso em conta, justificamos a seguir as escolhas tecnológicas feitas no processo de desenvolvimento do protótipo de *CoGPlat*:

- Descrições semânticas em **OWL-S**: OWL-S foi escolhido como padrão para a descrição semântica dos Serviços Web em *CoGPlat* por sua maior proximidade com OWL (padrão do W3C) e também por sua maior adoção na comunidade acadêmica e na indústria. Além disso, no período inicial do trabalho foi possível encontrar ferramentas de desenvolvimento (acadêmicas) apenas compatíveis com OWL-S;
- **MS .Net Framework / C# / Windows Workflow Foundation**: a escolha foi motivada por questões de preferência pessoal e também pelo conjunto de ferramentas de desenvolvimento disponíveis. O desenvolvimento baseado em JAVA (ou outra linguagem orientada a objetos com suporte a serviços) seria igualmente viável já que o modelo do *middleware* foi concebido de maneira independente de plataforma. As vantagens alcançadas com a adoção do WF foram descritas nos capítulos anteriores: suporte nativo a requisitos não funcionais, possibilidade de alterar os *workflows* dinamicamente em tempo de execução, boa integração com o arcabouço de funcionalidades oferecidas pelo *.Net Framework* etc.

Políticas de Autonomia e Estratégia de Composição

As *Políticas de Autonomia* determinam qual a estratégia de composição a ser adotada (orquestração e/ou coreografia). Quando uma entidade possui autoridade sobre outra, a primeira pode querer orquestrar os serviços internos à segunda. Já quando não existe tal hierarquia, um mecanismo baseado em coreografia pode ser a escolha mais apropriada. A Tabela 6.1 apresenta um resumo desta relação.

Tabela 6.1: Política de Autonomia x Estratégia de Composição

	Política	**Estratégia**
(I)	*Autonomia Fraca*	Orquestração
(II)	*Autonomia Moderada*	Orquestração + Coreografia
(III)	*Autonomia Forte*	Coreografia

Status de Modelagem e Implementação do Protótipo

A Tabela 6.2 apresenta o status da modelagem e da implementação das diferentes funcionalidades do middleware *CoGPlat*.

Sítio Web do Projeto

Informações adicionais sobre o projeto *CoGPlat* (documentação técnica, código fonte etc) podem ser encontrados em *http://www.cogplat.org*. Em caso de dúvidas, o e-mail para contato com o autor é *ivo [dot] santos [at] acm [dot] org*.

Tabela 6.2: Status da Modelagem e Implementação

	Modelagem	**Implementação**
Centro de **Serviços Transparentes**	Completa	*De acordo com a modelagem*
Centro de **Gerência de Metamodelos**	Completa	*De acordo com a modelagem*
Centro de **Rastreabilidade e Auditoria**	Completa	*De acordo com a modelagem*
Centro de **Governança e Democracia Eletrônica**	Parcial	*Apenas serviços abstratos*
Serviços de Apoio	Completa	*Oferecidos nativamente pelo WF e habilitados de acordo com as necessidades de cada aplicação*
Camada de **Descoberta e Execução de Serviços**	Completa	*Descoberta de serviços via UDDI externo não implementada*
Políticas de Interação	Completa	*Sugestão de melhoria: uso de alguma linguagem padrão para especificação de políticas*

6.4 Colaboração com o Instituto Fraunhofer-FOKUS

O projeto *CoGPlat* teve parte de seu desenvolvimento realizado no Instituto Fraunhofer-FOKUS (Berlim, Alemanha) com o suporte financeiro de bolsa de doutorado-sanduíche do programa conjunto CAPES-DAAD-CNPq, durante 12 meses. As atividades realizadas neste período consistiram da análise do "estado-da-arte"dos trabalhos na área de Governo Eletrônico, do estudo de diversos projetos em andamento no próprio centro de pesquisa visitado, da modelagem da plataforma e do início de sua implementação. Como resultado da boa colaboração e da pesquisa conjunta, dois artigos [Santos 05, Fluegge 06] foram publicados em conferências internacionais em co-autoria com pesquisadores do instituto FOKUS.

6.5 Trabalhos Futuros

Sugestões de trabalhos futuros incluem:

- a proposta e implementação de um conjunto maior de aplicações que explorem as diferentes funcionalidades da plataforma. Um exemplo interessante seria uma

implementação do chamado *Orçamento Participativo*, sistema adotado por diversas prefeituras no Brasil;

- a implementação de um módulo de planejamento automático de composições com técnicas da Inteligência Artificial;
- implementação sobre outras tecnologias para mostrar o caráter genérico da arquitetura do *middleware*;
- comparação com outros sistemas de governo eletrônico;
- agregação de serviços adicionais tal como tolerância a falhas e comunicação assíncrona;
- exploração mais profunda das funcionalidades de Governança e Democracia Eletrônica;
- estudo de aspectos de usabilidade, incluindo por exemplo a solicitação de serviços através de linguagem natural;
- proposta de ferramentas de auxílio ao desenvolvedor baseadas em MDA e nas estratégias do Capítulo 2, mas integradas ao contexto do *middleware CoGPlat*;
- proposta de diferentes estratégias de implantação do núcleo de *CoGPlat*, com suporte a configuração dinâmica baseadas no perfil das aplicações;
- a adoção de uma estratégia híbrida de composição que reaproveite composições bem sucedidas realizadas no passado sempre que possível. Isto poderia diluir o custo computacional pago durante o processo de composição dinâmica;
- a realização de testes de desempenho do *middleware* levando em conta diferentes configurações de rede e de carga, variando por exemplo parâmetros como o número de entidades participantes, a quantidade de serviços disponíveis e a distribuição dos serviços.

Outra extensão interessante a este trabalho, inspirada nas discussões do Capítulo 5, é a implementação de uma plataforma baseada em *CoGPlat* sobre Grades Computacionais orientadas a Serviços, também conhecidos como *Grid Services* [Forum 05].

Apêndice A

Lista de Publicações

Os artigos a seguir foram publicados a partir das contribuições e resultados do doutorado[1]:

- **SANTOS, I. J. G.**; MADEIRA, Edmundo Roberto Mauro. **Transparency in Citizen-Centric Services - A Traceability-based approach on the Semantic Web**. Em: *Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS)*, Barcelona - Espanha, 2008. p. 184-189.
- **SANTOS, I. J. G.**; MADEIRA, Edmundo Roberto Mauro. **E-Government and Grid Computing: Potentials and Challenges towards Citizen-Centric Services**. Em: *Proceedings of 9th International Conference on Enterprise Information Systems (ICEIS)*, Funchal, Madeira - Portugal, 2007. p. 144-148.
- FLUEGGE, Matthias; **SANTOS, I. J. G.**; TIZZO, Neil Paiva; MADEIRA, Edmundo Roberto Mauro. **Challenges and Techniques on the Road to Dynamically Compose Web Services**. Em: *Proceedings of the Sixth International Conference on Web Engineering (ICWE)*, 2006, Palo Alto, CA - EUA. ACM Press, 2006. p. 40-47.
- **SANTOS, I. J. G.**; MADEIRA, Edmundo Roberto Mauro; TSCHAMMER, Volker. **Towards Dynamic Composition of e-Government Services**. Em: *Proceedings of 5th IFIP Conference on e-Commerce, e-Business and e-Government*, Poznan - Polônia. IFIP Series, Springer, 2005. p. 173-185.
- **SANTOS, I. J. G.**; MADEIRA, Edmundo Roberto Mauro. **CoGPlat: Using Composition to Enable Collaborative e-Government Services**. Em: *EU-LAT Workshop on e-Government and e-Democracy*, 2004, Santiago - Chile. e-Government and e-Democracy: Progress and Challenges, 2004. v. 8. p. 17-27.

[1] A lista inclui somente os artigos publicados até o final de Outubro de 2008.

Apêndice B

Ontologia dos Serviços e Políticas

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY p3 "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY daml "http://www.daml.org/2001/03/daml+oil#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY p1 "http://www.owl-ontologies.com/assert.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY p5 "http://www.isi.edu/~pan/damltime/time-entry.owl#" >
    <!ENTITY service
     "http://www.daml.org/services/owl-s/1.2/Service.owl#" >
    <!ENTITY process
     "http://www.daml.org/services/owl-s/1.2/Process.owl#" >
    <!ENTITY p4 "http://www.daml.org/services/owl-s/1.2/Profile.owl#" >
    <!ENTITY p2 "http://www.owl-ontologies.com/Ontology1149586626.owl#" >
    <!ENTITY grounding
     "http://www.daml.org/services/owl-s/1.2/Grounding.owl#" >
    <!ENTITY drs
     "http://cs-www.cs.yale.edu/homes/dvm/daml/drsonto040520.owl#" >
    <!ENTITY shadow-rdf
     "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#" >
    <!ENTITY expr
     "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl#" >
]>

<rdf:RDF xmlns="http://www.cogplat.org/CoGPlatMetaModel#"
     xml:base="http://www.cogplat.org/CoGPlatMetaModel"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
```

```
 xmlns:p5="http://www.isi.edu/~pan/damltime/time-entry.owl#"
 xmlns:p4="http://www.daml.org/services/owl-s/1.2/Profile.owl#"
 xmlns:p3="http://www.w3.org/2003/11/swrlb#"
 xmlns:p2="http://www.owl-ontologies.com/Ontology1149586626.owl#"
 xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
 xmlns:expr="http://www.daml.org/services/owl-s/1.2/generic/Expression.owl#"
 xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
 xmlns:drs="http://cs-www.cs.yale.edu/homes/dvm/daml/drsonto040520.owl#"
 xmlns:shadow-rdf=
  "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"
 xmlns:process="http://www.daml.org/services/owl-s/1.2/Process.owl#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:swrl="http://www.w3.org/2003/11/swrl#"
 xmlns:grounding="http://www.daml.org/services/owl-s/1.2/Grounding.owl#"
 xmlns:service="http://www.daml.org/services/owl-s/1.2/Service.owl#"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
    <owl:imports rdf:resource=
     "http://www.daml.org/services/owl-s/1.2/Profile.owl"/>
    <owl:imports rdf:resource=
     "http://www.daml.org/services/owl-s/1.2/Process.owl"/>
</owl:Ontology>
<owl:Class rdf:ID="cpAnonymousnessIdentityPolicy">
    <rdfs:subClassOf rdf:resource="#cpIdentityManagementPolicy"/>
    <owl:disjointWith rdf:resource="#cpTheftProtectionIdentityPolicy"/>
    <owl:disjointWith rdf:resource="#cpNonRepudiationIdentityPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpAutonomyPolicy">
    <rdfs:subClassOf rdf:resource="#cpInteractionPolicy"/>
    <owl:disjointWith rdf:resource="#cpPrivacyPolicy"/>
    <owl:disjointWith rdf:resource="#cpIdentityManagementPolicy"/>
    <owl:disjointWith rdf:resource="#cpTraceabilityPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpCollaboration">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&p2;hasPrivacyPolicy"/>
            <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
```

```
        <owl:Restriction>
            <owl:onProperty rdf:resource="&p2;hasParticipant"/>
            <owl:cardinality rdf:datatype="&xsd;int">2</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cpEntity"/>
<owl:Class rdf:ID="cpIdentityManagementPolicy">
    <rdfs:subClassOf rdf:resource="#cpInteractionPolicy"/>
    <owl:disjointWith rdf:resource="#cpAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpPrivacyPolicy"/>
    <owl:disjointWith rdf:resource="#cpTraceabilityPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpInteractionPolicy"/>
<owl:Class rdf:ID="cpModerateAutonomyPolicy">
    <rdfs:subClassOf rdf:resource="#cpAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpStrongAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpWeakAutonomyPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpModerateTrustPolicy">
    <rdfs:subClassOf rdf:resource="#cpPrivacyPolicy"/>
    <owl:disjointWith rdf:resource="#cpTotalTrustPolicy"/>
    <owl:disjointWith rdf:resource="#cpNoTrustPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpNonRepudiationIdentityPolicy">
    <rdfs:subClassOf rdf:resource="#cpIdentityManagementPolicy"/>
    <owl:disjointWith rdf:resource="#cpAnonymousnessIdentityPolicy"/>
    <owl:disjointWith rdf:resource="#cpTheftProtectionIdentityPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpNoTrustPolicy">
    <rdfs:subClassOf rdf:resource="#cpPrivacyPolicy"/>
    <owl:disjointWith rdf:resource="#cpTotalTrustPolicy"/>
    <owl:disjointWith rdf:resource="#cpModerateTrustPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpOperation">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&p2;isOperationOf"/>
            <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:Class rdf:ID="cpPrivacyPolicy">
    <rdfs:subClassOf rdf:resource="#cpInteractionPolicy"/>
    <owl:disjointWith rdf:resource="#cpTraceabilityPolicy"/>
```

```
    <owl:disjointWith rdf:resource="#cpIdentityManagementPolicy"/>
    <owl:disjointWith rdf:resource="#cpAutonomyPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpService">
    <rdfs:subClassOf rdf:resource="&service;Service"/>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:Class rdf:ID="cpStrongAutonomyPolicy">
    <rdfs:subClassOf rdf:resource="#cpAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpModerateAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpWeakAutonomyPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpStrongTraceabilityPolicy">
    <rdfs:subClassOf rdf:resource="#cpTraceabilityPolicy"/>
    <owl:disjointWith rdf:resource="#cpWeakTraceabilityPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpTheftProtectionIdentityPolicy">
    <rdfs:subClassOf rdf:resource="#cpIdentityManagementPolicy"/>
    <owl:disjointWith rdf:resource="#cpAnonymousnessIdentityPolicy"/>
    <owl:disjointWith rdf:resource="#cpNonRepudiationIdentityPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpTotalTrustPolicy">
    <rdfs:subClassOf rdf:resource="#cpPrivacyPolicy"/>
    <owl:disjointWith rdf:resource="#cpModerateTrustPolicy"/>
    <owl:disjointWith rdf:resource="#cpNoTrustPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpTraceabilityPolicy">
    <rdfs:subClassOf rdf:resource="#cpInteractionPolicy"/>
    <owl:disjointWith rdf:resource="#cpPrivacyPolicy"/>
    <owl:disjointWith rdf:resource="#cpIdentityManagementPolicy"/>
    <owl:disjointWith rdf:resource="#cpAutonomyPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpWeakAutonomyPolicy">
    <rdfs:subClassOf rdf:resource="#cpAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpModerateAutonomyPolicy"/>
    <owl:disjointWith rdf:resource="#cpStrongAutonomyPolicy"/>
</owl:Class>
<owl:Class rdf:ID="cpWeakTraceabilityPolicy">
    <rdfs:subClassOf rdf:resource="#cpTraceabilityPolicy"/>
    <owl:disjointWith rdf:resource="#cpStrongTraceabilityPolicy"/>
</owl:Class>
<owl:ObjectProperty rdf:about="&p2;hasOperation">
    <rdfs:domain rdf:resource="#cpService"/>
    <rdfs:range rdf:resource="#cpOperation"/>
    <owl:inverseOf rdf:resource="&p2;isOperationOf"/>
</owl:ObjectProperty>
```

```
    <owl:ObjectProperty rdf:about="&p2;hasParticipant">
        <rdfs:domain rdf:resource="#cpCollaboration"/>
        <rdfs:range rdf:resource="#cpEntity"/>
        <owl:inverseOf rdf:resource="&p2;isParticipantOf"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="&p2;hasPrivacyPolicy">
        <rdfs:domain rdf:resource="#cpCollaboration"/>
        <rdfs:range rdf:resource="#cpPrivacyPolicy"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="&p2;hasService">
        <rdfs:domain rdf:resource="#cpEntity"/>
        <rdfs:range rdf:resource="#cpService"/>
        <owl:inverseOf rdf:resource="&p2;isServiceOf"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="&p2;isOperationOf">
        <rdfs:domain rdf:resource="#cpOperation"/>
        <rdfs:range rdf:resource="#cpService"/>
        <owl:inverseOf rdf:resource="&p2;hasOperation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="&p2;isParticipantOf">
        <rdfs:domain rdf:resource="#cpEntity"/>
        <rdfs:range rdf:resource="#cpCollaboration"/>
        <owl:inverseOf rdf:resource="&p2;hasParticipant"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="&p2;isServiceOf">
        <rdfs:domain rdf:resource="#cpService"/>
        <rdfs:range rdf:resource="#cpEntity"/>
        <owl:inverseOf rdf:resource="&p2;hasService"/>
    </owl:ObjectProperty>
    <cpWeakAutonomyPolicy rdf:about="&p2;WeakAutonomyPolicy_17"/>
    <owl:ObjectProperty rdf:about="&swrl;argument2"/>
</rdf:RDF>
```

Apêndice C

Interfaces do Barramento de Serviços

A seguir são apresentados:

1. Diagramas UML e código fonte em C# das interfaces do Barramento de Serviços de *CoGPlat*;

2. Todos os serviços do barramento são acessados através de SOAP/HTTP (Web Services) e sua interface pública é no formato WSDL.

Figura C.1: Interface: Transparent Services Center

```
[ServiceContract(Name = "ITransparentServicesCenter",
Namespace = "http://services.cogplat.org")]

public interface ITransparentServicesCenter
{
[OperationContract]
  int createConcreteComposition(String applicationOWLSRequestURI);

  [OperationContract]
  int startCompositionInstance(int pid);

  [OperationContract]
  int pauseCompositionInstance(int pid);

  [OperationContract]
  int resumeCompositionInstance(int pid);

  [OperationContract]
  int abortCompositionInstance(int pid);

  [OperationContract]
  int startAllCompositionInstances();

  [OperationContract]
  int pauseAllCompositionInstances();

  [OperationContract]
  int resumeAllCompositionInstances();

  [OperationContract]
  int abortAllCompositionInstances();

  [OperationContract]
  Boolean isPolicyEnabledInComposition(int concreteCompositionId,
   int policyId);

  [OperationContract]
  int enablePolicyInComposition(int concreteCompositionId,
   int policyId);

  [OperationContract]
  int disablePolicyInComposition(int concreteCompositionId,
   int policyId);
}
```

Figura C.2: Interface: Metamodel Management Center

```
[ServiceContract(Name="IMetamodelManagementCenter",
Namespace="http://services.cogplat.org")]

public interface IMetamodelManagementCenter
{
[OperationContract]
int registerComposition(String uri, Boolean isAbstract);

[OperationContract]
  int removeComposition(int composition_id);

  [OperationContract]
  int removeCompositionURI(String uri);

  [OperationContract]
  int registerOWLOntology(String uri);

  [OperationContract]
  int removeOWLOntology(int ontology_id);

  [OperationContract]
  int removeOWLOntologyURI(String uri);

  [OperationContract]
```

```
  int registerOWLSProfile(String uri);

  [OperationContract]
  int removeOWLSProfile(int profile_id);

  [OperationContract]
  int removeOWLSProfileURI(String uri);

  [OperationContract]
  int compositionLookup(String requestProfileURI, Boolean isAbstract,
   out String abstractCompositionURI);

  [OperationContract]
  int semWebServiceLookup(String requestProfileURI,
   out String serviceURI);

  [OperationContract]
  int externalUDDIServiceLookup(String uddiRequest,
   out String serviceURI);
}
```

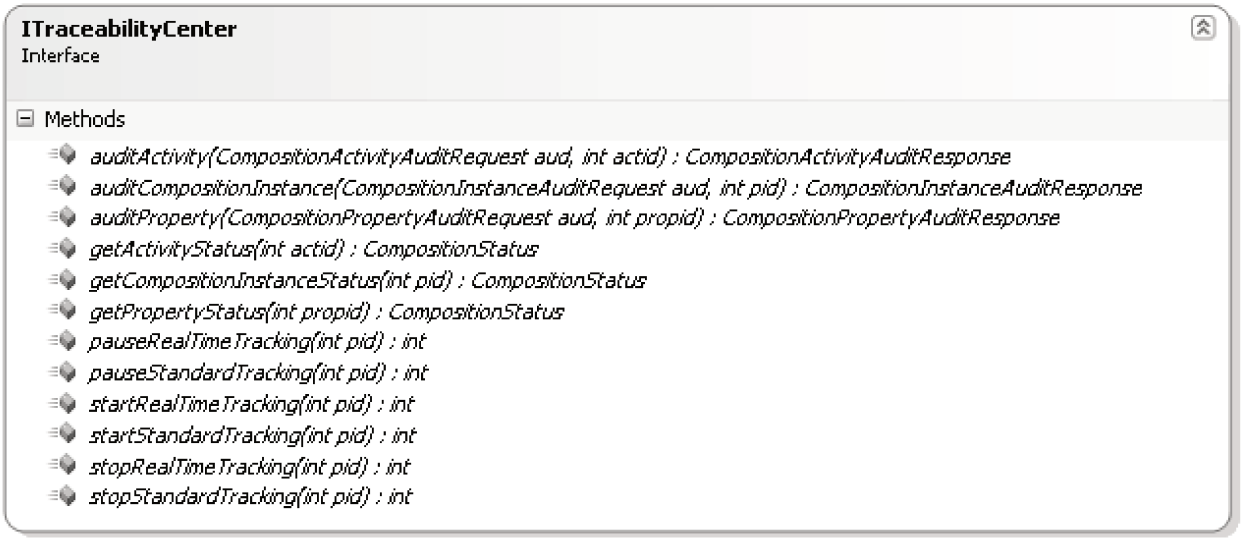

Figura C.3: Interface: Traceability and Auditing Center

```
[ServiceContract(Name = "ITraceabilityCenter",
Namespace = "http://services.cogplat.org")]
```

```
public interface ITraceabilityCenter
{
[OperationContract]
  int startRealTimeTracking(int pid);

  [OperationContract]
  int pauseRealTimeTracking(int pid);

  [OperationContract]
  int stopRealTimeTracking(int pid);

[OperationContract]
  int startStandardTracking(int pid);

  [OperationContract]
  int pauseStandardTracking(int pid);

  [OperationContract]
  int stopStandardTracking(int pid);

  [OperationContract]
  CompositionStatus getCompositionInstanceStatus(int pid);

  [OperationContract]
  CompositionStatus getActivityStatus(int actid);

  [OperationContract]
  CompositionStatus getPropertyStatus(int propid);

  [OperationContract]
  CompositionInstanceAuditResponse auditCompositionInstance(
   CompositionInstanceAuditRequest aud, int pid);

  [OperationContract]
  CompositionActivityAuditResponse auditActivity(
    CompositionActivityAuditRequest aud, int actid);

  [OperationContract]
  CompositionPropertyAuditResponse auditProperty(
   CompositionPropertyAuditRequest aud, int propid);
}
```

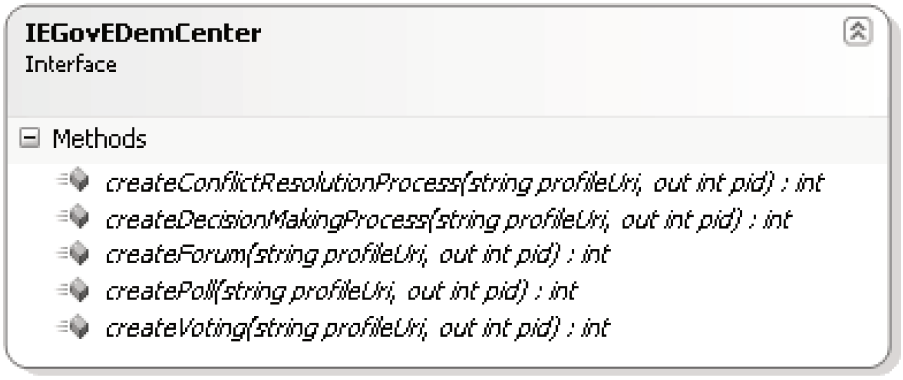

Figura C.4: Interface: E-Governance and E-Democracy Center

```
[ServiceContract(Name = "IEGovEDemCenter",
Namespace = "http://services.cogplat.org")]

public interface IEGovEDemCenter
{
[OperationContract]
  int createConflictResolutionProcess(string profileUri, out int pid);

  [OperationContract]
  int createDecisionMakingProcess(string profileUri, out int pid);

  [OperationContract]
  int createForum(string profileUri, out int pid);

  [OperationContract]
  int createPoll(string profileUri, out int pid);

  [OperationContract]
  int createVoting(string profileUri, out int pid);
}
```

Apêndice D

Serviço de Autorização para Construção de Casas

Exemplo de perfil OWL-S

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

<!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://services.cogplat.org/owl/BuildAuthorizationService.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;#">
```

```
  <owl:Ontology about="">
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&profile;" />
    <owl:imports rdf:resource="&process;" />
    <owl:imports rdf:resource="&actor;" />
    <owl:imports rdf:resource="&addParam;" />
    <owl:imports rdf:resource="http://services.cogplat.org/owl/cogplat_meta.owl"/>
    <owl:imports rdf:resource="http://services.cogplat.org/owl/civilConstruction.owl"/>
  </owl:Ontology>

  <profile:Profile rdf:ID="BuildAuthorizationService">
    <profile:serviceName>BuildAuthorizationService</profile:serviceName>
    <profile:textDescription>
This abstract composite service handles the authorization to build
a house process.
    </profile:textDescription>
    <!-- Descriptions of the parameters that will be used by IOPEs -->
      <profile:hasInput>
        <process:Input rdf:ID="CitizenData">
      <process:parameterType>http://services.cogplat.org/owl/civilConstruction.owl
      #RequesterData</process:parameterType>
      </process:Input>
      </profile:hasInput>

      <profile:hasInput>
        <process:Input rdf:ID="ConstructionAddress">
          <process:parameterType>http://services.cogplat.org/owl/civilConstruction.owl
          #Address</process:parameterType>
      </process:Input>
      </profile:hasInput>

      <profile:hasInput>
        <process:Input rdf:ID="ConstructionPlan">
          <process:parameterType>http://services.cogplat.org/owl/civilConstruction.owl
          #Plan</process:parameterType>
      </process:Input>
      </profile:hasInput>
      <profile:hasOutput>
        <process:UnConditionalOutput rdf:ID="BuildApprovalStatus">
          <process:parameterType>http://services.cogplat.org/owl/civilConstruction.owl
          #ApprovalStatus</process:parameterType>
        </process:UnConditionalOutput>
      </profile:hasOutput>
  </profile:Profile>
</rdf:RDF>
```

Referências Bibliográficas

[Agarwal 05] Vikas Agarwal, Koustuv Dasgupta, Neeran Karnik, Arun Kumar, Ashish Kundu, Sumit Mittal & Biplav Srivastava. *A service creation environment based on end to end composition of Web services*. Em WWW '05: Proceedings of the 14th international conference on World Wide Web, páginas 128–137, Chiba, Japão, 2005. ACM.

[Ahmed 06] Nahleen Ahmed. *An Overview of e-Participation Models*. Relatório técnico, United Nations Department of Economic and Social Affairs, Abril 2006.

[Ankolekar 06] Anupriya Ankolekar, David Martin, Deborah McGuinness, Sheila McIlraith, Massimo Paolucci & Bijan Parsia. *OWL-S' Relationship to Selected Other Technologies*. http://www.ai.sri.com/daml/services/owl-s/1.2/related.html, 2006.

[Antón 07] Annie I. Antón, Elisa Bertino, Ninghui Li & Ting Yu. *A roadmap for comprehensive online privacy policy management*. Communications of the ACM, vol. 50, no. 7, páginas 109–116, 2007.

[Barnickel 06] Nils Barnickel, Matthias Fluegge & Kay-Uwe Schmidt. *Interoperability in eGovernment through Cross-Ontology Semantic Web Service Composition*. Em Workshop Semantic Web for eGovernment / 3rd European Semantic Web Conference, Budva, Montenegro, Junho 2006.

[Belhajjame 08] Khalid Belhajjame, Suzanne M. Embury, Norman W. Paton, Robert Stevens & Carole A. Goble. *Automatic annotation of Web services based on workflow definitions*. ACM Trans. Web, vol. 2, no. 2, páginas 1–34, 2008.

[Berners-Lee 01] T. Berners-Lee, J. Hendler & O. Lassila. *The Semantic Web*. Scientific American, vol. 284, no. 5, páginas 34–43, Maio 2001.

[Berners-Lee 07] T. Berners-Lee. *The Semantic Web 'layercake' diagram*. http://www.w3.org/2007/03/layerCake.png, 2007. W3C.

[Borst 97] W. N. Borst. *Construction of Engineering Ontologies*. Tese de doutorado, University of Twente, Enschede, 1997.

[Brambilla 07] Marco Brambilla, Stefano Ceri, Federico Michele Facca, Irene Celino, Dario Cerizza & Emanuele Della Valle. *Model-driven design and development of semantic Web service applications*. ACM Trans. Internet Technol., vol. 8, no. 1, página 3, 2007.

[Bukovics 07] Bruce Bukovics. Pro wf: Windows workflow in .net 3.0. Apress, 2007.

[Bézivin 04] Jean Bézivin, Slimane Hammoudi, Denivaldo Lopes & Frédéric Jouault. *Applying MDA Approach for Web Service Platform*. Em Proc. of the 8th IEEE Intl. Enterprise Distributed Object Computing Conf. (EDOC 2004), páginas 58–70, Monterey, Califórnia, EUA, 2004.

[Cabral 06] Liliana Cabral, John Domingue, Stefania Galizia, Alessio Gugliotta, Vlad Tanasescu, Carlos Pedrinaci & Barry Norton. *IRS-III: A Broker for Semantic Web Services Based Applications*. Em 5th International Semantic Web Conference (ISWC), volume 4273 de *Lecture Notes in Computer Science*, páginas 201–214, Athens, GA, EUA, 2006. Springer.

[Casati 00] Fabio Casati, Ski Ilnicki, Li-Jie Jin, Vasudev Krishnamoorthy & Ming-Chien Shan. *eFlow: A Platform for Developing and Managing Composite e-Services*. HP Labs Technical Report HPL-2000-36, HP Software Technology Laboratory, Palo Alto, CA, Março 2000.

[Casati 01] F. Casati & M. Shan. *Dynamic and adaptive composition of e-services*. Information Systems, vol. 26, no. 3, páginas 143–163, 2001.

[CEFACT 03] CEFACT. *UN/CEFACT Modeling Methodology (UMM) User Guide CEFACT/TMG/N093.* http://www.unece.org/cefact/umm/umm_index.htm, 2003.

[Coalition 04] The OWL Services Coalition. *OWL-S: Semantic Markup for Web Services*. White paper - http://www.daml.org/services, Julho 2004.

[Curbera 03] F. Curbera, R. Khalaf, N. Mukhi, S. Tai & S. Weerawarana. *The next step in Web Services*. Communications of the ACM, vol. 46, no. 10, páginas 29–34, Outubro 2003.

[DAML 06] DAML. *OWL-S: Semantic Markup for Web Services*. http://www.daml.org/services/owl-s/, Março 2006.

[Davies 07] Jim Davies, Tomasz Janowski, Adegboyega Ojo & Aadya Shukla. *Technological foundations of electronic governance*. Em ICEGOV '07: Proceedings of the 1st international conference on Theory and practice of electronic governance, páginas 5–11, Macao, China, 2007. ACM.

[Dini 05] Paolo Dini, Neil Rathbone, Miguel Vidal, Pablo Hernandez, Pierfranco Ferronato, Gerard Briscoe & Stan Hendryx. *The Digital Ecosystem research Vision: 2010 and Beyond*. www.digital-ecosystems.org/events/2005.05/de_position_paper_vf.pdf, Julho 2005. Position Paper.

[Dogac 04] A. Dogac, Y. Kabak, G. Laleci, S. Sinir, A. Yildiz, S. Kirbas & Y. Gurcan. *Semantically enriched web services for the travel industry*. SIGMOD Rec., vol. 33, no. 3, páginas 21–27, 2004.

[Feier 05] Cristina Feier & John Domingue. *The Web Service Modeling Language WSML*. DERI International, WSML Final Draft, Abril 2005.

[Fellbaum 98] Christiane Fellbaum, editeur. Wordnet: An electronic lexical database (language, speech, and communication). The MIT Press, Maio 1998.

[Fensel 02] Dieter Fensel & Christoph Bussler. *The Web Service Modeling Framework WSMF*. Electronic Commerce Research and Applications, vol. 1, no. 2, páginas 113–137, 2002.

[Fensel 06] Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman & John Domingue. Enabling semantic web services: The web service modeling ontology. Springer-Verlag New York, Inc., Secaucus, NJ, EUA, 2006.

[Fileto 03] Renato Fileto, Ling Liu, Calton Pu, Eduardo Delgado Assad & Claudia Bauzer Medeiros. *POESIA: An ontological workflow approach for composing Web services in agriculture*. The VLDB Journal - The International Journal on Very Large Data Bases, vol. 12, no. 4, páginas 352–367, Novembro 2003.

[Flanders 06] Jon Flanders. *Atlas Workflow Designer*. http://www.masteringbiztalk.com/, 2006.

[Fluegge 04] Matthias Fluegge & Diana Tourtchaninova. *Ontology-derived Activity Components for Composing Travel Web Services*. Em International Workshop

on Semantic Web Technologies in Electronic Business (SWEB2004), Berlim, Alemanha, Outubro 2004.

[Fluegge 06] Matthias Fluegge, Ivo J. G. Santos, Neil Paiva Tizzo & Edmundo R. M. Madeira. *Challenges and techniques on the road to dynamically compose web services*. Em ICWE '06: Proceedings of the 6th international conference on Web engineering, páginas 40–47, Palo Alto, Califórnia, EUA, 2006. ACM Press.

[Forum 05] Global Grid Forum. *The Open Grid Services Architecture, Version 1.0*. http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf, Janeiro 2005.

[Foster 03] I. Foster & A. Iamnitchi. *On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing*. Em Proceedings of the 2nd International Workshop on Peer-toPeer Systems (IPTPS '03), páginas 129–138, Berkeley, CA, EUA, 2003.

[Foster 06] I. Foster. *Globus Toolkit Version 4: Software for Service-Oriented Systems*. Em IFIP International Conference on Network and Parallel Computing, volume 3779 de *LNCS*, páginas 2–13. Springer-Verlag, 2006.

[Frankel 05] David Frankel. *Scaling the Business Process Platform Up*. MDA Journal - http://www.bptrends.com, Dezembro 2005.

[Fujii 04] Keita Fujii & Tatsuya Suda. *Dynamic service composition using semantic information*. Em ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing, páginas 39–48, New York, NY, EUA, 2004. ACM.

[GOV3 06] GOV3. *Citizen Centric Government (White Paper)*. The GOV3 Foundation - Intel, www.intel.com/go/government, Fevereiro 2006.

[Gruber 93] T. R. Gruber. *A translation approach to portable ontology specifications*. Knowledge Acquisition, vol. 5, páginas 199–220, 1993.

[Gugliotta 08] Alessio Gugliotta, John Domingue, Liliana Cabral, Vlad Tanasescu, Stefania Galizia, Rob Davies, Leticia Gutiérrez-Villarías, Mary Rowlatt, Marc Richardson & Sandra Stincic. *Deploying Semantic Web Services-Based Applications in the e-Government Domain*. J. Data Semantics, vol. 10, páginas 96–132, 2008.

[IBM 05] IBM. *New to SOA and Web services*. http://www-128.ibm.com/developerworks/webservices/newto/, Julho 2005.

[IEEE 90] IEEE. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY, 1990.

[ISO 94] ISO. *ISO 8402:1994 - Quality management and quality assurance - Vocabulary*, 1994.

[ISO 05] ISO. *Use of UML for ODP system specifications*. ISO/IEC, JTC1/SC7 N3419. CD 19793.2, 2005.

[Jefferson 89] T. Jefferson. *Personal communication to R. Price*, 1789.

[Kagal 04] L. Kagal, T. Finin, M. Paolucci, Navcen Srinivasan, K. Sycara & G. Denker. *Authorization and privacy for semantic Web services*. Intelligent Systems, IEEE, vol. 19, no. 4, páginas 50–56, Julho-Agosto 2004.

[KBSt 06] KBSt. *SAGA - Standards and Architectures for eGovernment Applications - Version 3.0*. German Federal Ministry of Interior, www.kbst.bund.de/saga, Outubro 2006.

[Klusch 06] M. Klusch, B. Fries & K. Sycara. *Automated semantic web service discovery with OWLS-MX*. Em Proc. of the 5th Intl. joint conference on Autonomous agents and multiagent systems (AAMAS), páginas 915–922, Hakodate, Japão, 2006. ACM.

[Kopecký 07] Jacek Kopecký, Tomas Vitvar, Carine Bournez & Joel Farrell. *SAWSDL: Semantic Annotations for WSDL and XML Schema*. IEEE Internet Computing, vol. 11, no. 6, páginas 60–67, 2007.

[Küster 07] Ulrich Küster, Birgitta König-Ries, Mirco Stern & Michael Klein. *DIANE: an integrated approach to automated service discovery, matchmaking and composition*. Em WWW '07: Proceedings of the 16th international conference on World Wide Web, páginas 1033–1042, Banff, Alberta, Canadá, 2007. ACM.

[Lee 05] Sang M. Lee, Xin Tan & Silvana Trimi. *Current practices of leading e-government countries*. Commun. ACM, vol. 48, no. 10, páginas 99–104, 2005.

[Leite 98] J. C. Leite. *Imposto de Renda via Internet*. Em Anais do ENANPAD - Encontro Nacional da ANPAD, Foz do Iguaçu, PR, Brasil, 1998.

[Lenat 95] Douglas B. Lenat. *CYC: a large-scale investment in knowledge infrastructure*. Commun. ACM, vol. 38, no. 11, páginas 33–38, 1995.

[Lenk 02] K. Lenk & R. Traunmller. *Electronic government: Where are we heading?* Em EGOV 2002, volume 2456 de *LNCS*, páginas 1–9. Springer-Verlag, 2002.

[Lewis 06] Daniel Lewis. *What is web 2.0?* Crossroads, vol. 13, no. 1, páginas 3–3, 2006.

[Lin 05] Manshan Lin, Heqing Guo & Jianfei Yin. *Goal Description Language for Semantic Web Service Automatic Composition*. Em SAINT '05: Proceedings of the The 2005 Symposium on Applications and the Internet, páginas 190–196, Trento, Itália, 2005. IEEE Computer Society.

[Maad 05] Soha Maad, Brian Coghlan, John Ryan, Eamonn Kenny, Ronan Watson & Gabriele Pierantoni. *The Horizon of the Grid for e-Government*. Em eGovernment Workshop 05 (eGOV05), West London, UK, 2005.

[Marchionini 03] Gary Marchionini, Hanan Samet & Larry Brandt. *Digital Government*. Communications of the ACM, vol. 46, no. 1, páginas 25–27, Janeiro 2003.

[Martin 07] D. Martin & J. Domingue. *Semantic Web Services, Part 1*. Intelligent Systems, IEEE, vol. 22, no. 5, páginas 12–17, Sept.-Oct. 2007.

[Medjahed 03] Brahim Medjahed, Athman Bouguettaya & Ahmed Elmagarmid. *Composing Web services on the Semantic Web*. The VLDB Journal - The International Journal on Very Large Data Bases, vol. 12, no. 4, páginas 333–351, Novembro 2003.

[Medjahed 04] Brahim Medjahed. *Semantic Web Enabled Composition of Web Services*. Tese de doutorado, Virginia Polytechnic Institute and State University, 2004.

[Medjahed 05] Brahim Medjahed & Athman Bouguettaya. *Customized delivery of e-government Web services*. IEEE Intelligent Systems, vol. 20, no. 6, páginas 77–84, Novembro-Dezembro 2005.

[Milanovic 04] Nikola Milanovic & Miroslaw Malek. *Current Solutions for Web Service Composition*. IEEE Internet Computing, vol. 8, no. 6, páginas 51–59, Novembro-Dezembro 2004.

[Miller 95] George A. Miller. *WordNet: a lexical database for English*. Commun. ACM, vol. 38, no. 11, páginas 39–41, 1995.

[Mitra 06] Prasenjit Mitra, Chi-Chun Pan, Peng Liu & Vijayalakshmi Atluri. *Privacy-preserving semantic interoperation and access control of heterogeneous databases*. Em ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, páginas 66–77, Taipei, Taiwan, 2006. ACM.

[Moser 08] Oliver Moser, Florian Rosenberg & Schahram Dustdar. *Non-intrusive monitoring and service adaptation for WS-BPEL*. Em WWW '08: Proceeding of the 17th international conference on World Wide Web, páginas 815–824, Beijing, China, 2008. ACM.

[Mrissa 07] Michael Mrissa, Chirine Ghedira, Djamal Benslimane, Zakaria Maamar, Florian Rosenberg & Schahram Dustdar. *A context-based mediation approach to compose semantic Web services*. ACM Trans. Internet Technol., vol. 8, no. 1, página 4, 2007.

[Narayanan 02] Srini Narayanan & Sheila A. McIlraith. *Simulation, verification and automated composition of web services*. Em WWW '02: Proceedings of the 11th international conference on World Wide Web, páginas 77–88, Honolulu, Hawaii, EUA, 2002. ACM.

[OMG 03] OMG. *MDA Guide Version 1.01*. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, 2003.

[Papazoglou 03] M.P. Papazoglou & D. Georgakopoulos. *Service-Oriented Computing*. Communications of the ACM, vol. 46, no. 10, páginas 25–28, Outubro 2003.

[Patrascoiu 04] Octavian Patrascoiu. *Mapping EDOC to Web Services using YATL*. Em Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference (EDOC 2004), páginas 286–297, Monterey, Califórnia, EUA, 2004.

[Peltz 03] C. Peltz. *Web Services Orchestration and Choreography*. IEEE Computer, vol. 36, no. 10, páginas 46–52, 2003.

[Ponnekanti 02] Shankar R. Ponnekanti & Armando Fox. *SWORD: A developer toolkit for Web service composition*. Em Proceedings of the 11th World Wide Web Conference, Honolulu, EUA, 2002.

[Rao 04] Jinghai Rao & Xiaomeng Su. *A Survey of Automated Web Service Composition Methods*. Em Proceedings of 1st International Workshop on Semantic

Web Services and Web Process Composition, volume 3387 de *Lecture Notes in Computer Science*, páginas 43–54, San Diego, CA, EUA, Julho 2004. Springer.

[Ross-Talbot 05] Steve Ross-Talbot & N. Bharti. *Dancing with Web Services: W3C chair talks choreography*. http://searchwebservices.techtarget.com/, Março 2005.

[Ryu 08] Seung Hwan Ryu, Fabio Casati, Halvard Skogsrud, Boualem Benatallah & Régis Saint-Paul. *Supporting the dynamic evolution of Web service protocols in service-oriented architectures*. ACM Trans. Web, vol. 2, no. 2, páginas 1–46, 2008.

[Santos 05] Ivo J. G. Santos, Edmundo R. M. Madeira & Volker Tschammer. *Towards Dynamic Composition of e-Government Services - A Policy-based approach*. Em Proceedings of the 5th IFIP International Conference on e-Commerce, e-Business and e-Government (I3E), volume 189 de *IFIP*, páginas 173–185, Poznan, Polônia, Outubro 2005. Springer.

[Santos 06] Ivo J. G. Santos & Edmundo R. M. Madeira. *Applying Orchestration and Choreography of Web Services on Dynamic Virtual Marketplaces*. International Journal of Cooperative Information Systems (IJCIS), vol. 15, no. 1, páginas 57–85, Março 2006.

[Santos 07] Ivo J. G. Santos & Edmundo R. M. Madeira. *E-Government and Grid Computing: Potentials and Challenges towards Citizen-Centric Services*. Em Proc. of the 9th Intl. Conf. on Enterprise Information Systems (ICEIS), páginas 144–148, Portugal, 2007.

[Santos 08] Ivo J. G. Santos & Edmundo R. M. Madeira. *Transparency in Citizen-Centric Services - A Traceability-based Approach on the Semantic Web*. Em ICEIS 2008 - Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume SAIC, páginas 184–189, Barcelona, Espanha, Junho 2008.

[Schäfer 08] Michael Schäfer, Peter Dolog & Wolfgang Nejdl. *An environment for flexible advanced compensations of Web service transactions*. ACM Trans. Web, vol. 2, no. 2, páginas 1–36, 2008.

[Senger 06] H. Senger, F. A. B. Silva, M. de J. Mendes, R. Rondini & C. R. G. de Farias. *Grid Platforms for e-Democracy Applications*. Em Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06), páginas 334–339, Junho 2006.

[Shadbolt 06] N. Shadbolt, T. Berners-Lee & W. Hall. *The Semantic Web Revisited*. Intelligent Systems, IEEE, vol. 21, no. 3, páginas 96–101, 2006.

[Sirin 03] Evren Sirin, James Hendler & Bijan Parsia. *Semi-automatic composition of Web services using semantic descriptions*. Em Proceedings of Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003, páginas 17–24, Angers, France, 2003.

[Solanki 04] Monika Solanki, Antonio Cau & Hussein Zedan. *Augmenting semantic web service descriptions with compositional specification*. Em WWW '04: Proceedings of the 13th international conference on World Wide Web, páginas 544–552, New York, NY, EUA, 2004. ACM.

[SSP-SP 08] SSP-SP. *Delegacia Eletrônica*. http://www.ssp.sp.gov.br/bo/, 2008.

[Suchanek 07] Fabian M. Suchanek, Gjergji Kasneci & Gerhard Weikum. *Yago: a core of semantic knowledge*. Em WWW '07: Proceedings of the 16th international conference on World Wide Web, páginas 697–706, Banff, Alberta, Canadá, 2007. ACM.

[Tizzo 04] Neil Paiva Tizzo, José Renato Borelli, Manuel de Jesus Mendes, Luciano Damasceno, Aqueo Kamata, Adriana Figueiredo, Marcos Antonio Rodrigues & José Gonzaga Souza Junior. *Service Composition Applied to E-Government*. Em IFIP 18th World Computer Congress, Building the E-Service Society: E-Commerce, E-Business and E-Government, páginas 307–326, Toulouse, França, 2004. Kluwer Academic Publishers.

[Tokairim 03] Vera Tokairim & Agnaldo do Carmo Lopes. *E-poupatempo: ampliando os limites da prestação de serviços públicos*. Em VIII Congreso Internacional del CLAD sobre la Reforma del Estado y de la Administración Pública, Panamá, 2003.

[UNSPSC 08] UNSPSC. *United Nations Standard Products and Services Code*. http://www.unspsc.org, 2008.

[Uszok 04] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton & S. Aitken. *KAoS policy management for semantic Web services*. Intelligent Systems, IEEE, vol. 19, no. 4, páginas 32–41, Julho-Agosto 2004.

[W3C 04a] W3C. *OWL Web Ontology Language*. http://www.w3.org/TR/owl-features/, Fevereiro 2004.

[W3C 04b] W3C. *RDF Primer*. http://www.w3.org/TR/rdf-primer/, Fevereiro 2004.

[W3C 04c] W3C. *RDF Vocabulary Description Language 1.0: RDF Schema*. http://www.w3.org/TR/rdf-schema/, Fevereiro 2004.

[W3C 04d] W3C. *Web Services Glossary*. http://www.w3.org/TR/ws-gloss/, 2004.

[W3C 05] W3C. *Web Service Modeling Ontology (WSMO) Primer*. http://www.w3.org/Submission/WSMO-primer/, Junho 2005. Member Submission.

[W3C 07] W3C. *Semantic Annotations for WSDL and XML Schema (SA-WSDL)*. http://www.w3.org/TR/sawsdl/, Agosto 2007.

[Walsh 02] Aaron E. Walsh, editeur. Uddi, soap, and wsdl: The web services specification reference book. Prentice Hall Professional Technical Reference, 2002.

[Watson 01] Richard T. Watson & Bryan Mundy. *A Strategic Perspective of Electronic Democracy*. Communications of the ACM, vol. 44, no. 1, páginas 27–30, Janeiro 2001.

[Yu 07] Tao Yu, Yue Zhang & Kwei-Jay Lin. *Efficient algorithms for Web services selection with end-to-end QoS constraints*. ACM Trans. Web, vol. 1, no. 1, página 6, 2007.

[Zappen 08] James P. Zappen, Teresa M. Harrison & David Watson. *A new paradigm for designing e-government: web 2.0 and experience design*. Em dg.o '08: Proceedings of the 2008 international conference on Digital government research, páginas 17–26, Montreal, Canadá, 2008. Digital Government Society of North America.